Decomposition-based methods for Connectivity Augmentation Problems

by

Rian Neogi

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, 2021

Author's Declaration

Statement of Contributions

All the original results in this thesis are joint work with Joseph Cheriyan, Kanstantsin Pashkovich and Jack Dippel. References are given for any result proved in this thesis that is not original. This thesis uses figures that are borrowed from the papers of Adjiashvili [1] and Cecchetto et al. [6]. Permission to use these figures was granted by the authors of these papers.

Abstract

In this thesis, we study approximation algorithms for Connectivity Augmentation and related problems. In the Connectivity Augmentation problem, one is given a base graph $G = (V, E)$ that is k-edge-connected, and an additional set of edges $L \subseteq V \times V$ that we refer to as *links*. The task is to find a minimum cost subset of links $F \subseteq L$ such that adding F to G makes the graph $(k + 1)$-edge-connected. We first study a special case when $k = 1$, which is equivalent to the Tree Augmentation problem. We present a breakthrough result by Adjiashvili that gives an approximation algorithm for Tree Augmentation with approximation guarantee below 2, under the assumption that the cost of every link $\ell \in L$ is bounded by a constant. The algorithm is based on an elegant decomposition based method and uses a novel linear programming relaxation called the γ-bundle LP. We then present a subsequent result by Fiorini, Groß, Könemann and Sanità who give a $\frac{3}{2} + \epsilon$ approximation algorithm for the same problem. This result uses what are known as Chvátal-Gomory cuts to strengthen the linear programming relaxation used by Adjiashvili, and uses results from the theory of binet matrices to give an improved algorithm that is able to attain a significantly better approximation ratio. Next, we look at the special case when $k = 2$. This case is equivalent to what is known as the Cactus Augmentation problem. A recent result by Cecchetto, Traub and Zenklusen give a 1.393-approximation algorithm for this problem using the same decomposition based algorithmic framework given by Adjiashvili. We present a slightly weaker result that uses the same ideas and obtains a $\frac{3}{2} + \epsilon$ approximation ratio for the Cactus Augmentation problem. Next, we take a look at the integrality ratio of the natural linear programming relaxation for Tree Augmentation, and present a result by Nutov that bounds this integrality gap by $\frac{28}{15}$. Finally, we study the related Forest Augmentation problem that is a generalization of Tree Augmentation. There is no approximation algorithm for Forest Augmentation known that obtains an approximation ratio below 2. We show that we can obtain a $\frac{29}{15}$-approximation algorithm for Forest Augmentation under the assumption that the LP solution is half-integral via a reduction to Tree Augmentation. We also study the structure of extreme points of the natural linear programming relaxation for Forest Augmentation and prove several properties that these extreme points satisfy.

Acknowledgements

First and foremost, I would like to thank Joseph Cheriyan, my supervisor, for his guidance during my Master's program at University of Waterloo. I would also like to thank Kanstantsin Pashkovich and Jack Dippel for the various stimulating discussions on the Forest Augmentation problem that were held in the past year. Next, I would like to thank Chaitanya Swamy and Kanstantsin Pashkovich for their time and effort in reading this thesis. Finally, I would like to thank David Adjiashvili, Federica Cecchetto, Vera Traub and Rico Zenklusen for giving me permission to use figures from their papers in this thesis.

Table of Contents

List of Figures

Chapter 1

Introduction and Preliminaries

1.1 Introduction

One of the most fundamental and well-studied class of problems in Combinatorial Optimization is the class of *network design* problems where the task is find an optimal network subject to certain criteria. *Survivable network design* problems form a large subclass of network design problems. In survivable network design problem, one needs to find an optimal network subject to a robustness criteria, that is, the network should be well-connected even after the removal of certain nodes or edges. The class of survivable network design problems forms a subject of study in the field of approximation algorithms as most interesting problems in the class turn out to be NP-hard, and as such it is natural to consider the approximation variations of these problems. These problems have various applications in the domains of transportation and telecommunication. The subject of this thesis will be the study of a certain survival network design problem called the CONNECTIVITY AUGMENTATION problem.

We motivate our problem definition by an application. Suppose one needs to build a subway system for a city. A natural objective in building this subway system would be to minimize its cost, but at the same time, ensure that trains are able to find a route from any station A in the subway system to any other station B. That is, we want to find a minimum cost network such that the underlying graph is *connected*. This simple connectivity criterion is essential for passengers to be able to travel between any two stations in the city. While finding such a subway is computationally easy, such a simple criteria comes with certain drawbacks. It could be that, on a certain day, a station A might become unavailable to use for transit. This could be due to the fact that A is down for maintenance or it could be due to some system failure that may occur. In this case, every train that usually routes through A must need to find another route to its destination. Similarly, it could be the case that certain paths used by the subway become unavailable for use. For example, there might be a direct path in the subway connecting stations A and B which is down for maintenance on a particular day. In this case, any train that usually uses the path going from A to B must now find another route to its destination. We want to design our subway to satisfy a certain robustness criteria that says that even if a certain station or path fails, the subway network still remains connected, allowing passengers to be able to use to subway regardless of their destination. This robustness criteria can be generalized to a stronger requirement. For example, we may want that even if k stations or paths fail, the subway network still remains connected. In graph theoretical terms, this robustness criteria is equalivalent to ensuring that the underlying graph corresponding to the subway network is k-vertex-connected or k-edge-connected. In this thesis, we will be mostly concerned with the failure of direct paths between two stations in the subway, which corresponds to enforcing edge-connectivity in the underlying graph.

For an integer $k \in \mathbb{Z}$, a graph G is said to be *k-edge-connected* if G is connected and G remains connected even after removing at most $k - 1$ edges from G. The notion of k-edge-connectivity exactly captures the robustness criteria that we want. Standard results in graph theory tell us that a graph G is k-edge-connected if and only if for every pair of vertices $u, v \in V[G]$, there are at least k edge-disjoint $u - v$ paths in G. Following the motivation given earlier, our problem of building a robust subway system is equivalent to finding a k-edge-connected subgraph.

k-EDGE CONNECTED SPANNING SUBGRAPH
Input: Graph $G = (V, E)$, costs c_e for every edge $e \in E$, and an integer k.
Goal: Find $F \subseteq E$ of minimum cost such that (V, F) is k-edge-connected.

We won't be studying the above problem in this thesis. Instead, we will look at a related problem where the goal is to augment an existing network by adding edges so as to increase its connectivity. In the CONNECTIVITY AUGMENTATION problem, one is given a base graph $G = (V, E)$ that is k-edge-connected, and an additional set of edges $L \subseteq V \times V$ that we refer to as *links*. The task is to find a subset of links $F \subseteq L$ such that adding F to G makes the graph $(k + 1)$-edge-connected.

Connectivity Augmentation
Input: A k-edge-connected graph $G = (V, E)$, an additional set of links $L \subseteq V \times V$, and an integer k.
Goal: Find $F \subseteq L$ of minimum cardinality such that $(V, E \cup F)$ is $(k+1)$-edge-connected.

Following the real-life application of constructing a subway system, this problem corresponds to expanding the subway system so as to increase its robustness, and doing so in the cheapest way possible. One can additionally have costs c_ℓ on the links $\ell \in L$. This leads to the following weighted version of the problem.

Weighted Connectivity Augmentation
Input: A k-edge-connected graph $G = (V, E)$, an additional set of links $L \subseteq V \times V$, costs c_ℓ for every link $\ell \in L$, and an integer k
Goal: Find $F \subseteq L$ of minimum cost such that $(V, E \cup F)$ is $(k+1)$-edge-connected.

Consider the special case when $k = 1$. In this case, the base graph G is connected and the goal is to add links to make it 2-edge-connected. In fact, we may assume in this case that the base graph is a minimally connected, that is, it forms a spanning tree. Indeed, one can observe that if G contained a cycle C, then we may contract C into a single vertex to create a new graph G', and for any set of links $F \subseteq L$, we have that $G' + F$ is 2-edge-connected if and only if $G + F$ is 2-edge-connected. Thus the Connectivity Augmentation when specialized to $k = 1$ is equivalent to what is known as the Tree Augmentation problem, defined as follows.

Tree Augmentation
Input: A spanning tree $G = (V, E)$, an additional set of links $L \subseteq V \times V$.
Goal: Find $F \subseteq L$ of minimum cardinality such that $(V, E \cup F)$ is 2-edge-connected.

Similarly, specializing the Weighted Connectivity Augmentation problem by setting $k = 1$ leads to the following weighted version of Tree Augmentation.

Weighted Tree Augmentation
Input: A spanning tree $G = (V, E)$, an additional set of links $L \subseteq V \times V$, costs c_ℓ for every link $\ell \in L$.
Goal: Find $F \subseteq L$ of minimum cost such that $(V, E \cup F)$ is 2-edge-connected.

While there were many algorithms achieving approximation factor below two for Tree

Augmentation, the best known approximation algorithm for Weighted Tree Augmentation achieved factor two until the recent breakthrough work by Adjiashvili [1] in 2018 who gave a $1.964+\epsilon$ approximation algorithm for Weighted Tree Augmentation under the assumption that the costs are bounded by a constant. The ideas of Adjiashvili then led to a cascade of follow-up improvements in the approximation ratio for the problem for both the weighted and unweighted versions [6, 16, 21, 26, 30, 31]. In Chapter 2 we give an exposition of the algorithm of Adjiashvili, along with an improvement by Fiorini, Groß, Könemann, Sanita [16] who are able to achieve an approximation factor of $\frac{3}{2}+\epsilon$ for Weighted Tree Augmentation.

Now consider another specialization of the Connectivity Augmentation problem obtained by setting $k = 2$. Here, we are given a 2-edge-connected graph G and the goal is to add a set of links to make the graph 3-edge-connected. A 2-edge-connected graph is said to be a *cactus* if every edge is contained in exactly one cycle (see [17]). Just as in the case of $k = 1$, where can we apply reductions to make the base graph acyclic, in the case of $k = 2$, we can apply reductions to make the base graph into a cactus. Thus the Connectivity Augmentation problem, when specialized to $k = 2$, is equivalent to the following.

Cactus Augmentation
Input: A cactus graph $G = (V, E)$, an additional set of links $L \subseteq V \times V$.
Goal: Find $F \subseteq L$ of minimum cardinality such that $(V, E \cup F)$ is 3-edge-connected.

Classical results on the structure of minimum cuts imply that Connectivity Augmentation for odd k reduces to Tree Augmentation, and Connectivity Augmentation for even k reduces to Cactus Augmentation (see [13] and also [11, 24]). Moreover, Tree Augmentation can be reduced to Cactus Augmentation by simply making two parallel copies of every edge in the tree, so that the base graph becomes a cactus. Thus Cactus Augmentation is essentially equivalent to Connectivity Augmentation. Moreover, these reductions are approximation preserving, and so every approximation algorithm for Cactus Augmentation translates to an equivalent approximation algorithm for Connectivity Augmentation.

Following the algorithmic scheme given by Adjiashvili, a recent result by Cecchetto, Traub and Zenklusen [6] develops an algorithm for Cactus Augmentation that achieves an approximation factor of 1.393. The authors also provide a simpler $\frac{3}{2}+\epsilon$ approximation for the problem in the same paper. In Chapter 3, we give a high-level overview of this $\frac{3}{2}+\epsilon$ approximation algorithm.

Lastly, we consider a certain specialization of the k-edge connected spanning sub-

GRAPH problem known as the FOREST AUGMENTATION problem. Consider the special case of the k-EDGE CONNECTED SPANNING SUBGRAPH problem obtained by setting $k = 2$, thus the goal is to find a 2-edge-connected subgraph of the input graph G of minimum cost. The best known approximation algorithm for this problem achieves an approximation factor of two and can be obtained by a variety of techniques including iterative rounding and the primal-dual method [20, 23, 25]. For the unweighted version of the problem (that is, all edges have unit weight), better approximation algorithms are known [22, 29]. However, even if we restrict costs of the edges to be in $\{0, 1\}$, that is, we allow for zero weight edges along with the unit weight edges, no approximation factor better than two is known. Under this setting, consider the set $Z = \{e \in E[G] : c_e = 0\}$ of zero cost edges. We may assume that every optimal solution includes every edge in Z since we may always add any edge from Z to our solution without increasing its cost. Thus we may think of the graph $(V[G], Z)$ as our base graph, and the goal is to add edges from $E \setminus Z$ so as to make the graph 2-edge-connected. We may additionally assume that Z contains no cycles, since using the same argument as in TREE AUGMENTATION, we may always contract any cycle C in Z while preserving the optimal solution. Thus the base graph $(V[G], Z)$ may be assumed to be acyclic. Note that if the base graph $(V[G], Z)$ forms a spanning tree, then this is precisely the TREE AUGMENTATION problem. However, in general the base graph might be disconnected, in which case it forms a forest. This motivates the definition of the following problem.

FOREST AUGMENTATION
Input: A forest $G = (V, E)$, and an additional set of links $L \subseteq V \times V$.
Goal: Find $F \subseteq L$ of minimum cardinality such that $(V, E \cup F)$ is 2-edge-connected.

As argued earlier, FOREST AUGMENTATION is a specialization of the k-EDGE CONNECTED SPANNING SUBGRAPH problem obtained in the case when $k = 2$ and the cost c_e lies in $\{0, 1\}$, for every edge $e \in E[G]$. No approximation algorithm beating approximation factor two is known for FOREST AUGMENTATION. In Chapter 5, we explore the FOREST AUGMENTATION problem and prove some new partial results. While we do not beat factor two for the problem, our hope is that our study of the problem could lead to an improvement on the best approximation ratio for the problem in the future.

The rest of this thesis is structured as follows. In Section 1.2, we go over basic definitions and preliminaries that are used throughout the thesis. In Chapter 2, we cover the result of Adjiashvili for WEIGHTED TREE AUGMENTATION under the bounded costs assumption, along with the follow-up result by Fiorini, Groß, Könemann and Sanita. In Chapter 3, we give a high-level overview of the result by Cecchetto, Traub and Zenklusen. Then, in

Chapter 4, we cover a result by Nutov [26] that studies the integrality gap of the natural linear programming relaxation for the Tree Augmentation problem. In Chapter 5, we study the Forest Augmentation problem. We give an algorithm that beats approximation factor two for a special case and additionally, we analyze the structure of extreme point solutions of the natural linear programming relaxation for the problem. Finally, in Chapter 6, we conclude the thesis and provide some directions for future research.

1.2 Preliminaries

Graphs and Vectors

We use mostly standard notation for graphs and vectors. Graphs in this thesis will be undirected unless explicitly stated otherwise. Given a graph G, by $V[G]$, we denote the set of vertices of G, and by $E[G]$, we denote its set of edges. For a subset of vertices $S \subseteq V[G]$, by $G[S]$, we denote the subgraph of G induced on S. That is, $G[S]$ is the graph with vertex set S and edge set $\{uv \in E[G] : u, v \in S\}$. For a vertex $v \in V[G]$, by $G - v$, we denote the graph $G[V[G] \setminus \{v\}]$. For a set of vertices $S \subseteq V[G]$, by $G - S$, we denote the graph $G[V[G] \setminus S]$. For an edge $e \in E[G]$, by $G - e$, we denote the graph $(V[G], E[G] \setminus \{e\})$, and similarly, for a subset of edges $F \subseteq E[G]$, by $G - F$, we denote the graph $(V[G], E[G] \setminus F)$.

For a subset of vertices $S \subseteq V[G]$, by $\delta_G(S)$, we denote the set of edges with one endpoint in S and the other not in S, that is, $\delta_G(S) = \{uv \in E[G] : u \in S, v \notin S\}$. We drop the subscript G if the graph is clear from context. For a vertex $u \in V[G]$, we use $\delta_G(u)$ as a shorthand for $\delta_G(\{u\})$. Furthermore, if the graph G is clear from context and if $F \subseteq E[G]$ is a subset of edges of G, then we use $\delta_F(S)$ as a shorthand for $\delta_{G'}(S)$, where $G' = (V[G], F)$ denotes the graph G restricted to edges in F.

For a directed graph D, and a subset of vertices $S \subseteq V[D]$, we define $\delta^+_D(S) = \{uv \in E[D] : u \in S, v \notin S\}$ and $\delta^-_D(S) = \{uv \in E[D] : u \notin S, v \in S\}$. We drop the subscript D if the directed graph is clear from context. For a vertex $u \in V[D]$, we use $\delta^+_D(u)$ as a shorthand for $\delta^+_D(\{u\})$ and $\delta^-_D(u)$ as a shorthand for $\delta^-_D(\{u\})$.

For a graph G and an edge $e = uv \in E[G]$, the *contraction* of e results in a graph G/e obtained by removing vertices u, v from G and adding a new vertex w that is incident to all the edges in $(\delta(u) \cup \delta(v)) \setminus \{e\}$. For a subset of edge $F \subseteq E[G]$, by G/F we denote the graph obtained from G by contracting every edge in F. An edge $e \in E[G]$ is said to be a *bridge* of G if the removal of e from G increases the number of connected components of G.

Throughout this thesis, $\mathbb{R}$ denotes the set of real numbers and $\mathbb{Z}$ denotes the set of integers. We use $\mathbb{R}_+$ to denote the set of non-negative reals and $\mathbb{Z}_+$ to denote the set of non-negative integers. For a vector $x \in \mathbb{R}^E$ defined over the universe E and for a subset $F \subseteq E$ of the elements, we define $x(F) = \sum_{e \in F} x_e$. We denote the set of non-zero coordinates of x, called the *support* of x, as $supp(x) = \{e \in E : x_e \neq 0\}$. Moreover, the *restriction* of x on the subset F is defined to be the F-dimensional vector $y \in \mathbb{R}^F$ where $y_e = x_e$ for every $e \in F$. By $\chi^F \in \mathbb{R}^E$, we denote the indicator vector of F, that is, $\chi^F_e = 1$ if $e \in F$ and $\chi^F_e = 0$ otherwise.

Linear Programming Relaxations

Throughout this thesis, we use `OPT` to denote a (fixed) optimal solution for our problem and `opt` to denote the cost of this solution.

An often used technique in the field of approximation algorithms is using a linear programming relaxation of the problem as a lower bound. Most of the algorithms that we present in this thesis will use LP based lower bounds for their analysis. We now look at a natural LP relaxation for the family of Weighted Connectivity Augmentation problems. Recall that a graph G is k-edge-connected if and only if, for every pair of vertices $u, v \in V[G]$, there exists at least k edge-disjoint $u - v$ paths in G, which in turn happens if and only if every edge cut in the graph has cardinality at least k. This leads to the following natural LP relaxation for the Weighted Connectivity Augmentation problem with base graph $G = (V, E)$, set of links L and costs c_ℓ for every $\ell \in L$.

$$\begin{array}{lrcll} \text{minimize} & \sum_{\ell \in L} c_\ell x_\ell & & & \\ \text{subject to} & x(\delta_L(S)) & \geq & k + 1 - |\delta_E(S)| & \forall S \subset V, S \neq \emptyset \\ & x_\ell & \leq & 1 & \forall \ell \in L \\ & x_\ell & \geq & 0 & \forall \ell \in L \end{array} \tag{1.1}$$

We refer to this LP as the cut-LP. The constraints enforce that every cut $\delta(S)$ in the graph $(V, E \cup L)$ has size at least $k + 1$, implying that the graph $(V, E \cup L)$ is $(k + 1)$-edge-connected. Thus the integral solutions of the above LP correspond to exactly the feasible solutions $F \subseteq L$ for the Weighted Connectivity Augmentation problem. Note that we can specialize this relaxation to get an LP relaxation for Tree Augmentation or Cactus Augmentation by setting $k = 1$ or $k = 2$ respectively. While this LP has an exponential number of constraints, it can be solved in polynomial time using the ellipsoid method as it admits a polynomial time separation oracle as follows. Given an input vector

$x \in [0,1]^L$ to the separation oracle, in polynomial time, we compute a minimum cut $\delta(S^*)$ in the weighted graph $(V, E \cup L)$ where the weight for an edge $e \in E$ is 1 and the weight for a link $\ell \in L$ is x_ℓ. If the weight of the minimum cut $\delta(S^*)$ is at least $k+1$, then we have $x(\delta_L(S)) + |\delta_E(S)| \geq k+1$ for every $S \subseteq V$ and so every constraint in the LP is satisfied, and thus x is a feasible solution. Otherwise the weight of the cut $\delta(S^*)$ is less than $k+1$ in which case the vector x violates the constraint corresponding to S^*, which can be returned as a separating hyperplane. Thus the above LP has a polynomial time separation oracle and hence an optimal solution for it can be computed efficiently.

Total Unimodularity

Consider the following generic linear program,

$$\begin{array}{cl} \text{minimize (or maximize)} & c^\intercal x \\ \text{subject to} & Ax \geq b \\ & 0 \leq x \leq 1 \end{array}$$

Often, one is interested in finding integer optimal solutions to the above program. Unfortunately, the problem of optimizing over integer solutions of a linear program (known as the integer programming problem) is well-known to be NP-hard. On the other hand, there are many polynomial time algorithms that solve linear programs over the set of real solutions. It is well-known that if the above linear program has a solution, then it has one that is an extreme point of the polyhedron $P = \{x \in [0,1]^n : Ax \geq b\}$. If all the extreme points of P were to be integral, then one could find an integral optimal solution by finding the optimal solution over the reals. So it is interesting to study the conditions under which the extreme points of a polyhedron P become integral as it implies that we can solve the integer programming problem on these polyhedra efficiently.

One such condition is known as *total unimodularity.* A matrix $A \in \mathbb{R}^{m\times n}$ is said to be *totally unimodular* if every square submatrix B of A has determinant $\det(B) \in \{-1, 0, 1\}$. It is well-known that if a matrix is totally unimodular then all the extreme points of the polyhedron $P = \{x \in [0,1]^n : Ax \geq b\}$ are integral, for any $b \in \mathbb{Z}^m$ (for example, see [12]).

However, checking for total unimodularity is not easy as one needs to look at all possible submatrices. Thus one might ask for simple conditions under which a matrix becomes totally unimodular. The following theorem provides one such condition (see [12]).

Theorem 1.1. *Let A be a matrix with all entries in $\{-1, 0, +1\}$, where each column has at most one $+1$ and at most one -1. Then A is totally unimodular.*

The above theorem implies that in particular, incidence matrices of directed graphs are totally unimodular. This result can be extended as follows. Let $D = (V, E)$ be a directed graph and let $T = (V, E')$ be a directed spanning tree of D. Consider the matrix M having rows indexed by E' and columns indexed by E, where, for $e = uv \in E$ and $e' \in E'$, we have

$$M_{e',e} = \begin{cases} +1 & \text{if the } u - v \text{ path in } T \text{ uses } e' \text{ as a forward arc} \\ -1 & \text{if the } u - v \text{ path in } T \text{ uses } e' \text{ as a reverse arc} \\ 0 & \text{if the } u - v \text{ path in } T \text{ does not use } e' \end{cases}$$

Such a matrix is called a *network matrix.* Equivalently, M is a network matrix if M can be written as $M = T^{-1}R$ where $A = (TR)$ is the *truncated incidence matrix* of a directed graph D that is obtained from the incidence matrix of D by removing the row corresponding to some vertex $r \in V[D]$ (so that A has full-row rank), and T is a basis of A. It can be shown that network matrices are totally unimodular (see [12]).

Theorem 1.2. *Network matrices are totally unimodular.*

Additionally, the following theorem by Ghouila-Houri characterizes exactly when a matrix is totally unimodular (see also [17]).

Theorem 1.3 (Ghouila-Houri condition for total unimodularity)**.** *[19] A matrix $A \in \mathbb{R}^{m \times n}$ is totally unimodular if and only if for every subset of the rows $R \subseteq [m]$, there is a partition $R = R_1 \cup R_2$, such that for every $j \in [n]$, we have*

$$\sum_{i \in R_1} A_{ij} - \sum_{i \in R_2} A_{ij} \in \{-1, 0, 1\}$$

Or equivalently, the vector obtained by adding the rows in R_1 and subtracting the rows in R_2 has every coordinate in $\{-1, 0, 1\}$.

Chvátal-Gomory Cutting Planes

Let us suppose that we want to find a system of inequalities that defines the convex hull of a set S of integral vectors in $\mathbb{R}^n$. Quite often, it is easy to find a system of inequalities $Ax \geq b$ whose integral solutions are exactly the vectors in S. However, $Ax \geq b$ might contain fractional extreme points that do not lie in S which would imply that the system of inequalities $Ax \geq b$ does not define the convex hull of S. We would like refine this system of inequalities so as to cut off these fractional points, and hopefully get closer to attaining the integer hull of S.

One way to do this is a method known as *Chvátal-Gomory cutting planes*, or CG-cuts for short (see [12]). Given a system of m inequalities $Ax \geq b$, a CG-cut is an inequality of the form

$$\lambda^\intercal A x \geq \lceil \lambda^\intercal b \rceil \tag{1.2}$$

for some $\lambda \in \mathbb{R}^m_+$ such that $\lambda^\intercal A$ is an integral vector. Note that, since $\lambda \geq 0$, any solution x to the system must satisfy $\lambda^\intercal A x \geq \lambda^\intercal b$. Moreover, since $\lambda^\intercal A$ is integral, every *integral* solution x to the system satisfies the stronger inequality $\lambda^\intercal A x \geq \lceil \lambda^\intercal b \rceil$ given in (1.2). Thus the CG-cut (1.2) cuts off solutions x such that $\lambda^\intercal b \leq \lambda^\intercal A x < \lceil \lambda^\intercal b \rceil$. Since such solutions cannot be integral, the CG-cut preserves the feasibility of all the integral solutions of the system. Thus adding a CG-cut to the system of inequalities refines the system to as to prune away fractional points and brings it closer to defining the integer hull of feasible solutions.

T-joins

Let $G = (V, E)$ be a graph and let $T \subseteq V$ be such that $|T|$ is even. A T-join is a set of edges J such that

$$|J \cap \delta(v)| \equiv |T \cap \{v\}| \pmod 2, \text{ for all } v \in V$$

In other words, J is a T-join if the set of odd degree vertices of the subgraph (V, J) is exactly T.

A set $S \subseteq V$ is said to be T-odd if $|S \cap T|$ is odd. The set $\delta(S)$ for a T-odd set S is said to be a T-cut. T-cuts induce natural constraints on T-joins in the following way. Let J be a T-join and consider the subgraph $(S, J \cap E(S))$. It has an even number of nodes of odd degree, but in (V, J), S contains an odd number of nodes of odd degree (namely, those of $T \cap S$). Thus $|J \cap \delta(S)|$ must be an odd integer. In particular, we must have $|J \cap \delta(S)| \geq 1$. So the characteristic vector χ^J of J satsifies

$$x(\delta(S)) \geq 1, \text{ for all } T\text{-cuts } \delta(S) \tag{1.3}$$

Hence, the following linear program, which we call the T-join LP, provides a lower bound for the cost of any T-join.

$$\begin{array}{rrcll} \text{minimize} & \sum_{e \in E} c_e x_e & & & \\ \text{subject to} & x(\delta(S)) & \geq & 1 & \forall\ T\text{-cuts } \delta(S) \\ & x_e & \geq & 0 & \forall e \in E \end{array}$$

Edmonds and Johnson [14] proved that, if $c \geq 0$, then this lower bound is exact.

Theorem 1.4. *[14] Let $G = (V, E)$ be a graph and let $T \subseteq V$ be such that $|T|$ is even. Let $c \in \mathbb{R}^E$ be such that $c \geq 0$. Then the minimum cost of any T-join of G (with respect to costs c) is equal to the optimal value of the above LP.*

Additionally, it can be shown that the separation problem for the T-join LP can be solved in polynomial time. Given a graph $G = (V, E)$ and $T \subseteq V$ such that $|T|$ is even, and a non-negative vector $x \in \mathbb{R}^E$, the minimum T-cut problem asks to find T-cut Q such that $x(\delta(Q))$ is minimized. Padberg and Rao [27] give an algorithm that solves the minimum T-cut problem in polynomial time.

Theorem 1.5. *[27] There exists a polynomial time algorithm that solves the minimum T-cut problem.*

The above theorem can be used to design a separation oracle for the T-join LP as follows. Given a vector x we use the algorithm given by Theorem 1.5 to find a T-cut $\delta(S^*)$ that minimizes $x(\delta(S^*))$, if $x(\delta(S^*)) < 1$ then x violates the constraint $x(\delta(S^*)) \geq 1$ and we can return this constraint as a separating hyperplane. Otherwise we have $x(\delta(S)) \geq 1$ for every T-cut $\delta(S)$ which implies that x is feasible.

Chapter 2

An Approximation Algorithm for Tree Augmentation

In the Weighted Tree Augmentation problem, we are given a tree $G = (V, E)$ and an additional set of edges $L \subseteq V \times V$ called *links*, which have costs $c : L \to \mathbb{R}_+$. The goal is to find a subset of links $F \subseteq L$ such that $(V, E \cup F)$ is 2-edge-connected, and subject to that, $c(F)$ is minimized.

Weighted Tree Augmentation
Input: A tree $G = (V, E)$, a set of links $L \subseteq V \times V$, and costs c_ℓ for every link $\ell \in L$.
Goal: Find a minimum cost set of links $F \subseteq L$ such that the graph $(V, E \cup F)$ is 2-edge-connected.

This is a special case of Weighted Connectivity Augmentation where we wish to augment edge-connectivity of a graph from 1 to 2. This problem is known to be NP-hard [11, 18], even in the unweighted case. In this chapter, we present the breakthrough result by Adjiashvili [1] that beats approximation factor two for Weighted Tree Augmentation under the assumption that the costs are bounded by some constant M. This was the first improvement in the approximation ratio of the problem in over 35 years that did not restrict the structure of the tree or the links.

At a high level, the algorithm aims to decompose the input tree G into a collection of subtrees $T_1, \ldots, T_k$. Each of these subtrees will be easy to solve approximately. The algorithm then computes approximate solutions $F_1, \ldots, F_k$ to each of the subtrees $T_1, \ldots, T_k$ and then combines these solutions into an approximate solution $F \subseteq L$ for the original tree G.

The rest of this chapter is structured as follows. We provide some basic definitions, preliminaries and an overview of the algorithm in Section 2.1. In Section 2.2, we show how to decompose the input instance into several well-structured instances. In Section 2.3, we show how to solve these well-structured instances approximately, obtaining an approximation factor below two. In Section 2.4, we show how to use solutions of the well-structured instances in the decomposition to give an approximate solution for the original instance that attains an approximation factor of $1.964 + \epsilon$. This completes the exposition of the result by Adjiashvili [1]. In Section 2.5, we present a follow-up result by Fiorini, Groß, Könemann, Sanita [16] that improves the approximation factor to $\frac{3}{2} + \epsilon$ using Chvátal-Gomory cuts and a similar decomposition technique. Finally, in Section 2.6, we briefly go over several other follow-up results in the literature.

2.1 Preliminaries and Overview

Before proceeding, we mention certain assumptions that we make on the structure of the instance. As alluded to earlier, we consider the case of bounded costs. More formally, the cost c_ℓ of each link $\ell \in L$ lies in the range $[1, M]$ for some constant $M \geq 1$. Let $c_{max} = \max_{\ell \in L} c_\ell$ and let $c_{min} = \min_{\ell \in L} c_\ell$. The bounded cost assumption is equivalent to stating that the ratio c_{max}/c_{min} is at most M, as we can always multiply each cost c_ℓ by $1/c_{min}$ to make it so that $c_\ell \geq 1$ and $c_\ell \leq c_{max}/c_{min} \leq M$ for each $\ell \in L$.

There is also another assumption we make on the structure of the instance. For a tree $G = (V, E)$ and a link $\ell = uv \in V \times V$, we denote by P_ℓ^G the unique $u - v$ path in G. When G is clear from context, we drop the superscript and write P_ℓ. Given two links $\ell', \ell \in V \times V$, the link ℓ' is said to be a *shadow* of the link ℓ if $P_{\ell'} \subseteq P_\ell$. An instance (G, L) is said to be *shadow complete* if, for every $\ell \in L$, all shadows of ℓ are also in L. We assume that our instance is shadow complete. This is a safe assumption to make, since if the instance were not shadow complete, then all shadows of all links in L can be added to L. The cost of an added link ℓ is the minimum cost of any orignal link of which ℓ is a shadow. This operation does not change the cost of optimum solution of the instance. Indeed, any shadow ℓ that is added in the shadow completion operation can be replaced by one of the original links with minimum cost of which ℓ is a shadow, without increasing the cost of the solution, or affecting feasibility. We will henceforth assume that the instance is shadow complete.

We fix a vertex $r \in V$ to be the root of G. A link $\ell \in L$ is said to be a *cross-link* if r lies on the path P_ℓ, otherwise ℓ is said to be an *in-link*. Given two vertices $u, v \in V$, by $lca(u, v)$ we denote the least common ancestor of u, v. That is, $lca(u, v)$ is the unique vertex z on the $u-v$ path in G that minimizes the distance between r and z. A link $\ell \in L$ is said to be an *up-link* if $lca(u,v) \in \{u,v\}$. We denote the set of cross-links by $L^{cr} \subseteq L$, the set of in-links by $L^{in} \subseteq L$, and the set of up-links by $L^{up} \subseteq L$. Moreover, for a vector $x \in \mathbb{R}^L$, we let $x^{cr} \in \mathbb{R}^{L^{cr}}$ denote the vector x restricted to L^{cr} and let $x^{in} \in \mathbb{R}^{L^{in}}$ denote x restricted to L^{in}.

Consider the natural LP relaxation for Weighted Tree Augmentation that is obtained from the LP for Connectivity Augmentation (1.1) by setting $k = 1$.

$$\begin{array}{rrcll}
\text{minimize} & \displaystyle\sum_{\ell \in L} c_\ell x_\ell & & & \\
\text{subject to} & \displaystyle\sum_{\ell \in \delta_L(S)} x_\ell & \geq & 2 - |\delta_E(S)| & \forall S \subset V, S \neq \emptyset \\
& x_\ell & \leq & 1 & \forall \ell \in L \\
& x_\ell & \geq & 0 & \forall \ell \in L
\end{array}$$

We can considerably simplify this LP relaxation. Note that $|\delta_E(S)| \geq 1$ for any cut $\delta(S)$ since E forms the edges of a spanning tree. Moreover, if $|\delta_E(S)| \geq 2$, then the constraint corresponding to the set S is trivially satisfied. Hence we only need to consider constraints corresponding to cuts S such that $|\delta_E(S)| = 1$. Such cuts are precisely the cuts of connected components obtained by removing some edge $e \in E$ from G. For an edge $e \in E$, by $cov(e)$ we denote the set of links ℓ such that e lies in P_ℓ. We can now rewrite the above LP as follows.

$$\begin{array}{rrcll}
\text{minimize} & \displaystyle\sum_{\ell \in L} c_\ell x_\ell & & & \\
\text{subject to} & \displaystyle\sum_{\ell \in cov(e)} x_\ell & \geq & 1 & \forall e \in E \\
& x_\ell & \geq & 0 & \forall \ell \in L
\end{array}$$

We refer to this LP as the cut-LP (for Tree Augmentation). Here the constraint $\sum_{\ell \in cov(e)} x_\ell \geq 1$ is equivalent to the constraint $\sum_{\ell \in \delta_L(S)} x_\ell \geq 2 - |\delta_E(S)|$ for some set S such that $|\delta_E(S)| = 1$. Moreover, we have removed the $x_\ell \leq 1$ constraints as they are redundant since the right-hand side of each of the other constraints is 1.

We now present a simple 2-factor approximation algorithm.

Lemma 2.1. *[1] Suppose x is feasible for the cut-LP and suppose $supp(x) \subseteq L^{up}$. Then there is a feasible integral solution $F \subseteq L^{up}$ such that $c(F) \leq c^\intercal x$, that can be computed in polynomial time.*

Proof. We will prove that if $supp(x) \subseteq L^{up}$, then the constraint matrix of the LP (when restricted to columns in $supp(x)$) becomes totally unimodular. This will imply that the polyhedron defined by the LP has integral extreme points, and hence we can construct a feasible integral solution $F \subseteq L^{up}$ such that $c(F) \leq c^\intercal x$ by simply solving the LP.

To prove the total unimodularity of the constraint matrix, we use the Ghouila-Houri condition. Consider any subset of constraints $F \subseteq E$. We show that there exists a subset $F^+ \subseteq F$ such that the constraint obtained by adding all constraint corresponding to edges in F^+ and subtracting all constraints corresponding to edges in $F^- = F \setminus F^+$ is a constraint with coefficients in $\{-1, 0, 1\}$. Then by Theorem 1.3, the constraint matrix will be totally unimodular.

First, we contract all edges in $E \setminus F$ to obtain the graph $G' = G/(E \setminus F)$. Let F^+ be the set of edges in G' that are at an odd distance from r. That is, F^+ is the set of edges $e \in G'$ such that number of edges on the path connecting r to the closest endpoint of e is odd. Now consider any link $\ell \in L$, since ℓ is an up-link, the path P_ℓ alternates between edges in F^+ and edges in F^-. So we have,

$$|\{e \in F^+ : \ell \in cov(e)\}| - |\{e \in F^- : \ell \in cov(e)\}| \in \{-1, 0, 1\}$$

which completes the proof of the lemma. $\square$

From the above lemma we can design a simple 2-approximation algorithm.

Theorem 2.1. *[1] There exists a polynomial time algorithm that, when given a* Weighted Tree Augmentation *instance $(G = (V, E), L)$, returns a feasible solution F of cost $c(F) \leq 2c^\intercal x$.*

Proof. Consider a link $\ell = uv \in L \setminus L^{up}$ such that $\ell \in supp(x)$ and let $w = lca(u, v)$. We convert x to a solution x', where $x'_\ell = 0$ and $x'_{uw} = x_{uw} + x_\ell$ and $x'_{vw} = x_{vw} + x_\ell$, and $x'_{\ell'} = x_{\ell'}$ for all other links ℓ'. That is, x' is obtained from x by transferring the x-value from $\ell = uv$ to its two shadows uw and vw, and incurring an extra cost of $c_\ell x_\ell$ in the process. Note that x' is still feasible. Doing this operation for every link $\ell \in L \setminus L^{up}$ gives us a feasible solution x^* such that $supp(x^*) \subseteq L^{up}$ and $c^\intercal x^* \leq 2c^\intercal x$. Applying Lemma 2.1, we can compute an integral solution $F \subseteq L^{up}$ such that $c(F) \leq c^\intercal x^* \leq 2c^\intercal x$, proving the theorem. $\square$

Let $n = |V|$. The *detachment* of a graph $G = (V, E)$ at a vertex $v \in V$ is formed by removing v from G and adding $|deg(v)|$ vertices of degree 1, each of which is adjacent to a unique neighbour of v (see [17]). Recall that we want to proceed by decomposing the instance into subinstances that are easy to solve. What notion of 'easiness' should we use? The following lemma suggests that the number of leaves of the subtree could be a good measure of its easiness. It says that if the number of leaves in the instance is bounded by a constant, then we can solve the instance optimally in polynomial time.

Lemma 2.2. *[1] Given a* Weighted Tree Augmentation *instance* $(G = (V, E), L)$, *an optimal solution can be computed in time* $n^{p^{O(1)}}$, *where* p *is the number of leaves of* G.

Proof. Let W be the set of vertices in G that have degree at least three. Let $Q_1, \ldots, Q_k$ be the connected components of the graph obtained after detachment at every vertex in W. Note that each of $Q_1, \ldots, Q_k$ is a path. Since there are p leaves, the number of nodes in W is at most p, and hence $k = O(p)$.

Let $S^* \subseteq L$ be an optimal integral solution. For each pair of paths Q_i, Q_j, we claim that S^* contains at most two links that have one endpoint in Q_i and the other in Q_j. Indeed, if this were not the case, and there were at least three links connecting nodes in Q_i to nodes in Q_j, then the path of one link must be contained in the union of the paths of the other two. This contradicts the optimality of S^*, since we may remove one of the links and obtain a solution of smaller cost (recall that all links have cost at least one). Hence we conclude that S^* has at most $O(p^2)$ connecting nodes of different paths. All other links in S^* have endpoints within the same path.

The algorithm starts by guessing the $O(p^2)$ links in S^* that have endpoints in different paths, and including them in the solution. For each guess $F \subseteq L$, it remains to select the subset of links whose endpoints are contained in the same path. This reduces to solving k path covering problems, one for each subpath of Q_i consisting of the edges of Q_i that are not covered by F. For each path covering problem, the goal is to cover edges in the path using the links which have both endpoints in that path. We can solve these path covering problems in polynomial time using dynamic programming.

Thus the total runtime of the algorithm is $n^{O(p^2)}$, where p is the number of leaves of G. □

The above lemma suggests the following idea. We decompose the instance $(G = (V, E), L)$ into subinstances $T_1, \ldots, T_k$ each of which have a constant number of leaves. We then use Lemma 2.2 to solve each of these instances optimally and then combine these solutions to get a solution of the original instance. How do we decompose the instance? A natural way to do so is to simply pick an edge $e = uv \in E$ and break our instance down to the subtrees G^u and G^v, where G^u is the connected component of $G \setminus e$ containing the vertex u and G^v is the connected component of $G \setminus e$ containing the vertex v. Both G^u

and G^v induce their own subinstance where we wish to cover only the edges within one subtree using links with both endpoints in that subtree. Note that since the instance is shadow complete and since the instance has a feasible solution, it follows that both G^u and G^v have feasible solutions. Thus G^u and G^v make for good candidate subinstances. So we can proceed by decomposing G into G^u and G^v (plus the edge $e = uv$), and then recusively find a decomposition $T_1, ..., T_{k'}$ of G^u and a decomposition $T_{k'+1}, \dots, T_k$ of G^v and take the union to get the final decomposition $T_1, \dots, T_k$ of G.

To bound the cost of the final solution computed for G, we need to bound the costs of the solutions computed for G^u and G^v. This might not be so easy, as there are examples where the cost of optimum solution for G^u and G^v can be as expensive as the cost of optimal solution for G. See Fig. 2.1 for one such example. It turns out that we can use the LP solution to guide us in our decomposition step and help us bound the cost of the final solution. Let x be a feasible LP solution for the instance $(G = (V, E), L)$. The vector x then naturally splits into feasible solutions x^u, x^v of G^u, G^v as follows. For each link $\ell = pq$ in the support of x such that $p \in V[G^u]$ and $q \in V[G^v]$, we transfer the x-value of ℓ to its two shadows $pu \in G^u$ and $qv \in G^v$ incurring an additional cost of $c_\ell x_\ell$ in the process. Doing this for every such link ℓ with one endpoint in G^u and the other in G^v gives a vector that naturally splits into two solutions x^u, x^v for G^u and G^v respectively. More formally, we define the vector x^u as follows. For each link $\ell = pq \in L$ we have,

$$x^u_\ell = \begin{cases} x_\ell & \text{if } p, q \in V[G^u] \setminus \{u\} \\ 0 & \text{if } p \notin V[G^u] \text{ or } q \notin V[G^u] \\ x_\ell + \sum_{\ell' \in cov(e), q \in \ell'} x_{\ell'} & \text{if } p = u \text{ and } q \in V[G^u] \end{cases}$$

The vector x^v is defined symmetrically. Note that we have $c^\intercal x^u + c^\intercal x^v = c^\intercal x + \sum_{\ell \in cov(e)} c_\ell x_\ell$. Thus the increase in the cost of the LP solutions of G^u and G^v versus the cost of the LP solution of G is bounded by the term $\sum_{\ell \in cov(e)} c_\ell x_\ell$. So, to make sure that we incur only a small loss when splitting, we need a way to control the term $\sum_{\ell \in cov(e)} c_\ell x_\ell$. Consider the set of edges e for which $\sum_{\ell \in cov(e)} x_\ell$ is large (i.e. more than some fixed constant B), and hence $\sum_{\ell \in cov(e)} c_\ell x_\ell$ is also large. We call these edges the *heavy edges* with respect to the solution x. We will show that we can find a set of links $L_0 \subseteq L$ of small cost that covers all the heavy edges. Once we find such a set L_0, we can add L_0 to our solution and contract any cycles that are formed. This will result in a graph G' that has no heavy edge, since every heavy edge must have been present in one of these cycles. That is, for any edge e in the resulting graph G', we have $\sum_{\ell \in cov(e)} x_\ell \leq B$, and thus $\sum_{\ell \in cov(e)} c_\ell x_\ell \leq MB$. This means that we can apply the splitting operation on any edge in G' and incur a loss of at most MB, which we choose to be small enough to be

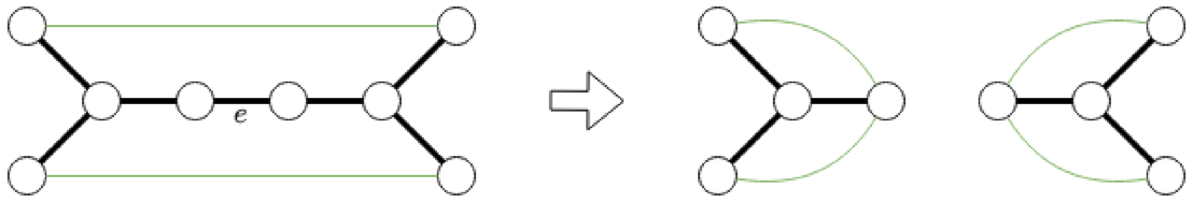


Figure 2.1: An example of an instance (G, L) and an edge e such that the cost of the optimal solution for both the subinstances obtained after removing the edge e is equal to the cost of the optimal solution of (G, L). The tree edges are given by thick lines and the links are given by thin lines. All links have cost 1 in this example. Shadows of links are not shown for the sake of clarity.

neglected in the final analysis. Note that the original LP solution x remains feasible for the new graph G' since contractions preserve the feasibility of the constraint $\sum_{\ell \in cov(e)} x_\ell \geq 1$ for any non-contracted edge e.

We can now refine our decomposition idea by incorporating the previous discussion. Given an instance and an accompanying LP solution $(G = (V, E), L, x)$, we decompose it into subinstances $(T_1, L_1, x^1), \ldots, (T_k, L_k, x^k)$ each with an accompanying feasible LP solution, and incur only a small increase in the cost of the LP in the process. We then compute an optimal solution F_i for each subinstance (T_i, L_i) using Lemma 2.2, and combine these solutions to produce a final solution for the original instance. One hassle is that we don't know how to bound the cost of the solution F_i by the cost of the corresponding cut-LP solution x^i. This is crucial since we need to use the cost of the LP solution as a lower bound, as the cost incurred during splitting is charged to the cost of the LP. To alleviate this issue, we can add constraints to ensure that the integrality gap of each subinstance (T_i, L_i) is 1, that is, the cost of an optimal integral solution is equal to the cost of an optimal LP solution for each subinstance. For any subset of edges $F \subseteq E$, let $cov(F)$ denote the set of links that cover at least one edge in F and let $\texttt{OPT}(F)$ denote the cost of an optimal set of links that cover every edge in F. Since the number of leaves in T_i is at most p, a constant, and since the number of subtrees of G with at most p leaves is at most n^p, a polynomial in n, we can simply add the constraint

$$\sum_{\ell \in cov(E[T])} c_\ell x_\ell \geq \texttt{OPT}(E[T])$$

for each subtree T of G that has at most p leaves. Note that these constraints can be added in polynomial time since there are at most n^p subtrees of T having at most p leaves, and so the number of constraints is polynomially bounded and furthermore, we can compute $\texttt{OPT}(E[T])$ in polynomial time using Lemma 2.2.

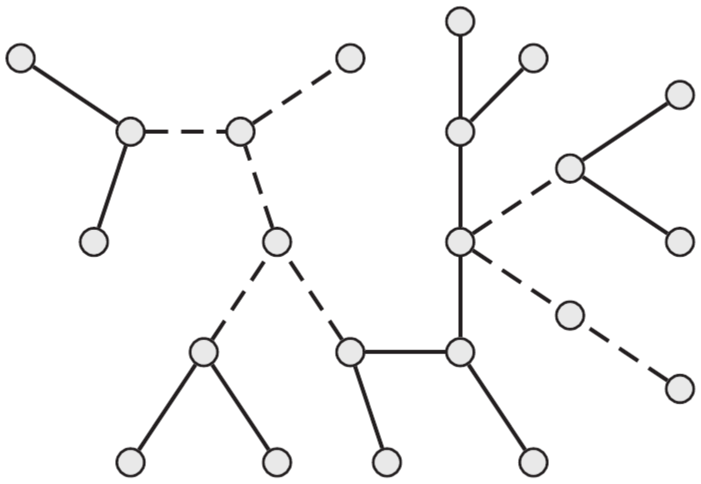


Figure 2.2: [1] The dashed edges can be obtained as a union of three paths in the tree. Hence they represent a 3-bundle.

However one problem with these constraints is that they are not preserved under contractions. Indeed, if G' is obtained from G by contracting edges, then a subtree with at most p leaves in G' might not correspond to a subtree with at most p leaves in G since this subtree might contain compound nodes that are created during the contraction. We crucially need the feasibility of our LP to be preserved under contractions since we contract edges in the graph after adding the set L_0 of links that cover the heavy edges, and we want the LP to be feasible for the resulting graph. To work around this issue, Adjiashvili uses a stronger set of constraints that imply the subtree constraints. For any $\gamma \in \mathbb{Z}_+$, we define a γ-bundle to be the union of a collection of γ paths in G. See Fig. 2.2 for an example of a 3-bundle. Let the set of all γ-bundles in G be denoted by $\mathcal{B}_\gamma$. Note that the paths of a γ-bundle need not be disjoint. In particular, they need not be distinct either, so we have $\mathcal{B}_{\gamma-1} \subseteq \mathcal{B}_\gamma$. We add the bundle-constraint

$$\sum_{\ell \in cov(B)} c_\ell x_\ell \geq \texttt{OPT}(B)$$

for each γ-bundle B of G, for some constant γ to be decided later. That is, we use the following LP, which we call the γ-bundle LP.

$$\begin{array}{rrcll} \text{minimize} & \sum_{\ell \in L} c_\ell x_\ell & & & \\ \text{subject to} & \sum_{\ell \in cov(e)} x_\ell & \geq & 1 & \forall e \in E \\ & \sum_{\ell \in cov(B)} c_\ell x_\ell & \geq & \mathtt{OPT}(B) & \forall B \in \mathcal{B}_\gamma \\ & x_\ell & \geq & 0 & \forall \ell \in L \end{array} \tag{2.1}$$

Note that, any subtree T of G with at most p leaves is a p-bundle, as it is the union of the p leaf to r paths, for any vertex $r \in V[T]$. Thus the γ-bundle constraints imply the subtree constraints. The γ-bundle constraints have the same problem as the subtree constraints; that they are not preserved under contractions. However, we will be able to show that any γ-bundle in any contracted graph is a $O(\gamma)$-bundle in the original graph G. This will allow us to use γ-bundle constraints for any contracted graph, as long as we have enforced $O(\gamma)$-bundle constraints in the original graph.

Before we proceed into providing details, we need to address one final important issue with our decomposition strategy. Since each subtree in our decomposition has at most p leaves, the total number of subtrees in the decomposition can be as bad as $O(\frac{n}{p})$. Since p is a constant, this is essentially $O(n)$. This means that the small cost incurred in each splitting operation is incurred for a total of $O(n)$ times, and thus the total cost incurred throughout the decomposition is significant. Ideally, we would like to split at most k times for some small enough k, so that the total loss incurred in the splitting is small enough to be negligible compared to the cost of the solution. To achieve this, we will change the type of subinstances we are decomposing into. Enforcing each subinstance to have at most p leaves is a bit too ambitious. Instead, we will decompose into subinstances that are in some sense just one level more complex than having at most p leaves. We call such instances *β-simple.*

Definition 2.1 (β-simple instances)**.** *Let $\beta \in \mathbb{Z}_+$ and let x be a fractional solution to the cut-LP for the instance $(G = (V, E), L)$. We call the pair (G, x) β-simple if there exists a root $r \in V$ the removal of which results in a forest with trees $K_1, \ldots, K_t$ such that for each $j \in [t]$,*

- *K_j has at most β leaves.*
- *$\sum_{\ell \in L, \ell \in V[K_j] \times V[K_j]} c_\ell x_\ell \leq \beta$.*

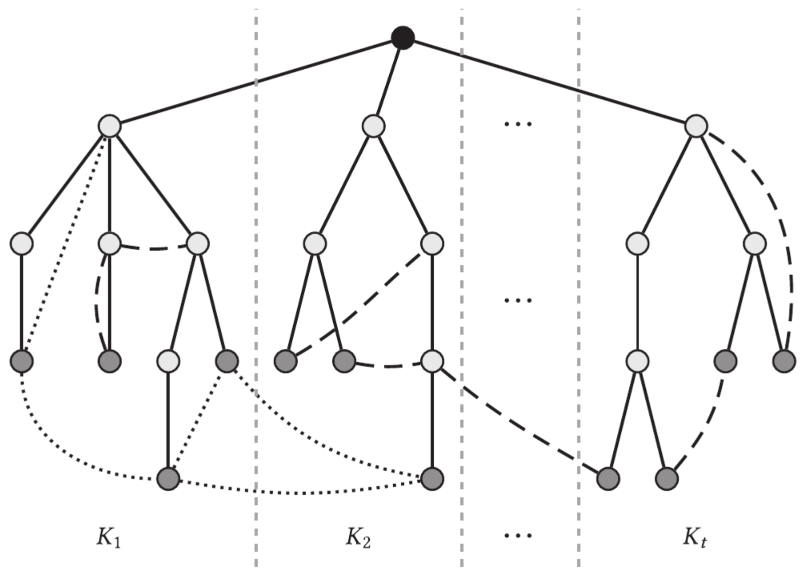


Figure 2.3: [1] An example of a 4-simple instance. The dark node is the 4-center. The shown links represent the support of x, with integral (dashed) and half-integral (dotted) links. The number of leaves and the total fractional value in each subtree is at most 4.

Furthermore, the vertex r is called the β-center of (G, x).

So β-simple instances are one level more complex than bounded leaf instances in the sense that they are instances composed of multiple bounded leaf instances joined by a root. See Fig. 2.3 for an example of a 4-simple instance. There is an additional second condition that says that the cost of the LP induced on each component of $G - r$ is small (at most β). This is a technical condition that will help us in the analysis of the decomposition step. Note that, while each subtree hanging from the root r of a β-simple instance has at most β leaves, the instance itself could have $O(n)$ leaves since there is no bound on the degree of the root r. If we were to do the splitting operation for one more level, that is, if we were to apply the splitting operation on each edge incident to the β-center r, we would get subtrees in our decomposition that all have at most β leaves. However, this final level of splitting might be expensive, since the degree of the vertex r might be as bad as $O(n)$.

We won't be able to solve β-simple instances optimally, but we will be able to do the next best thing: We will solve these instances *approximately* attaining an approximation factor better than 2. We will then be able to combine these approximate solutions of the β-simple instances in the decomposition to provide an approximate solution for the original instance.

This concludes the high-level overview of the algorithm. To summarize, the algorithm first computes a solution x to the γ-bundle LP (2.1) for the instance. The algorithm then proceeds by finding a set of links $L_0 \subseteq L$ of small cost that cover every heavy edge of the graph, and includes L_0 into the solution, contracting away all the heavy edges in the process. The resulting graph has only light edges, which means that we can start applying the splitting operation without incurring too much cost. The algorithm then splits the instance into a collection of β-simple subinstances $(T_1, L_1, x^1), \dots, (T_k, L_k, x^k)$, after which it finds approximate solutions $F_1, \dots, F_k$ for each of these β-simple instances and combines them to obtain a solution for the original instance.

2.2 Decomposition into β-simple instances

2.2.1 Covering heavy edges

Now we are ready to give the formal details of the algorithm. We start by describing the decomposition step. We first solve the γ-bundle LP on the input instance $(G = (V, E), L, c)$.

$$\begin{array}{rrcll} \text{minimize} & \displaystyle\sum_{\ell \in L} c_\ell x_\ell & & & \\ \text{subject to} & \displaystyle\sum_{\ell \in cov(e)} x_\ell & \geq & 1 & \forall e \in E \\ & \displaystyle\sum_{\ell \in cov(B)} c_\ell x_\ell & \geq & \texttt{OPT}(B) & \forall B \in \mathcal{B}_\gamma \\ & x_\ell & \geq & 0 & \forall \ell \in L \end{array}$$

Recall that we want to decompose the instance into β-simple instances using the splitting operation, which for the sake of convenience, we describe again.

Definition 2.2 (Splitting)**.** *Let $(G = (V, E), L, c)$ be a* Weighted Tree Augmentation *instance and let $x \in \mathbb{R}^L$. Let $e = uv \in E$ be an edge of G. Let G^u and G^v be the trees obtained by removing e from G, where G^u is the tree containing the vertex u and G^v is the tree containining the vertex v. The* splitting *of x at e produces two vectors $x^u, x^v \in \mathbb{R}^L$ defined as follows. For each $\ell = pq \in L$, we have*

$$x^u_\ell = \begin{cases} x_\ell & \text{if } p, q \in V[G^u] \setminus \{u\} \\ 0 & \text{if } p \notin V[G^u] \text{ or } q \notin V[G^u] \\ x_\ell + \sum_{\ell' \in cov(e), q \in \ell'} x_{\ell'} & \text{if } p = u \text{ and } q \in V[G^u] \end{cases}$$

The vector x^v is defined symmetrically.

That is, for every link $\ell = pq$ with endpoints $p \in G^u$ and $q \in G^v$, we move the x-value of ℓ to its two shadows pu and qv, incurring a cost of $c_\ell x_\ell$ in the process. Note that $supp(x^u) \subseteq V[G^u] \times V[G^u]$ and $supp(x^v) \subseteq V[G^v] \times V[G^v]$. Moreover, note that if x is a feasible cut-LP solution for the instance over G, then both x^u and x^v are feasible cut-LP solutions for the instances over G^u and G^v respectively. Whenever we apply the splitting operation, we also root the subinstances G^u and G^v at arbitrary vertices, so that in-links, cross-links and up-links are well-defined for the subinstances.

Splitting incurs a cost of $\sum_{\ell \in cov(e)} c_\ell x_\ell$. We want this cost incurred to be small. Since the costs c_ℓ for each link $\ell \in L$ is bounded by a constant M, we have $\sum_{\ell \in cov(e)} c_\ell x_\ell \leq M \sum_{\ell \in cov(e)} x_\ell$ for every edge $e \in E$. Consider an edge $e \in E$ such that $\sum_{\ell \in cov(e)} x_\ell \leq B$, then we have $\sum_{\ell \in cov(e)} c_\ell x_\ell \leq MB$. We call such edges *light* edges.

Definition 2.3 (Heavy and light edges)**.** *An edge $e \in E$ is said to be* light *with respect to an LP solution x if $\sum_{\ell \in cov(e)} x_\ell \leq B$, otherwise e is said to be* heavy*.*

So splitting on light edges incurs a cost of at most MB. Ideally, we would like to split only on light edges. To ensure this, we will first show how to deal with heavy edges. That is, we will show how to compute a set of links $L_0 \subseteq L$ with small cost such that L_0 covers every heavy edge in G. This is achieved by the following lemma.

Lemma 2.3. *[1] Given a* Weighted Tree Augmentation *instance $(G = (V, E), L, c)$ and a feasible solution x to the bundle-LP, there exists a polynomial time algorithm that outputs a set of links $L_0 \subseteq L$ with cost $c(L_0) \leq \frac{2}{B} c^\intercal x$, such that for any edge e not covered by L_0, we have $x(cov(e)) \leq B$, that is, e is light with respect to x.*

Proof. Let $E^h = \{e \in E : x(cov(e)) > B\}$ be the set of heavy edges. First, we contract the edges $E \setminus E^h$ to obtain a subinstance G^h of G whose edge set is E^h. Thus the solution x covers every edge in G^h by a factor of at least B. Hence the vector $y = \frac{1}{B} x$ is a feasible solution to the cut-LP for the instance over G^h. Now using Theorem 2.1, we can compute a set of links $L_0 \subseteq L$ that cover all edges of G^h and has cost at most $c(L_0) \leq 2c^\intercal y = \frac{2}{B} c^\intercal x$. The set of edges not covered by L_0 are edges that do not exist in G^h and hence lie in $E \setminus E^h$, that is, they are light with respect to x. This finishes the proof. □

2.2.2 Decomposing the instance

Once we cover all heavy edges, and contract any cycles formed, all the remaining edges in our graph are light. So we can split on any edge and incur only a small additional cost during the splitting. To ensure that the decomposition has all the properties that we want, we will split on special edges that we call *thin* edges.

Definition 2.4 (Thin edge)**.** *An edge $uv \in E[T]$ is called α-thin with respect to x if $\sum_{\ell \in L, \ell \in V[G^q] \times V[G^q]} c_\ell x_\ell \geq \alpha$ for each $q \in \{u, v\}$.*

In other words, an edge e is α-thin if the cost of the LP solution is large when induced on the connected components of $G - e$. The algorithm recursively finds an α-thin edge for $\alpha = \frac{2BM}{\epsilon}$ and splits on that edge. If there is no such thin edge the algorithm will output the subtree as part of the decomposition.

The following theorem tells us that the resulting decomposition $(T^1, z^1), \ldots, (T^k, z^k)$ has the properties we want. The first, second and third properties in the theorem relate to $(T^1, z^1), \ldots, (T^k, z^k)$ being the desired decomposition of G into β-simple subinstances. The fourth property is a consequence of the working of the algorithm. The fifth property says that the total fractional cost of the subinstances in the decomposition amount to at most an ϵ factor more than the fractional cost of the original instance, that is, the decomposition step incurs only a negligible loss in approximation. Finally, we also need to be able to translate the bundle constraints on G to bundle constraints for each of the subinstances. This is achieved by the final property.

Theorem 2.2. *[1] Let $\epsilon \in (0, \frac{1}{4})$ and $B > 1$. Let $(G = (V, E), L)$ be a* Weighted Tree Augmentation *instance and let x be a solution to the bundle-LP of this instance such that every edge in E is light with respect to x. Then there is a polynomial time algorithm that computes a decomposition $(T^1, z^1), \ldots, (T^k, z^k)$ such that*

1. *$T^1, \ldots, T^k$ are vertex-disjoint subtrees of G, and z^j is a cut-LP solution for T^j, for every $j \in [k]$.*
2. *The subtrees $T^1, \ldots, T^k$ are obtained by removing exactly $k - 1$ edges from G*
3. *(T^j, z^j) is β-simple for every $j \in [k]$, for $\beta = \frac{6BM}{\epsilon}$*
4. *$c^\intercal z^j \geq \alpha = \frac{2BM}{\epsilon}$ for every $j \in [k]$*
5. *$\sum_{j=1}^{k} c^\intercal z^j \leq (1 + \epsilon) c^\intercal x$*
6. *For any $j \in [k]$ and $F \subseteq E[T^j]$, we have $\sum_{\ell \in cov(F)} c_\ell z^j_\ell \geq \sum_{\ell \in cov(F)} c_\ell x_\ell$.*

Proof. As mentioned, the algorithm recursively tries to find a α-thin edge uv, for $\alpha = \frac{2BM}{\epsilon}$. If such an edge exists, the algorithm applies the splitting operation on the edge uv and recurses on the subinstances (G^u, x^u) and (G^v, x^v). Otherwise, if no α-thin edge exists, the algorithm will output (G, x) to be part of the decomposition.

Clearly this algorithm runs in polynomial time. Let $(T^1, z^1), \ldots, (T^k, z^k)$ be the decomposition that is returned by the algorithm. We need to prove that this decomposition satisfies properties (1)-(6) in the statement of the theorem. Properties (1), (2) and (4) hold by the definition of the decomposition.

Now we prove property (5). Observe that the total number of splittings is exactly $k - 1$, the number of subinstances output by the algorithm minus one. Since we split only on edges e that are light, every splitting incurs a loss of at most $\sum_{\ell \in cov(e)} c_\ell x_\ell \leq M \sum_{\ell \in cov(e)} x_\ell \leq MB$. Thus we have $\sum_{j=1}^{k} c^\intercal z^j \leq c^\intercal x + kMB$. To show property (5), it suffices to prove that $kMB \leq \epsilon c^\intercal x$. By property (4), each subinstance (T^j, z^j) of the decomposition satisfies $c^\intercal z^j \geq \alpha = \frac{2MB}{\epsilon}$. Thus, $\sum_{j=1}^{k} c^\intercal z^j \geq \frac{2kMB}{\epsilon}$. Now we have,

$$c^\intercal x \geq \sum_{j \in [k]} c^\intercal z^j - kMB \geq \frac{2kMB}{\epsilon} - kMB = kMB \left(\frac{2}{\epsilon} - 1 \right)$$

Thus we get,

$$kMB \leq c^\intercal x \cdot \frac{1}{\left(\frac{2}{\epsilon} - 1\right)} = c^\intercal x \cdot \left(\frac{\epsilon}{2 - \epsilon} \right) \leq \epsilon \cdot c^\intercal x$$

where the last inequality uses the fact that $\epsilon \leq \frac{1}{4}$. This proves property (4).

Now we prove property (6). Fix a $j \in [k]$ and $F \subseteq E[T^j]$. Observe that, in any splitting operation performed by the algorithm, whenever the fractional value of some link ℓ in $cov(F)$ is decreased to 0, the fractional value of some shadow ℓ' of ℓ is increased by the same amount. Moreover, this shadow also lies in $cov(F)$, since $F \subseteq E[T^j]$ and T^j was returned by the algorithm. Thus the value of $cov(F)$ does not decrease during the splitting operations, for any $j \in [k]$ and any $F \subseteq E[T^j]$.

Lastly, we need to prove property (3). We need to prove that every instance (T^j, z^j) is β-simple for $\beta = \frac{6BM}{\epsilon}$. Since every edge is light with respect to x, we have $x(cov(e)) \leq B$ for all $e \in E$. Note that the spliting operation does not increase the coverage of any edge with respect to x. More specifically, if (G^u, x^u) and (G^v, x^v) are obtained from splitting (G, x) at edge e, we have $x^u(cov(e)) = x(cov(e))$ for all $e \in E(G^u)$ and $x^v(cov(e)) = x(cov(e))$ for all $e \in E(G^v)$.

Fix a pair (T, z) obtained in the decomposition. By property of the decomposition, the instance (T, z) has no α-thin edges for $\alpha = \frac{2BM}{\epsilon}$. This implies, that for every $uv \in E(T)$, there exists an endpoint $q \in \{u, v\}$ such that

$$\sum_{\ell \in L, \ell \in V[T^q] \times V[T^q]} c_\ell z_\ell < \frac{2BM}{\epsilon} \tag{2.2}$$

Orient each edge uv from $p \in \{u, v\}$ to $q \in \{u, v\}$ so that the endpoint q satisfies (2.2). If both endpoints u, v satisfy (2.2), then we may orient uv in an arbitrary manner.

After orienting T in such a way, we obtain a directed tree $\overrightarrow{T}$, which must contain a node $r \in V[T]$ with in-degree 0. We claim that r is a β-center of T and hence (T, z) is β-simple. By the construction of $\overrightarrow{T}$, and since $\frac{6BM}{\epsilon} > \frac{2BM}{\epsilon}$, the second property of Definition 2.1 holds. It remains to prove the first property. Assume for contradiction that the first property does not hold, that is, some component K of $T \setminus \{r\}$ has more than $\frac{6BM}{\epsilon}$ leaves. Since each link has cost at least 1, and since each link can cover at most two leaves, it follows that any fractional solution covering K must have cost more than $\frac{3BM}{\epsilon}$. Now let e' be the edge connecting r to K. Since e' is light with respect to z, and since the cost of every link is bounded by M, the total cost of all links covering e' is at most BM. Note that z fractionally covers K using only links in $L' = (V[K] \times V[K]) \cup cov(e')$. So we have,

$$\sum_{\ell \in L'} c_\ell z_\ell < \frac{2BM}{\epsilon} + BM \leq \frac{3BM}{\epsilon}.$$

This contradicts the fact that any fractional solution covering K must have cost more than $\frac{3BM}{\epsilon}$. Hence we conclude that every connected component of $T \setminus \{r\}$ must have at most $\frac{6BM}{\epsilon}$ leaves. This finishes the proof of property (3), and hence also the theorem. □

2.2.3 Combining the solutions

The set of links in L can be partitioned into the set of edges covered by L_0, the set of edges in some subtree T^j, and the set of $k-1$ edges that are removed during the splitting operations. In the following section, we will show how to find solutions $S_1, \ldots, S_k$ that cover the edges in the respective subtrees $T^1, \ldots, T^k$. As we have seen in Lemma 2.3, the cost of L_0 is at most $O(\epsilon)c^\intercal x$ (we will choose $B = O(\frac{1}{\epsilon})$). In this subsection, we will show that we can find a subset of links $L_1 \subseteq L$ that covers each of the $k-1$ edges removed during the splitting operations such that the cost of L_1 is at most $O(\epsilon)OPT$. Thus the total cost of L_0 and L_1 is at most $O(\epsilon)OPT$, and hence can be neglected as ϵ can be chosen to be as small as possible.

Lemma 2.4. *[1] Given a* Weighted Tree Augmentation *instance* $(G = (V, E), L)$ *and a decomposition* $(T^1, z^1), \ldots, (T^k, z^k)$ *computed by Theorem 2.2, there is a polynomial time algorithm that outputs a set of links* $L_1 \subseteq L$ *of cost at most* $c(L_1) \leq \frac{\epsilon}{6} c^\intercal x$*, that covers each of the* $k-1$ *links that are not present in the decomposition.*

Proof. To cover the $k-1$ cut edges, we simply pick one link in L_1 covering e for every edge $e \in E$ that was removed during the splitting operations. Thus we have $|L_1| \leq k-1$. Since $c_\ell \leq M$ for every $\ell \in L$, we have $c(L_1) \leq M(k-1)$. By property (4) in Theorem 2.2, each of the solutions $z^1, \ldots, z^k$ has cost at least $\alpha = \frac{2BM}{\epsilon}$. This implies that the total cost $c^\intercal z^1 + c^\intercal z^2 + \ldots + c^\intercal z^k$ is at least $\frac{2kBM}{\epsilon}$. Thus we have,

$$c(L_1) \leq M(k-1) < kM = \frac{\epsilon}{2B} \frac{2kBM}{\epsilon} \leq \frac{\epsilon}{2B} \sum_{i=1}^{k} c^\intercal z^i$$

By property (3) in Theorem 2.2, we have $\sum_{i=1}^{k} c^\intercal z^i \leq (1+\epsilon) c^\intercal x$, which implies,

$$c(L_1) \leq \frac{\epsilon}{2B} (1+\epsilon)\, c^\intercal x \leq \left(\frac{\epsilon + \epsilon^2}{2B} \right) c^\intercal x \leq \frac{\epsilon}{6} c^\intercal x$$

where the last inequality uses the fact that $B \geq 6$ and $\epsilon^2 \leq \epsilon$. This completes the proof of the lemma. $\square$

2.3 Rounding β-simple instances

In this section, we will show how to pick solutions $S_1, \ldots, S_k$ for each of the β-simple subtrees $T^1, \ldots, T^k$ in the decomposition. As it turns out, we won't have a single algorithm that rounds β-simple instances. Instead, we will have two algorithms, each of which only achieves an approximation factor of 2 in the worst case. But picking the better of the two solutions given by the two algorithms allows us to beat factor 2 for β-simple instances. Borrowing terminology from [6], we call these two algorithms the *backbone procedures.*

2.3.1 First backbone procedure

The first backbone procedure is a natural one. Recall that the algorithm given by Lemma 2.2 solves any instance optimally provided that it has a bounded number of leaves. By the definition of β-simple instances, there exists a vertex $r \in V$ such that each connected component of $G - r$ has at most β leaves. So we can use Lemma 2.2 to solve each connected component optimally, and then take the union of the solutions to produce a solution for the entire tree. The goal is to use bundle constraints to argue that the cost of the solution returned by Lemma 2.2 for any subtree is at most the cost of the fractional LP solution induced on that subtree. However, technicalities arise due to the heavy edge covering step in which we have contracted edges in the graph, which may break the bundle constraints. In the following lemma, we show that we can deal with these technicalities and we can compute a solution S for a β-simple instance in the decomposition that has cost at most $c(S) \le c^\intercal x^{in} + 2c^\intercal x^{cr} + \delta$, where δ is an additional term whose sum over all instances in the decomposition is negligible.

Lemma 2.5. *[1] Let x be a solution to the γ-bundle LP for $\gamma \ge \frac{28MB}{\epsilon}$. There is an algorithm that, given a β-simple instance (T, z) from the decomposition of (G, x), runs in time $n^{(BM)^{O(1)}}$, and computes a solution $S \subseteq L$ of cost at most*

$$c(S) \le c^\intercal x^{in} + 2c^\intercal x^{cr} + |V^0 \cap V[T]|$$

where V^0 are the compound nodes in the tree that are formed due to the contractions of the links L_0 obtained in the heavy edge covering step of the decomposition algorithm.

Proof. Let r be the β-center of (T, z) and let $H^1, \dots, H^m$ be the subtrees obtained from T after detachment at r. By the β-simple property, each subtree H^j has at most $\beta + 1$ leaves (where the additional leaf corresponds to the root r). Now, using Lemma 2.2, we can compute a set of links S^j for the instance over H^j such that $c(S^j) = OPT(E[H^j])$, i.e., S^j is an optimal integral solution for H^j. Then we return the union $S^1 \cup \ldots \cup S^m$ as our solution. Clearly, the algorithm runs in polynomial time. It remains to prove that the returned solution has cost at most $c(S) \le c^\intercal x^{in} + 2c^\intercal x^{cr} + |V^0 \cap V[T]|$.

To do this, we will first decompose the solution z into solutions $y^1, \dots, y^m$ of the subtrees $H^1, \dots, H^m$. Each cross-link $\ell = uv \in L$ is replaced in the fractional solution z by its two shadows ur and vr, by adding z_ℓ to z_{ur} and z_{vr} and setting z_ℓ to 0. Doing this for each cross-link, we obtain a solution with no cross-link in its support. Formally, we create a new solution y from z, defined as follows. For every link $\ell = pq$, we have,

$$y_\ell = \begin{cases} z_\ell & \text{if } \ell \in supp(z) \cap L^{in}, r \notin \ell \\ z_\ell + \sum_{\ell' \in supp(z^{cr}), q \in \ell'} z_{\ell'} & \text{if } p = r, q \in V[T] \setminus \{r\} \\ 0 & \text{otherwise} \end{cases}$$

Note that $c^\intercal y = c^\intercal z^{in} + 2c^\intercal z^{cr}$. Since y has no cross-links in its support, we can treat y as a disjoint union of solutions $y^1, \ldots, y^m$ for each of the subtrees $H^1, \ldots, H^m$. More formally, for each $j \in [m]$, the vector y^j is defined as follows.

$$y^j_\ell = \begin{cases} y_\ell & \text{if } \ell \in V[H^j] \times V[H^j] \\ 0 & \text{otherwise} \end{cases}$$

Now, we will show how to exploit the bundle constraints to prove that each vector y^j can be rounded to an integer solution for H^j, at a negligible loss in cost.

Claim 2.1. *If x is an optimal solution to the γ-bundle LP for $\gamma \geq \frac{28BM}{\epsilon}$, then*

$$OPT(E[H^j]) \leq c^\intercal y^j + |V^0 \cap V[H^j]|.$$

Proof. Since (T, z) is β-simple, we have $\sum_{\ell \in (V[H^j] \setminus \{r\}) \times (V[H^j] \setminus \{r\})} c_\ell y_\ell \leq \frac{6BM}{\epsilon}$. Every link that is in $V[H^j] \times V[H^j]$ but not in $(V[H^j] \setminus \{r\}) \times (V[H^j] \setminus \{r\})$ must lie in $cov(e)$ where e is the unique edge in H^j incident to r. Since every edge is light, we have $\sum_{\ell \in cov(e)} c_\ell x_\ell \leq BM$. Thus the total cost of all links in H^j is $c^\intercal y^j \leq \frac{6BM}{\epsilon} + BM \leq \frac{7BM}{\epsilon}$. It follows from the feasibility of y^j and Theorem 2.1 that $OPT(E[H^j]) \leq \frac{14BM}{\epsilon}$. Now, if H^j contains at least $\frac{14BM}{\epsilon}$ compound nodes, then $|V^0 \cap V[H^j]| \geq \frac{14BM}{\epsilon} \geq OPT(E[H^j])$, and we are done.

Otherwise, the number of compound nodes in H^j is at most $\frac{14BM}{\epsilon}$. In this case, we will prove that $E[H^j]$ is a $\frac{28BM}{\epsilon}$-bundle in G, and since $\gamma \geq \frac{28BM}{\epsilon}$, the γ-bundle LP on G contains the constraint

$$\sum_{\ell \in cov(E[H^j])} c_\ell x_\ell \geq OPT(E[H^j])$$

From which it will follow that,

$$c^\intercal y^j = \sum_{\ell \in cov(E[H^j])} c_\ell y^j_\ell \geq \sum_{\ell \in cov(E[H^j])} c_\ell z_\ell \geq \sum_{\ell \in cov(E[H^j])} c_\ell x_\ell \geq OPT(E[H^j])$$

where the second last inequality follows from property (6) in Theorem 2.2

Since (T, z) is β-simple, the number of leaves in H^j is at most $\frac{6BM}{\epsilon} + 1 \leq \frac{7BM}{\epsilon}$. Let $W \subseteq V[H^j]$ be the set of nodes of degree at least 3 in H^j. Since the number of leaves is at most $\frac{7BM}{\epsilon}$, it follows that $|W| \leq \frac{7BM}{\epsilon}$. Let $Q_1, \ldots, Q_t$ be the paths obtained after detachment on H^j at each of the nodes in W. Observe that $t \leq 2 \cdot \frac{7BM}{\epsilon} = \frac{14BM}{\epsilon}$.

Now each Q_i itself is a union of paths in G, separated by components contracted in the heavy edge covering phase of the algorithm, where we contracted the edges in G that are covered by the set of links L_0 given by Lemma 2.3 (see Fig. 2.4). Since every path in

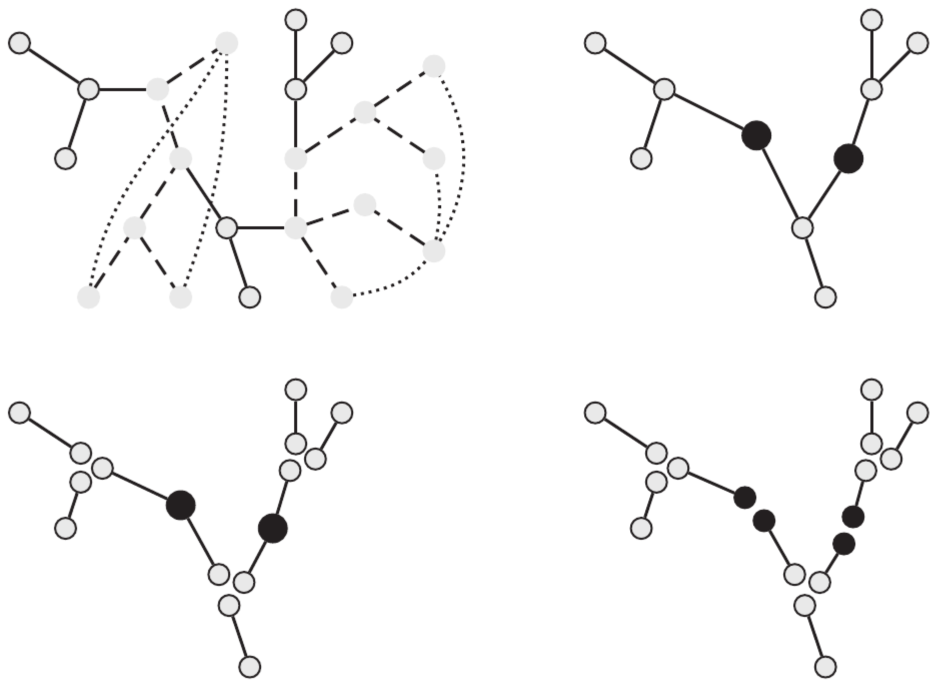


Figure 2.4: [1] Figure depicting the proof of Claim 2.1. Top Left: The part of the original tree corresponding to H^j. The dashed edges are contracted as they are heavy edges, resulting in the creation of two compound nodes. The dotted links are included in L_0. Top Right: The subtree H^j. The dark nodes are the compound nodes. Bottom Left: Decomposition H^j into 7 paths after detachment at W. Bottom Right: The resulting paths are further subdivided by detachment at compound nodes to obtain 9 paths, thus forming a 9-bundle in the original tree G.

$Q_1, \dots, Q_t$ does not contain nodes of degree larger than two in the interior, each compound node can lie in the interior of at most one path in the decomposition. Since the total number of compound nodes is at most $\frac{14BM}{\epsilon}$, we conclude that H^j is a union of at most $\frac{28BM}{\epsilon}$ paths in G. That is, $E[H^j]$ is a γ-bundle in G. This ends the proof of the claim. □

The proof of the lemma now follows from the above claim and the fact that $y = \sum_{j=1}^{m} y^j$ and $\sum_{j=1}^{m} |V^0 \cap V[H^j]| = |V^0 \cap V[T]|$. □

Note that, in the worse case, Lemma 2.5 does not beat approximation factor two for β-simple instances. Indeed, it could be the case that the support of x lies entirely within the cross-links L^{cr}, in which case we will have $x^{cr} = x$ and $x^{in} = 0$, and the guarantee on the cost of the returned solution is only $2c^{\intercal}x$.

2.3.2 Second backbone procedure

The first backbone procedure fails to give us a good approximation when $c^{\intercal}x^{cr}$ is close to $c^{\intercal}x$. That is, the total cost of the links in the support is dominated by the cross-links. Consider an extreme case: Suppose that *all* the links in the support are cross-links. As it turns out, this extreme case has a simple $\frac{4}{3}$-factor approximation algorithm. We call these types of instances *star shaped*.

Definition 2.5 (Star-shaped instances)**.** *An instance is said to be star-shaped if there exists a node $r \in V$ such that $r \in P_\ell$ for every $\ell \in L$.*

Hence if we root a star-shaped instance at the vertex r, every link becomes a cross-link with respect to r. First, we prove an important lemma characterizing the feasibility of solutions for star-shaped instances. An edge is called a *leaf edge* if it is incident to a leaf in the graph.

Lemma 2.6. *[1] A solution $S \subseteq L$ for a star-shaped instance is feasible if and only if S covers all leaf edges.*

Proof. Clearly, any feasible solution must cover all leaf edges. Conversely, consider a set of links $S \subseteq L$ that covers all the leaf edges. We will prove that S is a feasible solution for the instance. Consider any edge $e \in E$. We will show that it is covered by S. Let $r \in V$ be the root of the star-shaped instance so that for every link $\ell \in L$, we have $r \in P_\ell$. There exists some leaf $u \in V$ such that e lies on the $u - r$ path in G. Let e' be the unique leaf edge incident to u. Consider the link $\ell' \in S$ that covers e'. Since the instance is star-shaped, we have $r \in P_{\ell'}$. Hence all edges on the $u - r$ path in G are in $P_{\ell'}$, in particular, we have $e \in P_{\ell'}$ and thus e is covered. This completes the proof of the lemma. $\square$

Lemma 2.6 tells us that in any star-shaped instance, the non-leaf edges are redundant and thus can be contracted away. The resulting graph G' then becomes a star. We will show how to design a $\frac{4}{3}$-approximation algorithm for such an instance via a reduction to the EDGE COVER problem, defined as follows.

EDGE COVER
Input: Graph $G = (V, E)$ and edge costs c_e for every edge $e \in E$.
Goal: Find a minimum cost subset of edges $F \subseteq E$ such that every vertex in V is incident to at least one edge in F.

Lemma 2.6 suggests the following reduction to the EDGE COVER problem. We create a vertex in G'' for each leaf in the star G', and create an edge between two vertices in G'' if the corresponding leaves share a link in G'. Links between a leaf and the root r of G' can be modeled by self-loops in G''. We will then be able to show that G'' has an edge cover if and only if G' has a solution of the same cost.

EDGE COVER admits a $\frac{4}{3}$-approximation that uses the following linear programming relaxation, which we call the fractional edge cover LP.

$$\begin{array}{rrcll} \text{minimize} & \displaystyle\sum_{e\in E} c_e x_e & & & \\ \text{subject to} & \displaystyle\sum_{e\in \delta(u)} x_e & \geq & 1 & \forall u \in V \\ & x_e & \geq & 0 & \forall e \in E \end{array}$$

The following classical result from polyhedral theory states that we can round a fractional edge cover into an integral one, paying at most a factor of $\frac{4}{3}$ of the cost (see [28]).

Lemma 2.7. *Given an* EDGE COVER *instance* $(G = (V, E), c)$ *and a feasible solution* x *to the fractional edge cover LP, there is a polynomial time algorithm that outputs an integral edge cover of cost at most* $\frac{4}{3}c^\intercal x$.

Now we are ready to formally give the $\frac{4}{3}$-approximation for star-shaped instances via the reduction to EDGE COVER.

Lemma 2.8. *[1] Given a star-shaped instance* (G, L, c)*, there exists a polynomial time algorithm, that given a feasible solution* x *to the cut-LP, outputs an integral solution* S *for* (G, L, c) *of cost at most* $c(S) \leq \frac{4}{3}c^\intercal x$.

Proof. Contract every edge of G that is not a leaf edge to obtain a graph G'. Note that any solution of the WEIGHTED TREE AUGMENTATION instance (G', L) covers all leaf edges in G and hence, by Lemma 2.6, is a solution to the instance (G, L) also. Thus it suffices to find a $\frac{4}{3}$ approximation for (G', L). Note that since we have contracted all non-leaf edges,

G' is a star graph consisting of a root vertex r and leaves $v_1, \dots, v_k$, that are also leaves of G. We will show how to reduce the instance (G', L) to the Edge Cover problem.

We create a graph G'' with vertex set $V[G''] = \{v_1, \dots, v_k\}$. For each link $\ell \in L$ between two leaves v_i, v_j in G', we add the edge $v_i v_j$ to G'' with cost c_ℓ. For each link $\ell \in L$ between a leaf v_i and the root r, we add the self-loop $v_i v_i$ to G'' with the same cost. Now we claim that every edge cover solution of G'' corresponds to a Weighted Tree Augmentation solution for (G', L) of the same cost and vice versa. Indeed, if $F \subseteq E[G'']$ is an edge cover, then $F \subseteq L$ is a solution to (G', L) since, for every vertex $v_i \in V[G'']$, the edge cover F must either contain an edge $v_i v_j$ for some $j \neq i$, in which case the link $v_i v_j$ covers the edge $v_i r$, or F must contain the self-loop $v_i v_i$, in which case the link $v_i r$ covers the edge $v_i r$. Conversely, if $F' \subseteq L$ is a Weighted Tree Augmentation solution for the instance (G', L), then in order to cover the edge $v_i r \in E[G']$, the solution F' must either contain a link $v_i v_j$ for some $j \neq i$, or the link $v_i r$. In either case, the link covering $v_i r$ contains v_i as its endpoint. Thus $F' \subseteq E[G'']$ is also an edge cover of G''.

So, to prove the lemma, it suffices to find an integral edge cover solution for G'' with cost at most $\frac{4}{3} c^\intercal x$. Note that, since x is feasible for the cut-LP for the instance (G, L, c), it is also feasible for the cut-LP for the instance (G', L, c). Moreover, we observe that x is a feasible solution to the edge cover LP over G''. To see this, consider any constraint $\sum_{e \in \delta(v)} x_e \geq 1$, for $v \in V[G'']$, of the edge cover LP over G''. By construction of G'', we have that e is in $\delta(v)$ if and only if e covers the edge vr in G'. Thus $\sum_{e \in \delta(v)} x_e = \sum_{\ell \in cov(vr)} x_\ell \geq 1$, where the last inequality follows from the feasiblity of x for the cut-LP over the instance (G', L, c). Thus every constraint $\sum_{e \in \delta(v)} x_e \geq 1$ of the edge cover LP is satisfied by x. Now, using Lemma 2.7, we can find an integral edge cover solution for G'' with cost at most $\frac{4}{3} c^\intercal x$, completing the proof of the lemma. □

Using the above lemma for star-shaped instances as a subroutine, the following lemma shows how to design an approximation algorithm for instances where the cost of the cross-links dominate the total cost of the fractional solution. Intuitively, the algorithm tries to apply Theorem 2.1 on the in-links and Lemma 2.8 on the cross-links. So, when the fractional cost is dominated by the cross-links, the approximation factor attained by the algorithm is closer to that of Lemma 2.8, allowing us to beat factor two.

Lemma 2.9. *[1] For any $\lambda > 1$, let x be a solution to the cut-LP. Then there is a polynomial time algorithm that outputs an integral solution S of cost at most $c(S) \leq 2\lambda c^\intercal x^{in} + \frac{4}{3}\left(\frac{\lambda}{\lambda - 1}\right) c^\intercal x^{cr}$.*

Proof. Let $E_\lambda = \{e \in E : x^{in}(cov(e)) \geq \frac{1}{\lambda}\}$ be the set of edges covered by a fraction of at least $\frac{1}{\lambda}$ by in-links. Now the vector $y = \lambda x^{in}$ covers every edge in E_λ completely, i.e., by a fraction of at least 1. Using the 2-approximation algorithm given by Theorem 2.1, we can compute a solution $S^1 \subseteq L$ that covers all edges in E_λ and has cost at most

$$c(S^1) \leq 2c^\intercal y = 2\lambda c^\intercal x^{in}.$$

Now contract all edges in E_λ to obtain a new tree G' with the edge set $E \setminus E_\lambda$. By definition of E_λ, every edge in G' is covered by a fraction of at most $\frac{1}{\lambda}$ by in-links, and thus must be covered by a fraction of at least $1 - \frac{1}{\lambda} = \frac{\lambda-1}{\lambda}$ by cross-links. That is, we have $x^{cr}(cov(e)) \geq \frac{\lambda-1}{\lambda}$ for each $e \in E(G')$. Thus the vector $y' = \frac{\lambda}{\lambda-1}x^{cr}$ is a feasible fractional solution for the instance on G'. We remove every link $\ell \in L \setminus supp(y')$. Note that this maintains the feasibility of the solution y' for the instance over G'. Since $supp(y') \subseteq L^{cr}$, it follows that the resulting instance is star-shaped, as every link that is not removed is a cross-link. Using the $\frac{4}{3}$-approximation algorithm given by Lemma 2.8, we can compute a solution $S^2 \subseteq L$ that covers all edges in $E \setminus E_\lambda$ and has cost at most

$$c(S^2) \leq \frac{4}{3}c^\intercal y' = \frac{4}{3}\left(\frac{\lambda}{\lambda-1}\right)c^\intercal x^{cr}.$$

Hence $S = S^1 \cup S^2$ covers all edges in E and so is a feasible solution for the instance on G. Moreover, we have,

$$c(S) \leq c(S^1) + c(S^2) \leq 2\lambda c^\intercal x^{in} + \frac{4}{3}\left(\frac{\lambda}{\lambda-1}\right)c^\intercal x^{cr}$$

which completes the proof of the lemma. □

Note that, once again, Lemma 2.9 gives only an approximation factor of 2 in the worst case. However, when the cost on the in-links in the support of x is small compared to the cost of the cross-links in the support of x (which is precisely the scenario when Lemma 2.5 fails to give a good approximation), the approximation given by Lemma 2.9 is closer to $\frac{4}{3}$.

2.4 The overall algorithm

Let $\mu = \frac{8(23+3\sqrt{5})}{121} \approx 1.964$. We will show that using the two backbone procedures combined will allow us to achieve approximation factor μ for Weighted Tree Augmentation for β-simple instances. Using the decomposition theorem from the previous section, we will then be able to achieve approximation factor $\mu + \epsilon$ for Weighted Tree Augmentation for general instances, for any $\epsilon > 0$. We prove this formally in the following theorem. Recall that we use `opt` to denote the cost of an optimal solution for the instance.

Theorem 2.3. *[1] Let $(G = (V, E), L, c)$ be a* Weighted Tree Augmentation *instance where $c_\ell \in [1, M]$ for every $\ell \in L$. There exists an algorithm that runs in time $n^{M^{O(1)}}$ and outputs a solution S^* of cost at most $(\mu + \epsilon)\texttt{opt}$ for any $\epsilon > 0$.*

Proof. Let $B = \frac{6}{\epsilon}$. Let x be a solution to the γ-bundle LP for $\gamma \geq \frac{28MB}{\epsilon}$. Using Lemma 2.3, we first compute a set of links L_0 of cost at most $c(L_0) \leq \frac{2}{B}c^\intercal x = \frac{\epsilon}{3}c^\intercal x$ that covers all heavy edges. After adding L_0 to the graph and contracting any cycles formed, the resulting graph G' contains only light edges with respect to x. Now we apply Theorem 2.2 and Lemma 2.4 to find a decomposition $(T^1, z^1), \ldots, (T^k, z^k)$ into β-simple subinstances, and a subset of links L_1 of cost at most $c(L_1) \leq \frac{\epsilon}{6}c^\intercal x$ that covers all the edges removed during the decomposition.

For the j-th instance (T, z) in the decomposition $(T^1, z^1), \ldots, (T^k, z^k)$, we compute the ratio $\omega^j = \frac{c^\intercal z^{cr}}{c^\intercal z^{in}}$. Define

$$\omega^* = 1 - \frac{2(5 - 2\sqrt{5})}{25 + 2\sqrt{5}} \approx 0.964$$

If $\omega^j < \omega^*$, we apply Lemma 2.5 on (T, z) to get a solution $S^j \subseteq L$ of cost $c(S^j) \leq c^\intercal z^{in} + 2c^\intercal z^{cr} + |V^0 \cap V[T]|$. Otherwise, we apply Lemma 2.9 on (T, z), with $\lambda = 3 + \sqrt{5}$, to get a solution $S^j \subseteq L$ of cost $c(S^j) \leq 2\lambda c^\intercal z^{in} + \frac{4}{3}\frac{\lambda}{\lambda-1}c^\intercal z^{cr}$. Thus, for each $j \in [k]$, we have

$$c(S^j) \leq \min\left\{c^\intercal z^{in} + 2c^\intercal z^{cr} + |V^0 \cap V[T]|, 2\lambda c^\intercal z^{in} + \frac{4}{3}\frac{\lambda}{\lambda - 1}c^\intercal z^{cr}\right\} \leq \mu c^\intercal z^j + |V^0 \cap V[T]|.$$

Now $S = S^1 \cup \ldots \cup S^j$ has cost

$$c(S) \leq \sum_{j=1}^{k} c(S^j) \leq \mu \sum_{j=1}^{k} c^\intercal z^j + |V_0| \leq \mu(1 + \epsilon)c^\intercal x + |V^0| \leq \mu\left(1 + \epsilon + \frac{\epsilon}{3}\right)c^\intercal x$$

where we used the fact that $|V^0| \leq c(L_0) \leq \frac{\epsilon}{3}c^\intercal x$. Now we return the solution $L_0 \cup L_1 \cup S$ which has cost at most

$$c(L_0) + c(L_1) + c(S) \leq \frac{\epsilon}{3}c^\intercal x + \frac{\epsilon}{6}c^\intercal x + \mu\left(1 + \frac{4}{3}\epsilon\right)c^\intercal x \leq \mu(1 + O(\epsilon))c^\intercal x$$

which concludes the proof of the theorem. □

2.5 An improvement using Chvátal-Gomory cuts

In this section, we discuss the improvement by Fiorini, Groß, Könemann and Sanità [16], who give a $\frac{3}{2}$-factor approximation for WEIGHTED TREE AUGMENTATION under the bounded costs assumption. They achieve this by improving the approximation ratio of the second backbone procedure, the one that rounds nearly star shaped instances.

2.5.1 Adding Chvátal-Gomory cuts

Recall that the second backbone procedure uses a $\frac{4}{3}$-approximation for star-shaped instances which in turn uses a $\frac{4}{3}$-approximation for EDGE COVER via the integrality gap of the fractional edge cover polyhedron. However, EDGE COVER can be solved optimally in polynomial time. Moreover, adding the CG-cut $\lambda^\intercal Ax \geq \lceil \lambda^\intercal b \rceil$ for every $\lambda \in \{0, \frac{1}{2}\}^m$ to the fractional edge cover polyhedron makes it integral. We call such CG-cuts the $\{0, \frac{1}{2}\}$-CG cuts of a polyhedron.

Definition 2.6 ($\{0, \frac{1}{2}\}$-CG cuts)**.** *Given a polyhedron $P = \{x \in \mathbb{R}^n : Ax \geq b\}$, a $\{0, \frac{1}{2}\}$-CG cut of P is an inequality of the form $\lambda^\intercal Ax \geq \lceil \lambda^\intercal b \rceil$, where $\lambda \in \{0, \frac{1}{2}\}^m$ and $\lambda^\intercal A \in \mathbb{Z}^n$.*

That is, $\{0, \frac{1}{2}\}$-CG cuts are a specialization of CG-cuts where the coordinates of the vector λ are restricted to be in $\{0, \frac{1}{2}\}$. As mentioned previously, adding all the $\{0, \frac{1}{2}\}$-CG cuts to the fractional edge cover polyhedron makes the polyhedron integral.

If we are able to use a polynomial time algorithm to optimally solve EDGE COVER rather than using the $\frac{4}{3}$-approximation algorithm in the second backbone procedure, we will be able to get rid of the $\frac{4}{3}$ factor on the $c^\intercal x^{cr}$ term in Lemma 2.9, thus improving the final approximation ratio. Unfortunately, we can't directly use a polynomial time algorithm, as there is no guarantee that the returned solution is bounded by a small enough factor of the cost of the LP solution. Since the fractional edge cover polyhderon becomes integral after adding its $\{0, \frac{1}{2}\}$-CG cuts, and since the fractional edge cover polyhedron is equivalent to the cut-LP for the star-shaped instances (this fact is inherent in the proof of Lemma 2.8), a natural idea would be to augment the cut-LP by adding all its $\{0, \frac{1}{2}\}$-CG cuts so that the resulting LP for star-shaped instances becomes integral, and then we would be able to round star-shaped instances with no loss in approximation. This is precisely the innovation of Fiorini, Groß, Könemann and Sanità.

For the cut-LP, a $\{0, \frac{1}{2}\}$-CG cut is of the form

$$\sum_{e \in E} \lambda_e x(cov(e)) + \sum_{\ell \in L} \mu_\ell x_\ell \geq \left\lceil \sum_{e \in E} \lambda_e \right\rceil \tag{2.3}$$

where λ is the multiplier for the edge covering constraints and μ is the multiplier for the non-negativity constraints. Note that, for any fixed λ, there is a unique μ such that the coefficients of the left-hand side are integral.

Let $K = supp(\lambda) = \{e \in E : \lambda_e = \frac{1}{2}\}$. Since G is a tree, there exists a (not necessarily connected) set S such that $\delta_G(S) = K$. Therefore, the right-hand side of (2.3) is equal to $\lceil |\delta_G(S)/2| \rceil$. This implies that the cut is redundant whenever $|\delta_G(S)|$ is even.

Let $\pi(S)$ denote the multiset of links ℓ such that P_ℓ intersects $\delta_G(S)$, in which the multiplicity of the link ℓ is defined to be $\lceil \frac{1}{2}|P_\ell \cap \delta_G(S)| \rceil$. Now, assuming that $|\delta_G(S)|$ is odd, we can rewrite (2.3) as

$$x(\pi(S)) \geq \frac{|\delta_G(S)| + 1}{2} \tag{2.4}$$

Thus all non-redundant $\{0, \frac{1}{2}\}$-CG cuts of the cut-LP are of the above form. Let $\mathcal{S}$ be the collection of sets S such that $|\delta_G(S)|$ is odd. We add all the $\{0, \frac{1}{2}\}$-CG cuts to the cut-LP to obtain the following LP, which we refer to as the odd-cut LP.

$$\begin{array}{rrcll} \text{minimize} & \sum_{\ell \in L} c_\ell x_\ell & & & \\ \text{subject to} & x(\pi(S)) & \geq & \frac{|\delta_G(S)| + 1}{2} & \forall S \in \mathcal{S} \\ & x_\ell & \geq & 0 & \forall \ell \in L \end{array}$$

Note that the above LP includes the edge covering constraints $\sum_{\ell \in cov(e)} x_\ell \geq 1$ for each edge $e \in E$ since if we consider the set S to be one of the connected components of $G - e$, then $x(\pi(S))$ is equal to $x(cov(e))$ and $|\delta_G(S)| = 1$. Hence the constraint $x(\pi(S)) \geq \frac{|\delta_G(S)|+1}{2}$ is equivalent to the edge covering constraint $x(cov(e)) \geq 1$ corresponding to e.

Let us take a moment to understand how the $\{0, \frac{1}{2}\}$-CG cuts might affect the structure of the LP solution. In particular, we are interested in how they affect the cross-links (since star-shaped instances have only cross-links). Root the tree at an arbitrary vertex $r \in V$ so that the cross-links and in-links are defined with respect to r, and suppose that r has odd degree. Consider the cut $\delta_G(r)$. The constraint corresponding to this cut is a non-redundant $\{0, \frac{1}{2}\}$-CG cut since r has odd degree. Every cross-link and every link incident to r has coefficient 1 in this constraint, and all other links have coefficient 0. The constraint then asserts that the sum of x-values of these links is at least $\frac{|\delta_G(r)|+1}{2}$. Indeed any integral solution must contain at least $\frac{|\delta_G(r)|+1}{2}$ such links since $|\delta_G(r)|$ is odd and any such link can cover at most 2 edges in $\delta_G(r)$. However, there can be fractional solutions that cover $\delta_G(r)$ using links with total x-value strictly less than $\frac{|\delta_G(S)|+1}{2}$. See Fig. 2.5 for such an example.

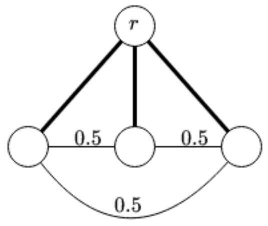


Figure 2.5: A fractional solution that is cut by the $\{0, \frac{1}{2}\}$-CG constraint corresponding to $\delta(r)$. The underlying tree is a star on 3 leaves and the links form a triangle on the leaves. Each link has fractional value 0.5.

Thus, the $\{0, \frac{1}{2}\}$-CG constraint cuts off solutions such as the one shown in Fig. 2.5 that are able to fractionally cover $\delta_G(r)$ with a low cost.

There is an alternate interpretation of the odd-cut constraints as T-join constraints of an auxilliary graph. Let T be the set of odd-degree vertices of G. Recall that a set of edges F is said to be a T-join if a vertex v has odd degree in the graph induced on F if and only if $v \in T$. Let $H = (V, E \cup L)$ be the graph obtained from G by including all the links. Let x be a solution to the odd-cut LP over G. Consider the vector $(x, y) \in \mathbb{R}^L \times \mathbb{R}^E = \mathbb{R}^{E[H]}$ where y is defined so that $y_e = x(cov(e)) - 1$ for each $e \in E$. Let S be a set such that $|\delta_G(S)|$ is odd. Since T is the set of odd degree vertices, $|\delta_G(S)|$ being odd is equivalent to stating that $|T \cap S|$ is odd. That is, S is a T-odd cut. Now we can rewrite the odd-cut constraint (2.4) as

$$\begin{aligned}
& & x(\pi(S)) &\geq \frac{|\delta_G(S)|+1}{2} \\
&\Leftrightarrow & \frac{1}{2}\sum_{e\in\delta_G(S)} x(cov(e)) + \frac{1}{2}x(\delta_L(S)) &\geq \frac{|\delta_G(S)|+1}{2} \\
&\Leftrightarrow & \sum_{e\in\delta_G(S)} (x(cov(e)) - 1) + x(\delta_L(S)) &\geq 1 \\
&\Leftrightarrow & \sum_{e\in\delta_G(S)} y_e + x(\delta_L(S)) &\geq 1 \\
&\Leftrightarrow & (x, y)(\delta_H(S)) &\geq 1
\end{aligned}$$

This holds for every set S that is a T-odd cut. Hence the odd-cut constraints are precisely the T-cut constraints (1.3) on the vector (x, y) over the graph H. A nice consequence of this is that the odd-cut constraints can be separated in polynomial time using the algorithm by Padberg and Rao [27] (Theorem 1.5). This allows us to solve the odd-cut LP in polynomial time.

2.5.2 Improving the second backbone procedure

An integer matrix $M \in \mathbb{Z}^{m \times n}$ is said to be the incidence matrix of a bidirected graph if $\sum_{i=1}^{n} |M_{ij}| \leq 2$ for every column $j \in [m]$. A *binet matrix* is any matrix of the form $B = S^{-1}R$ where $M = (SR)$ is an incidence matrix of a bidirected graph with full-row rank and R is a basis of M. Binet matrices are a generalization of network matrices that we defined in Section 1.2 and were introduced by Appa and Kotnyek [2, 3].

Appa, Kotnyek, Papalamprou and Pitsoulis [4] showed that polyhedra defined by binet matrices are described by their $\{0, \frac{1}{2}\}$-CG cuts, that is, adding the $\{0, \frac{1}{2}\}$-CG cuts to a polyhedron $\{x : Ax \geq b, x \geq 0\}$, where A is a binet matrix, makes the polyhderon integral.

Theorem 2.4. *[4] For every binet matrix $A \in \mathbb{Z}^{m \times n}$ and every vector $b \in \mathbb{Z}^m$, the integer hull of the polyhedron $P = \{x \in \mathbb{R}^n : Ax \geq b, x \geq 0\}$ is described by its $\{0, \frac{1}{2}\}$-CG cuts.*

We will show that when the instance has only up-links and cross-links, then the coefficient matrix A of the cut-LP is a binet matrix. This implies that the odd-cut LP is integral if the instance has only up-links and cross-links, allowing us to round a fractional solution at no cost.

Theorem 2.5. *[16] The odd-cut LP is integral for* Weighed Tree Augmentation *instances that contain only up-links and cross-links.*

Proof. By Theorem 2.4, it suffices to show that the coefficient matrix A of the cut-LP is a binet matrix when the instance contains only up-links and cross-links.

Let r be the root of G so that the cross-links and up-links are defined with respect to r. For a directed edge $e = uv$, let $z(e) \in \mathbb{R}^{V \setminus \{r\}}$ denote the truncated incidence vector of e defined as $z(e)_a = -1$ if $a = u$, $z(e)_a = 1$ if $a = v$ and $z(e)_a = 0$ otherwise. That is, $z(e)$ is obtained from the indicator vector corresponding to the arc uv by removing the coordinate corresponding to the vertex r. Direct all the edges away from the root r and let $R \in \mathbb{R}^{(V \setminus \{r\}) \times E}$ denote the truncated incidence matrix of the resulting directed graph $\overrightarrow{G}$. That is, R is the matrix with $|E|$ columns, containing a column $z(e)$ for each edge $e \in E$.

Now, we define a matrix $S \in \mathbb{R}^{(V \setminus \{r\}) \times L}$ that encodes the links. For a link $\ell = uv \in L$, if ℓ is an up-link with $u = lca(u, v)$, we define the vector $s(\ell)$ to be the (truncated) incidence vector of the directed link uv directed from u to v. Otherwise ℓ is a cross-link, and we define $s(\ell)$ to be the incidence vector of the *undirected* edge uv, that is, $s(\ell)_u = s(\ell)_v = 1$ and $s(\ell)_w = 0$ for all other coordinates $w \in V \setminus \{r\}$. The matrix S is then defined to the matrix consisting of the columns $s(\ell)$ for each link $\ell \in L$.

Consider the matrix $M = (SR)$. Since $\sum_{a \neq r} |S_{a\ell}| \leq 2$ for all $\ell \in L$ and $\sum_{a \neq r} |R_{ae}| \leq 2$ for all $e \in E$, it follows that M is an incidence matrix of a bidirected graph. Moreover, R is a basis of M (since the columns of R are linearly independent and $|E| = |V \setminus \{r\}|$). Thus it follows that the matrix $A = R^{-1}S$ is a binet matrix.

We claim that $A = R^{-1}S$ is the constraint matrix of the cut-LP over G. Indeed it can be observed that, for any link $\ell \in L$, the sums of the columns $z(e)$ of R that correspond to the tree edges that are in P_ℓ is equal to the corresponding column in S. Thus we have $RA = S$, or equivalently, $A = R^{-1}S$. This completes the proof of the theorem. □

Now we are ready to give the improved approximation for nearly star-shaped instances.

Lemma 2.10. *[16] Let $(G = (V, E), L, c)$ be a* Weighted Tree Augmentation *instance and x be a feasible solution to the odd-cut LP for G. Then there exists a polynomial time algorithm that computes a solution $S \subseteq L$ with cost $c(S) \leq 2c^\intercal x^{in} + c^\intercal x^{cr}$*

Proof. Let $\ell = uv \in L^{in} \setminus L^{up}$ be an in-link that is not an up-link. Let $w = lca(u, v)$ be the least common ancestor of u and v in G. We modify the solution x by increasing x_{uw} and x_{vw} by x_{uv} and dropping x_{uv} to 0. That is, we transfer the x-value from uv to its two shadows uw and vw, incurring an additional cost of $c_\ell x_\ell$ in the process. Doing this for every $\ell \in L^{in} \setminus L^{up}$ results in an instance that contains only up-links and cross-links. Moreover, the resulting vector y has cost at most $c^\intercal y \leq 2c^\intercal x^{in} + c^\intercal x^{cr}$.

Now, *supp*(y) only contains up-links and cross-links. We re-solve the odd-cut LP with the set of links restricted to *supp*(y) to obtain another odd-cut LP solution z. By Theorem 2.5, the solution z is integral and has cost at most

$$c^\intercal z \leq c^\intercal y \leq 2c^\intercal x^{in} + c^\intercal x^{cr}$$

which completes the proof of the lemma. □

2.5.3 The improved algorithm

Now we show how to use the odd-cut LP and the improved backbone procedure to get a better approximation for Weighted Tree Augmentation. We start by adding the bundle constraints to the odd-cut LP to get the following LP, which we call the odd-cut bundle LP.

$$\begin{array}{llcll}
\text{minimize} & \displaystyle\sum_{\ell\in L} c_\ell x_\ell & & & \\
\text{subject to} & x(\pi(S)) & \geq & \dfrac{|\delta_G(S)|+1}{2} & \forall S \in \mathcal{S} \\
& \displaystyle\sum_{\ell\in cov(B)} c_\ell x_\ell & \geq & \mathtt{OPT}(B) & \forall B \in \mathcal{B}_\gamma \\
& x_\ell & \geq & 0 & \forall \ell \in L
\end{array}$$

The above LP can be solved in polynomial time since both the odd-cut constraints and the bundle constraints can be separated over in polynomial time. The algorithm computes a solution x to the odd-cut bundle LP and proceeds in the same way as in the old algorithm. We first cover all the heavy cuts using Lemma 2.3 and contract any cycles formed to produce an instance that has only light cuts. Then we recursively find an α-thin edge e and apply the splitting operation on e. This results in a decomposition $(T^1, z^1), \ldots, (T^k, z^k)$ of β-simple instances as before. For each subinstance (T, z) in the decomposition, we compare the ratio of $c^\intercal z^{cr}$ versus $c^\intercal z^{in}$. If this ratio is dominated by the in-links, then we use Lemma 2.5 to get a good approximation. Otherwise the ratio is dominated by the cross-links. Now, instead of using Lemma 2.5, we will use the new rounding procedure given by Lemma 2.10 on the instance (T, z). This completes the description of the algorithm.

Note that Lemma 2.10 uses the $\{0, \frac{1}{2}\}$-CG cuts of the cut-LP of the subinstance (T, z). That is, we require that the vector z in the decomposition must be an odd-cut LP solution for T. So, to be able to use the lemma, we will need to prove that these $\{0, \frac{1}{2}\}$-CG cuts are preserved under the heavy edge contraction and splitting operations. This is done in the following lemma.

Lemma 2.11. *[16] Let $(G = (V, E), L, c)$ be a* Weighted Tree Augmentation *instance and let x be a solution to the odd-cut bundle-LP over this instance. Then for every subinstance (T, z) in the decomposition computed by Theorem 2.2, the vector z is a feasible solution to the odd-cut LP over T.*

Proof. Let G be a graph and let x be a solution to the odd-cut LP over G. First, we will prove that the odd-cut constraints are preserved under contractions. That is, we will show that if a graph G' is obtained from G by contracting edges, the vector x is a feasible solution to the odd-cut LP over G'. Consider a set $S \subseteq V[G']$ such that $|\delta_{G'}(S)|$ is odd.

Let $S' \subseteq V[G]$ be the corresponding set in G so that $\delta_G(S') = \delta_{G'}(S)$. The existence of S' is guaranteed since G' is obtained from G by contractions. Now we have,

$$\sum_{\ell \in L} \left\lceil \frac{1}{2}|P_\ell^{G'} \cap \delta_{G'}(S)| \right\rceil x_\ell = \sum_{\ell \in L} \left\lceil \frac{1}{2}|P_\ell^{G} \cap \delta_{G}(S')| \right\rceil x_\ell \geq \frac{|\delta_G(S')|+1}{2} = \frac{|\delta_{G'}(S)|+1}{2}$$

Thus the odd-cut constraint corresponding to S in G' is satisfied.

Now we will prove that the odd-cut constraints are preserved under the splitting operation. That is, we will prove that for any edge $e = uv$, the vector x^u is feasible for the odd-cut LP over G^u and the vector x^v is feasible for the odd-cut LP over G^v, where $(G^u, x^u), (G^v, x^v)$ are the subinstances obtained after splitting on the edge e. We will give the proof for (G^u, x^u), and the proof for (G^v, x^v) will follow by symmetry.

Consider a set $S \subseteq V[G^u]$ such that $|\delta_{G^u}(S)|$ is odd. Without loss of generality, by potentially complementing S, we may assume that $u \notin S$. Since $u \notin S$, it follows that $|\delta_{G^u}(S)| = |\delta_G(S)|$. For any link $\ell \in L$, define $\alpha_\ell = \left\lceil \frac{1}{2}|P_\ell^{G^u} \cap \delta_{G^u}(S)| \right\rceil = \left\lceil \frac{1}{2}|P_\ell^{G} \cap \delta_{G}(S)| \right\rceil$ to be the coefficient of the link ℓ in the constraint corresponding to S. We have,

$$\begin{aligned}
\sum_{\ell \in L} \alpha_\ell x_\ell^u &= \sum_{\ell \in L, \ell \subseteq V[G^u]\setminus\{u\}} \alpha_\ell x_\ell + \sum_{\ell = pu \in L} \alpha_\ell \left(x_\ell + \sum_{\ell' \in cov(e), p \in \ell'} x_{\ell'} \right) \\
&= \sum_{\ell \in L, \ell \subseteq V[G^u]\setminus\{u\}} \alpha_\ell x_\ell + \sum_{\ell = pu \in L} \alpha_\ell x_\ell + \sum_{\ell = pu \in L} \sum_{\ell' \in cov(e), p \in \ell'} \alpha_\ell x_{\ell'} \\
&= \sum_{\ell \in L, \ell \subseteq V[G^u]\setminus\{u\}} \alpha_\ell x_\ell + \sum_{\ell = pu \in L} \alpha_\ell x_\ell + \sum_{\ell \in L, \ell \in cov(e)} \alpha_\ell x_\ell \\
&= \sum_{\ell \in L} \alpha_\ell x_\ell \\
&\geq \frac{|\delta_{G^u}(S)|+1}{2}
\end{aligned}$$

Here the third equality uses the fact that $\alpha_\ell = \alpha_{\ell'}$ for any $\ell = pu \in L$ and $\ell' \in cov(e)$ such that $p \in \ell'$, the fourth equality follows from the fact that, for any link $\ell \subseteq V[G^v]$, we have $\alpha_\ell = 0$, and the last equality uses the feasibility of x for the odd-cut LP over G and the fact that $|\delta_G(S)| = |\delta_{G^u}(S)|$. Hence the odd-cut constraint corresponding to S is satisfied for G^u.

Since every subinstance (T, z) in the decomposition computed by Theorem 2.2 is obtained by a sequence of contraction and splitting operations, it follows that z is a solution for the odd-cut LP of T, finishing the proof of the lemma. $\square$

Using the improved approximation for nearly star-shaped instances, we are now able to get a better approximation ratio for WEIGHTED TREE AUGMENTATION.

Theorem 2.6. *[16] Let $(G = (V, E), L, c)$ be a* WEIGHTED TREE AUGMENTATION *instance. There exists a polynomial time algorithm that returns a solution $S \subseteq L$ such that $c(S) \leq (\frac{3}{2} + \epsilon)\texttt{opt}$, for any $\epsilon > 0$.*

Proof. By Lemma 2.11, any β-simple instance (T, z) in the decomposition is such that z is a solution to the odd cut LP over T. We will prove that we can construct an integral solution S for T of cost at most $c(S) \leq \frac{3}{2}c^\intercal z + \delta$, where δ is a term whose sum over all β-simple instances in the decomposition is at most $\epsilon \cdot \texttt{opt}$. The rest of the proof is then similar to the proof of Theorem 2.3, so we omit it.

To construct the desired $\frac{3}{2}$-approximation algorithm for β-simple instances, we look at the ratio of $c^\intercal z^{in}$ and $c^\intercal z$. If $c^\intercal z^{in} \leq \frac{1}{2}c^\intercal z$, then we use Lemma 2.10 to get a solution S such that

$$c(S) \leq 2c^\intercal z^{in} + c^\intercal z^{cr} = c^\intercal z^{in} + (c^\intercal z^{in} + c^\intercal z^{cr}) = c^\intercal z^{in} + c^\intercal z \leq \frac{3}{2}c^\intercal z$$

Otherwise $c^\intercal z^{in} \geq \frac{1}{2}c^\intercal z$, which implies $c^\intercal z^{cr} \leq \frac{1}{2}c^\intercal z$. Now we use Lemma 2.5 to get a solution S such that

$$c(S) \leq c^\intercal z^{in} + 2c^\intercal z^{cr} + \delta = (c^\intercal z^{in} + c^\intercal z^{cr}) + c^\intercal z^{cr} + \delta = c^\intercal z + c^\intercal z^{cr} + \delta \leq \frac{3}{2}c^\intercal z + \delta$$

where δ is a term whose sum over all β-simple instances in the decomposition is at most $\epsilon \cdot \texttt{opt}$. This completes the proof of the theorem. □

2.6 Further Improvements

Since the breakthrough result by Adjiashvili in 2017, there have been several improvements to the approximation ratio of the WEIGHTED TREE AUGMENTATION problem. In the previous section, we described one such improvement that utilizes $\{0, \frac{1}{2}\}$-CG cuts to obtain a $\frac{3}{2} + \epsilon$ factor approximation. This result improves on the second backbone procedure to achieve the approximation ratio. Another result by Grandoni, Kalaitzis and Zenklusen [21] makes an improvement on the first backbone procedure under the assumption that all links have unit cost. Together with the improved second backbone procedure using $\{0, \frac{1}{2}\}$-CG cuts, this leads to an approximation ratio of 1.458 for (unweighed) TREE AUGMENTATION. This result was later improved by Cecchetto, Traub and Zenklusen [6] who give a 1.393 approximation for Tree Augmentation by solving the more general Cactus Augmentation problem. We cover a weaker version of this last result in the next chapter.

An improvement in another direction was made by Nutov [26] who gives a $\frac{12}{7} + \epsilon$ approximation for Weighted Tree Augmentation using an algorithm that runs in time $2^{O(\frac{M}{\epsilon^2})}$. This running time is polynomial even when $M = O(\log n)$, thus it gives an approximation when costs that are logarithmic in the size of the input, rather than a constant. A recent result by Traub and Zenklusen [31] gives an approximation factor $1 + \ln 2 + \epsilon < 1.7$ for the problem with costs of arbitrary size. This was later improved in the same year by Traub and Zenklusen [30], who give a $\frac{3}{2} + \epsilon$ approximation. This result uses a local search based approach, which is considerably different than the decomposition based approach of Adjiashvili.

Chapter 3

An Approximation Algorithm for Cactus Augmentation

In this chapter, we briefly go over a $(\frac{3}{2}+\epsilon)$-approximation algorithm for the Cactus Augmentation problem given by Cecchetto, Traub and Zenklusen [6]. Due to the sequence of reductions described in Section 1.1, this also implies a $(\frac{3}{2}+\epsilon)$-approximation algorithm for the Connectivity Augmentation problem. For the sake of brevity, we only give a high-level overview of this result and skip most of the proofs.

A 2-edge-connected graph G is said to be a *cactus* if every edge of G is contained in exactly one cycle. In the Cactus Augmentation problem, we are given a cactus graph G and an additional set of edges $L \subseteq V \times V$, which we refer to as *links*. The task is to find a subset of links $F \subseteq L$ of minimum size such that adding F to G makes the graph 3-edge-connected. See Fig. 3.1 for an example of a Cactus Augmentation instance.

Cactus Augmentation
Input: A cactus graph $G = (V, E)$, an additional set of links $L \subseteq V \times V$.
Goal: Find $F \subseteq L$ of minimum cardinality such that $(V, E \cup F)$ is 3-edge-connected.

The algorithm uses the same three step decomposition framework as that of Adjiashvili that we covered in Chapter 2. The first step is to decompose the instance into a particular type of well-structured instance, which we call *k-wide* instances. The second step is to find an approximate solution for each k-wide instance. Following the scheme of Adjiashvili, this is done using two subroutines, which are referred to as *backbone procedures*. Each subroutine by itself gives an approximation factor of two in the worst case, but it can be

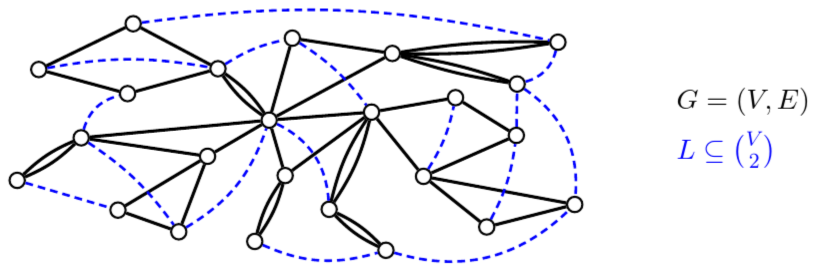


Figure 3.1: [6] An example of a Cactus Augmentation instance. The thick lines depict the base graph, which is a cactus, and the links are depicted by dashed lines.

shown that using both the procedures combined allows us to beat factor two for k-wide instances. Finally, in the last step, the algorithm combines the solutions of the k-wide subinstances to produce a solution of the original instance. This is done in such a way that the final solution costs only a negligible amount more than the union of the solutions for the subinstances. While the algorithm also works in the weighted setting under the bounded weight assumption, for simplicity, we present the results in the unweighted setting.

The rest of this chapter is divided as follows. In Section 3.1, we provide a high-level overview of the algorithm and give a formal definition of the well-structured instance that we want to decompose into. In Section 3.2, we show how to decompose the given instance into these well-structured instances and show how we can combine the solutions of the subinstances to obtain a solution for the original instance. Finally in Section 3.3, we show how to solve these well-structured instances approximately, obtaining an approximation factor below two, and completing the algorithm.

3.1 Overview of the algorithm

We start by describing the first step of the algorithm. We wish to decompose the given instance $\mathcal{I} = (G = (V, E), L)$ into well-structures instances that can be solved approximately in polynomial time. Following the ideas in Chapter 2, it would be good to find a parameter for which we can solve the entire instance optimally in polynomial time, provided that this parameter is bounded. As it turns out, one such parameter is the number of terminals of the graph. A *terminal* of a cactus graph is a vertex of degree two. Note that, since a cactus graph is also 2-edge-connected, it follows that every vertex in the graph has degree at least two. Thus terminals are exactly those vertices with minimum possible degree, and they

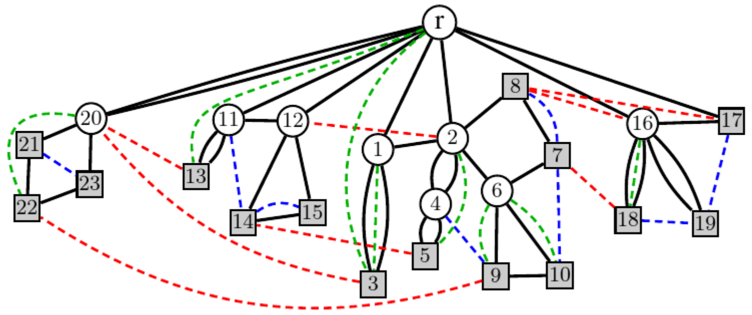


Figure 3.2: [6] A 6-wide instance with center r and four principal subcacti. The terminals are denoted by grey squares and the links are depicted by dashed lines.

can be thought of as the analog of a leaf in a tree. Let $n = |V|$ be the number of vertices in G. The following lemma by Basavaraju et al. [5] motivates the use of the number of terminals as our parameter.

Lemma 3.1. *[5]* Weighted Cactus Augmentation *can be solved optimally in time* $3^p n^{O(1)}$*, where* p *denotes the number of terminals of* G*.*

This lemma can be thought of as the analog of Lemma 2.2 from Chapter 2. Ideally, we would like to decompose our instance into several subinstances, each of which has a bounded number of terminals, and then we would be able to use Lemma 3.1 to optimally solve each subinstance. However, as in Chapter 2, trying to obtain such a decomposition proves to be a bit too ambitious. Instead, we will decompose to another type of subinstance which is, in some sense, one level higher in complexity than having a bounded number of terminals. We call such instances k*-wide*.

Definition 3.1 (k-wide instances)**.** *For an integer* $k \in \mathbb{Z}_+$*, a* Cactus Augmentation *instance* $(G = (V, E), L)$ *is said to be* k-wide *if there exists a vertex* $r \in V$ *such that each connected component of* $G - r$ *contains at most* k *terminals of* G*. We call the vertex* r *a center of* G*. Moreover, for each* $W \subseteq V \setminus \{r\}$ *that forms a connected component of* $G - r$*, we call the subgraph* $G[W \cup \{r\}]$ *a principal subcactus of* G*.*

See Fig. 3.2 for an example of a k-wide instance. k-wide instances can be thought of as the analog of β-simple instances from Chapter 2. The decomposition step is achieved by the following theorem.

Theorem 3.1. *[6] Let $\alpha \geq 1$ and $\epsilon > 0$. Given an α-approximation algorithm $\mathcal{A}$ for $\frac{32(8+3\epsilon)}{\epsilon^2}$-wide* Cactus Augmentation *instances, there is an $\alpha(1+\epsilon)$-apporixmation algorithm $\mathcal{B}$ for (general)* Cactus Augmentation *instances that calls $\mathcal{A}$ at most polynomially many times and performs further operations that take polynomial time.*

The above theorem gives us a blackbox reduction from general Cactus Augmentation instances to Cactus Augmentation instances that are k-wide, where $k = O(1)$. We go over more details as to how this theorem is proved in Section 3.2.

By Theorem 3.1, it suffices to develop an α-approximation algorithm for k-wide instances. Following the framework of Adjiashvili, we provide two backbone procedures that solve k-wide instances. First, we partition the set of links L into two types: in-links and cross-links. A link $\ell \in L$ is said to be an *in-link* if both its endpoints lie in the same principal subcactus, otherwise it is said to be a *cross-link*. Let L^{in} denote the set of in-links and let L^{cr} denote the set of cross-links.

The first backbone procedure follows the same idea as Adjiashvili's first backbone procedure (Lemma 2.5). However, since we use a round-or-cut approach (described later), we do not require any sort of bundle constraints and as such the first backbone procedure is greatly simplified. Recall that $\texttt{OPT} \subseteq L$ denotes some fixed optimal solution of the given k-wide instance. The first backbone procedure is described in the following lemma.

Lemma 3.2. *[6] For any* Weighted Cactus Augmentation *instance $(G = (V, E), L, c)$, there exists an algorithm that runs in time $3^p n^{O(1)}$, where p denotes the number of terminals of G, and computes a solution $F \subseteq L$ of cost $|F| \leq |\texttt{OPT}| + |\texttt{OPT} \cap L^{cr}|$.*

Proof. For each principal subcactus, we compute an optimal solution using Lemma 3.1 for the Cactus Augmentation instance induced on this principal subcactus. We then return the union $F \subseteq L$ of all these solutions. Observe that F is a feasible solution for the original k-wide instance, and the running time bound follows from Lemma 3.1. The claimed cost guarantee on F holds because F is not more costly than returning, for each subcactus, the links of $\texttt{OPT}$ with at least one endpoint in that subcactus (excluding the center). This leads to a cost of $|F| \leq |\texttt{OPT}| + |\texttt{OPT} \cap L^{cr}|$ since each in-link appears in exactly one subcactus and each cross-link appears in exactly two. □

The above lemma can be thought of as the analog of Lemma 2.5 for the Cactus Augmentation setting. Note that, in the worst case, we may have that $\texttt{OPT}$ uses only cross-links and thus $\texttt{OPT} = \texttt{OPT} \cap L^{cr}$, in which case the above lemma only gives us an approximation guarantee of two.

The second backbone procedure is significantly more involved, and uses the ideas of Section 2.5 based on Chvátal-Gomory cuts. It uses a polyhedron P_{cross} that is a relaxation of the convex hull of all feasible integral solutions. This polyhedron can be efficiently separated over, and hence is solvable. The second backbone procedure computes an optimal solution over this polyhedron and rounds it. Let $P_I = conv\{\chi^F : F \subseteq L$ such that $(V, E \cup F)$ is 3-edge-connected$\}$ be the convex hull of integral solutions of $\mathcal{I}$. We state the lemma now and give more details about the proof in Section 3.3.

Lemma 3.3. *[6] Given a* Weighted Cactus Augmentation *instance* $(G = (V, E), L, c)$, *there is a efficiently solvable polyhedron* $P_{cross} \supseteq P_I$ *such that there exists a polynomial time algorithm that given any* $x \in P_{cross}$ *computes an integral solution* $F \subseteq L$ *with cost* $|F| \leq x(L) + x(L^{in})$.

This leads to the following corollary.

Corollary 3.1. *[6] There exists a polynomial time algorithm that, given a* Weighted Cactus Augmentation *instance* $(G = (V, E), L, c)$, *returns a solution* $F \subseteq L$ *with cost* $|F| \leq |\texttt{OPT}| + |\texttt{OPT} \cap L^{in}|$.

Again note that, in the worst case, we could have that $\texttt{OPT} \subseteq L^{in}$, in which case the above corollary gives us an approximation guarantee of two. However, since L^{in} and L^{cr} partition the set of links, we can show that running both the backbone procedures and returning the cheaper of the two solutions beats factor 2 for k-wide instances. This in turn implies an approximation factor below 2 for general Cactus Augmentation instances, via Theorem 3.1.

Theorem 3.2. *[6] For any* $\epsilon > 0$, *there is a* $(\frac{3}{2} + \epsilon)$*-approximation algorithm for* Cactus Augmentation.

Proof. By Theorem 3.1, it suffices to provide a $\frac{3}{2}$-approximation algorithm for $O(1)$-wide Cactus Augmentation instances. We do this by computing the solutions F_1 and F_2 given by Lemma 3.2 and Corollary 3.1 respectively and returning the cheaper of the two. Since L^{cr} and L^{in} form a partition of L, we have either $|\texttt{OPT} \cap L^{cr}| \leq \frac{1}{2}|\texttt{OPT}|$ or $|\texttt{OPT} \cap L^{in}| \leq \frac{1}{2}|\texttt{OPT}|$. If $|\texttt{OPT} \cap L^{cr}| \leq \frac{1}{2}|\texttt{OPT}|$ then the solution F_1 given by Lemma 3.2 has cost,

$$|F_1| \leq |\texttt{OPT}| + |\texttt{OPT} \cap L^{cr}| \leq \frac{3}{2}|\texttt{OPT}|$$

Otherwise $|\texttt{OPT} \cap L^{in}| \leq \frac{1}{2}|\texttt{OPT}|$ and the solution F_2 given by Corollary 3.1 has cost,

$$|F_2| \leq |\texttt{OPT}| + |\texttt{OPT} \cap L^{in}| \leq \frac{3}{2}|\texttt{OPT}|$$

In either case, we will return a solution with cost at most $\frac{3}{2}|\mathtt{OPT}|$, giving a $\frac{3}{2}$-approximation for $O(1)$-wide instances. This completes the proof of the theorem. $\square$

3.2 Decomposition into k-wide instances

In this section, we go over the proof of Theorem 3.1 at a high-level. Recall that $P_I = conv\{\chi^F : F \subseteq L \text{ such that } (V, E \cup F) \text{ is 3-edge-connected}\}$ is the convex hull of integral solutions of $\mathcal{I}$. To obtain the blackbox reduction as stated in the theorem, we use Adjiashvili's splitting approach along with a round-or-cut framework. More precisely, given a point $x \in [0,1]^L$, we use x to obtain a decomposition of the input instance $\mathcal{I} = (G = (V, E), L)$ into a k-wide subinstances $\mathcal{I}_1, \ldots, \mathcal{I}_q$ such that, given α-approximation solutions $F_1, \ldots, F_q$ for $\mathcal{I}_1, \ldots, \mathcal{I}_q$, one of the following holds.

1. Either we can compute a $(\alpha + \epsilon)$-approximate solution F for $\mathcal{I}$.
2. Or we can produce a vector $h \in \mathbb{R}^L$ such that $h^\intercal x < h^\intercal \overline{x}$ for every $\overline{x} \in P_I$, thus certifying that $x \notin P_I$.

The above implies Theorem 3.1 by the round-or-cut framework as follows. First, we use binary search to guess the value of $|\mathtt{OPT}|$. We then use the ellipsoid method to find a feasible solution for the polytope $\{x \in [0,1]^L : x(L) = |\mathtt{OPT}| \text{ and } x \in P_I\}$ by using the above procedure as a separation oracle. Indeed, given $x \in [0,1]^L$, either (1) applies, in which case we have found an $(\alpha + \epsilon)$-approximate solution which can be returned as our final solution, or (2) applies in which case we can return a separating hyperplane that separates x from the convex hull of integral solutions of $\mathcal{I}$. Since the ellipsoid method terminates in polynomial time, (1) must apply for some call to the ellipsoid method for the correct guess of $|\mathtt{OPT}|$.

We now describe how to decompose our instance into k-wide subinstances such that either (1) or (2) always hold. Let $\mathcal{C}_G \subseteq 2^{V \setminus \{r\}}$ denote the cuts C of G such that $|\delta_E(C)| = 2$. Following the ideas in Chapter 2, we apply a splitting operation at well-chosen cuts of $\mathcal{C}_G$. Splitting at a cut $C \in \mathcal{C}_G$ leads to two subinstances $\mathcal{I}_C$ and $\mathcal{I}_{V \setminus C}$, where $\mathcal{I}_C$ is the instance obtained from $\mathcal{I}$ by contracting all vertices in $V \setminus C$ into a single vertex, and $\mathcal{I}_{V \setminus C}$ is the instance obtained from $\mathcal{I}$ by contracting all vertices in C into a single vertex. See Fig. 3.3 for a figure depicting the splitting operation.

The vector x then naturally decomposes into two vectors x^C and $x^{V \setminus C}$ for $\mathcal{I}_C$ and $\mathcal{I}_{V \setminus C}$ respectively. Formally, we define x^C as follows. For each link $\ell = pq \in L$ we have,

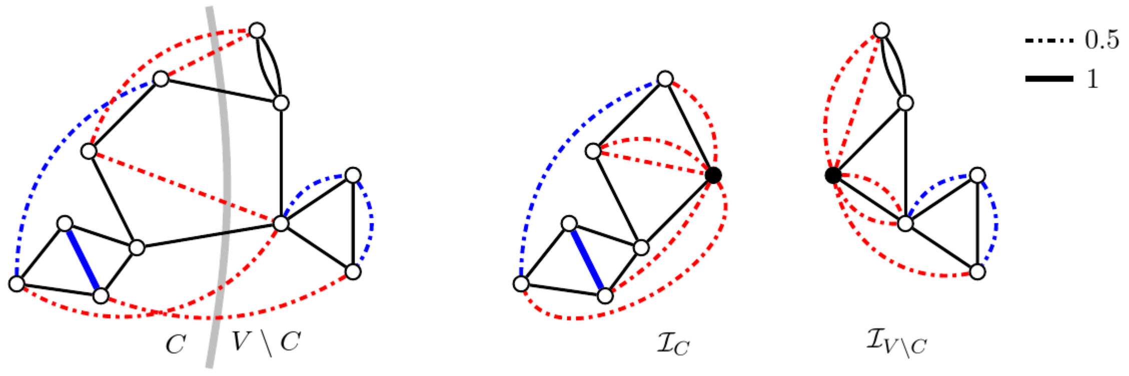


Figure 3.3: [6] Example of splitting of a Cactus Augmentation instance $\mathcal{I}$ (together with a point $x \in [0,1]^L$) along a cut $C \in \mathcal{C}_G$ into subinstances $\mathcal{I}_C$ and $\mathcal{I}_{V\setminus C}$.

$$x_\ell^C = \begin{cases} x_\ell & p \in C \text{ or } q \in C \\ 0 & \text{otherwise} \end{cases}$$

The vector $x^{V\setminus C}$ is defined symmetrically. Observe that the links in $\delta_L(C)$ appear in both subinstances produced by the splitting, whereas other links appear in only a single subinstance. Thus we have $x^C(L) + x^{V\setminus C}(L) \le x(L) + x(\delta_L(C))$. Just as in Chapter 2, to bound the increase in the cost of the subinstances, we need to ensure that we split on cuts such that $x(\delta_L(C))$ is small. We call such cuts *light*.

Definition 3.2 (Light and heavy cuts)**.** *A cut $C \in \mathcal{C}_G$ is said to be* light *with respect to x if $x(\delta_L(C)) \le \frac{16}{\epsilon}$, and C is said to be* heavy *with respect to x otherwise.*

Just as in Chapter 2, we show that we can compute a set of links of negligible cost that cover all the heavy cuts of G. This is implied by the following lemma.

Lemma 3.4. *[6] Let $\mathcal{I} = (G = (V,E), L)$ be a* Cactus Augmentation *instance. Let $r \in V$ and let $\mathcal{W} \subseteq \mathcal{C}_G$. Then the LP*

$$\begin{array}{lrcll} \textit{minimize} & x(L) & & & \\ \textit{subject to} & x(\delta_L(C)) & \ge & 1 & \forall W \in \mathcal{W} \\ & x_\ell & \ge & 0 & \forall \ell \in L \end{array} \tag{3.1}$$

has integrality gap at most 8. Moreover, given a solution x to the above LP, one can compute an integral solution with cost at most $8x(L)$ in polynomial time.

The above lemma is proved via a reduction to a certain type of rectangle hitting problem. To obtain a cheap set of links that cover all the heavy cuts, we apply Lemma 3.4 with $\mathcal{W} = \{C \in \mathcal{C}_G : C \text{ is heavy}\}$. Then, by the definition of heavy cuts, the vector $y = \frac{\epsilon}{16}x$ is feasible for the LP (3.1). Thus we get an integral solution L_H that covers every heavy cut of G with cost at most $8y(L) = \frac{\epsilon}{2}x(L)$.

Once we cover all heavy cuts by links in L_H, we start splitting on the remaining light cuts. To ensure that the total cost incurred over all splitting operations is controlled, we will split only on cuts which we call *big*, defined below. Let $T = \{v \in V : |\delta_E(v)| = 2\}$ denote the set of terminals of G.

Definition 3.3 (Small and big cuts)**.** *A set $C \subseteq V$ is said to be* small *if $|C \cap T| \leq \frac{k}{2}$, otherwise C is said to be* big*.*

Now we are ready to define the types of cuts that the algorithm chooses to split on.

Definition 3.4 (Splittable instances)**.** *Let $\mathcal{I} = (G = (V, E), L)$ be a* Cactus Augmentation *instance with root $r \in V$ and let $x \in [0, 1]^L$. Then $\mathcal{I}$ is* splittable *at a cut $C \in \mathcal{C}_G$ if C is light with respect to x, C is big and $C \neq V \setminus \{r\}$. Furthermore, $\mathcal{I}$ is said to be* unsplittable *if there exists no cut $C \in \mathcal{C}_G$ such that $\mathcal{I}$ is splittable at C.*

The reason that we do not allow splitting at $C = V \setminus \{r\}$ is that the subinstance $\mathcal{I}_C$ in this case will be identical to $\mathcal{I}$, while $\mathcal{I}_{V \setminus C}$ is a trivial instance on two vertices. Hence we do not make progress when splitting at $C = V \setminus \{r\}$.

We can think of splittable cuts as the analog of α-thin edges from Chapter 2. Indeed, if the instance contains no splittable cuts, then one can show that it has the desired k-wide structure.

Lemma 3.5. *[6] Every unsplittable instance is k-wide.*

Finally, it remains to describe how to recombine the solutions for the subinstances to obtain a $(\alpha + \epsilon)$-approximation solution for the original instance. Suppose $\mathcal{I}_C$ and $\mathcal{I}_{V \setminus C}$ are obtained from splitting across a cut C on instance $\mathcal{I}$, and suppose we have solutions F_C and $F_{V \setminus C}$ for $\mathcal{I}_C$ and $\mathcal{I}_{V \setminus C}$ respectively. Unlike Tree Augmentation, we cannot simply take the union $F_C \cup F_{V \setminus C}$ since it might not be a feasible solution for $\mathcal{I}$, as is highlighted in Fig. 3.4. Nevertheless, it can be shown that one can bound the number of links that need to be added to $F_C \cup F_{V \setminus C}$ as follows.

Lemma 3.6. *[6] Given a feasible* Cactus Augmentation *instance $\mathcal{I} = (G = (V, E), L)$, a cut $C \in \mathcal{C}_G$, and solutions $F_C, F_{V \setminus C} \subseteq L$ for $\mathcal{I}_C$ and $\mathcal{I}_{V \setminus C}$ respectively, in polynomial time, one can compute a set of links $F \subseteq L$ such that*

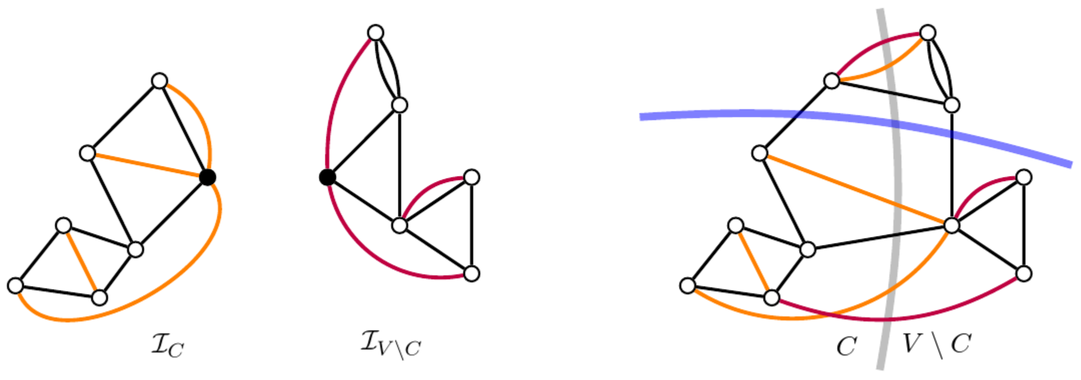


Figure 3.4: [6] This figure shows an example where we have feasible solutions for $\mathcal{I}_C$ and $\mathcal{I}_{V\setminus C}$, but their union is not a feasible solution for $\mathcal{I}$ because the cut shown in blue is not covered.

1. *$F_C \cup F_{V\setminus C} \cup F$ is a feasible solution for $\mathcal{I}$*
2. *$|F| \leq |\delta_L(C) \cap F_C| - 1$*

If the instance is splittable, the algorithm proceeds by finding a well-chosen cut $C \in \mathcal{C}_G$ such that $\mathcal{I}$ is splittable across C and then applies the splitting operation on the cut C to produce two subinstances $\mathcal{I}_C$ and $\mathcal{I}_{V\setminus C}$. The algorithm then recurses on these subinstances, finding solutions F_C and $F_{V\setminus C}$ for $\mathcal{I}_C$ and $\mathcal{I}_{V\setminus C}$ respectively, then uses Lemma 3.6 to obtain a solution for $\mathcal{I}$. Otherwise, if the instance is unsplittable, we know that the instance is k-wide by Lemma 3.5. Here the algorithm uses the α-factor approximation algorithm for k-wide instances to find a solution $F \subseteq L$. If $|F| \leq \alpha x(L)$, then the algorithm outputs F as an α-approximate solution. Otherwise $|F| > \alpha x(L)$, which implies that the optimum solution of $\mathcal{I}$ has cost greater than $x(L)$ (since F was an α-approximate solution for $\mathcal{I}$). Now the algorithm returns $h = 1$ as a separating hyperplane, since $1^\intercal x < 1^\intercal \overline{x}$ for every $\overline{x} \in P_I$

To control the cost of merging solutions when using Lemma 3.6, the algorithm looks for cuts such that $|\delta_L(C) \cap F_C| = O(\frac{1}{\epsilon^2})$. Following the round-or-cut framework, one can show that either one can find a splittable cut $C \in \mathcal{C}_G$ such that $|\delta_L(C) \cap F_C| = O(\frac{1}{\epsilon^2})$, or one can find a separating hyperplane that separates x from the convex hull of integral solutions. This completes the high-level description of the proof of Theorem 3.1.

3.3 Solving k-wide instances

The first backbone procedure is already given by Lemma 3.2. In this section, we go over the proof of the second backbone procedure at a high-level. Consider the cut-LP for Cactus Augmentation.

$$\begin{array}{rrcl} \text{minimize} & \sum_{\ell \in L} x_\ell & & \\ \text{subject to} & x(\delta_L(S)) & \geq & 1 \quad \forall S \in \mathcal{C}_G \\ & x_\ell & \geq & 0 \quad \forall \ell \in L \end{array}$$

Note that this LP can be solved in polynomial time, as it has a polynomial number of constraints since $|\mathcal{C}_G| \leq |E|^2$. Let P_{cut} denote the polyhedron consisting of feasible points of the above LP. We wish to use the approach from Section 2.5 by adding $\{0, \frac{1}{2}\}$-CG cuts to P_{cut}. However, it is unclear how one might separate over these $\{0, \frac{1}{2}\}$-CG cuts as there are exponentially many of them, and we cannot use the results from binet matrices here. Nevertheless, we can show that efficient separation is possible for the $\{0, \frac{1}{2}\}$-CG cuts of a carefully selected subset of the constraints of the cut-LP. Together with an appropriately designed rounding procedure, this will allow us to prove Lemma 3.3.

Two sets $A, B \subseteq V$ are said to *cross* if each of the sets $A \setminus B, B \setminus A, A \cap B$ are non-empty. A family $\mathcal{L}$ of subsets of V is said to be *laminar* if no two sets in $\mathcal{L}$ cross. For any sets $S, S' \in \mathcal{L}$, we say that S' is a *child* of S if $S' \subset S$ and there is no other set $R \in \mathcal{L}$ such that $S' \subset R \subset S$. If S' is a child of S, we refer to S as the *parent* of S'. It can be seen that under this parent-child relationship, the family $\mathcal{L}$ forms a rooted forest $\mathcal{F}$. Every vertex of $\mathcal{F}$ corresponds to a set in $\mathcal{L}$ and the parent and children of this vertex in $\mathcal{F}$ correspond to the parent and children of the corresponding set in $\mathcal{L}$.

One way to obtain a well-structured subset of constraints for which we can efficiently separate over the $\{0, \frac{1}{2}\}$-CG cuts is by considering the set of constraints corresponding to a laminar sub-family of $\mathcal{C}_G$, as this reduces to Tree Augmentation. More precisely, if $\mathcal{L} \subseteq \mathcal{C}_G$ is a laminar family, then the following LP relaxation, where we only want to cover cuts in $\mathcal{L}$,

$$\begin{array}{rrcl} \text{minimize} & \sum_{\ell \in L} x_\ell & & \\ \text{subject to} & x(\delta_L(S)) & \geq & 1 \quad \forall S \in \mathcal{L} \\ & x_\ell & \geq & 0 \quad \forall \ell \in L \end{array}$$

is equivalent to the cut-LP of a Tree Augmentation instance. Indeed, consider the Tree Augmentation instance $(G = (V', E'), L')$ on vertex set $V' = \mathcal{L} \cup \{V\}$. That is, the vertex set V' includes all the sets in $\mathcal{L}$ plus a new set V, corresponding to the entire vertex set of G in the Cactus Augmentation instance. The edges of G' include the edges of the rooted forest $\mathcal{F}$ corresponding to $\mathcal{L}$, which encodes the parent-child relationships of the sets of $\mathcal{L}$, plus the edges RS for every set $S \in \mathcal{L}$ which does not have a parent. For every link $\ell = uv \in L$, we add a corresponding link AB to L', where A is the inclusionwise smallest set in $\mathcal{L}$ containing u and B is the inclusionwise smallest set in $\mathcal{L}$ containing v. Now it can be seen that the resulting Tree Augmentation instance is equivalent to the problem of covering cuts in $\mathcal{L}$ using links in L. Indeed, one can observe that a link $\ell \in L$ covers a cut $S \in \mathcal{L}$ if and only if the corresponding link $\ell' \in L'$ covers the edge $SP \in E'$, where P is the unique parent of the set S in G'.

Let $P^{\mathcal{L}}$ denote the polyhedron consisting of feasible points of the above LP and let $P^{\mathcal{L}}_{CG}$ be the polyhedron obtained from $P^{\mathcal{L}}$ by adding all its $\{0, \frac{1}{2}\}$-CG cuts. Since the problem of covering a laminar family is equivalent to Tree Augmentation, we can separate over all $\{0, \frac{1}{2}\}$-CG cuts of this polyhedron using the results from Section 2.5. For a particular choice of the laminar family $\mathcal{L}$, we will solve the LP where the objective is to minimize $x(L)$ subject to $x \in P_{cut} \cap P^{\mathcal{L}}_{CG}$, which is the polyhedron obtained from P_{cut} by adding all the $\{0, \frac{1}{2}\}$-CG cuts for the subset of constraints corresponding to sets in $\mathcal{L}$.

We will carefully choose the laminar family $\mathcal{L}$ so that we can design a good rounding procedure for LP of minimizing $x(L)$ over the polyhedron $P_{cut} \cap P^{\mathcal{L}}_{CG}$. To this end, we introduce a directed LP from which we can derive both a laminar family $\mathcal{L}$ as well as a way to round solutions for the polyhedron $P_{cut} \cap P^{\mathcal{L}}_{CG}$.

This LP is inspired by the 2-approximation algorithm for Tree Augmentation given by Theorem 2.1, which replaces each in-link that is not an up-link by two up-links that cover the same set of cuts. In our case, we will replace each link $uv \in L$ by two directed links (u, v) and (v, u). Let $\overrightarrow{L} = \bigcup_{uv \in L} \{(u, v), (v, u)\}$ be the resulting set of links. We then consider the following linear program of a cut covering problem on the resulting directed graph.

$$\begin{array}{rrcll} \text{minimize} & \displaystyle\sum_{\ell \in \overrightarrow{L}} x_\ell & & & \\ \text{subject to} & x(\delta^-_{\overrightarrow{L}}(S)) & \geq & 1 & \forall C \in \mathcal{C}_G \\ & x_\ell & \geq & 0 & \forall \ell \in \overrightarrow{L} \end{array} \tag{3.2}$$

For the corresponding TREE AUGMENTATION instance, the above directed LP is equivalent to replacing every in-link by two up-links, where each of the two directions of an original link corresponds to one of the two up-links. Indeed, it can be seen that for any in-link $\ell = uv \in L$ that corresponds to a link $AB \in L'$, with $C = lca(A, B)$, a set $S \in \mathcal{L}$ is covered by one of the directed links (u, v) or (v, u) if and only if the edge $SP \in E'$ is covered by one of the up-links AC or BC which are the shadows of AB, where P denotes the parent of S in G'. Using standard uncrossing arugments, one can show that the LP (3.2) is integral.

Lemma 3.7. *[6] The linear program (3.2) is integral for any* CACTUS AUGMENTATION *instance.*

The above lemma provides a 2-approximation for CACTUS AUGMENTATION as follows. Let x be a solution to the cut-LP for the given CACTUS AUGMENTATION instance. Using x, we can construct a solution z to (3.2) where $z_{(u,v)} = z_{(v,u)} = x_{uv}$ for every $uv \in L$. Note that $z(L) = 2x(L)$. Now, by Lemma 3.7, there exists an integral optimal solution z^* for (3.2), corresponding to a set of links $S \subseteq L$. Thus S is a feasible solution for the CACTUS AUGMENTATION instance, and we have $|S| = z^*(L) \leq z(L) = 2x(L)$, proving that S is a two approximate solution.

Note that this 2-approximation algorithm by itself is insufficient for our purposes since we wish to get an approximation factor strictly below 2 for k-wide instances. We will instead follow the strategy of Lemma 2.10 that uses a 2-approximation on the in-links and rounds the cross-links without any loss.

To find the laminar family $\mathcal{L}$ for which we add $\{0, \frac{1}{2}\}$-CG cuts, we consider the dual of (3.2) given below.

$$\begin{array}{llcll} \text{maximize} & \displaystyle\sum_{C \in \mathcal{C}_G} y_C & & & \\ \text{subject to} & \displaystyle\sum_{C \in \mathcal{C}_G : \ell \in \delta^-_{\overrightarrow{L}}(C)} y_C & \leq & 1 & \forall \ell \in \overrightarrow{L} \\ & y_C & \geq & 0 & \forall C \in \mathcal{C}_G \end{array} \tag{3.3}$$

Using uncrossing techniques, one can compute an optimal solution y for the above dual LP such that $supp(y)$ is laminar. However, this is not sufficient for our purposes, and we need and additional property called *minimality*.

Definition 3.5 (Minimal solution)**.** *A solution $y \in \mathbb{R}^{\mathcal{C}_G}$ to (3.3) is* minimal *if for any $\epsilon > 0$ and any two sets $C_1, C_2 \in \mathcal{C}_G$ with $C_1 \subset C_2$, the point $y - \epsilon\chi^{\{C_2\}} + \epsilon\chi^{\{C_1\}}$ is not a feasible solution to (3.3) .*

That is, minimality means that we cannot move dual weight from any set to a strictly smaller one. Again, using uncrossing techniques, we can show that we can compute an optimal dual solution y^* to (3.3) that is minimal and has laminar support.

Lemma 3.8. *[6] One can compute, in polynomial time, an optimal solution y^* to (3.3) such that y^* is minimal and supp(y^*) is laminar.*

The following lemma shows how to use y^* to bound the cost required to complete a set of cross-links to a complete solution. In words, the lemma says that a set of cross-links $R \subseteq L^{cr}$ can be completed into a solution at a cost no higher than the total y^*-load on the cuts in $\mathcal{C}_G$ not crossed by R.

Lemma 3.9. *[6] Let $R \subseteq L^{cr}$. Then one can efficiently compute a set $F \subseteq L$ such that*

1. *$R \cup F$ is a* Cactus Augmentation *solution*

2. *$|F| \leq \sum_{C \in \mathcal{C}_G : \delta_L(C) \cap R = \emptyset} y^*_C$, where y^* is the solution given by Lemma 3.8*

Proof sketch. Consider the vector y obtained from y^* by restricting to sets $C \in \mathcal{C}_G$ such that $R \cap \delta_L(C) = \emptyset$. Consider the instance $\mathcal{I}'$ obtained from $\mathcal{I}$ by adding R to the base graph, and then applying reductions so that the base graph becomes a cactus. It can now be shown that y is an optimal solution to the LP (3.3) over the resulting instance $\mathcal{I}'$. The proof of this fact crucially uses the minimality of y^* and we omit this proof for the sake of brevity. Instead, we will show how it implies the lemma.

The point y being an optimal solution of (3.3) implies, by strong duality, that that there exists a solution x for (3.2) with the same cost. Moreover, this solution x is integral by Lemma 3.7. That is, the vector x corresponds to an integral solution $F \subseteq L$ for the instance $\mathcal{I}'$ such that

$$|F| = \sum_{C \in \mathcal{C}_G : \delta_L(C) \cap R = \emptyset} y^*_C.$$

Furthermore, observe that $R \cup F$ is a solution for the instance $\mathcal{I}$, which completes the proof of the lemma. □

Following the ideas and lemmas shown so far, we provide an overview of the proof of Lemma 3.3.

Proof sketch of Lemma 3.3. Define $\mathcal{L} = supp(y^*)$ where y^* is the minimal solution to (3.3) with laminar support given by Lemma 3.8. This is the laminar sub-family for which we add the $\{0, \frac{1}{2}\}$-CG cuts. Note that the polyhedron $P_{CG}^{\mathcal{L}}$ can be separated over in polynomial time using the results in Section 2.5 since $P^{\mathcal{L}}$ is the polyhedron of a Tree Augmentation instance. We will show that $P_{cross} = P_{cut} \cap P_{CG}^{\mathcal{L}}$ is the desired polyhedron, as required in Lemma 3.3. Observe that P_{cross} can be separated over in polynomial time since both P_{cut} and $P_{CG}^{\mathcal{L}}$ can be separated over in polynomial time.

Let $x \in P_{cross}$, we need to show that we can compute, in polynomial time, a solution $F \subseteq L$ such that $|F| \leq x(L) + x(L^{in})$. Note that we can interpret x as a fractional solution to the Tree Augmentation instance over the laminar family $\mathcal{L}$, that is, x satisfies,

$$x(\delta_L(C)) \geq 1, \forall C \in \mathcal{L}$$

We now consider an auxilliary problem. We replace each in-link with two directed links in opposing directions, that is, we replace L^{in} with the directed link set

$$\overrightarrow{L_{in}} = \bigcup_{uv \in L^{in}} \{(u, v), (v, u)\}$$

The auxilliary problem seeks to cover cuts in $\mathcal{L}$ using links in $\overrightarrow{L_{in}} \cup L^{cr}$, where a cut $C \in \mathcal{L}$ is covered by a link $\ell \in \overrightarrow{L_{in}} \cup L^{cr}$ if either $\ell \in L^{cr}$ and $\ell \in \delta_L(C)$ or $\ell \in \overrightarrow{L_{in}}$ and $\ell \in \delta_L^-(C)$. Following the connection between Tree Augmentation and covering laminar families, it can be shown that this auxilliary problem is equivalent to a Tree Augmentation instance that contains only cross-links (corresponding to links in L^{cr}) and up-links (corresponding to links in $\overrightarrow{L_{in}}$).

Now, we follow the proof strategy of Lemma 2.10. Let $z \in \mathbb{R}^{L^{cr} \cup \overrightarrow{L_{in}}}$ be defined as $z_\ell = x_\ell$ for $\ell \in L_{cross}$ and $z_{(u,v)} = z_{(v,u)} = x_{uv}$ for $\ell = uv \in L^{in}$. Now we observe that z is feasible for the natural LP relaxation for the auxilliary problem, that is, we have,

$$z(\delta_{L^{cr}}(C)) + z(\delta^-_{\overrightarrow{L_{in}}}(C)) \geq 1, \forall C \in \mathcal{L}$$

Again, following the connection between Tree Augmentation and covering laminar families, z can be intepreted as a fractional solution for the Tree Augmentation problem over $\mathcal{L}$ whose support contains only cross-links and up-links, where every link $\ell \in supp(z) \cap L^{cr}$ corresponds to a cross-link, and every directed link $(u, v) \in supp(z) \cap \overrightarrow{L_{in}}$ corresponds to one of the up-links that are shadows of the in-link uv. Moreover, because $x \in P_{CG}^{\mathcal{L}}$, the vector z also fulfills the $\{0, \frac{1}{2}\}$-CG constraints for the Tree Augmentation problem over $\mathcal{L}$. This allows us to invoke Theorem 2.5 to obtain a set of links $F^{\mathcal{L}} \subseteq L^{cr} \cup \overrightarrow{L_{in}}$ such that $F^{\mathcal{L}}$ is a solution to the auxilliary problem, and $|F^{\mathcal{L}}| \leq z(L) = x(L) + x(L^{in})$.

Note that although $F^{\mathcal{L}}$ covers every cut in $\mathcal{L}$, it might not cover other cuts in $\mathcal{C}_G$ and hence might not correspond to a feasible Cactus Augmentation solution. Nevertheless, we can show that, using Lemma 3.9, we can replace $\overrightarrow{F_{in}} = F^{\mathcal{L}} \cap \overrightarrow{L_{in}}$ by a set of links $S \subseteq L$ of size at most $|\overrightarrow{F_{in}}|$ such that $F^{cr} \cup S$ is a solution to the Cactus Augmentation instance, where $F^{cr} = F^{\mathcal{L}} \cap L^{cr}$. We return $F^{cr} \cup S$ as our solution. The desired cost bound on $F^{cr} \cup S$ now follows,

$$|F^{cr} \cup S| = |F^{cr}| + |S| \leq |F^{cr}| + |\overrightarrow{F_{in}}| = |F^{\mathcal{L}}| \leq x(L) + x(L^{in})$$

$\square$

Chapter 4

Integrality gap of the cut-LP

In this chapter, we present a result by Nutov [26] that bounds the integrality gap of the cut-LP of the Tree Augmentation problem with unit costs. Unlike the previous results covered in this thesis, this result does not use a decomposition based method, rather it uses an iterative greedy algorithm. We first write down the LP that we are interested in, and its dual.

$$\begin{array}{rrcll} \text{minimize} & \displaystyle\sum_{\ell \in L} x_\ell & & & \\ \text{subject to} & \displaystyle\sum_{\ell \in cov(e)} x_\ell & \geq & 1 & \forall e \in E \\ & x_\ell & \geq & 0 & \forall \ell \in L \end{array}$$

Its dual is:

$$\begin{array}{rrcll} \text{maximize} & \displaystyle\sum_{e \in E} y_e & & & \\ \text{subject to} & \displaystyle\sum_{e \in P_\ell} y_e & \leq & 1 & \forall \ell \in L \\ & y_e & \geq & 0 & \forall e \in E \end{array}$$

Cheriyan, Karloff, Khandekar and Könemann [8] proved that there is an infinite sequence of unit cost Tree Augmentation instances $(G_1, L_1), (G_2, L_2), \ldots$ such that, for each $k \in \mathbb{Z}$, the ratio of the optimum integral solution to the optimum fractional solution of the instance (G_k, L_k) is at least $\frac{k+1}{2k/3+1}$. , thus proving a lower bound on the integrality gap of the cut-LP that approaches $\frac{3}{2}$. In this section we will provide an upper bound on the integrality gap that was shown by Nutov [26]. We will prove that the integrality gap of the cut-LP is at most $\frac{28}{15}$ using a *dual fitting* method. That is, we will show how to construct an integral feasible primal solution J and a (possible infeasible) dual solution $y \in \mathbb{R}_+^E$ that has the following properties.

- y fully pays for J, that is, $|J| \leq 1^\intercal y$.
- y violates the dual constraints by a factor of at most $\rho = \frac{28}{15}$. That is, $\sum_{e \in P_\ell} y_e \leq \rho$, for all $\ell \in L$.

These two properties imply the bound on the integrality gap as follows. Let $\texttt{opt}_{\texttt{LP}}$ denote the optimum value of the cut-LP. We have,

$$\frac{1^\intercal y}{\rho} \leq \texttt{opt}_{\texttt{LP}} \leq |J| \leq 1^\intercal y \leq \rho \cdot \texttt{opt}_{\texttt{LP}}$$

Here, the first and last inequalities hold since, by the second property, $\frac{y}{\rho}$ is feasible for the dual. The second inequality holds since J is a feasible solution to the primal, and the third inequality follows from the first property. From this, it follows that $|J| \leq \rho \cdot \texttt{opt}_{\texttt{LP}}$. Hence J is an integral Tree Augmentation solution of cost at most $\rho = \frac{28}{15}$ times the optimum cut-LP solution, proving that the integrality ratio of the cut-LP is at most $\frac{28}{15}$.

The rest of this chapter is split into two sections. Section 4.1 defines necessary notation and definitions required for the algorithm, and Section 4.2 describes how the algorithm constructs the dual vector y satisfying the required primal-dual properties.

4.1 Preliminaries

Root the graph G at an arbitrary vertex $r^* \in V$. We employ a iterative greedy algorithm. The algorithm iteratively finds a pair T, J', where T is a subtree of G and J' covers all edges in T. The algorithm will then contract the edges of T and add J' to the partial solution J, and recurse.

We refer to the nodes obtained via contractions as *compound nodes*, and the remaining nodes as *original nodes*. The set of non-leaf compound nodes is denoted as C.

To find T and J', the algorithm maintains a matching M on the leaves of G that are original nodes. A subtree T is said to be *M-compatible* if, for all $uv \in M$, either $u, v \in V[T]$ or $u, v \in V[G \setminus T]$. That is, T is *M-compatible* if no matching edge in M crosses the cut $\delta_L(V[T])$. The algorithm will only do contractions on subtrees T that are M-compatible. Let $V[M] = \{u, v \in V : uv \in M\}$ be the set of vertices matched by M. For a subtree T of G, we define $M[T] = \{uv \in M : u, v \in V[T]\}$ to be the matching edges of M within T.

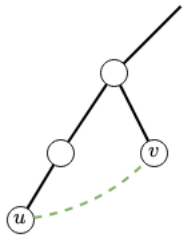


Figure 4.1: Figure depicting a twin-link. The link $\ell = uv$, denoted by a dashed line, is a twin-link as the contraction of P_ℓ results in a new leaf in the graph.

We let H denote the set of leaves of G. A link ℓ between two leaves is said to be a *twin-link* if the compound node obtained by contraction of the subtree P_ℓ is a leaf in the contracted graph. In other words, $\ell = uv$ is a twin-link if every vertex in P_ℓ except for u, v and $lca(u, v)$ has degree two in G, and $lca(u, v)$ has degree three in G (see Fig. 4.1). The least common ancestor $lca(u, v)$ of the endpoints of a twin-link $\ell = uv$ is called a *stem*. We denote the set of twin-links by L^{twin}. Observe that no two twin-links can share a common endpoint.

Let U denote the set of leaves of G unmatched by M. We say that a subtree T is *semi-closed* if is it M-compatible and there is no link between a vertex in $U \cap V[T]$ and another vertex in $V \setminus V[T]$. In other words, T is semi-closed if there is no link ℓ with one endpoint in T and the other outside T such that either $\ell \in M$ or ℓ is incident to a vertex in $U \cap V[T]$.

There is a certain type of semi-closed subtree that we would like to avoid, which we call a *dangerous* subtree. We say that a semi-closed subtree is *dangerous* if $|M[T]| = |U \cap V[T]| = 1$, $|C \cap V[T]| = 0$, and if w is the leaf of T unmatched by M and $\ell = uv \in M$ is the unique matching edge in $M[T]$, then $uw \in L$ for some endpoint u of ℓ such that uw is not a twin-link, and there is a link between the other endpoint v and another vertex outside T. See Figure 4.2 for examples of dangerous subtrees. Note that since every dangerous subtree is also semi-closed, there cannot be a link between the unmatched vertex w and any vertex outside T.

There are two types of iterative contractions that the algorithm uses. First, the algorithm tries to find a link ℓ with both endpoints in U. If such a link exists, the algorithm adds ℓ to the partial solution J and contracts the subtree P_ℓ. We call this operation the *greedy contraction*. Note that P_ℓ forms an M-compatible subtree since both endpoints of ℓ are in U. If the algorithm cannot find a link ℓ with both endpoints in U (that is, there are no greedy contractions available), then the algorithm will find a *non-dangerous*

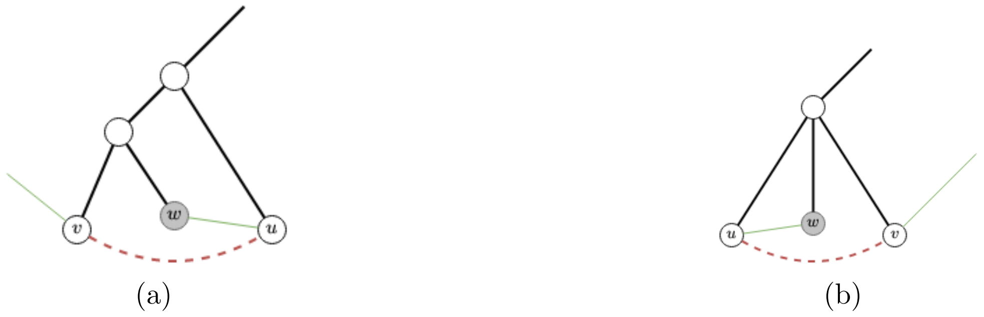


Figure 4.2: Figure showing two examples of dangerous semi-closed subtrees. The dashed line denotes the unique link $uv \in M$. The shaded vertex w is the unique unmatched vertex in the subtree.

semi-closed subtree T and a subset of links J' covering T and proceed by adding J' to the partial solution J and contracting the edges of T. We call this operation the *semi-closed contraction*. So in other words, the algorithm iteratively looks for either a greedy contraction or a semi-closed contraction and terminates when all edges in the graph have been contracted, at which point the algorithm will return the partial solution J as the final solution. The following lemma, proved in an earlier paper by Even, Feldman, Kortsarz and Nutov [15], guarantees that the algorithm is always able to find either a greedy contraction or a semi-closed contraction.

Lemma 4.1. *[15] Suppose that M has no twin-links and that the current tree G was obtained from the initial tree by sequentially applying either a greedy contraction or a semi-closed contraction. Then, if G has no greedy contraction, there exists a polynomial time algorithm that finds a non-dangerous semi-closed subtree T of G and a subset of links $J' \subseteq L$ covering T of size $|J'| = |M[T]| + |U \cap V[T]|$.*

The above lemma always returns a semi-closed subtree that is *non-dangerous*. This fact is crucial to the analysis. We skip the proof of Lemma 4.1 and instead refer readers to [15]. Observe that whenever the algorithm contracts some edge e of G, the algorithm also adds a link covering e to J, thus it follows that the final solution J computed by the algorithm is feasible, and it only remains to bound the cost of the solution. In the following section, we prove that the size of the solution returned by the algorithm is at most $|J| \leq \frac{28}{15}\texttt{opt}_{\texttt{LP}}$.

4.2 Constructing the dual solution

We describe how contruct the dual vector y satisfying the properties necessary for the dual fitting scheme. The algorithm maintains a dual vector y and updates this vector after every semi-closed contraction. We want the dual vector y to satisfy certain properties after every contraction step and we will say the vector y is *good* if it satisfies these properties. Before we can describe these properties, we need to first define some terminology.

Given a vector $y \in \mathbb{R}^E$ and a link $\ell \in L$, the *load* on the link ℓ with respect to G and y is defined to be

$$\sigma(\ell) = \sum_{e \in P_\ell^{G^*}} y_e$$

That is, the load on a link is defined to be the left-hand side of the constraint corresponding to the link in the dual. Recall that we need to ensure that the load on ℓ is at most $\frac{28}{15}$ for every link $\ell \in L$. Note here that the load is defined with respect to the original graph G^* and not the current graph G.

Given a vector $y \in \mathbb{R}^E$ and a node c obtained by contracting a (possibly trivial) subtree S of G^*, the *credit* $\pi(c)$ of the node c with respect to G and y is defined to be the sum of the dual variables y of the edges of the subtree S and the parent edge of c minus the number of links used by the algorithm to contract S into c. That is, if J' was the set of links used to contract S, and if e is the parent edge of c, we have

$$\pi(c) = \sum_{e' \in E[S]} y_{e'} + y_e - |J'|.$$

Note that, if c has no parent edge, then by convention we assume that $y_e = 0$, that is, $\pi(c) = \sum_{e' \in E[S]} y_{e'} - |J'|$ in this case. For a subset $S \subseteq V$, we define $\pi(S) = \sum_{v \in S} \pi(v)$.

For a link ℓ, the *level* $\phi(\ell)$ of ℓ is the number of compound nodes and original leaves that are endpoints of ℓ. By definition, we have $\phi(\ell) \in \{0, 1, 2\}$. Note that if both endpoints of ℓ lie in the same compound node, then ℓ is a loop and $\phi(\ell) = 2$.

Now we are ready to describe the properties that we want the dual vector y to satisfy. Recall that r^* is the arbitrarily designated root of G. Any contraction of a subtree T containing r^* results in a new (compound) root of G, and we will then continue to refer to this new root as r^*.

Definition 4.1. *Given a* Tree Augmentation *instance $(G = (V, E), L)$ that is obtained from the original instance $(G^* = (V^*, E^*), L)$ by a sequence of greedy contractions and semi-closed contractions, a vector $y \in \mathbb{R}_+^L$ is said to be* good *for G if the following holds.*

1. *$\pi(r^*) \geq 0$, and $\pi(c) \geq 1$ if $c \neq r^*$ and either c is a leaf that is unmatched by M or c is a compound node.*

2. *For any link $\ell \in L$, we have*
   - $\sigma(\ell) \leq \frac{28}{15}$ *if* $\phi(\ell) = 2$
   - $\sigma(\ell) \leq \frac{16}{15}$ *if* $\phi(\ell) = 1$
   - $\sigma(\ell) = 0$ *if* $\phi(\ell) = 0$

We will start with a good vector y and at every step we maintain the fact that the currect vector y is good for the current tree G. At the end of the algorithm, the tree G will consist of a single node r^*, and every link will be a self-loop of this node. The good-ness of y will then imply that $\sigma(\ell) \leq \frac{28}{15}$ for every link $\ell \in L$. Thus $\frac{y}{\rho}$ is a feasible dual solution for the original graph G^*. Moreover, since $\pi(r^*) \geq 0$, we will have that $\sum_{e \in E} y_e \geq |J|$, implying that the dual solution y pays for the solution J.

Now we are ready to show how to construct an initial good assignment y and how to update y at each contraction step. We first describe the initial assignment to y. We use notation y_v to denote the dual variable y_e where $e \in E$ is the parent edge of v. Let $L[H]$ denote the set of links $\ell \in L$ with both endpoints in H. The algorithm first computes a maximal matching M on the set of leaves H among the non-twin-links (that is, M is a maximal matching in the graph $(H, L[H] \setminus L^{twin})$), and a maximal matching J on the nodes in H that are unmatched by M (that is, J is a maximal matching in the graph $(H, L[H] \setminus V[M])$). Note that, since M is maximal matching among the non-twin-links, it follows that every link in J is a twin-link. The initial assignment of y is as follows.

$$y_v = \begin{cases} 1 & \text{if } v \text{ is a leaf that is unmatched by } M \text{ and } J \\ 4/5 & \text{if } v \text{ is matched by } M \\ 14/15 & \text{if } v \text{ is matched by } J \\ 2/15 & \text{if } v \text{ is a stem of a twin-link in } J \end{cases} \tag{4.1}$$

After the algorithm intializes y as above, the algorithm proceeds to contract the subtree P_ℓ for every link ℓ in J. We call this step the *twin-link contraction step.* Note that, by the definition of twin-links, it follows that any compound node formed in this step is a leaf in the resulting graph. The following claim tells us that the initial dual assignmentment is good for the graph obtained after the twin-link contraction step.

Claim 4.1. *After the twin-link contraction step, the initial dual solution y is good.*

Proof. First we prove that $\pi(c) \geq 1$ for any vertex c that is either an unmatched leaf or a compound node. Suppose that c is an original unmatched leaf, then we have $y_c = 1$ in the initial assignment, and hence $\pi(c) \geq y_c = 1$ as desired. Otherwise c is a compound node. The only compound nodes are ones obtained by contracting P_ℓ for a twin-link $\ell = uw$ in J. Let v be the stem of this twin-link ℓ. Now we have $\pi(c) \geq y_u + y_w + y_v - 1 = \frac{14}{15} + \frac{14}{15} + \frac{2}{15} - 1 = 1$.

Now we will prove that the second property in Definition 4.1 holds. Let G' be the graph obtained from G^* after the twin-link contraction step. Note that the only edges that have non-zero y value are either leaf edges of the graph G', or are edges contained inside of a compound node of G'. Consider a link $\ell \in L$. If $\phi(\ell) = 0$ then ℓ must lie between two non-leaf nodes of G', which implies $\sigma(\ell) = 0$ as desired.

Now suppose $\phi(\ell) = 1$. Let $\ell = uv$ and let v be the endpoint of ℓ that is an original leaf or a compound node of G'. If v is an original leaf then, since the other endpoint u is not a leaf of G', the only non-zero term in $\sum_{e \in P_\ell^{G^*}} y_e$ corresponds to y_v, which can be at most 1. Thus $\sum_{e \in P_\ell^{G^*}} y_e \leq 1$. Otherwise if v is a compound node, then v must have been obtained by a twin-link contraction. In this case, $P_\ell^{G^*}$ can contain at most two edges e such that y_e is non-zero, one edge that is the parent of the stem of the twin-link, that has y value $\frac{2}{15}$, and another edge that is a leaf edge of one of the endpoints of the twin-link that has y value $\frac{14}{15}$. Thus $\sum_{e \in P_\ell^{G^*}} y_e \leq \frac{16}{15}$. In either case, we have $\sum_{e \in P_\ell^{G^*}} y_e = \sigma(\ell) \leq \frac{16}{15}$ as desired.

It remains to consider the case when $\phi(\ell) = 2$. We break this case into a variety of subcases and show that in each subcase we have $\sigma(\ell) \leq \frac{28}{15}$. Let $\ell = uv$. If at least one endpoint u or v is an internal vertex of G^*, then are done since $P_\ell^{G^*}$ can contain the parent edges of at most two stems and at most one leaf, which implies that $\sigma(\ell) = \sum_{e \in P_\ell^{G^*}} y_e \leq \frac{2}{15} + \frac{2}{15} + 1 = \frac{19}{15} \leq \frac{28}{15}$. Thus we may assume that both endpoints u, v of ℓ are leaves of the original graph G^*.

Suppose that both endpoints u, v are unmatched in M and J. Then, if uv is not a twin-link, the set of links $M \cup \{uv\}$ is a matching in the graph $(H, L[H] \setminus L^{twin})$, contradicting the maximality of M, and if uv is a twin-link then $J \cup \{uv\}$ is a matching in the graph $(H, L[H] \setminus V[M])$, contradicting the maximality of J. Hence, at least one endpoint is matched either by M or J.

Now suppose that one endpoint u is matched by J and the other endpoint v is unmatched in both M and J, then, if uv were not a twin-link, the set of edges $M \cup \{uv\}$ is matching in $(H, L[H] \setminus L^{twin})$, a contradiction to the maximality of M. Thus uv must be a twin-link. Consider the link $uw \in J$ that matches the endpoint u. Note that uw is also a twin-link, as otherwise $M \cup \{uw\}$ would be a matching in $(H, L[H] \setminus L^{twin})$, a contradiction to the maximality of M. Thus uv and uw are two twin-links that share a common endpoint u, a contradiction.

Now suppose that one endpoint u is matched by M and the other endpoint v is unmatched in both M and J, then observe that $P_\ell^{G^*}$ cannot contain the parent edge of any stem of a twin-link in J, and hence contains at most two edges e with non-zero y value, one corresponding to y_u and the other corresponding to y_v. So we have $\sigma(\ell) = y_u + y_v = \frac{4}{5} + 1 \le \frac{28}{15}$.

Now suppose both u and v are matched by M. Again, observe that $P_\ell^{G^*}$ cannot contain the parent edge of a stem of a twin-link in J, and thus contains at most two edges with non-zero y values, which are the parent edges of u and v. So we have $\sigma(\ell) = \frac{4}{5} + \frac{4}{5} = \frac{8}{5}$.

Now suppose both u and v are matched by J. In this case $\ell = uv$ must be a twin-link, as otherwise $M \cup \{uv\}$ would be a matching in the graph $(H, L[H] \setminus L^{twin})$, contradicting the maximality of M. This implies that $\ell \in J$, as otherwise there must be some other twin-link $\ell' \in J$ that matches u, but then the two twin-links ℓ and ℓ' share an endpoint, a contradiction. Thus ℓ is a twin-link of J. Now $P_\ell^{G^*}$ has exactly two edges with non-zero y-values, namely, the parent edges of the endpoints u and v. So we have $\sigma(\ell) = \frac{14}{15} + \frac{14}{15} = \frac{28}{15}$.

The only remaining case to consider is when one endpoint u is matched by M and the other endpoint v is matched by J. Here $P_\ell^{G^*}$ can contain at most three edges with non-zero y value, one parent edge from each endpoint and potentially one parent edge of a stem of a twin-link. Now we have $\sigma(\ell) \le \frac{4}{5} + \frac{14}{15} + \frac{2}{15} = \frac{28}{15}$. Therefore, in each case we have $\sigma(\ell) \le \frac{28}{15}$ as desired. Hence the second property of Definition 4.1 also holds, completing the proof of the claim. $\square$

Now we show how to modify the dual vector y at every contraction step so as to maintain the condition that y is good for the current graph G. Recall that there are two types of contractions that the algorithm executes, greedy contractions and semi-closed contractions. As it turns out, we do not need to modify y after a greedy contraction. This is proved in the next claim.

Claim 4.2. *Let G be obtained from the initial tree G^* by a sequence of greedy or semi-closed contractions, and let y be dual variables that are good for G. Suppose G' was obtained from G by a greedy contraction. Then y is good for G' also.*

Proof. Let $\ell = uv \in L$ be the contracted link. Let σ' be the loads with respect to G' and y and let π' be the credits with respect to G' and y. By definiton of greedy contraction, we have $u, v \in U$ and hence $\pi(u) \ge 1$ and $\pi(v) \ge 1$. Thus, we have $\pi'(c) = \pi(u) + \pi(v) - 1 \ge 1$, where c is the new node obtained in the contraction. Thus property (1) in Definition 4.1

is satisfied with respect to G' and y. Note that, due to the contraction, the level of any link $\ell' \in L$ can only increase. Moreover, since we do not change the dual vector y, we have $\sigma'(\ell') = \sigma(\ell')$ for all $\ell' \in L$. Thus property (2) in Definition 4.1 for G' and y' follows by the fact that y is good for G, and the fact that load upper bounds in property (2) are increasing with the level of the link. This completes the proof of the claim. □

It remains to show how to modify y after a semi-closed contraction of a subtree T. The next claim shows that if T contains at least one non-leaf compound node, or if T contains at least two matching edges of M, then we do not need to modify y.

Claim 4.3. *Let G be obtained from the initial tree G^* by a sequence of greedy or semi-closed contractions, and let y be dual variables that are good for G. Suppose G' was obtained from G by a semi-closed contraction of a subtree T given by Lemma 4.1 such that either $|C \cap V[T]| \geq 1$ or $|M[T]| \geq 2$. Then y is good for G' also.*

Proof. Let π' be the node credits with respect to G' and y and let σ' be the loads with respect to G' and y. Let c be the node created by the contraction of T'. Consider some vertex $v \in V[T]$ matched by M. Then v must be an original node as every contraction done by the algorithm is M-compatible, and hence $y_v = \frac{4}{5}$ by the initial assignment. Moreover, for any $v \in U \cup C$, we have $\pi(v) \geq 1$ since y is good. Thus we have,

$$\begin{aligned}
\pi'(c) &\geq \pi(C \cap V[T]) + \frac{8}{5}|M[T]| + \pi(U \cap V[T]) - (|M[T]| + |U \cap V[T]|) \\
&\geq |C \cap V[T]| + \frac{8}{5}|M[T]| + |U \cap V[T]| - (|M[T]| + |U \cap V[T]|) \\
&= |C \cap V[T]| + \frac{3}{5}|M[T]| \\
&\geq 1
\end{aligned}$$

where the first inequality uses the fact that the solution J' given by Lemma 4.1 has size $|J'| = |M[T]| + |U \cap V[T]|$ and the last inequality uses the fact that either $|C \cap V[T]| \geq 1$ or $|M[T]| \geq 2$. Thus property (1) in Definition 4.1 holds. As in Claim 4.2, property (2) follows from the fact that we do not change the dual vector y, completing the proof of the claim. □

The remaining case is when T does not have any non-leaf compound node and T contains at most one matching edge in M. The following claim tells us that if T contains no matching edges, then we can modify the dual vector y to another vector y' that is good for the new graph obtained by the contraction.

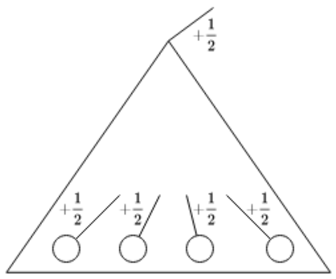


Figure 4.3: Figure depicting the change in the dual variables as in Claim 4.4. The variables corresponding to the leaf edges of T are increased by $\frac{1}{2}$ and the variable corresponding to the parent edge of T is increased by $\frac{1}{2}$.

Claim 4.4. *Let G be obtained from the initial tree G^* by a sequence of greedy or semi-closed contractions, and suppose that G has no greedy contraction. Let y be dual variables that are good for G. Suppose G' was obtained from G by a semi-closed contraction of a subtree T given by Lemma 4.1 such that $|C \cap V[T]| = 0$ and $|M[T]| = 0$. Then in polynomial time, one can construct a dual vector y' that is good for G'.*

Proof. Let r be the root of T. Consider the dual vector y' obtained from y by increasing y_v by $\frac{1}{2}$ for every leaf v of T, and increasing y_r by $\frac{1}{2}$. More formally, y' is defined as

$$y'_v = \begin{cases} y_v + \frac{1}{2} & \text{if } v = r \text{ or } v \text{ is a leaf of } T \\ y_v & \text{otherwise} \end{cases}$$

Note that it could be the case that $r = r^*$ in which case y_r is undefined (as the root r^* does not have a parent edge), in which case we simply increase the y-value of each leaf in T by $\frac{1}{2}$. We claim that the vector y' obtained this way is good for G'.

Let π' be the node credits with respect to G' and y' and let σ' be the loads with respect to G' and y'. Let c be the node created by the contraction of T. Let $C' = C \cap V[T]$ and $U' = U \cap V[T]$. We have,

$$\begin{aligned} \pi'(c) &\geq \pi(C') + \frac{8}{5}|M[T]| + |U'| - (|M[T]| + |U'|) + \left(\frac{1}{2}|U'| + \frac{1}{2}\right) \\ &\geq \frac{1}{2}|U'| + \frac{1}{2} \\ &\geq 1 \end{aligned}$$

where the last inequality follows from the fact that $|U \cap V[T]| \geq 1$ for any subtree T where $|M[T]| = 0$. Note that when $r = r^*$, the credit drops by $\frac{1}{2}$ since $\pi'(c)$ loses the

$\frac{1}{2}$ term obtained from the increase of y_r, but we still have $\pi'(c) \geq \frac{1}{2} \geq 0$ as required in Definition 4.1 since c becomes the new root r^* in this case. This ends the proof of property (1).

To prove property (2), it is sufficient to consider links ℓ with at least one endpoint in T, since $\sigma'(\ell) = \sigma(\ell)$ and $\phi'(\ell) = \phi(\ell)$ holds for other links. Since $|M[T]| = 0$, it follows that every leaf of T is unmatched in M. Suppose there was a link $\ell = uv$ between two leaves u, v of T. Then, since all leaves of T are in U (as $|M[T]| = 0$), it follows that $u, v \in U$ and hence $\ell = uv$ would be candidate link for a greedy contraction, a contradiction to the assumption that there are no greedy contractions in G. Hence there is no link between any two leaves of T. Moreover, since T is semi-closed, it follows that there are no links between any leaf v of T and a vertex outside T. Thus it follows that every link ℓ with at least one endpoint in T must have at least one endpoint as a non-leaf vertex of T. In this case, we have $\phi'(\ell) \geq \phi(\ell) + 1$ since the endpoint of ℓ that is a non-leaf vertex of T becomes a compound node after the contraction. Due to the increase in the level of ℓ, if we can show that $\sigma(\ell)$ increases by at most $\frac{12}{15}$ we will be done, since $\frac{12}{15}$ is the minimum increase in the upper-bounds on the loads in Definition 4.1. In fact, we will show that $\sigma(\ell)$ increases by at most $\frac{1}{2}$. Note that P_ℓ either contains at most one leaf of T or contains r, but not both. Then, by definition of y', we have $\sum_{e \in P_\ell} y'_e \leq \sum_{e \in P_\ell} y_e + \frac{1}{2}$ which implies $\sigma'(\ell) \leq \sigma'(\ell) + \frac{1}{2}$. Hence property (2) is satisfied and we conclude that y' is good for G'. □

Thus the only remaining case is when T contains zero non-leaf compound nodes and exactly one matching edge. We break this down further into more subcases depending on the number of unmatched nodes in T. The following claim says that if T has at least 2 unmatched nodes, then we will be done.

Claim 4.5. *Let G be obtained from the initial tree G^* by a sequence of greedy or semi-closed contractions, and let y be dual variables that are good for G. Suppose G' was obtained from G by a semi-closed contraction of a subtree T given by Lemma 4.1 such that $|M[T]| = 1$ and $|U \cap V[T]| \geq 2$. Then in polynomial time, one can construct a dual vector y' that is good for G'.*

Proof. Let r be the root of T. Let $\ell = uv$ be the unique link of T that is in M. Consider the dual vector y' obtained from y by decreasing y_u and y_v by $\frac{2}{5}$, increasing y_w by $\frac{2}{5}$ for every unmatched leaf $w \neq u, v$ of T, and increasing y_r by $\frac{2}{5}$. More formally, y' is defined as

$$y'_w = \begin{cases} y_w - \frac{2}{5} & \text{if } w = u \text{ or } w = v \\ y_w + \frac{2}{5} & \text{if } w = r \text{ or } w \text{ is an unmatched leaf of } T \\ y_w & \text{otherwise} \end{cases}$$

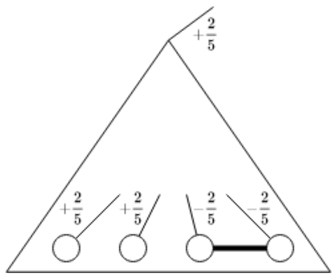


Figure 4.4: Figure depicting the change in the dual variables as in Claim 4.5. The bold edge corresponds to the unique link uv in T that is in M.

Note that if $r = r^*$ then y_r is undefined, so in this case we simply decrease y_u and y_v by $\frac{2}{5}$ and increase y_w by $\frac{2}{5}$ for every unmatched leaf w of T. We claim that y' is good for G'.

Let π' be the node credits with respect to G' and y' and let σ' be the loads with respect to G' and y'. Let c be the node created by the contraction of T. Let $C' = C \cap V[T]$ and $U' = U \cap V[T]$. We have,

$$\begin{aligned}
\pi'(c) &\geq \pi(C') + \frac{8}{5}|M[T]| + |U'| - (|M[T]| + |U'|) + \left(\frac{2}{5}|U'| + \frac{2}{5} - \frac{4}{5}\right) \\
&\geq \frac{3}{5}|M[T]| + \frac{2}{5}|U'| - \frac{2}{5} \\
&\geq 1
\end{aligned}$$

where the last inequality follows from the fact that $M[T] \geq 1$ and $|U \cap V[T]| \geq 2$. Note that if $r = r^*$ then the credit drops by $\frac{2}{5}$ since $\pi'(c)$ loses the $\frac{2}{5}$ term obtained from the increase of y_r, but we still have $\pi'(c) \geq \frac{3}{5} \geq 0$ as needed in Definition 4.1 since c becomes the new root r^* in this case. This finishes the proof of property (1).

To prove property (2), it is sufficient to consider links ℓ with at least one endpoint in T, since $\sigma'(\ell) = \sigma(\ell)$ and $\phi'(\ell) = \phi(\ell)$ holds for other links. As in the proof of Claim 4.4, there cannot be a link between two unmatched leaves of T as that would violate the assumption that G has no greedy contraction. Consider a link ℓ between two leaves of T. If ℓ is the unique matching edge uv of $M[T]$ then by construction of y', we have $\sigma'(\ell) = \sigma(\ell) - \frac{4}{5}$ and so property (2) holds for ℓ. Otherwise, it must be that case that ℓ is incident to one unmatched node $p \in V[T]$ and another matched node $w \in \{u, v\}$, then by construction of y', we have $y'_w = y_w - \frac{2}{5}$ and $y'_p = y_p + \frac{2}{5}$ and the y-values of all other edges in $P_\ell^{G^*}$ remain the same, which implies $\sigma'(\ell) = \sigma(\ell)$ and thus property (2) holds for ℓ.

Now consider a link ℓ incident to one leaf of T and another vertex outside T. Since T is semi-closed, it follows that ℓ must be incident to a matched leaf $w \in \{u, v\}$ of T. By construction of y', we have $y'_w = y_w - \frac{2}{5}$ and $y'_r = y_r + \frac{2}{5}$ and the y-values of all other edges in $P_\ell^{G^*}$ remain the same, which implies $\sigma'(\ell) = \sigma(\ell)$ and thus property (2) holds for ℓ.

The only links left to consider are links ℓ that are incident to a non-leaf vertex of T. In this case we have $\phi'(\ell) \geq \phi(\ell) + 1$ since the non-leaf vertex of T incident to ℓ becomes a compound node after contraction. Due to the increase in the level of ℓ, if we can show that $\sigma(\ell)$ increases by at most $\frac{12}{15}$ we will be done since $\frac{12}{15}$ is the minimum increase in the upper-bounds on the loads in Definition 4.1. In fact, we will show that $\sigma(\ell)$ increases by at most $\frac{2}{5}$. Note that P_ℓ either contains at most one leaf of T or contains r, but not both. Now, by definition of y', we have $\sum_{e \in P_\ell} y'_e \leq \sum_{e \in P_\ell} y_e + \frac{2}{5}$ which implies $\sigma'(\ell) \leq \sigma'(\ell) + \frac{2}{5}$. Hence property (2) holds for ℓ in every case, completing the proof of the claim. □

Finally, the only case remaining is when T has no compound nodes, exactly one matching edge and exactly one unmatched leaf. In this case, T must have exactly three leaves. Thus, ignoring the degree two vertices of T, there are finitely many configurations of T to consider. These configurations are enumerated in Fig. 4.5. For each of these configurations, the next claim shows how to update the duals.

Claim 4.6. *Let G be obtained from the initial tree G^* by a sequence of greedy or semi-closed contractions, and let y be dual variables that are good for G. Suppose G' was obtained from G by a semi-closed contraction of a subtree T given by Lemma 4.1 such that $|M[T]| = |U \cap V[T]| = 1$. Then in polynomial time, one can construct a dual vector y' that is good for G'.*

Proof. We first show how to construct y'. Since $|M[T]| = |U \cap V[T]| = 1$, it follows that T has exactly three leaves. Let u, v, w be the leaves of T so that $uv \in M$ is the unique matching link in T and w is the unique unmatched leaf. Note that there is no link between w and $V \setminus V[T]$ since T is semi-closed. Let r be the root of T. Contract every degree two vertex in T to get a tree T' with no vertices of degree two. Since T' has exactly three leaves, it follows that T' is one of two trees. Either T' is a star with three leaves or T' is a binary tree with two internal vertices and three leaves.

Case 1. T' is a star. First, suppose that neither uw nor vw are links in L. In this case, we obtain y' from y by increasing y_w by $\frac{2}{5}$ (see Fig. 4.5a). Now suppose that both uw and vw are links in L. In this case, since T is non-dangerous, there cannot be a link with one endpoint in $\{u, v\}$ and another endpoint outside T. We now obtain y' from y by increasing y_r by $\frac{2}{5}$ (see Fig. 4.5b). Finally, consider the case when $vw \in L$ and $uw \notin L$ (the case when $vw \notin L$ and $uw \in L$ is symmetric). Here, since T is non-dangerous, there cannot be a link between u and another vertex outside T. We then obtain y' from y by increasing y_u and y_w by $\frac{2}{5}$ and decreasing y_v by $\frac{2}{5}$ (see Fig. 4.5c).

Case 2. T' is a binary tree with two internal vertices. Let s be the internal vertex of T' that is not the root. Note that the two children of s in T' cannot be matched by M, as otherwise s would be the stem of a twin-link in M, a contradiction to the fact that M has no twin-links. Thus the unique unmatched vertex w must be a child of s in T'. Let the other child of s in T' be u so that the remaining leaf v is a child of r. First suppose that both uv is not in L. In this case, we obtain y' from y by increasing y_w and y_s by $\frac{2}{5}$ and decreasing y_u by $\frac{2}{5}$ (see Fig. 4.5d). Next, suppose that $vw \in L$. In this case, since T is non-dangerous, there cannot be a link between u and another vertex outside T. Here we obtain y' from y by increasing both y_s and y_r by $\frac{2}{5}$ and decreasing y_v by $\frac{2}{5}$ (see Fig. 4.5e).

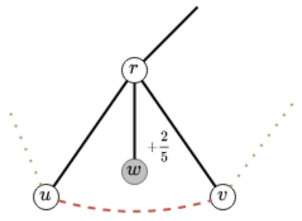
Now we claim that y' constructed as above is good for G'. Let π' be the node credit with respect to G' and y' and let σ' be the loads with respect to G' and y'. Let c be the node created by the contraction of T'. Note that in every case, the difference $1^\intercal y' - 1^\intercal y$ is exactly $\frac{2}{5}$. So we have,

$$\begin{aligned}\pi'(c) &\geq \pi(C \cap V[T]) + \frac{8}{3}|M[T]| + |U \cap V[T]| - (|M[T]| + |U \cap V[T]|) + \frac{2}{5} \\ &\geq \frac{3}{5}|M[T]| + \frac{2}{5} \\ &= 1\end{aligned}$$

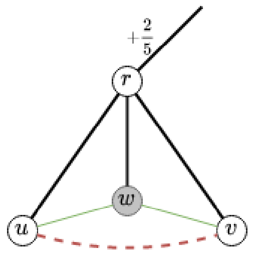
where the last equality uses the fact that $|M[T]| = 1$. Hence property (1) in Definition 4.1 holds.

To prove property (2), it is sufficient to consider links ℓ with at least one endpoint in T', since $\sigma'(\ell) = \sigma(\ell)$ and $\phi'(\ell) = \phi(\ell)$ holds for other links. Consider a link $\ell = pq \in L$. Note that in the construction of y', in every case we increase the value of y_e by $\frac{2}{5}$ for at most two edges e of T. This implies $\sigma'(\ell) \leq \sigma(\ell) + \frac{4}{5}$. If $\phi'(\ell) \geq \phi(\ell) + 1$ then we are done since the increase in the level of ℓ allows the increase in the load of ℓ by at least $\frac{4}{5}$. Otherwise $\phi'(\ell) = \phi(\ell)$, that is, the level of ℓ remains the same after the contraction. This implies that no endpoint p, q of ℓ is an internal vertex of T. If both p and q are leaves of T then one can verify (by looking at all possible leaf-to-leaf links in Fig. 4.5) that $\sigma'(\ell) \leq \sigma(\ell)$. Otherwise one endpoint p is a leaf and the other endpoint q lies outside T. Again one can verify (by looking at all possible leaf-to-outside links in Fig. 4.5) that $\sigma'(\ell) \leq \sigma(\ell)$. This completes the proof of property (2) and hence also the claim. □

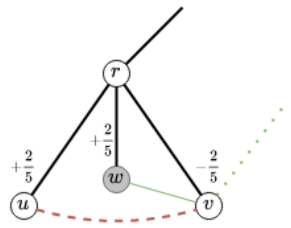
Now we summarize everything in the following lemma. The lemma formally gives the algorithm and combines the above claims to show that the algorithm gives us the desired result.



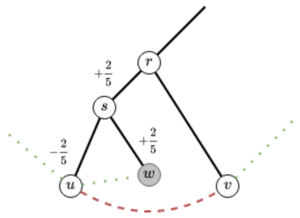
(a) Case when T' is a star and $vw, uw \notin L$.



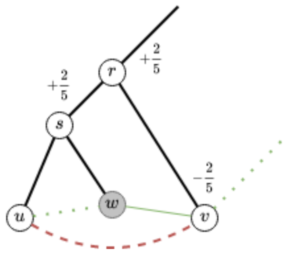
(b) Case when T' is a star and $vw, uw \in L$. There cannot be a link between u, v and another vertex outside T since T is non-dangerous.



(c) Case when T' is a star and $vw \in L$, $uw \notin L$. There cannot be a link between u and another vertex outside T since T is non-dangerous.



(d) Case when T' is a binary tree and $vw \notin L$.



(e) Case when T' is a binary tree and $vw \in L$. There cannot be a link between u and another vertex outside T since T is non-dangerous.

Figure 4.5: Figure showing the construction of the dual vector y' in Claim 4.6. The dashed line denotes the unique link $uv \in M$. The shaded vertex w is the unique unmatched vertex in the subtree. Bold lines denote the edges in T', which correspond to paths of arbitrary length in T. The dotted lines denote possible links that may or may not exist. Note that since T is semi-closed, there cannot be a leaf-to-outside link incident to w.

Lemma 4.2. *[26] Given a* Tree Augmentation *instance $(G^* = (V^*, E^*), L)$, there exists an algorithm that runs in polynomial time and computes a solution J and a dual vector $y \in \mathbb{R}_+^L$ such that*

- $|J| \leq 1^\intercal y$.
- $\sum_{e \in P_\ell^{G^*}} y_e \leq \frac{28}{15}$, *for all $\ell \in L$.*

Proof. The algorithm starts by finding a maximal matching M of the graph $(H, L[H] \setminus L^{twin})$, and a maximal matching J of the graph $(H, L[H] \setminus V[M])$. The initial dual vector y is set as in Eq. (4.1). The algorithm then proceeds by contracting all the twin-links in J. By Claim 4.1, y is good for the graph G' obtained by contracting the twin-links.

The algorithm then iteratively finds either a greedy contraction or a semi-closed contraction. If the algorithm finds a greedy contraction, then by Claim 4.2, the vector y is good for the graph obtained after the contraction. Otherwise, there does not exist any greedy contraction. Note that, since M has no twin-links of G^* and since every contraction is M-compatible, it follows that M has no twin-links in the current tree G also. Lemma 4.1 then gives us a semi-closed subtree T and a subset of links J' such that J' covers every edge in T and $|J'| = |M[T]| + |U \cap V[T]|$. If $|C \cap V[T]| \geq 1$ or $|M[T]| \geq 2$ then by Claim 4.3, the vector y is good for the graph obtained after the contraction. Otherwise $|C \cap V[T]| = 0$ and $|M[T]| \leq 1$. If $|M[T]| = 0$ then the vector y' given by Claim 4.4 is good for the graph obtained after the contraction. Otherwise, if $|M[T]| = 1$ and $|U \cap V[T]| \geq 2$ then the vector y' given by Claim 4.5 is good for the graph obtained after the contraction. The only remaining case is when $|M[T]| = 1$ and $|U \cap V[T]| = 1$. Now the vector y' computed by Claim 4.6 is good for the contracted graph.

Thus, at every contraction step, we can compute a vector y' that is good for the contracted graph. At the end of the algorithm, the current graph G consists of a single node r^* and all links are self-loops of r^*. Moreover, we have a dual vector y that is good for G and an integral solution J covering every edge of G. Since all links are self-loops, we have $\phi(\ell) = 2$ for all $\ell \in L$ which implies $\sigma(\ell) = \sum_{e \in P_\ell^{G^*}} y_e \leq \frac{28}{15}$. Moreover, we have $\pi(r^*) = \sum_{e \in E^*} y_e - |J| \geq 0$ which implies $1^\intercal y = \sum_{e \in E^*} y_e \geq |J|$. This completes the proof of the lemma. $\square$

Using the primal-dual scheme and the above lemma, we obtain the following theorem.

Theorem 4.1. *[26] The integrality gap of the cut-LP of a* Tree Augmentation *instance is at most $\frac{28}{15}$. Moreover, given a* Tree Augmentation *instance and a cut-LP solution $x \in \mathbb{R}^L$, there exists a polynomial time algorithm that returns an integral solution J for the instance such that $|J| \leq \frac{28}{15} 1^\intercal x$.*

Chapter 5

The Forest Augmentation Problem

In this chapter, we take a look at the related Forest Augmentation problem.

Forest Augmentation
Input: A forest $G = (V, E)$ and an additional set of edges, called links, $L \subseteq V \times V$.
Goal: Find $F \subseteq L$ of minimum cardinality such that the graph $(V, E \cup F)$ is 2-edge-connected.

Forest Augmentation is a generalization of the unweighted Tree Augmentation problem where instead of a tree, the base graph G is a forest. The natural LP relaxation for the problem is given below. We call this LP the cut-LP (for Forest Augmentation).

$$\begin{array}{rrcll} \text{minimize} & \sum_{\ell \in L} x_\ell & & & \\ \text{subject to} & x(\delta_L(S)) & \geq & 2 - |\delta_E(S)| & \forall S \subset V, S \neq \emptyset \\ & x_\ell & \leq & 1 & \forall \ell \in L \\ & x_\ell & \geq & 0 & \forall \ell \in L \end{array}$$

Observe that if $|\delta_E(S)| \geq 2$ for some subset S, then the constraint corresponding to S is trivially satisfied. Thus, it only suffices to consider the constraints corresponding to sets S such that $|\delta_E(S)| \leq 1$. Let $\mathcal{C}_1$ denote the family of cuts $S \subset V, S \neq \emptyset$, such that $|\delta_E(S)| = 1$ and let $\mathcal{C}_2$ denote the family of cuts $S \subset V, S \neq \emptyset$ such that $|\delta_E(S)| = 0$. Let $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$. We refer to the cuts in $\mathcal{C}_1$ as the 1-cuts of G, and the cuts in $\mathcal{C}_2$ as the 2-cuts of G, and the cuts that are not in either $\mathcal{C}_1$ or $\mathcal{C}_2$ are referred to as the 0-cuts of G. Now, we can rewrite the above LP as follows.

$$\begin{array}{llcll} \text{minimize} & \displaystyle\sum_{\ell\in L} x_\ell & & & \\ \text{subject to} & x(\delta_L(S)) & \geq & 1 & \forall S \in \mathcal{C}_1 \\ & x(\delta_L(S)) & \geq & 2 & \forall S \in \mathcal{C}_2 \\ & x_\ell & \leq & 1 & \forall \ell \in L \\ & x_\ell & \geq & 0 & \forall \ell \in L \end{array}$$

While there are several approximation algorithms beating factor two for special cases of Forest Augmentation, such as when G is a tree [1,6,7,30], a matching [9,10], or an empty graph [22,29], the best known approximation algorithm for Forest Augmentation achieves an approximation ratio of two. This approximation factor can be achieved using a variety of techniques such as iterative rounding or the primal-dual method [20,23,25], but there has been no improvement in the approximation ratio of this problem since 1992. In this chapter, we study the Forest Augmentation problem in an attempt to beat approximation factor two. We give a partial result that says that we can beat factor two if the LP solution is half-integral. Additionally, we also study the structure of extreme points of LP solutions and show several properties that these extreme points satisfy. Our hope is that our work will help in designing better approximation algorithms for Forest Augmentation in the future.

The rest of this chapter is divided as follows. In Section 5.1, we give an algorithm that uses the results in Chapter 4 to obtain an approximation ratio $\frac{29}{15}$ for instances in which the cut-LP solution is half-integral. In Section 5.2, we study the structure of extreme points of the cut-LP relaxation and prove various properties that they satisfy.

5.1 Beating factor two in the half-integral case

We now present a $\frac{29}{15}$-factor approximation algorithm for the case when the solution x to the cut-LP is half-integral, that is, $x_\ell \in \{0, \frac{1}{2}, 1\}$ for every link $\ell \in L$. The algorithm is based on the following idea. Since we have an algorithm that gives an approximation factor of $\frac{28}{15}$ for a Tree Augmentation instance (Theorem 4.1) with respect to a cut-LP solution y, a natural idea would be to convert our Forest Augmentation instance $(G = (V, E), L)$ into a Tree Augmentation instance $(G' = (V, E'), L')$ and convert the cut-LP solution x to a solution y for the cut-LP of Tree Augmentation, and then use Theorem 4.1. A natural way to convert our Forest Augmentation instance to a Tree Augmentation is by adding, to the solution, a spanning tree of G that contains all the edges in E. The following lemma does exactly this.

Lemma 5.1. *Suppose there exists a set of links $S \subseteq L$ such that $E \cup S$ is a spanning tree of G. Let x be an optimum solution to the cut-LP for the* Forest Augmentation *instance over G. Then, in polynomial time, one can compute an integral solution F of cost at most $|S| + \frac{28}{15}x(L) - \frac{28}{15}x(S)$.*

Proof. Let (P) denote the cut-LP for the Tree Augmentation instance over the tree $(V, E \cup S)$, with links $L \setminus S$. Consider the vector $y \in \mathbb{R}^{L \setminus S}$ defined such that $y_e = x_e$ for all $e \in L \setminus S$. Observe that y is feasible for (P). Now using Theorem 4.1, in polynomial time, we can compute an integral solution $F \subseteq L \setminus S$ such that $(V, E \cup S \cup F)$ is 2-edge-connected and $|F| \leq \frac{28}{15}y(L \setminus S)$. We return $S \cup F$ as our solution. Now we have $|S \cup F| = |S| + |F| \leq |S| + \frac{28}{15}y(L \setminus S) = |S| + \frac{28}{15}(x(L) - x(S))$. This completes the proof of the lemma. $\square$

Now we show an approximation algorithm for the half-integral case. First, we remove all links in L that have $x_e = 0$. Note that the instance still remains feasible after removing these edges. Let T be a spanning tree of G such that $E \subseteq E[T]$. Since the graph is connected, such a tree always exists, and can be found in polynomial time. Let $T' = E[T] \setminus E$. Since the number of components of G is p, it follows that $|T'| = |E[T] \setminus E| = p-1$. Moreover, since x is half-integral, and since we have removed the edges with an x-value of 0, we have $x(T') \geq \frac{1}{2}|T'| = \frac{p-1}{2}$. Since $E \cup T' = T$ is a spanning tree, using the above lemma, we can compute an integral solution S^* of cost at most

$$|S^*| \leq |T'| + \frac{28}{15}x(L) - \frac{28}{15}x(T') \leq (p-1) + \frac{28}{15}x(L) - \frac{28}{15}\frac{(p-1)}{2} = \frac{1}{15}(p-1) + \frac{28}{15}x(L)$$

Using the bound $p - 1 \leq x(L)$, we get $|S^*| \leq \frac{29}{15}x(L)$. Thus we have proved the following theorem.

Theorem 5.1. *Let $(G = (V, E), L)$ be a* Forest Augmentation *instance, and let $x \in \mathbb{R}^L$ be a solution to the cut-LP of this instance such that $x_\ell \in \{0, \frac{1}{2}, 1\}$ for every $\ell \in L$. Then there exists a polynomial time algorithm that computes an integral solution S^* for the instance, such that $|S^*| \leq \frac{29}{15}x(L)$.*

Note that the above theorem does not generalize to the weighted setting since we crucially use Theorem 4.1 which assumes unit costs.

5.2 Structure of extreme point solutions

Recall that two sets $A, B \subseteq V$ are said to *cross* if each of the sets $A \setminus B, B \setminus A, A \cap B$ are non-empty, and a family $\mathcal{L}$ of subsets of V is said to be *laminar* if no two sets in $\mathcal{L}$ cross. A set S is said to be *tight* with respect to a vector x^* if $x^*(\delta_L(S)) + |\delta_E(S)| = 2$. Fix a basic feasible solution x^* to the cut-LP of our instance. Since the goal is to design an approximation algorithm, we may assume that $x^*_\ell > 0$ for every $\ell \in L$, as we may remove all links $\ell \in L$ such that $x^*_\ell = 0$. Similarly, we may also assume that $x^*_\ell < 1$ for every $\ell \in L$, as we may include any link $\ell \in L$ such that $x^*_\ell = 1$ by adding it to E and contracting any cycles formed, and then recomputing the basic feasible solution x^* for the resulting graph. Thus, throughout the remainder of this section, we assume that $0 < x^*_\ell < 1$ for every $\ell \in L$. It is well known that one can find a linearly independent family of tight sets using standard uncrossing arguments (see for example [32]).

Lemma 5.2 (Theorem 11.21 in [32]). *For any basic feasible solution x^* to the cut-LP of a* Forest Augmentation *instance $(G = (V, E), L)$, there is a collection $\mathcal{L}$ of subsets of vertices with the following properties:*

1. *Every set $S \in \mathcal{L}$ is tight with respect to x^*.*
2. *The set of vectors $\{\chi^{\delta_L(S)} : S \in \mathcal{L}\}$ are linearly independent.*
3. $|\mathcal{L}| = |L|$
4. *The collection $\mathcal{L}$ is laminar.*

Fix a family $\mathcal{L}$ of tight cuts satisfying properties (1)-(4) as in Lemma 5.2. In this section, we will show that this laminar family satisfies various nice properties. We will show that $\mathcal{L}$ contains a leaf S such that $|\delta_L(S)| \leq 3$. Furthermore, we will show that we can find, in polynomial time, another laminar family $\mathcal{L}'$ such that every set $S \in \mathcal{L}'$ is 2-edge-connected. Our hope is that these results on the structure of the laminar family arising from extreme points of the cut-LP will help in designing better approximation algorithms for Forest Augmentation.

Recall that, for any sets $S, S' \in \mathcal{L}$, we say that S' is a *child* of S if $S' \subset S$ and there is no other set $R \in \mathcal{L}$ such that $S' \subset R \subset S$. If S' is a child of S, we refer to S as the *parent* of S'. A set $S \in \mathcal{L}$ is said to be a *leaf* if S has no children. Equivalently, $S \in \mathcal{L}$ is a leaf if it is an inclusion-wise minimal set of $\mathcal{L}$. It can be seen that under this parent-child relationship, the family $\mathcal{L}$ forms a rooted forest $\mathcal{F}$. Every vertex of $\mathcal{F}$ corresponds to a set in $\mathcal{L}$ and the parent and children of this vertex in $\mathcal{F}$ correspond to the parent and children of the corresponding set in $\mathcal{L}$. We further augment this forest by adding links. For every link $uv \in L$, we add a link in $\mathcal{F}$ with endpoints A and B, where A and B are the inclusionwise smallest sets in $\mathcal{L}$ containing vertices u and v respectively.

Claim 5.1. *Let x^* be a basic feasible solution to the cut-LP and suppose that $0 < x^*_\ell < 1$ for every $\ell \in L$. Let $\mathcal{L}$ be the laminar family given by Lemma 5.2 and let $S \in \mathcal{L}$ be such that S has a unique child R. Then S is incident to at least two links in $\mathcal{F}$.*

Proof. Note that we have $\delta_L(S) \neq \delta_L(R)$ as otherwise we would have $\delta_L(S) = \delta_L(R)$ and thus $\chi^{\delta_L(S)} = \chi^{\delta_L(R)}$, contradicting the linear independence of $\mathcal{L}$. So there exists at least one link in $\delta_L(S)$ that is not in $\delta_L(R)$ or vice versa. First, suppose that there exists a link ℓ such that $\ell \in \delta_L(S)$ and $\ell \notin \delta_L(R)$, then S is the inclusionwise smallest set containing an endpoint of ℓ. If $x^*(\delta_L(S)) \leq x^*(\delta_L(R))$, then there must exist a link $\ell' \in \delta(R)$ and $\ell' \notin \delta(S)$ as otherwise we would have $x^*(\delta_L(R)) \leq x^*(\delta_L(S)) - x^*_\ell < x^*(\delta_L(S)) \leq x^*(\delta_L(R))$, a contradiction. Now S is the inclusionwise smallest set in $\mathcal{L}$ containing an endpoint of ℓ' also. Thus ℓ and ℓ' are incident to S in $\mathcal{F}$. Otherwise $x^*(\delta_L(S)) > x^*(\delta_L(R))$. Since both S and R are tight, this is only possible when S is a 2-cut and R is a 1-cut. So we have $x^*(\delta_L(S)) = 2$ and $x^*(\delta_L(R)) = 1$. Since $x^*_\ell < 1$ for all $\ell \in L$, there must exist at least two links ℓ', ℓ'' that are in $\delta(S)$ and not in $\delta(R)$. Then S is the inclusionwise smallest set containing endpoints of ℓ' and ℓ'' implying that ℓ' and ℓ'' are incident to S in $\mathcal{F}$.

Now suppose that there exists a link $\ell \in \delta(R)$ such that $\ell \notin \delta(S)$. The proof for this case is largely symmetric to the proof of the previous case. Suppose $x^*(\delta(R)) \leq x^*(\delta(S))$, then there must also exist a link $\ell' \in \delta(S)$ such that $\ell' \notin \delta(R)$. So ℓ, ℓ' are links incident to S in $\mathcal{F}$. Otherwise $x^*(\delta(R)) > x^*(\delta(S))$ implying that $x^*(\delta(R)) = 2$ and $x^*(\delta(S)) = 1$. Since $x^*_\ell < 1$ for every $\ell \in L$, there must exist at least two links ℓ', ℓ'' such that $\ell', \ell'' \in \delta(R)$ and $\ell', \ell'' \notin \delta(S)$. Thus ℓ', ℓ'' are incident to S in $\mathcal{F}$.

In every case we have found two links that are incident to S in $\mathcal{F}$, completing the proof of the claim. $\square$

Now we use a simple counting argument to show that there exists a leaf $S \in \mathcal{L}$ such that $|\delta_L(S)| \leq 3$.

Lemma 5.3. *Let x^* be a basic feasible solution to the cut-LP and suppose that $0 < x^*_\ell < 1$ for every $\ell \in L$. Let $\mathcal{L}$ be the laminar family given by Lemma 5.2. Then there exists a leaf $S \in \mathcal{L}$ such that $|\delta_L(S)| \leq 3$.*

Proof. Suppose for contradiction that $|\delta_L(S)| \geq 4$ for every leaf $S \in \mathcal{L}$. Then every leaf S of $\mathcal{L}$ is incident to at least four links in $\mathcal{F}$ since every link in $|\delta_L(S)|$ has S as the smallest set in $\mathcal{L}$ containing an endpoint of it. Let q denote the number of leaves in $\mathcal{L}$, let r denote the number of sets in $\mathcal{L}$ with exactly one child, and let d denote the number of sets in $\mathcal{L}$ with at least two children. So we have $q + r + d = |\mathcal{L}|$. Since the average degree of a forest is strictly less than 2, it follows that $q > d$. Thus we have $2q + r > |\mathcal{L}|$. By Claim 5.1, every set with a single child is incident to at least two links. Thus the total number of links is $|L| \geq \frac{1}{2}(2r + 4q) = r + 2q > |\mathcal{L}|$, contradicting the fact that $|\mathcal{L}| = |L|$ by Lemma 5.2. $\square$

Now we will show that we can find, in polynomial time, a laminar family $\mathcal{L}$ such that every set $S \in \mathcal{L}$ is 2-edge-connected. Note that this laminar family $\mathcal{L}$ might not be the same as the family given by Lemma 5.2. For a set of edges (or links) F, and for subsets $A, B \subseteq V$, by $\delta_F(A, B)$, we denote the set of edges in F with one endpoint in A and the other in B. Let $H = (V, E \cup L)$ be the graph obtained from G by including all the links. We use $\delta_H(A, B)$ as a shorthand for $\delta_{E[H]}(A, B) = \delta_{E \cup L}(A, B)$. Let x^* be an optimal solution for the cut-LP. Consider the dual of the cut-LP,

$$\begin{array}{lrcll}
\text{maximize} & \displaystyle\sum_{C \in \mathcal{C}_1} y_C + \sum_{C \in \mathcal{C}_2} 2y_C + \sum_{\ell \in L} z_\ell & & & \\
\text{subject to} & \displaystyle\sum_{C \in \mathcal{C}: e \in \delta_L(C)} y_C + z_\ell & \leq & 1 & \forall e \in L \\
& y_C & \geq & 0 & \forall C \in \mathcal{C} \\
& z_\ell & \leq & 0 & \forall \ell \in L
\end{array}$$

Here the y variables correspond to the cut constraints and the z variables correspond to the upper-bound constraints $x_\ell \leq 1$. Note that, since we assume that $x^*_\ell < 1$ for all $\ell \in L$, by complementary slackness it follows that $z^*_\ell = 0$ for all $\ell \in L$, for any optimal dual solution (y^*, z^*). Recall that a set $S \in \mathcal{C}$ is said to be tight if $|\delta_E(S)| + x^*(\delta_L(S)) = 2$. For a cut $C \subset V, C \neq \emptyset$, let $f(C)$ denote the connectivity demand on C. That is, $f(C) = 1$ if $C \in \mathcal{C}_1$, $f(C) = 2$ if $C \in \mathcal{C}_2$ and $f(C) = 0$ otherwise. Note that the objective function of the dual is $\sum_{C \in \mathcal{C}_1} y_C + \sum_{C \in \mathcal{C}_2} 2y_C = \sum_{C \subset V, C \neq \emptyset} f(C) y_C$.

It is well-known that one can find an optimal dual solution (y^*, z^*) in polynomial time such that the support of y is laminar. We observe the following fact about the dual. Suppose there exists a set $S \in \mathit{supp}(y^*)$, and $S' \in \mathcal{C}$ such that $\delta_L(S') \subseteq \delta_L(S)$. Then we can reduce y^*_S by $\epsilon > 0$ and increase $y^*_{S'}$ by ϵ, and the solution will still be feasible. This is because $\delta_L(S') \subseteq \delta_L(S)$ and so the set of constraints that the variable $y^*_{S'}$ participates in is a subset of the set of constraints the variable y^*_S participates in. Thus, we may obtain a certain type of *minimal* laminar family by iteratively finding a set $S' \subset S$ for some $S \in supp(y^*)$ such that $\delta_L(S') \subseteq \delta_L(S)$ and then applying the above procedure to remove S from the support of y^*. It turns out that such a laminar family is exactly the type of family for which $H[S]$ is 2-edge-connected for every $S \in \mathcal{L}$.

We also want to be able to find such a minimal family in polynomial time. The natural way of obtaining such a laminar family, as alluded to earlier, is by starting with the laminar family $supp(y^*)$ for some optimal solution (y^*, z^*), and iteratively using the above procedure so long as there exists $S' \subset S$ for some $S \in supp(y^*)$ such that $\delta_L(S') \subseteq \delta_L(S)$. However, we cannot enumerate over all subsets $S' \subset S$ in polynomial time. As it turns out, it suffices to enumerate over subsets $S' \subset S$ such that $\delta_E(S')$ contains a unique edge $e \in E$ that is a bridge of the subgraph $H[S]$ (recall that an edge e is said to be a *bridge* if the removal of e increases the number of connected components of the graph). Following these ideas, we formalize the type of family that we require, which we refer to as a *proper laminar family.*

Definition 5.1. *[Proper family] A laminar family of sets $\mathcal{L}$ is said to be proper if for each $S \in \mathcal{L}$, there does not exist $S' \in \mathcal{C}$ such that*

1. $S' \subset S$
2. $\delta_L(S') \subseteq \delta_L(S)$
3. *For any child C of S, either $C \subseteq S'$ or $C \subseteq S \setminus S'$*
4. *There exists an edge $e \in E$ in $\delta_E(S')$ that is a bridge in the subgraph $H[S]$.*

Intuitively, $\mathcal{L}$ is proper if we cannot find a pair S, S' such that $S \in \mathcal{L}$ and $\delta_L(S') \subseteq \delta_L(S)$, but there are also extra conditions about the children of S and edges in $\delta(S')$ that are required for the proof to go through. The added condition that there must exist an edge $e \in E$ in $\delta_E(S')$ that is a bridge in the subgraph $H[S]$ is crucially needed to be able to compute a proper family in polynomial time. If such a pair S, S' does exist, then $\mathcal{L}$ is not proper, and we refer to S, S' as a witnessing pair (that witnesses the non-properness of $\mathcal{L}$).

We will first prove that in any proper laminar family of sets $\mathcal{L}$, the graph $H[S]$ is 2-edge-connected for every $S \in \mathcal{L}$. To do this, we first show that every tight set induces a connected subgraph of H.

Lemma 5.4. *Let $S \in \mathcal{C}$ be a tight set. Then $H[S]$ is connected.*

Proof. Suppose for contradiction that $H[S]$ is not connected. Then there exists a non-empty subset $C \subset S$ such that $\delta_H(C, S \setminus C) = \emptyset$. Then we must have $\delta_H(C) \subseteq \delta_H(S)$ and $\delta_H(S \setminus C) \subseteq \delta_H(S)$. Due to feasibility of x^*, we have $|\delta_E(C)| + x^*(\delta_L(C)) \geq 2$ and $|\delta_E(S \setminus C)| + x^*(\delta_L(S \setminus C)) \geq 2$. This implies $|\delta_E(S)| + x^*(\delta_L(S)) \geq 4$, a contradiction to the tightness of S. □

Now we are ready to show that, if $\mathcal{L}$ is proper, then every set in $\mathcal{L}$ induces a 2-edge-connected subgraph of H.

Lemma 5.5. *Suppose $x^*_e < 1$ for each $e \in L$. Let $\mathcal{L}$ be a laminar collection of tight sets that is proper. Then for any $S \in \mathcal{L}$, the subgraph $H[S]$ is 2-edge-connected.*

Proof. Fix a set $S \in \mathcal{L}$. By Lemma 5.4, $H[S]$ is connected. To prove 2-edge-connectivity, it suffices to show that there is no bridge in $H[S]$. We prove this by induction on the depth of S in the rooted forest $\mathcal{F}$ corresponding to $\mathcal{L}$.

In the base case, S is a leaf. Suppose for contradiction that there exists $e \in E \cup L$ such that e is a bridge of $H[S]$. Let $S' \subset S$ be such that $\delta_E(S', S \setminus S') = \{e\}$.

Case 1. $e \in L$. In this case, we will show that $x^*_e = 1$, contradicting the assumption on x^*. Suppose for contradiction that $x^*_e < 1$. Since $|\delta_E(S')| + x^*(\delta_L(S')) \geq 2$ by feasibility of x^*, and since $x^*(\delta_L(S', S \setminus S')) = x^*_e < 1$, we must have $|\delta_E(S', V \setminus S)| + x^*(\delta_L(S', V \setminus S)) > 1$. Symmetrically, we also have $|\delta_E(S \setminus S', V \setminus S)| + x^*(\delta_L(S \setminus S', V \setminus S)) > 1$. But then we have $|\delta_E(S)| + x^*(\delta_L(S)) > 2$, a contradiction to the tightness of S.

Case 2. $e \in E$. In this case, we will show that either $S' \in \mathcal{C}$ and $\delta_L(S') \subseteq \delta_L(S)$, or $S \setminus S' \in \mathcal{C}$ and $\delta_L(S \setminus S') \subseteq \delta_L(S)$, contradicting the fact that $\mathcal{L}$ is proper. Note that one of S' or $S \setminus S'$ must be a 1-cut or a 2-cut. Otherwise, both are 0-cuts which implies there are two edges in both $\delta_E(S')$ and $\delta_E(S \setminus S')$. But since $|\delta_E(S', S \setminus S')| = 1$, two of these edges must cross $\delta_E(S)$, a contradiction to the fact that $S \in \mathcal{C}$ and hence S is either a 1-cut or a 2-cut. Thus one of S' or $S \setminus S'$ is in $\mathcal{C}$. Assume that $S' \in \mathcal{C}$. The case when $S \setminus S' \in \mathcal{C}$ is symmetric. We will show that $\delta_L(S') \subseteq \delta_L(S)$. To see this, note that any link in $\delta_L(S')$ cannot cross the cut $(S', S \setminus S')$ since $\delta_H(S', S \setminus S') = \{e\}$, thus every link in $\delta_L(S')$ must cross S. Hence $\delta_L(S') \subseteq \delta_L(S)$, which contradicts the fact that $\mathcal{L}$ is proper.

Now we prove the induction step. By induction hypothesis, we can assume that for any child C of S, the graph $H[C]$ is 2-edge-connected. Suppose for contradiction that there exists $e \in E \cup L$ such that e is a bridge of $H[S]$. Let $S' \subset S$ be such that $\delta_E(S', S \setminus S') = \{e\}$. If $e \in L$, then the same argument as in the base case implies that $x^*_e = 1$, a contradiction. Now suppose $e \in E$. As in the base case, we will contradict the assumption that $\mathcal{L}$ is proper. By the same argument as in the base case, we may assume that one of either S' or

$S \setminus S'$ is in $\mathcal{C}$. Assume that $S' \in \mathcal{C}$. The case when $S \setminus S' \in \mathcal{C}$ is symmetric. We will show that $\delta_L(S') \subseteq \delta_L(S)$ and, for each child C of S, either $C \subseteq S'$ or $C \subseteq S \setminus S'$, contradicting the fact that $\mathcal{L}$ is proper. To see that $\delta_L(S') \subseteq \delta_L(S)$, note that any link in $\delta_L(S')$ cannot cross the cut $(S', S \setminus S')$ since $\delta_H(S', S \setminus S') = \{e\}$, thus every link in $\delta_L(S')$ must cross S. Hence $\delta_L(S') \subseteq \delta_L(S)$.

Finally, we need to show that, for any child C of S, either $C \subseteq S'$ or $C \subseteq S \setminus S'$. Suppose for contradiction, that there exists some child C of S such that $C \cap S' \neq \emptyset$ and $C \cap (V \setminus S') \neq \emptyset$. By induction hypothesis, $H[C]$ is 2-edge-connected, thus there are at least two edges of H crossing the cut $(C \cap S', C \cap (V \setminus S'))$ of $H[C]$. Both these edges also cross S' in $H[C]$, a contradiction to the fact that $|\delta_H(S', S \setminus S')| = 1$. □

Now we prove that we can find such a proper laminar family in polynomial time. Here, we crucially use the property in Definition 5.1 that says that there exists an edge $e \in \delta_E(S')$ that is a bridge of $H[S]$.

Lemma 5.6. *In polynomial time, we can compute an optimal dual solution y^* such that supp(y^*) is a proper laminar collection of tight sets.*

Proof. Let (y^*, z^*) be an optimal dual solution such that *supp*(y^*) is laminar. This can be computed in polynomial time. We will show that, as long as *supp*(y^*) is not proper, we can keep applying a procedure that modifies y^* until *supp*(y^*) becomes proper at which point we return *supp*(y^*).

First, we show how to check whether *supp*(y^*) is proper or not in polynomial time, and if *supp*(y^*) is not proper, how to find a witnessing pair S, S'. For every set $S \in \mathcal{L}$ and every bridge e of $H[S]$ such that $e \in E$, let $S' \subset S$ be a set such that $\delta(S') = \{e\}$ in $H[S]$. We check whether $\delta_L(S') \subseteq \delta_L(S)$, and if, for each child C of S, either $C \subseteq S'$ or $C \subseteq S \setminus S'$. If we find such an S', then we have found a witnessing pair of cuts S, S' that show that *supp*(y^*) is not proper, otherwise the laminar family *supp*(y^*) must be proper and we can halt and return y^*. This can be done in polynomial time since the number of sets in $\mathcal{L}$ is polynomial in $|V|$, and since there can be at most $|E|$ bridges in $H[S]$.

Now we describe the procedure that modifies y^* as long as *supp*(y^*) is not proper. Given a witnessing pair of cuts S, S', consider the dual solution y' obtained from y^* as follows,

$$y'_C = \begin{cases} 0 & \text{if } C = S \\ y^*_C + y^*_S & \text{if } C = S' \text{ or } C = S \setminus S' \\ y^*_C & \text{otherwise} \end{cases}$$

That is, we transfer the y-value on S to the y-values of both S' and $S \setminus S'$. We will first prove that (y', z^*) is feasible for the dual. Consider any constraint $\sum_{\ell \in \delta(C)} y_C + z_\ell \leq 1$ for some $\ell \in L$. Since $\delta_L(S')$ is disjoint from $\delta_L(S \setminus S')$, and since $\delta_L(S'), \delta_L(S \setminus S') \subseteq \delta_L(S)$, we have $\sum_{\ell \in \delta(C)} y'_C + z^*_\ell \leq \sum_{\ell \in \delta(C)} y^*_C + z^*_\ell \leq 1$, and thus (y', z^*) is feasible.

Now we will prove that (y', z^*) is in fact an optimal solution to the dual. Let $\epsilon = y^*_S$. Note that the cost of y' is $\sum_{C \in \mathcal{C}} f(C) y^*_C + \epsilon(f(S') + f(S \setminus S') - f(S))$. Observe that $f(S') + f(S \setminus S') \geq f(S)$ for any two sets $S' \subset S$ such that $S', S \in \mathcal{C}$. Thus we conclude that the cost of y' is at least the cost of y^*. If $f(S') + f(S \setminus S') > f(S)$, then the cost of y' is strictly greater than the cost of y^*, a contradiction to the optimality of y^*. Thus we must have $f(S') + f(S \setminus S') = f(S)$, which implies that (y', z^*) is an optimal solution.

Moreover, note that *supp*(y') is laminar since for each child C of S, either $C \subseteq S'$ or $C \subseteq S \setminus S'$, which implies that *supp*(y') is a laminar family where the parent of S' and $S \setminus S'$ is now the parent of S in *supp*(y^*). Thus we have obtained another optimal dual solution y' such that *supp*(y') is laminar, but $S \notin$ *supp*(y'). Furthermore, note that every set in *supp*(y') is tight due to complementary slackness. We iterate this procedure on y' by finding sets that violate the proper-ness of *supp*(y') if any exist, and removing them in the same way. To see that this procedure terminates in polynomial time, consider the potential function $\sum_{C \in supp(y^*)} |C|^2$. At every step, this potential drops by $|S|^2 - (|S'|^2 + |S \setminus S'|^2)$ since we remove the set S from the support of y^* and potentially add the sets S' and $S \setminus S'$ into the support of y^*. This potential drop at least 1 since $|S'| + |S \setminus S'| = |S|$ and $1 \leq |S'| < |S|$, and hence the procedure terminates in polynomial time. Once the procedure terminates, we are left with an optimal solution (y^*, z^*) such that *supp*(y^*) is a proper laminar family of tight cuts, as desired. $\square$

Chapter 6

Conclusion

In this thesis, we surveyed three decomposition based approximation algorithms for Connectivity Augmentation problems. We first covered the original decomposition based method by Adjiashvili for Weighted Tree Augmentation, which was the first algorithm to beat factor two for the problem. We then surveyed two additional decomposition based algorithms that built upon the observations and techniques of Adjiashvili's algorithm. We covered an improvement to Adjiashvili's algorithm that was given by Fiorini, Groß, Könemann and Sanità, and an algorithm for Cactus Augmentation by Cecchetto, Traub and Zenklusen, that draws from Adjiashvili's work.

Additionally, we covered a result of Nutov that bounds the integrality gap of the cut-LP relaxation for unweighted Tree Augmentation. We also studied the related Forest Augmentation problem, for which no approximation ratio better than 2 is known. We showed some partial progress for this problem: We showed that we can achieve approximation ratio $\frac{29}{30}$ in the case when the LP solution is half-integral, and we showed various properties that are satisfied by extreme points of the natural LP relaxation for the problem.

We hope that our study of Forest Augmentation leads to further progress on the approximation ratio for the problem. One possible idea would be to try some sort of induction on the sets of the proper laminar family given by Lemma 5.6. Indeed, by Lemma 5.5, every set $S \in \mathcal{L}$ induces its own feasible subinstance since $H[S]$ is 2-edge-connected. Perhaps solutions $F_1, \ldots, F_k$ for the children $C_1, \ldots, C_k$ of a set $S \in \mathcal{L}$ could be combined into a solution F of S with a small increase in cost. Doing this recursively would lead to a solution for the entire graph. However, it is unclear how one might bound the cost of such a solution.

Another possible future research direction would be to try to apply Adjiashvili's decomposition based approach to Forest Augmentation. While Adjiashvili's approach heavily relies on a tree-like structure, which is inherent in Tree Augmentation, the result of Cecchetto, Traub and Zenklusen for Cactus Augmentation show that it is possible to use this approach even when there is no explicit tree-like structure in the problem. One possible idea could be to use the laminar family given by the extreme points of the cut-LP to guide in the decomposition process, as laminar families have inherent tree-like structure.

References

[1] David Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. *ACM Transactions on Algorithms (TALG)*, 15(2):1–26, 2018.

[2] Gautam Appa and Balázs Kotnyek. Rational and integral k-regular matrices. *Discrete Mathematics*, 275(1-3):1–15, 2004.

[3] Gautam Appa and Balázs Kotnyek. A bidirected generalization of network matrices. *Networks: An International Journal*, 47(4):185–198, 2006.

[4] Gautam Appa, Balázs Kotnyek, Konstantinos Papalamprou, and Leonidas Pitsoulis. Optimization with binet matrices. *Operations research letters*, 35(3):345–352, 2007.

[5] Manu Basavaraju, F. Fomin, P. Golovach, Pranabendu Misra, M. Ramanujan, and S. Saurabh. Parameterized algorithms to preserve connectivity. In *ICALP*, 2014.

[6] Federica Cecchetto, Vera Traub, and Rico Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 370–383, 2021.

[7] J. Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. *Algorithmica*, 80:608–651, 2017.

[8] J. Cheriyan, H. Karloff, R. Khandekar, and Jochen Könemann. On the integrality ratio for tree augmentation. *Oper. Res. Lett.*, 36:399–401, 2008.

[9] Joe Cheriyan, Jack Dippel, Fabrizio Grandoni, Arindam Khan, and Vishnu V Narayan. The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm. *Mathematical Programming*, 182(1):315–354, 2020.

[10] Joseph Cheriyan, Robert Cummings, Jack Dippel, and J Zhu. An improved approximation algorithm for the matching augmentation problem. *arXiv preprint arXiv:2007.11559*, 2020.

[11] Joseph Cheriyan, Tibor Jordán, and Ramamoorthi Ravi. On 2-coverings and 2-packings of laminar families. In *European Symposium on Algorithms*, pages 510–520. Springer, 1999.

[12] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. Combinatorial optimization, 1997.

[13] Efim A Dinitz, Alexander V Karzanov, and Michael V Lomonosov. On the structure of the system of minimum edge cuts in a graph. *Issledovaniya po Diskretnoi Optimizatsii*, pages 290–306, 1976.

[14] Jack Edmonds and Ellis L Johnson. Matching, Euler tours and the Chinese postman. *Mathematical programming*, 5(1):88–124, 1973.

[15] Guy Even, Jon Feldman, Guy Kortsarz, and Zeev Nutov. A 1.8 approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Transactions on Algorithms (TALG)*, 5(2):1–17, 2009.

[16] Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. Approximating weighted tree augmentation via chvátal-gomory cuts. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 817–831. SIAM, 2018.

[17] András Frank. *Connections in Combinatorial Optimization.* Oxford Lecture Series in Mathematics and its Applications, Volume 38. Oxford University Press, 2011.

[18] Greg N Frederickson and Joseph Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM Journal on Computing*, 10(2):270–283, 1981.

[19] Alain Ghouila-Houri. Caractérisation des matrices totalement unimodulaires. *Comptes Redus Hebdomadaires des Séances de l'Académie des Sciences (Paris)*, 254:1192–1194, 1962.

[20] M. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1992.

[21] Fabrizio Grandoni, Christos Kalaitzis, and Rico Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 632–645, 2018.

[22] Christoph Hunkenschröder, Santosh Vempala, and Adrian Vetta. A 4/3-approximation algorithm for the minimum 2-edge connected subgraph problem. *ACM Transactions on Algorithms (TALG)*, 15(4):1–28, 2019.

[23] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[24] S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. In *ICALP*, 1992.

[25] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. In *STOC '92*, 1992.

[26] Zeev Nutov. On the tree augmentation problem. *Algorithmica*, 83(2):553–575, 2021.

[27] Manfred W Padberg and M Ram Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982.

[28] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Algorithms and Combinatorics, Volume 24. Springer-Verlag, Berlin Heidelberg, 2003.

[29] András Sebö and J. Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34:597–629, 2014.

[30] Vera Traub and R. Zenklusen. Local search for weighted tree augmentation and Steiner tree. *ArXiv*, abs/2107.07403, 2021.

[31] Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. *arXiv preprint arXiv:2104.07114*, 2021.

[32] David P Williamson and David B Shmoys. *The design of approximation algorithms.* Cambridge University Press, 2011.