A Primal-Dual Algorithm On 2-Steiner Graphs

by

Matthew Buckley

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, 2018

Abstract

The Steiner Tree Problem is a fundamental network design problem, where the goal is to connect a subset of terminals of a given network at minimum cost. A major open question regarding this problem, is proving that the integrality gap of a certain linear program relaxation, called the bidirected cut relaxation (BCR), is strictly smaller than 2.
In this thesis, we prove that (BCR) has integrality gap at most $\frac{5}{3}$ for a subset of instances, which we call 2-Steiner instances, via a primal-dual method.

Acknowledgements

I would like to thank Laura Sanità for agreeing to be my supervisor. Her expertise and enthusiasm for this problem helped guide our analysis. She also provided many helpful comments to keep the analysis on track as well as to simplify the final result.
I am also thankful to Chaitanya Swamy and Jochen Köenemann for agreeing to read my thesis and providing many insightful comments.
Lastly, I would like to thank Sanchit Kalhan, Ramin Mousavi, and Maxwell Levit. They offered encouragement in difficult times during the thesis process.

Dedication

This is dedicated to my parents, Bill and Paula, as well as my sister Chelsea, for their continual support and words of encouragement.

Table of Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

A fundamental problem in the area of network design is the *Steiner Tree* problem, in which the goal is to connect a set of terminals of a network at minimum cost. In particular, we are given a graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}_+$ and a partition of the vertices $V = R \cup S$. The vertices of R are referred to as terminals, while the vertices of S are called Steiner vertices. Our goal is to find a tree T spanning R that minimizes the sum of the cost of the edges. The Steiner Tree problem has been extensively studied in the literature and has applications in logistics, telecommunication, transport networks, and biology [5, 18]. The Steiner Tree problem is known to be NP-hard [14]. Determining the relationship between P and NP is arguably the most important open question in the fields of algorithms and theoretical computer science. Many researchers and practitioners have tried to discover a solution to this problem. The question was initially proposed by Cook [4] and still remains an important open problem today [11]. One approach to tackling NP-hard problems, is via approximation algorithms. An α-approximation algorithm is guaranteed to find a solution within a factor α of the optimal value and it does so in polynomial time. The theory of approximation algorithms has been applied extensively to the Steiner tree problem and determining the approximability of this problem is a major focus of current research. It is known that it is NP-hard to achieve an approximation ratio of $\frac{96}{95}$ [3], and it admits a 2-approximation algorithm via a reduction to the minimum spanning tree problem [15]. In a sequence of papers, the approximation was improved from 2 to $\ln(4)$ [20, 24, 27, 31], which is currently the best known bound. The $\ln(4)$-approximation algorithms are presented in [1, 17] respectively and they require solving a large linear program (LP) relaxation of the problem, known as the *hypergraphic* relaxation (HYP) [2, 22]. It is NP-hard to obtain a solution to (HYP), but it is possible to obtain an $(1 + \epsilon)$-approximate solution, [17], for any $\epsilon > 0$. Computing this approximate solution is the bottleneck of both $\ln(4)$-approximation algorithms. Thus a more efficient method of obtaining an approximate solution would substantially reduce the runtimes of both of these algorithms.
A more compact LP relaxation for the Steiner Tree problem is given by the so-called *bidirected cut relaxation* (BCR) [8, 30]. Only a trivial upperbound of 2 is known for (BCR), however this LP can be solved much faster than (HYP) and it is widely believed that the *integrality gap* is strictly less than 2. Improving this upperbound is an important open question which is listed in [29] as one of the 10 big open questions in the area of approximation algorithms.

It is known that (HYP) is a stronger relaxation than (BCR) and has integrality gap at most ln(4) [17]. Thus, one obvious way to tackle the open question regarding (BCR) is to compare the integrality gap of this LP with that of (HYP). In recent years, many researchers have investigated this relationship and the papers [2, 13, 17] have shown that the integrality gaps of (BCR) and (HYP) are equal for quasi-bipartite instances. G is said to be quasi-bipartite if every component of $G[S]$ is a single vertex, that is, the Steiner vertices are pairwise non-adjacent. Subsequently, [9] showed that the two LPs are equivalent for a larger class of instances: all instances where any Steiner vertex has at most 2 Steiner neighbours. Unfortunately, [9] also shows that there are instances in which exactly one Steiner vertex having degree 3 in $G[S]$ guarantees that equivalence does not hold. Thus, any progress in this direction would require substantial new ideas.

Another way to attack the open question in [29] is to work with (BCR) directly. In this context, [25] uses a primal-dual approach to show that (BCR) has integrality gap at most $\frac{3}{2}$ in quasi-bipartite instances.

In this thesis, we generalize the approach of [25] to develop a primal-dual algorithm that proves an integrality gap of at most $\frac{5}{3}$ for (BCR) in 2-Steiner instances. These are instances in which the set of Steiner vertices induces a graph where every component has at most 2 vertices. That is, the graph induced by the Steiner vertices is the union of a matching and singleton vertices. Due to the contributions of [9] it is known that these instances have a smaller integrality gap. However, our contributions here are twofold: (i) to show that a direct proof can be achieved via a primal-dual method on (BCR) and (ii) shedding some new light on the challenges of generalizing this method to arbitrary instances. We now turn to a brief overview of some of the research done on this problem as it relates to our current work.

1.1 Related Results

The Steiner Tree problem has received much attention in the literature, especially within the last twenty years. In this section, we will focus on only a small subset of this work that is most relevant to this thesis.

One well-known result is a 2-approximation for the Steiner Tree problem on metric graphs. The algorithm, as stated in [15] consists of computing a minimum spanning tree of $G[R]$. Since this result, there have been many improvements in approximation algorithms for the problem and many papers have focused on (BCR) to develop these algorithms.

One important topic for any LP relaxation is characterizing when the LP is *integral*. An LP is said to be integral if all extreme points of the polyhedron defined by the LP are integral. It was shown that (BCR) is integral if $|R| = 2$ [10], or if $R = V$ [7, 12]. Moreover, in [16, 23, 28] it is shown that (BCR) integrality also holds for a special class of graphs known as series-parallel graphs. Thus, it is possible to solve the Steiner tree problem in polynomial time for such instances. However, even if a graph does not satisfy these properties, there is still hope of developing an approximation algorithm using (BCR).

In [25] the authors use (BCR) to develop a primal-dual $\frac{3}{2} + \epsilon$ approximation algorithm for the Steiner Tree problem on quasi-bipartite graphs. This also showed that the integrality gap of (BCR) is at most $\frac{3}{2}$ for quasi-bipartite instances. These results laid the foundation for other work, such as the approximation algorithm found in

[26]. In addition, this thesis is a generalization of the results in [25] and closely follows the methods used in this paper.
Since the results in [25, 26], there have been many important developments in approximation algorithms for the Steiner Tree problem. In [31] the author presents an $\frac{11}{6}$ approximation algorithm. These results were used in [24] where the authors present a $\frac{5}{3}$ randomized approximation algorithm on 3-uniform hypergraphs. $\mathcal{H} = (\mathcal{V}, \mathcal{F})$ is a hypergraph if for each $f \in \mathcal{F}$, $f \subseteq V$. $\mathcal{H}$ is 3-uniform if for each $f \in \mathcal{F}$, $|f| = 3$. In [?] the authors develop an LP relaxation for the Steiner Tree problem and show that if every component of $G[S]$ has at most b nodes than the LP has an integrality gap of at most $\frac{2b+1}{b+1}$. Then, in order to obtain a Steiner Tree solution, they lose a small factor in the approximation algorithm.
A major breakthrough in this area was given in [27] where the authors present a 1.55 approximation algorithm for the general Steiner Tree problem which improves to a 1.28 approximation in the quasi-bipartite case. In this result, the authors use the notion of a full component which has led to more approximation algorithms. The most recent can be found in [1, 9] in which the authors present the currently best $\ln(4)$-approximation algorithm using a clever rounding algorithm. Although our approximation ratio is strictly worse than this guarantee, this thesis provides a purely combinatorial algorithm for the problem with an efficient runtime. It also provides insights relating to the structure of the problem.

1.2 Thesis Organization

The layout and ideas of this work are derived largely from the results of [25], and our work serves to generalize their results from quasi-bipartite graphs to 2-Steiner graphs. In particular, the three algorithms presented in this work are generalizations of those found in [25].
We give a $(\frac{5}{3}+\epsilon)$-approximation for the Steiner Tree Problem on 2-Steiner graphs via a primal-dual method. Such methods use a linear program and exploit LP duality and in our case we use (BCR) as the underlying LP.
In chapter 2, we start by presenting (BCR). We then discuss a notion of symmetry, which will be important in the algorithm and its analysis. The chapter is concluded by proving two results relating to this definition.
In chapter 3 we first present our algorithm in two stages. First, we describe algorithm 1, which either terminates with a Steiner Tree or terminates unsuccessfully because of one of the vertices in S. Then, algorithm 2 iteratively applies algorithm 1 to decide which vertices of S will be included in our Steiner Tree. It does so by considering the cause of algortihm 1's early termination. We end the chapter by proving three results relating to algorithm 1.
Chapter 4 discusses laminarity and how it applies to algorithm 2. This provides the basis for the remaining chapters and the results in this chapter allow us to show the correctness of algorithm 2.
In chapter 5 we assume that algorithm 2 terminates. Under this assumption we show that the tree returned by our algorithm is a Steiner Tree and we prove the approximation factor.
Chapter 6 shows that algorithm 2 indeed terminates. We do this by introducing the Pruning Algorithm, which allows us to analyze what happens when algorithm 1 terminates unsuccessfully. We conclude the chapter by giving a polynomial imple-

mentation of algorithm 2.
The thesis ends with some concluding remarks in which we also discuss some open questions and offer ideas for future work.

Chapter 2

Preliminaries

In the Steiner Tree problem we are given a connected undirected graph $H = (V, E_H)$ with cost function $c_H : E_H \to \mathbb{R}_+$ and a partition of the vertices $V = R \cup S$. The goal is to find a minimum cost tree T that spans the vertices of R.
For a vertex $u \in V$ we will denote the neighbours of u in H by $N_H(u)$. We will say that H is 2-Steiner if $|N_H(u) \cap S| \leq 1$ for each $u \in S$. It is well known that for the Steiner tree problem we can work with the graph $G = (V, E)$ which is the metric completion of H. That is, G is a complete graph on V where the cost of an edge $\{a, b\}$ of G is the length of a shortest ab-path in H with respect to costs c_H. We will denote the new costs by c. Then c satisfies the triangle inequality. That is, for $u, v, w \in V$, $c_{uv} \leq c_{uw} + c_{wv}$. This follows simply from the fact that c_{uv} is the length of a shortest path from u to v while $c_{uw} + c_{wv}$ is the length of a particular uv-path that uses w. Note that G is not a 2-Steiner graph, but it arises from a 2-Steiner graph H. This property will prove useful in showing the approximation ratio in chapter 5.
In the following, for $V' \subseteq V$, we will denote by $G[V']$ the subgraph of G induced by the vertices V'.

2.1 The Bidirected Cut Relaxation

For each edge $e = \{u, v\} \in E$ replace e by the two arcs uv and vu. Call the resulting set of arcs $\vec{E}$. We then extend c in the natural way so that $c_{uv} = c_{vu}$ for each $\{u, v\} \in E$. For a set of vertices $K \subseteq V$ we define

$$\delta^{out}(K) = \{ab \in \vec{E} \mid a \in K, b \notin K\}$$

and

$$\delta^{in}(K) = \{ab \in \vec{E} \mid a \notin K, b \in K\}$$

Now choose an arbitrary root vertex $r \in R$.

Definition 2.1. *We will say a set $C \subseteq V$ is a valid set if $C \cap R \neq \emptyset$ and $r \in V \setminus C$.*

Given a specified root vertex the bidirected cut relaxation (BCR) for the Steiner Tree problem is:

$$\min \sum_{e \in \vec{E}} c_e x_e$$

(BCR)

$$\text{s.t.} \quad \sum_{e \in \delta^{out}(C)} x_e \geq 1 \qquad \forall \text{ valid C}$$

$$x_e \geq 0 \qquad \forall\, e \in \vec{E}$$

Then the dual is given by:

$$\max \sum_{\text{valid C}} Y_C$$

(D)

$$\text{s.t.} \quad \sum_{C:\ e \in \delta^{out}(C)} Y_C \leq c_e \qquad \forall\, e \in \vec{E}$$

$$Y_C \geq 0 \qquad \forall \text{ valid C}$$

Now suppose T is an optimal Steiner Tree of G. Then by definition we have $R \subseteq V(T)$. However we may also have $V(T) \cap S \neq \emptyset$ and so $V(T) = R \cup I$ for some $I \subseteq S$. If we know I then the Steiner Tree problem just reduces to a minimum spanning tree problem. So our goal is to find I. This idea can be initially attributed to [19]. In [19], the authors develop a Steiner Tree heuristic that starts from a minimum spanning tree of $G[R]$. Their algorithm then recursively finds vertex subsets L such that if we add any node of L to our spanning tree, then the cost of our tree decreases. Such an algorithm is not a polytime algorithm. However, in [26], these techniques were applied to develop an approximation algorithm for quasi-bipartite graphs.
Following [25], we will develop an algorithm that will start with $I = \emptyset$. Then in each iteration we will add or subtract vertices from I in order to improve the Steiner tree. So suppose we have a set $I \subseteq S$ in some iteration.

Definition 2.2. *We call a set $X \subseteq S$ a residual set for $I \subseteq S$ if $X \cap I \neq \emptyset$ and $H[X]$ is connected.*

Notice that if X is a residual set, then by definition X does not contain a terminal. That is, X is not a proper set. As H is 2-Steiner we have that $H[X]$ is a single vertex or an edge. When talking about residual sets, we will omit the term "for $I \subseteq S$" if it is clear from the context. In chapter 6 we will show that we want to include the vertices of I in our Steiner tree. So our algorithm will grow dual around these vertices.
Furthermore, in our algorithm we will not pick a root vertex $r \in R$ initially. To account for this we define a notion similar to valid sets.

Definition 2.3. *A set $C \subseteq V$ is a proper set if $C \cap R \neq \emptyset$ and $(V \setminus C) \cap R \neq \emptyset$.*

Since our algorithm does not pick a root initially, we cannot identify valid sets and so we will instead grow dual for proper sets and residual sets. Notice that only valid sets appear in the objective value of (D) and so any proper set that is not valid will not affect the objective value. Thus we will refer to dual variables γ_K where K is a proper set or a residual set. For convenience we will call K a dual set.

Definition 2.4. *We will say that the amount of dual felt by an arc $e = uv$ is $\sum_{K:e\in\delta^{out}(K)} \gamma_K$ and we will say that e is tight if $\sum_{K:e\in\delta^{out}(K)} \gamma_K = c_e$.*

With this definition, we modify the constraints of (D) to $\sum_{K:e\in\delta^{out}(K)} \gamma_K \le c_e$ for each arc e. Note that here the summation is over all dual sets containing e, these are proper sets or residual sets. So our new primal dual pair becomes

$$\text{(P')} \qquad \min \sum_{e\in\vec{E}} c_e x_e$$

$$\text{s.t.} \quad \sum_{e\in\delta^{out}(K)} x_e \ge \begin{cases} 1 & \text{if } K \text{ is valid} \\ 1 & \text{if } K \text{ is residual} \\ 0 & \text{otherwise} \end{cases} \qquad \forall \text{ dual sets } K \qquad \begin{matrix}(1)\\(2)\\(3)\end{matrix}$$

It's dual is given by (D') where,

$$\text{(D')} \qquad \max \sum_{\text{valid } C} \gamma_C + \sum_{\text{residual } X} \gamma_X$$

$$\text{s.t.} \quad \sum_{K:e\in\delta^{out}(K)} \gamma_K \le c_e \qquad \forall\ e \in \vec{E}$$

$$\gamma_K \ge 0 \qquad \forall \text{ dual sets } K$$

Notice that constraints (3) in (P') are trivially satisfied as $x_e \ge 0$ for all arcs e in any feasible solution x. Throughout the remainder of this thesis, we will work with (P') and (D'). However, if K is a residual set then K does not contain any terminals, and so a Steiner Tree solution is not required to use vertices of K. Thus (P') is not a relaxation of the Steiner Tree problem. In chapter 5, we will use (BCR) to bound the cost of our solution to (P').
Now in the algorithm we will be able to continue raising variables for proper sets and residual sets as long as the total dual felt by an arc does not exceed its cost. We will then show that the set of tight edges gives a feasible solution to (P').
Now suppose at a particular point during the algorithm we have a dual solution γ.

Definition 2.5. *A set $K \subseteq V$ is unsatisfied with respect to dual solution γ if K is a dual set and for each $e \in \delta^{out}(K)$ we have $\sum_{K:e\in\delta^{out}(K)} \gamma_K < c_e$*

That is, K is an unsatisfied set if there are no tight edges in $\delta^{out}(K)$. Thus, if K is a valid or residual set, then the set of tight edges does not give a feasible solution to (P').

2.2 Dual Symmetry

Given $I \subseteq S$ our algorithm will work with (P') and (D'), and initialize all dual variables to 0. As we have not yet picked a root vertex, we will grow a collection of dual sets that enforces a notion of symmetry, which will prove useful in the analysis.

Definition 2.6. *We will say dual symmetry holds if* $\sum\limits_{K:v\in K}\gamma_K = \sum\limits_{K:u\in K}\gamma_K$ *for each* $u, v \in R\cup I$.

For $u, v \in R\cup I$ let $P = x_0x_1x_2...x_n$ be a directed uv-path where $u = x_0$ and $v = x_n$. Also, let $\bar{P} = x_n...x_2x_1x_0$ be the reversed directed vu-path.

Definition 2.7. *We will say path symmetry holds for* P *if*

$$\sum_{e\in\vec{E}(P)}\Big(\sum_{K:e\in\delta^{out}(K)}\gamma_K\Big) = \sum_{e\in\vec{E}(\bar{P})}\Big(\sum_{K:e\in\delta^{out}(K)}\gamma_K\Big)$$

Definition 2.8. *If all arcs of* P *are tight then we will say that* P *is a tight path.*

Notice that from this definition, if path symmetry holds for P and no arc feels more dual than its cost, then all arcs of P are tight if and only if all arcs of $\bar{P}$ are tight. In chapter 4 we will show that this property does indeed hold for any dual solution obtained from our algorithm.
The following lemma illustrates an important link between dual symmetry and path symmetry. The next result generalizes one found in [25]. We present a proof for completeness.

Lemma 2.9. *Let* $u, v \in R\cup I$ *and let* P *be a directed* uv*-path. Then* $\sum\limits_{K:v\in K}\gamma_K = \sum\limits_{K:u\in K}\gamma_K$ *if and only if path symmetry holds for* P.

Proof. Notice that we have

$$\sum_{K:v\in K}\gamma_K = \sum_{K:u\in K}\gamma_K \iff \sum_{K:v\in K,u\notin K}\gamma_K = \sum_{K:u\in K,v\notin K}\gamma_K \tag{2.1}$$

But the amount of dual felt by arcs on P from a dual variable γ_K is $\gamma_K|\vec{E}(P)\cap\delta^{out}(K)|$. So the total dual felt by arcs of P is $\sum_K\gamma_K|\vec{E}(P)\cap\delta^{out}(K)|$. Now for any dual set K we have:

- If $u, v \in K$ or $u, v \notin K$ then

$$|\vec{E}(P)\cap\delta^{out}(K)| = |\vec{E}(\overline{P})\cap\delta^{out}(K)| \tag{2.2}$$

- If $u \in K$, $v \notin K$ then

$$|\vec{E}(P)\cap\delta^{out}(K)| = |\vec{E}(\overline{P})\cap\delta^{out}(K)| + 1 \tag{2.3}$$

- If $u \notin K$, $v \in K$ then

$$|\vec{E}(P)\cap\delta^{out}(K)| = |\vec{E}(\overline{P})\cap\delta^{out}(K)| - 1 \tag{2.4}$$

Thus we have,

$$
\begin{aligned}
&\sum_{e\in\vec{E}(P)}\Big(\sum_{K:e\in\delta^{out}(K)}\gamma_K\Big)=\sum_{e\in\vec{E}(\bar{P})}\Big(\sum_{K:e\in\delta^{out}(K)}\gamma_K\Big)\\
\iff &\sum_K|\vec{E}(P)\cap\delta^{out}(K)|\gamma_K=\sum_K|\vec{E}(\overline{P})\cap\delta^{out}(K)|\gamma_K\\
\iff &\sum_{K:u\in K,v\notin K}|\vec{E}(P)\cap\delta^{out}(K)|\gamma_K\ +\sum_{K:v\in K,u\notin K}|\vec{E}(P)\cap\delta^{out}(K)|\gamma_K\\
&=\sum_{K:u\in K,v\notin K}|\vec{E}(\overline{P})\cap\delta^{out}(K)|\gamma_K\ +\sum_{K:v\in K,u\notin K}|\vec{E}(\overline{P})\cap\delta^{out}(K)|\gamma_K\\
\iff &\Big(\sum_{K:u\in K,v\notin K}|\vec{E}(\overline{P})\cap\delta^{out}(K)|\gamma_K+\sum_{K:v\in K,u\notin K}|\vec{E}(\overline{P})\cap\delta^{out}(K)|\gamma_K\Big)\\
&+\Big(\sum_{K:u\in K,v\notin K}\gamma_K-\sum_{K:v\in K,u\notin K}\gamma_K\Big)\\
&=\sum_{K:u\in K,v\notin K}|\vec{E}(\overline{P})\cap\delta^{out}(K)|\gamma_K\ +\sum_{K:v\in K,u\notin K}|\vec{E}(\overline{P})\cap\delta^{out}(K)|\gamma_K\\
\iff &\sum_{K:u\in K,v\notin K}\gamma_K=\sum_{K:v\in K,u\notin K}\gamma_K\\
\iff &\sum_{K:u\in K}\gamma_K=\sum_{K:v\in K}\gamma_K
\end{aligned}
$$

Where the third line follows from (2.2), the fourth line follows from (2.3) and (2.4), and the fifth line follows from (2.1).
Thus, path symmetry holds for P if and only if $\sum_{K:v\in K}\gamma_K=\sum_{K:u\in K}\gamma_K$ □

The following result now follows immediately.

Corollary 2.10. *Dual symmetry implies path symmetry for every directed path connecting two vertices of $R\cup I$.*

Chapter 3

The Algorithm

We now want to develop a primal-dual algorithm to solve the Steiner Tree problem on 2-Steiner graphs. The main idea of the algorithm is as follows: suppose we are given a set $I \subseteq S$ and suppose further that we have decided to include the vertices of I in a Steiner Tree solution. In this way, we can treat the vertices of I as terminals and now we must decide if we will add vertices of $S \setminus I$ to possibly improve the solution, this decision will be taken by running algorithm 1. That is, we will run algorithm 1 to decide whether to produce a spanning tree of $R \cup I$ or augment I.
Algorithm 1 has three phases: initialization, the symmetric phase, and the deletion phase. The symmetric phase is the heart of our algorithm as it allows us to maintain dual symmetry, as will be shown in corollary 3.5.
In initialization, we start with $\gamma_K = 0$ for each dual set K. Recall that K is either a proper set or a residual set. We also initialize $\vec{L}$ to a list of all tight arcs. As all dual variables are 0, $\vec{L}$ will only consist of arcs having cost 0 at initialization.
In the symmetric phase we identify minimally unsatisfied sets at each iteration.

Definition 3.1. *A set $C \subseteq V$ is minimally unsatisfied if it is an unsatisfied set and is minimal subject to inclusion.*

Note by definition, minimally unsatisfied sets can be either proper sets or residual sets. In this way, the vertices in I are treated just like terminals.
In Theorem 6.15, we will show that minimally unsatisfied sets can be found efficiently. So in each iteration of the symmetric phase we grow minimally unsatisfied sets until an arc becomes tight. If there is a tight arc that has its tail in $R \cup I$ or has both ends in $S \setminus I$, then we add it to our current set of arcs and redefine the minimally unsatisfied sets. Otherwise we stop algorithm 1 and will say that algorithm 1 terminates unsuccessfully. Note that we give preference to edges of H and so because of the triangle inequality, the algorithm on G and on H are equivalent, where G is the metric completion of H. In addition, we give preference to arcs having their tail in $R \cup I$.
If algorithm 1 does not terminate unsuccessfully then we proceed to the deletion phase. In this phase we pick some terminal to be the root. We then perform a reverse delete where we examine the arcs in the reverse order to how they were added. For an arc $e \in \vec{L}$ if $\vec{L} \setminus \{e\}$ contains an in-arborescence directed into r and spanning R, then we delete the arc e. Finally we output the tree T.
If algorithm 1 terminates unsuccessfully it is because an arc uv became tight where $u \in S \setminus I$ and $v \in R \cup I$. In lemma 4.3, we will show that if K_1 and K_2 are two vertex sets with $\gamma_{K_1} > 0$ and $\gamma_{K_2} > 0$ then K_1 and K_2 do not cross at any node

Algorithm 1: Algorithm 1

```
Input: Vertex sets $R$ and $I$

Initialization
  • For each dual set $K$, $\gamma_K \leftarrow 0$
  • Initialize $\vec{L}$ to contain all tight arcs

Symmetric Phase
  • While $\exists$ an unsatisfied set, do:
      – Raise dual variables of all minimally unsatisfied sets uniformly, until
        some arc $uv$ becomes tight[1]
      – If $u \in S \setminus I$ and $v \in R \cup I$ then STOP
      – Else append $uv$ to $\vec{L}$

Deletion Phase
  • Pick an arbitrary $r \in R$ as the root
  • For edges $e \in \vec{L}$, in reverse order do:
      – If each $v \in R$ has a path to $r$ in $\vec{L} \setminus \{e\}$ then set $\vec{L} \leftarrow \vec{L} - e$
  • Set $\vec{T} \leftarrow \vec{L}$, output corresponding undirected tree T
```

of $R \cup I$. This property will allow us to show that our algorithm terminates in Chapter 6. Now notice that when the arc uv becomes tight, $u \in S \setminus I$. So the above guarantee does not necessarily hold for u. Thus we will stop running Algorithm 1 in this case. However, since the arc uv became tight we will add u to our set I. So we define $Q \subseteq S \setminus I$ to be the vertices v for which there is a tight vu-path in $G[S \setminus I]$. Note that, trivially, $u \in Q$. Theorem 4.8, will show that Q can be used to connect multiple minimally unsatisfied sets.
In chapter 6 we will show that a minimum spanning tree on $R \cup I \cup Q$ has lower cost than a minimum spanning tree on $R \cup I$. This means that we can improve our Steiner tree solution by adding Q to I. This immediately suggests algorithm 2.

In algorithm 2 we start with $I = \emptyset$. At each iteration we run algorithm 1. If algorithm 1 terminates unsuccessfully then we determine the set Q as described above. We then add Q to I and move to the next iteration where we reinitialize algorithm 1 with all dual sets at 0.
If algorithm 1 terminates successfully then we are given a tree T. Note that T spans R because of the deletion phase in algorithm 1.
Now we pick a vertex $s \in I$ with minimum degree for T. If this degree is 0 or 1 then we simply remove it. This does not increase the cost of T. If the degree is 2, let w_1, w_2 be the neighbours of s in T. We replace the edges $\{w_1, s\}$ and $\{w_2, s\}$ with the edge $\{w_1, w_2\}$. By the triangle inequality this does not increase the cost of the tree. So we can remove s from I.
If s was removed from I then we reinitialize algorithm 1 in the next iteration of

1. We give priority to arcs corresponding to edges of H over edges of $E(G) \setminus E(H)$. Among those arcs, we give priority to edges having their tail in $R \cup I$ over arcs having their tail in S. That is, in algorithm 1, if we take arc uv where $u \in S \setminus I$ and $v \in R \cup I$, then there is a minimally unsatisfied set containing u such that there is no arc from this set to the minimally unsatisfied set containing v such that this arc has its tail in $R \cup I$.

Algorithm 2: Algorithm 2

```
1. Initialize $I = \emptyset$

2. Run Algorithm 1

    • If Algorithm 1 terminates unsuccessfully because arc $ab$ became tight
      with $a \in S \setminus I$ and $b \in R \cup I$, then:
        (a) Let $Q \subseteq S \setminus I$ be the vertices $v$ for which there is a tight $va$-path in
            $G[S \setminus I]$
        (b) Add $Q$ to $I$
        (c) Return to the beginning of step 2

    • Otherwise, let T be the tree returned by Algorithm 1
        (a) Pick a vertex $s \in I$ with $\deg_T(s)$ minimum.
              i. If $\deg_T(s) \geq 3$, proceed to step 3
             ii. If $\deg_T(s) \leq 2$, remove $s$ from $I$ and return to the beginning of
                 step 2

3. Output tree T returned by Algorithm 1 when every Steiner vertex has degree
   at least 3
```

algorithm 2. Otherwise, $\deg_T(s) \geq 3$ and so every vertex of I has degree at least 3 in T. Then we output T.
In chapter 5 we will show that if algorithm 2 terminates then T is within a factor of $\frac{5}{3}$ of optimal.

3.1 Properties of Algorithm 1

An important property, which will be key in the analysis, is that of laminarity. Let $L \subseteq V$.

Definition 3.2. *A collection of dual sets, $\mathcal{S}$, is L-laminar if for any $S_1, S_2 \in \mathcal{S}$ either $(S_1 \cap L) \cap (S_2 \cap L) = \emptyset$ or $(S_1 \cap L) \subseteq (S_2 \cap L)$ or $(S_2 \cap L) \subseteq (S_1 \cap L)$.*

Definition 3.3. *$\mathcal{S}$ is L-disjoint if for any $S_1, S_2 \in \mathcal{S}$ we have $(S_1 \cap L) \cap (S_2 \cap L) = \emptyset$.*

Note, in each iteration of algorithm 2 we run algorithm 1. So we will start by proving some properties of algorithm 1. The next three results are generalizations of results in [25].

Lemma 3.4. *During the symmetric part of algorithm 1, minimally unsatisfied sets are $R \cup I$-disjoint.*

Proof. Let C be any minimally unsatisfied set with respect to the dual γ and let $v \in C \cap (R \cup I)$. Now define $C_v \subseteq V$ to be the set of vertices reachable from v by a tight path. That is, for any $a \in C_v$ there is a directed va-path consisting of tight edges.
Now if there is an $p \in C_v \setminus C$ then because $v \in C$ and $p \notin C$, by definition of C_v,

there must be a tight arc in $\delta^{out}(C)$. But this contradicts that C is unsatisfied so we must have $C_v \subseteq C$.
Next suppose that there is a $q \in C \setminus C_v$ so that $C_v \subset C$. But by definition of C_v, $\delta^{out}(C_v)$ does not contain any tight arcs. So because $v \in R \cup I$, C_v is an unsatisfied set contained in C, contradicting the minimality of C. Thus we have $C = C_v$
So suppose as a contradiction that C^1 and C^2 are two distinct minimally unsatisfied sets such that there is a $u \in R \cup I$ with $u \in C^1 \cap C^2$. Then we have $C^1 = C_u = C^2$, a contradiction. Thus, minimally unsatisfied sets are $R \cup I$-disjoint. □

Corollary 3.5. *During the symmetric part of algorithm 1, the dual is symmetric and for each $v \in R \cup I$, v is contained in exactly one minimally unsatisfied set.*

Proof. We will prove the result by induction on the number of iterations of algorithm 1. Note that when algorithm 1 initializes $\gamma_K = 0$ for each dual set K. Thus, dual symmetry holds at intialization. So by lemma 2.9, path symmetry holds for any two vertices of $R \cup I$. So either all pairs of vertices of $R \cup I$ are connected by a path of cost 0 or each such vertex is contained in at least one minimally unsatisfied set. In the first case algorithm 1 terminates, so we can assume each vertex of $R \cup I$ is in at least one minimally unsatisfied set. By lemma 3.4, each vertex of $R \cup I$ is in at most one minimally unsatisfied set and so we have that each such vertex is in exactly one minimally unsatisfied set. But now we grow all minimally unsatisfied sets uniformly and so dual symmetry holds throughout this first iteration.
Now consider some iteration $t \geq 2$ and suppose that the result holds in all iterations before t. By lemma 3.4, for each $v \in R \cup I$, v is contained in at most one minimally unsatisfied set at the beginning of iteration t. However, if we define C_v to be the set of vertices that are reachable from v by a tight path, then by the proof of lemma 3.4, either C_v is minimally unsatisfied or $R \cup I \subseteq C_v$.
So consider the case that $R \cup I \subseteq C_v$ at the beginning of iteration t and choose any $u \in R \cup I$. Then by definition of C_v, there is a tight path from v to u. However, dual symmetry held at the end of the previous iteration and so by lemma 2.9, we also have a tight path from u to v. As this is true for each node of $R \cup I$, algorithm 1 terminates.
Thus, we can instead suppose that C_v is minimally unsatisfied. By the above argument, C_u is also minimally unsatisfied for each $u \in R \cup I$. Thus, each vertex of $R \cup I$ is contained in exactly one minimally unsatisfied set during this iteration. Lastly, notice that dual symmetry holds at the beginning of this iteration and we uniformly grow minimally unsatisfied sets throughout the iteration. Thus dual symmetry holds during iteration t. The result now follows by induction. □

The notion of dual symmetry turns out to be a crucial property which will play a key role in the analysis. In the next chapter we will use this definition to enforce a concept of laminarity and analyze the role of the set Q in algorithm 2.

Chapter 4

The Importance of Laminarity

As mentioned in the previous chapter, laminarity is an important property of our algorithm. In this chapter we will show that dual sets with positive γ-values form an $R \cup I$ laminar family in each iteration of algorithm 2. Using this fact, we will then be able to show in chapter 6 that algorithm 2 terminates.
So consider an iteration of algorithm 2 and let $I \subseteq S$ be the set of selected Steiner vertices. In this iteration we run algorithm 1. By lemma 3.4, minimally unsatisfied sets are $R \cup I$-disjoint during this iteration. We will use this property to derive important consequences for algorithm 1.

4.1 Laminarity in Algorithm 1

In algorithm 1 we focus on raising dual variables for minimally unsatisfied sets in each iteration.

Definition 4.1. *We will say a set $K \subseteq V$ is a loaded set for dual solution γ if K is a dual set with $\gamma_K > 0$.*

That is, K is a loaded dual set if it is currently a minimally unsatisfied set or was a minimally unsatisfied set at some earlier point during algorithm 1. Our first result will allow us to guarantee that when algorithm 1 terminates successfully there is a tight path between any two vertices of $R \cup I$. This generalizes the corresponding result found in [25]

Lemma 4.2. *Suppose K is a minimally unsatisfied set during the symmetric part of algorithm 1. Then for any two vertices $u, v \in K \cap (R \cup I)$, there is a tight directed path from u to v when restricted to $G[K \cap (R \cup I)]$.*

Proof. By lemma 3.4, minimally unsatisfied sets are $R \cup I$-disjoint. So for any $U \subset K$ such that $u \in U$, $v \notin U$, U is not an unsatisfied set by the minimality of K. Thus $\delta^{out}(U)$ contains a tight arc. As U was an arbitrary subset of K we have that there is a tight uv-path, P, using vertices of K. Moreover, note that algorithm 1 has not yet terminated unsuccessfully. So in particular, there is no tight arc ab where $a \in S \setminus I$ and $b \in R \cup I$. As $u, v \in R \cup I$, we have that P is actually a path in $G[K \cap (R \cup I)]$. □

In particular, this result shows that if K is a loaded set during the symmetric part of algorithm 1, then K is strongly connected when restricted to $G[R \cup I]$ and only tight edges. With this, we can now show that loaded sets form a $R \cup I$-laminar family.

Lemma 4.3. *Loaded dual sets form a $R \cup I$-laminar family at all times during the symmetric part of algorithm 1.*

Proof. We will prove the result by induction on the number of iterations of the symmetric part of algorithm 1. When the algorithm is initialized each vertex of $R \cup I$ forms a minimally unsatisfied set and so the result holds throughout this iteration. So now consider an iteration $t \geq 2$ and suppose that U_1 and U_2 are loaded dual sets with $U_1 \neq U_2$. Then $U_1 \cap (R \cup I) \neq \emptyset$ and $U_2 \cap (R \cup I) \neq \emptyset$. Thus U_i is either an unsatisfied set or it is contained in a minimally unsatisfied set for $i \in \{1, 2\}$. Now notice that by the description of algorithm 1, in a given iteration we only grow minimally unsatisfied sets. Moreover, a set K remains minimally unsatisfied until an arc of $\delta^{out}(K)$ becomes tight. In addition, we never decrease dual variables during algorithm 1 and so when we start growing γ_K, K is minimally unsatisfied and it remains minimally unsatisfied until an arc of $\delta^{out}(K)$ becomes tight. At which point, K is satisfied. So, because U_i is a loaded set, if U_i is unsatisfied, then U_i is a minimally unsatisfied set. Thus there is a minimally unsatisfied set K_i such that $U_i \subseteq K_i$ for $i \in \{1, 2\}$.
First suppose that $K_1 \neq K_2$. Then K_1 and K_2 are distinct minimally unsatisfied sets, so by lemma 3.4, K_1 and K_2 are $R \cup I$-disjoint. Thus U_1 and U_2 are $R \cup I$-disjoint.
Otherwise, suppose that $K_1 = K_2$.
Then let $LS_{K_1} = \{K \subseteq K_1 \mid K \cap (R \cup I) \neq \emptyset, \gamma_K > 0\}$. Note that $U_1, U_2 \in LS_{K_1}$ by the choice of K_1 and the fact that $K_1 = K_2$. Now let t' be the last iteration where $\gamma_{K_1} = 0$. At iteration t, $\gamma_{K_1} > 0$ and so we have $t' < t$. Moreover, each element of LS_{K_1} is a loaded dual set. As stated above, for a dual set K, when we start growing γ_K, K is minimally unsatisfied and it remains minimally unsatisfied until an arc of $\delta^{out}(K)$ becomes tight.
But now notice that we start growing γ_{K_1} after iteration t', by the choice of t'. Also, K_1 is minimally unsatisfied in iteration t. So in all iterations between t' and t, K_1 is minimally unsatisfied. Thus the dual sets in $LS_{K_1} \setminus \{K_1\}$ have stopped growing by iteration t' and so by the induction hypothesis $LS_{K_1} \setminus \{K_1\}$ is a $R \cup I$-laminar family. So by definition of LS_{K_1} we have that LS_{K_1} is a $R \cup I$-laminar family. Thus we have $U_1 \cap (R \cup I) \subseteq U_2 \cap (R \cup I)$ or $U_2 \cap (R \cup I) \subseteq U_1 \cap (R \cup I)$.
Thus in both cases U_1 and U_2 do not cross. So loaded dual sets from a $R \cup I$-laminar family. $\square$

Note that if the dual felt by an arc is at most its cost at the beginning of algorithm 1, then this will hold throughout algorithm 1. The reason is, as soon as an arc e becomes tight we redefine minimally unsatisfied sets so that if C is a minimally unsatisfied set then $e \notin \delta^{out}(C)$. Thus the dual felt by e will not increase after e becomes tight. But when algorithm 1 initializes, $\gamma_K = 0$ for each dual set K. As $c_e \geq 0$ for each arc e, we have that no arc is overtightened during algorithm 1. We can now prove the following result.

Lemma 4.4. *Suppose that $u, v \in R \cup I$ during an execution of algorithm 1 and let P be a directed path from u to v. Then at all times during algorithm 1, P is a tight path if and only if $\bar{P}$ is a tight path, where $\bar{P}$ is the reversal of P.*

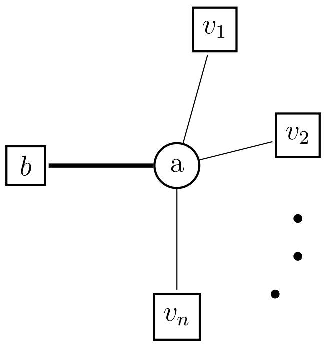


Figure 4.1: The Crossing Sets of Lemma 4.7

Proof. Consider an arbitrary point during the execution of algorithm 1 and suppose that P is a tight path. By corollary 3.5, dual symmetry holds for $R \cup I$ so that by corollary 2.10, path symmetry holds for P. That is,

$$\sum_{e \in \vec{E}(P)} \Big(\sum_{K: e \in \delta^{out}(K)} \gamma_K \Big) = \sum_{e \in \vec{E}(\bar{P})} \Big(\sum_{K: e \in \delta^{out}(K)} \gamma_K \Big) \tag{4.1}$$

However, as argued above, no arc is overtightened during the execution of algorithm 1 and each arc of P is tight. Moreover, for $u, v \in V$, $c_{uv} = c_{vu}$. Combining this with (4.1) shows that each arc of $\bar{P}$ is also tight so that $\bar{P}$ is a tight path.
Repeating this analysis shows that whenever $\bar{P}$ is a tight path, P is also tight. □

4.2 Important Connectivity Properties

We will now assume that algorithm 1 terminates unsuccessfully because some arc ab became tight where $a \in S \setminus I$ and $b \in R \cup I$. We now define the following.

Definition 4.5. *For a loaded dual set K, we say that K is an outermost loaded dual set if there is no other loaded dual set K' such that $K \subseteq K'$*

Note that by lemma 4.3, loaded dual sets form a $R \cup I$-laminar family and so the notion of an outermost loaded dual set is well-defined. We also give the following.

Definition 4.6. *For an outermost loaded dual set K we will say that w identifies K for any $w \in K \cap (R \cup I)$*

Again, because of lemma 4.3, each vertex of $R \cup I$ identifies a unique outermost loaded dual set. The following two results analyze the dual felt by arcs of $\delta^{out}(\{a\})$ when the arc ab becomes tight.

Lemma 4.7. *Suppose the arc ab becomes tight during the symmetric part of algorithm 1 and causes algorithm 1 to terminate unsuccessfully for $a \in S \setminus I$ and $b \in R \cup I$. Then the arc ba is also tight.*

Proof. Assume as a contradiction that ab is tight but that there is no tight ba-path. Now because ab went tight but $a \in S \setminus I$ there must be loaded sets crossing at a. To see this, suppose as a contradiction that no loaded sets cross at a. As $a \in S \setminus I$, any loaded dual set containing a also contains a node of $R \cup I$. Moreover, as the arc ab is tight, there is a loaded dual set containing a and not b. So suppose that $c \in V \setminus \{a, b\}$ be contained in this dual set such that the arc ca is tight. Then, because no loaded dual sets cross at a, any loaded dual set containing a also contains c. But the arc ca is tight so we have

$$\sum_{K:c\in K, a\notin K} \gamma_K = c_{ac}$$

Also, the arc ab is tight, and whenever a is contained in a loaded dual set, c is also contained in a loaded dual set. So we have

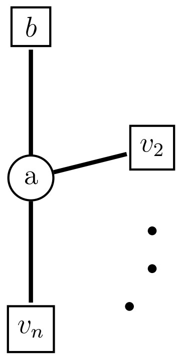
$$\sum_{K:a,c\in K} \gamma_K = c_{ab}$$

But now by the triangle inequality $c_{bc} \leq c_{ab} + c_{ac}$. Moreover, as every loaded dual set containing a also contains c we have that b and c are not contained in a dual set together. Thus

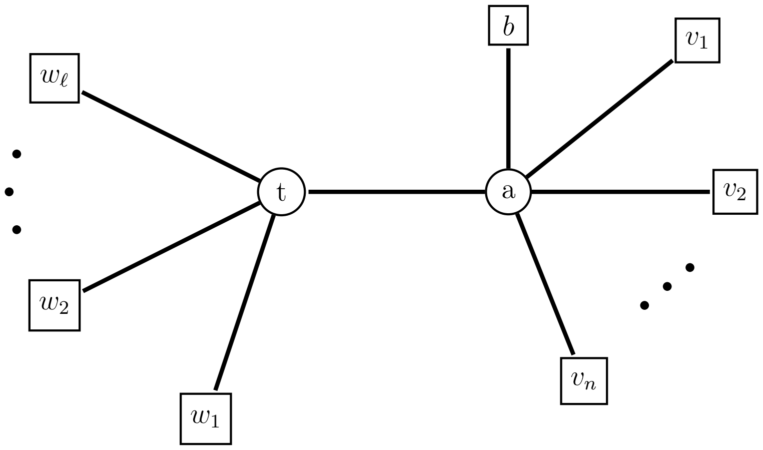
$$\sum_{K:c\in K, b\notin K} \gamma_K = \sum_{K:c\in K} \gamma_K = \sum_{K:c\in K, a\notin K} \gamma_K + \sum_{K:a,c\in K} \gamma_K = c_{ac} + c_{ab} \geq c_{bc}$$

Thus the arc cb is tight. If $c \in R \cup I$ then notice that in algorithm 1 we give preference to edges that have both ends in $R \cup I$. Also, the arc bc is tight no later than when the arc ab becomes tight, thus we would not stop algorithm 1, a contradiction. Otherwise, $c \in S \setminus I$, but then we can repeat the same argument, to show that there must be at least two loaded dual sets crossing at c, and thus there are at least two loaded dual sets crossing at a.
So let $v_1, ..., v_n$ be vertices of $R \cup I$ that identify distinct outermost loaded sets $K_1, ..., K_n$ crossing at a. Note that b is not contained in a loaded set with v_i for any $i \in \{1, ..., n\}$. To see this, suppose as a contradiction, b and v_i are contained in a loaded set K. Now if $v_i \in S \setminus I$ then because $v_i \in K$ and K is a loaded set there must be a tight $v'v_i$-path for some vertex $v' \in K \cap (R \cup I)$. Otherwise $v_i \in R \cup I$ so set $v' = v_i$. But now $b, v' \in K \cap (R \cup I)$ and K is a loaded set. Thus, K was a minimally unsatisfied set at some point during algorithm 1. So by lemma 4.2, there is a tight bv'-path in K. This now gives a tight bv_i-path in K. However, by assumption the arc va is tight and so we have a tight ba-path, a contradiction.
Thus we have that b is not contained in a loaded set with v_i for any $i \in \{1, ..., n\}$. Note, by assumption, $b \in R \cup I$. An example is shown in figure 4.1. (Note that the thin edges of figure 4.1 correspond to tight paths).
Now observe that the arc ab is tight and so there must be dual sets crossing at a that do not contain b. That is, $n \geq 1$. So consider node v_1. This identifies an outermost loaded dual set K_1 crossing at a. But note that we only define minimally unsatisfied sets to grow around nodes of $R \cup I$. So because both v_1 and a are contained in a minimally unsatisfied set, there must be a tight path from v_1 to a. Moreover, by assumption, the arc ab is tight. Thus there is a tight path from v_1 to b, P_1, and P_1 uses the edge ab. But note that $v_1, b \in R \cup I$. Thus, by lemma 4.4, $\bar{P_1}$ is a tight path. Lastly, notice that the path P_1 contains the arc ab, thus the arc ba is tight as required. $\square$

In algorithm 2, if an arc ab becomes tight where $a \in S \setminus I$ and $b \in R \cup I$ then algorithm 1 terminates unsuccessfully. We then compute $Q \subseteq S \setminus I$ and add the nodes of Q to our set I. Our next result justifies why it is a good idea to add Q to I.



(a) Case 1 of Theorem 4.8



(b) Case 2 of Theorem 4.8

Figure 4.2: The cases of Theorem 4.8

Theorem 4.8. *Suppose algorithm 1 terminates unsuccessfully when arc ab becomes tight for $a \in S \setminus I$, $b \in R \cup I$, and let $Q \subseteq S \setminus I$ be the set determined in algorithm 2. Then for any $c \in R \cup I$, if there is a tight path from c to a vertex of Q that only uses vertices of $R \cup I \cup Q$ then there is a tight path from Q to c that only uses vertices of $R \cup I \cup Q$ when algorithm 1 terminates. Moreover, Q is strongly connected in the graph of tight arcs.*

Proof. By lemma 4.7, we have that ba is also tight. Note that ab went tight and caused algorithm 1 to terminate unsuccessfully. However, $a \in S \setminus I$, so there must be loaded sets crossing at a. Let $v_1, ..., v_n$ be the nodes of $R \cup I$ such that the arc $v_i a$ is tight for each $i \in \{1, ..., n\}$. As the arc ba is tight we have that $b \in \{v_1, ..., v_n\}$ and so $n \geq 1$. After relabelling, we can assume that $b = v_1$.
Now note that for any other loaded set crossing at a that is not identified by one of $v_1, ..., v_n$ then because G is 2-steiner all such sets are identified by some vertex $t \in S \setminus I$ such that $\{a, t\} \in E_H$. That is, $|Q| \in \{1, 2\}$. So we have two cases:

Case 1: $|Q| = 1$.
Then $Q = \{a\}$ and is clearly strongly connected. In addition, any loaded dual set crossing at a contains one of $v_1, ..., v_n$. But now $b = v_1$ so that the arc av_1 is tight. Thus, there must be additional dual sets crossing at a that load dual for the arc av_1. That is, $n \geq 2$. This case is illustrated in figure 4.2(a).

We will start by showing that the arc av_i is tight for each $i \in \{2, ..., n\}$. Observe that arc $v_i a$ is tight, by definition of v_i. So together with the arc av_1 this gives a tight path from v_i to v_1 through a, call this path P_i. Moreover, $v_1, v_i \in R \cup I$. Thus, by lemma 4.4, $\bar{P}_i$ is also a tight path. Also, as $v_i a \in \vec{E}(P)$, we have that $av_i \in \vec{E}(\bar{P})$ so that the arc av_i is tight.
Thus, for each $i \in \{1, ..., n\}$, the arc av_i is tight. So consider any other vertex $c \in R \cup I$ such that there is a tight path from c to Q using nodes of $R \cup I \cup Q$, let P be such a path. We know that $Q = \{a\}$ and so $V(P) \setminus \{a\} \subseteq R \cup I$. Moreover, $v_1, ..., v_n$ are the only vertices of $R \cup I$ that are joined to a by a tight arc. Thus the last vertex of P before a is v_i for some $i \in \{1, ..., n\}$. So let P' be the subpath of P from c to v_i. Then P' is a tight between two nodes of $R \cup I$ and so by lemma 4.4, $\bar{P}'$ is also tight. Now the arc av_i followed by $\bar{P}'$ gives a tight path from a to c using nodes of $R \cup I \cup Q$. In particular, we have a tight path from Q to c using nodes of $R \cup I \cup Q$.

Case 2: $|Q| = 2$
Then $Q = \{a, t\}$ for some $t \in S \setminus I$. Moreover, we have chosen $v_1, ..., v_n$ to be all nodes of $R \cup I$ that are joined to a by a tight arc. Thus, if K is any dual set not containing any of $v_1, ..., v_n$ and crossing at a, then K must contain t. In addition, by the description of algorithm 2, we have included t in Q because there is a tight path from t to a in $H[S \setminus I]$. Thus because G is 2-Steiner, the arc ta is tight. So let $w_1, ..., w_\ell \in R \cup I$ be the vertices joined to t by a tight arc that are distinct from $v_1, ..., v_n$. Note that algorithm 1 terminated unsucessfully because the arc ab became tight. As $a \in S \setminus I$ and $b \in R \cup I$, by the priority rule used in algorithm 1 there must be two minimally unsatisfied sets loading the arc ab. Otherwise, there is only one minimally unsatisfied set containing a and not b, let v be the first node in this set such that the arc va became tight. Then note that

$$\sum_{K: v \in K, a, b \notin K} = c_{av}$$

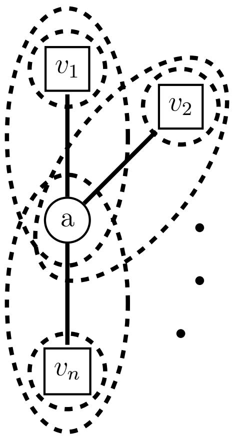
and

$$\sum_{K: a, v \in K, b \notin K} = c_{ab}$$

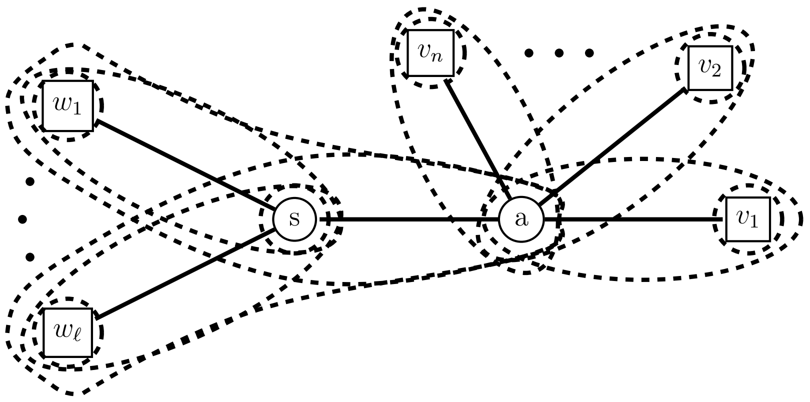
Thus

$$\sum_{K: v \in K, a \notin K} = c_{av} + c_{ab} \leq c_{bv}$$

by the triangle inequality. So the arc bv is tight and thus we would have taken the arc bv instead of ab by the priority rule of algorithm 1, a contradiction. Thus $n + \ell \geq 2$. This case is illustrated in figure 4.2(b).
We have that $b = v_1$, so now consider v_i for $i \in \{2, ..., n\}$. Then as the arc $v_i a$ is tight we have that $v_i a v_1$ gives a tight $v_i v_1$-path. As in case 1, we now have that $v_1 a v_i$ is also a tight path so that in particular the arc av_i is tight.
Next, consider w_i for $i \in \{1, ..., \ell\}$ By the choice of w_i the arc $w_i t$ is tight. Moreover, we have shown that the arcs ta and av_1 are tight. Also, we have chosen the vertices so that $w_1, ..., w_\ell, v_1, ..., v_n$ are all distinct. Thus $P_i = w_i t a v_1$ is a tight path from w_i to v_1. However, $a, t \in S$ and G is 2-Steiner. Thus $v_1, w_i \in R \cup I$. So by lemma 4.4, the path $\bar{P}_i$ is also tight. As $w_i t \in \vec{E}(P_i)$ we have that $tw_i \in \vec{E}(\bar{P}_i)$ so that the arc tw_i is tight as desired. Note that in addition, $at \in \vec{E}(\bar{P}_i)$ so that this arc is also



(a) Case for $|Q| = 1$



(b) Case for $|Q| = 2$

Figure 4.3: Possible crossings at Q

tight and thus Q is strongly connected.
So now consider $c \in R \cup I$ such that there is a tight path P from c to a node in Q with $V(P) \subseteq R \cup I \cup Q$. Then $V(P) \setminus \{a, t\} \in R \cup I$. Now let u be the last vertex on P in $R \cup I$ and let P' be the subpath of P from c to u. Then by definition, $v_1, ..., v_n$ are all vertices joined to a by a tight arc and $w_1, ..., w_\ell$ are all vertices joined to t by a tight arc. So we have that $u \in \{v_1, ..., v_n, w_1, ..., w_\ell\}$. Thus, as shown above we have that there is a tight arc from a vertex of Q to u. Moreover, $c, u \in R \cup I$ and P' is a tight cu-path with $V(P') \subseteq R \cup I$. By lemma 4.4, $\bar{P}'$ is also tight. Thus we have a tight path from Q to c contained in $R \cup I \cup Q$.

Thus in both cases we obtain the desired result so that this completes the proof. □

Note that in algorithm 1 we give preference to arcs whose tail is a vertex of $R \cup I$. But the arc ab became tight where $a \in S \setminus I$. As proved in lemma 4.7, when the arc ab becomes tight there must be at least two minimally unsatisfied sets crossing at a which do not contain b. By lemma 4.7, there is also a minimally unsatisfied set containing b which crosses at a. Combining this with Theorem 4.8, we have one of the two cases depicted in figure 4.3, where $n \geq 2$ in figure 4.3(a) and $n + \ell \geq 2$ in figure 4.3(b). We have thus proven the following.

Observation 4.9. *If algorithm 1 terminates unsuccessfully then Q is used to connect* *at least 3 components of $R \cup I$ when restricted to tight edges. Moreover, this new component is also strongly connected when restricted to tight edges in $G[R \cup I \cup Q]$.*

This now provides the intuition for why we have defined Q in this way and why we choose to add Q to our set I in algorithm 2. In chapter 6, we will make this intuition formal and show that indeed, adding Q decreases the cost of a spanning tree of $R \cup I$. Before proving this, we first show in chapter 5 that if algorithm 2 is guaranteed to terminate, it will output a $\frac{5}{3}$-approximate solution. Chapter 5 will also use the results of this chapter in order to show the approximation ratio.

Chapter 5

The Approximation Factor

In this chapter we will show the approximation factor, assuming that algorithm 2 terminates. For the remainder of this chapter we suppose that algorithm 2 terminates with dual solution γ and tree T.

5.1 A Minimum Cost $R \cup I$-Spanning Tree

The next proposition shows that γ is feasible for (D').

Proposition 5.1. *For an arc $e = uv \in \vec{E}(G)$, the total dual felt by e is at most c_e at all times during the execution of algorithm 2.*

Proof. Note that in each iteration of algorithm 2 we run algorithm 1 and start by setting $\gamma_K = 0$ for each dual set K. As $c_e \geq 0$ for each arc e, no arc feels more than its cost at initialization.
Now we only alter dual variables during the symmetric phase. In this phase we uniformly grow minimally unsatisfied sets in each iteration. By definition of an unsatisfied set, the dual values raised in a particular iteration only affect the dual felt by arcs that are not tight. Moreover, by the description of algorithm 1, we stop raising dual variables as soon as an arc becomes tight. It thus follows that during the symmetric phase of algorithm 1, no arc is overtightened. As no arc was overtightened at initialization, we have that no arc feels more dual than its cost during algorithm 1. As mentioned above, this now implies that the desired property holds throughout algorithm 2. □

We would now like to analyze the cost of this dual solution. Recall that a dual set K can refer to a proper set or a residual set. In algorithm 1, we do not pick a root vertex initially and so we only consider proper sets and residual sets. Once a root is selected, the proper sets are partitioned into valid and invalid sets. However, only valid sets and residual sets appear in the objective function of (D').
Now consider any iteration of algorithm 2. We run the symmetric part of algorithm 1 until there are no more minimally unsatisfied sets. Thus, by lemma 4.2, when the symmetric phase ends there is a directed vr-path using edges of $\vec{L}$ for each $v \in R \setminus \{r\}$. Now during the deletion phase of algorithm 1 we remove arcs of $\vec{L}$ as long as there is still a tight vr-path for each $v \in R \setminus \{r\}$. Thus when algorithm 1 terminates, if C is a valid set then there is a vertex $v \in C \cap (R \setminus \{r\})$ and $r \notin C$. Thus $|\delta^{out}(C) \cap \vec{L}| \geq 1$.

Moreover, when algorithm 2 terminates, for each $u \in I$ we have $\deg_T(u) \geq 3$. Also $u \in I \subseteq S$ and $r \in R$ with $S \cap R = \emptyset$. Thus $u \neq r$. Moreover note that each edge of T corresponds to an arc of $\vec{L}$. But now if $\delta^{out}(\{u\}) \cap \vec{L} = \emptyset$ then because $u \neq r$, u is not used to connect any vertex of $R \setminus \{r\}$ to r. Thus all the arcs in $\delta^{in}(\{u\}) \cap \vec{L}$ would be deleted during the deletion phase of algorithm 1 in the last iteration of algorithm 2. Thus we would have $\deg_T(u) = 0$, a contradiction. Thus, when algorithm 2 terminates for each $u \in I$ we have $|\delta^{out}(\{u\}) \cap \vec{L}| \geq 1$.
Lastly consider a residual set X so that $X \cap I \neq \emptyset$. Now suppose $u \in X \cap I$. If $\delta^{out}(X) \cap \vec{L} = \emptyset$ then because $r \notin X$, none of the vertices of X are used to connect a vertex of $R \setminus \{r\}$ to r. Thus, for each $s \in X$ we will delete all arcs with an end in s during the reverse delete step. So we will have $\delta^{out}(\{s\}) \cap \vec{L} = \emptyset$ and $\delta^{in}(\{s\}) \cap \vec{L} = \emptyset$ so that $\deg_T(s) = 0$. In particular $\deg_T(u) = 0$, a contradiction. Thus we have $|\delta^{out}(X) \cap \vec{L}| \geq 1$.
So given that $\vec{L}$ are the arcs corresponding to the tree output by algorithm 2, define x for each arc $e \in \vec{E}(G)$ as

$$x_e = \begin{cases} 1 & \text{if } e \in \vec{L} \\ 0 & \text{otherwise} \end{cases}$$

Then clearly $x_e \geq 0$ for each $e \in \vec{E}(G)$. Also by the above analysis we have $\sum\limits_{e \in \delta^{out}(C)} x_e \geq 1$ for each valid set C. Moreover, for each residual set X we have $\sum\limits_{e \in \delta^{out}(X)} x_e \geq 1$. Thus x is feasible for (P'). Also note that for any $u \in S \setminus I$, $\delta^{out}(\{u\}) \cap \vec{L} = \emptyset$ as otherwise u would have been added to I during algorithm 2. Thus during the last iteration of algorithm 2, when the symmetric part of algorithm 1 terminates u is not used to connect any vertex $v \in R \setminus \{r\}$ to r. So when algorithm 2 terminates $\delta(\{u\}) \cap E(T) = \emptyset$. Thus T is a spanning tree on $R \cup I$. Now using complementary slackness we will show that T is a minimum spanning tree of $R \cup I$. The following result follows from the seminal work of Edmonds [8] and was also shown in [25]. We report here a proof for completeness.

Lemma 5.2. *If algorithm 2 terminates with tree T and set $I \subseteq S$ then*

$$\sum_{e \in E(T)} c_e = \sum_{\text{valid } C} \gamma_C + \sum_{\text{residual } X} \gamma_X$$

Moreover, T is a minimum spanning tree of $R \cup I$.

Proof. Let $\vec{L}$ be the set of arcs in $\vec{T}$ when algorithm 2 terminates. Now define x for each $e \in \vec{E}(G)$ as

$$x_e = \begin{cases} 1 & \text{if } e \in \vec{L} \\ 0 & \text{otherwise} \end{cases}$$

Then as discussed above, x is feasible for (P'). Moreover, if γ is the dual solution returned by algorithm 2 then by lemma 5.1 and the above discussion γ is feasible for (D'). Also observe that $\sum\limits_{e \in E(T)} c_e = \sum\limits_{e \in \vec{L}} c_e = \sum\limits_{e \in \vec{E}(G)} c_e x_e$. Now the complementary slackness conditions for the pair (P'), (D') are:

1. For $e \in \vec{E}$ either $x_e = 0$ or $\sum\limits_{K : e \in \delta^{out}(K)} \gamma_K = c_e$

2. For valid C either $\gamma_C = 0$ or $\sum_{e \in \delta^{out}(C)} x_e = 1$
3. For residual X either $\gamma_X = 0$ or $\sum_{e \in \delta^{out}(X)} x_e = 1$
4. For all other dual sets K either $\gamma_K = 0$ or $\sum_{e \in \delta^{out}(K)} x_e = 0$

Now note that if $x_e > 0$ then $e \in \vec{L}$. However, in the last iteration of algorithm 2 when algorithm 1 is run it terminates successfully (as otherwise we would add a new vertex to I and begin a new iteration of algorithm 2). So when e was added to $\vec{L}$ it was a tight arc and we do not decrease any dual variables during the last iteration of algorithm 2. Thus e remains tight when algorithm 2 terminates. So condition 1 holds for x and γ.
Now we will show that conditions 2 and 3 hold. So suppose K is a valid set or a residual set. Because x is feasible we have

$$\sum_{e \in \delta^{out}(K)} x_e \geq 1$$

So suppose as a contradiction that $\gamma_K > 0$ and there are two arcs $e, e' \in \vec{L}$ such that $e, e' \in \delta^{out}(K)$. Assume that e' was added to $\vec{L}$ after e. However, because $\gamma_K > 0$, K was a minimally unsatisfied set at some point during the last iteration of algorithm 2. Thus, by lemma 4.2, for any $a, b \in K \cap (R \cup I)$ there is a tight path from a to b contained in $G[K \cap (R \cup I)]$. Now suppose that $e = uv$. Then because $|\delta^{out}(K') \cap \vec{L}| \geq 1$ for each valid set K', there is still a directed path from u to r among the arcs of $\vec{L} \setminus \{e'\}$. As e is a tight arc we have that $u \in R \cup I$. Also, by lemma 4.2, for any $t \in K \cap (R \cup I)$ there is a tight path from t to u, using the vertices of K. And so for any $t \in K \cap (R \cup I)$ there is a tight path from t to r in $\vec{L} \setminus \{e'\}$. Thus e' is not needed to connect vertices of C to r and so e' will be deleted during the reverse delete step, a contradiction. As K was either a valid set or residual set, we have that conditions 2 and 3 hold.
Lastly, consider condition 4. If K is a dual set that is not valid or residual then K is a proper set with $r \in K$. But then if there are any arcs in $\delta^{out}(K) \cap \vec{L}$, they are not needed to connect any terminal to r. So these arcs would be deleted during the reverse delete step. Thus $|\delta^{out}(K) \cap \vec{L}| = 0$ and so condition 4 holds.
So all of the complementary slackness conditions hold and so we have

$$\sum_{e \in E(T)} c_e = \sum_{e \in \vec{E}(G)} c_e x_e = \sum_{\text{valid } C} \gamma_C + \sum_{\text{residual } X} \gamma_X$$

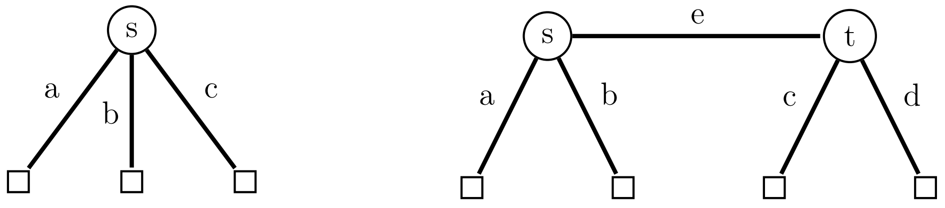
Now, let T' be any other $(R \cup I)$-spanning tree. Then if we direct all edges of T' towards r and define x' for each $e \in \vec{E}(G)$ as

$$x'_e = \begin{cases} 1 & \text{if } e \in \vec{E}(T') \\ 0 & \text{otherwise} \end{cases}$$

then because T' is a spanning tree of $R \cup I$, x' is feasible for (P'). But by the optimality of x we have

$$\sum_{e \in E(T)} c_e = \sum_{e \in \vec{E}(G)} c_e x_e \leq \sum_{e \in \vec{E}(G)} c_e x'_e = \sum_{e \in E(T')} c_e$$

Thus T is a minimum spanning tree of $R \cup I$. □



(a) A 3 edge 1-star

(b) A 5 edge 2-star

Figure 5.1: The 1-star and 2-star from lemma 5.4

5.2 The $\frac{5}{3}$ Ratio

Let $G' = (V', E')$ be an arbitrary graph.

Definition 5.3. *We say that G' is a k-star, for $k \in \mathbb{N}$, if G' has a path P on k vertices such that by contracting P to a single vertex, the resulting graph is a star.*

We will use this definition and the previous result to derive the approximation factor as well as an integrality gap bound for (BCR).
Note that any Steiner tree corresponds to a feasible solution for (BCR). However, only $\sum_{\text{valid } C} \gamma_C$ is a dual bound on the optimal value of a Steiner tree. So we need to bound $\sum_{\text{residual } X} \gamma_X$.

Lemma 5.4. *If algorithm 2 terminates with tree T and set $I \subseteq S$ then*

$$\sum_{\textit{residual } X} \gamma_X \le \frac{2}{5} \sum_{e \in E(T)} c_e$$

Proof. Note that every component of $H[I]$ is a single vertex or a path with one edge. Moreover, algorithm 1 gives priority to arcs of $\vec{H}$. So suppose that $a, b \in S$ are not adjacent in H. Then these nodes are joined by a path P in H with at least two vertices. By definition of G we have that

$$\sum_{e \in \vec{E}(P)} c_e \le c_{ab}$$

But notice that if K is a dual set with $ab \in \delta^{out}(K)$ then at least one arc of P is also contained in $\delta^{out}(K)$. So at any point during algorithm 1, if the arc ab is tight then we also have that P is a tight path. So by our priority rule, the arc ab will never be used in algorithm 1. Thus, every component of $T[I]$ is also either a single vertex or a path with one edge. Now let K be a component of $T[I]$.
If K is a single vertex s then $\deg_T(s) \ge 3$ and so s is incident to at least 3 vertices of R. Thus s and all the edges incident to s induce a 1-star, $\bar{T}_K$, with at least 3 edges. By possibly removing some edges we obtain a 1-star $\widetilde{T}_K$ with exactly 3 edges: a, b, c. $\widetilde{T}_K$ is shown in figure 5.1(a).
Then notice that $\sum_{f \in E(\widetilde{T}_K)} c_f \le \sum_{f \in E(\bar{T}_K)} c_f$. So consider residual sets X with $s \in X$. Edges a, b, and c all feel dual from γ_X for each such set X. However, by lemma 5.1 no arc feels more than its cost when the last iteration of algorithm 2 terminates. Then

because dual variables are not changed after this last iteration we have $\sum\limits_{X:s\in X} \gamma_X \leq c_i$ for each $i \in \{a, b, c\}$. Thus,

$$3\Big(\sum_{X:s\in X} \gamma_X\Big) \;\leq\; c_a + c_b + c_c = \sum_{f\in E(\widetilde{T}_K)} c_f \;\leq\; \sum_{f\in E(\bar{T}_K)} c_f$$
$$\implies \sum_{X:s\in X} \gamma_X \;\leq\; \frac{1}{3}\Big(\sum_{f\in \bar{T}_K} c_f\Big) \;\leq\; \frac{2}{5}\Big(\sum_{f\in \bar{T}_K} c_f\Big)$$

Otherwise, K is a path with one edge: $\{s, t\}$. However, $\deg_T(s) \geq 3$ and $\deg_T(t) \geq 3$ so both s and t are adjacent to at least two vertices of R. Moreover, $\delta_T(\{s\}) \cap \delta_T(\{t\}) = \{s, t\}$ as T is a tree. Thus s and t together with edges adjacent to one of these two vertices induces a 2-star, $\bar{T}_K$ in which both s and t have degree at least 3. By possibly deleting edges we can obtain a 2-star, $\widetilde{T}_K$ in which both s and t have degree 3. Moreover, $\delta_{\widetilde{T}_K}(\{s\}) = \{a, b, e\}$ and $\delta_{\widetilde{T}_K}(\{t\}) = \{c, d, e\}$. $\widetilde{T}_K$ is depicted in figure 5.1(b).
Then notice that $\sum\limits_{f\in E(\widetilde{T}_K)} c_f \;\leq\; \sum\limits_{f\in E(\bar{T}_K)} c_f$. Now the maximum cost edge is adjacent to one of s or t. So assume the maximum cost edge is $i \in \{a, b, c, d, e\}$ and that it is adjacent to s, without loss of generality. Note that because i is the maximum cost edge we have

$$\sum_{j\in\{a,b,c,d,e\}\setminus\{i\}} c_j \leq \sum_{j\in\{a,b,c,d,e\}} c_j$$

But now notice that edges c and d feel dual from γ_X for each residual X with $t \in X$. So,

$$\sum_{X:t\in X} \gamma_X \leq c_j \qquad \forall\; j \in \{c, d\}$$

Also, for $j \in \{a, b, e\}\setminus\{i\}$, edge j feels dual from γ_X for each residual X with $s \in X$, $t \notin X$. So,

$$\sum_{X:s\in X, t\notin X} \gamma_X \leq c_j \qquad \forall\; j \in \{a, b, e\} \setminus \{i\}$$

Thus we have

$$\begin{aligned}2\Big(\sum_{X:X\cap\{s,t\}\neq\emptyset} \gamma_X\Big) = 2\Big(\sum_{X:t\in X} \gamma_X\Big) + 2\Big(\sum_{X:s\in X, t\notin X} \gamma_X\Big) &\leq \sum_{j\in\{a,b,c,d,e\}\setminus\{i\}} c_j \\ &\leq \frac{4}{5}\Big(\sum_{j\in\{a,b,c,d,e\}} c_j\Big) \\ &\leq \frac{4}{5}\Big(\sum_{f\in \bar{T}_K} c_f\Big)\end{aligned}$$

Thus we have

$$\sum_{X:X\cap\{s,t\}\neq\emptyset} \gamma_X \leq \frac{2}{5}\Big(\sum_{f\in \bar{T}_K} c_f\Big)$$

Thus for each component K of $T[I]$ we have $\sum\limits_{X:X\cap V(K)\neq\emptyset} \gamma_X \;\leq\; \frac{2}{5}\Big(\sum_{f\in E(\bar{T}_K)} c_f\Big)$. However each component of $T[I]$ induces a 1-star or a 2-star and these graphs are all edge disjoint. Moreover, clearly for each $s \in I$, s is contained in some component of $T[I]$. So let $\mathcal{K}_I$ be the set of components of $T[I]$. Then, we have,

$$\sum_{\text{residual } X} \gamma_X \;\leq\; \sum_{K \in \mathcal{K}_I} \Big(\sum_{X : X \cap V(K) \neq \emptyset} \gamma_X \Big) \;\leq\; \frac{2}{5} \Big[\sum_{K \in \mathcal{K}_I} \Big(\sum_{f \in E(\bar{T}_K)} c_f \Big) \Big] \;\leq\; \frac{2}{5} \Big(\sum_{f \in E(T)} c_f \Big)$$

□

With these results, we now have all the ingredients for the approximation factor and the integrality gap bound.

Theorem 5.5. *If algorithm 2 terminates successfully with tree T then*

$$\sum_{e \in E(T)} c_e \leq \frac{5}{3} \sum_{\textit{valid } C} \gamma_C$$

Proof. Let T' be an optimal Steiner Tree for the given instance. Also, let r be the root chosen in the last iteration of algorithm 2 and direct all edges of T' towards r. Let the resulting arcs be $\vec{E}(T')$. Now define x' for each $e \in \vec{E}(G)$ as

$$x'_e = \begin{cases} 1 & \text{if } e \in \vec{E}(T') \\ 0 & \text{otherwise} \end{cases}$$

Then, by definition of x', $\sum_{e \in \vec{E}(G)} c_e x'_e \;=\; \sum_{e \in E(T')} c_e$. Also, because the directed version of T' is an arborescence directed into r and spanning R, x' is feasible for (P). But now if x^* is the optimal solution for (P) and Y^* is the optimal solution for (D) then by strong duality

$$\sum_{e \in \vec{E}(G)} c_e x^*_e \;=\; \sum_{\text{valid } C} Y^*_C$$

However, if γ is the dual solution returned by algorithm 2 then by lemma 5.1 ,γ is feasible for (D'). So define $Y_C = \gamma_C$ for each valid set C. Then Y is feasible for (D). Thus,

$$\sum_{e \in E(T')} c_e \;=\; \sum_{e \in \vec{E}(G)} c_e x'_e \;\geq\; \sum_{e \in \vec{E}(G)} c_e x^*_e \;=\; \sum_{\text{valid } C} Y^*_C \;\geq\; \sum_{\text{valid } C} Y_C$$

Moreover, by lemma 5.2, $\sum_{\text{valid } C} \gamma_C \;+\; \sum_{\text{residual } X} \gamma_X \;=\; \sum_{e \in E(T)} c_e$.
and by definition $\sum_{\text{valid } C} \gamma_C \;=\; \sum_{\text{valid } C} Y_C$
Also, by lemma 5.4, $\sum_{\text{residual } X} \gamma_X \;\leq\; \frac{2}{5} \Big(\sum_{e \in E(T)} c_e \Big)$.
Thus we have,

$$\sum_{e \in E(T)} c_e \;=\; \sum_{\text{valid } C} \gamma_C \;+\; \sum_{\text{residual } X} \gamma_X \;\leq\; \sum_{\text{valid } C} \gamma_K \;+ \frac{2}{5} \Big(\sum_{e \in E(T)} c_e \Big)$$
$$\implies \sum_{e \in E(T)} c_e \;\leq\; \frac{5}{3} \Big(\sum_{\text{valid } C} \gamma_C \Big)$$

□

Note that in this proof we showed if we restrict γ to valid sets then it is feasible for (D). As any solution to (D) gives a lower bound on the cost of an optimal Steiner Tree, we now have the desired approximation factor. In the next chapter, we discuss how to implement the algorithm in polynomial time.

Chapter 6

Algorithmic Guarantees

In this chapter we will show that algorithm 2 terminates. To do this we will introduce an "uncrossing" procedure, which will be applied only when algorithm 1 terminates unsuccessfully. This procedure, called the Pruning Algorithm, will be crucial for our analysis.

6.1 The Pruning Algorithm

We now introduce the pruning algorithm, which is depicted in algorithm 3. In this algorithm, we start by picking a set $Q \subseteq S \backslash I$ and a dual solution γ to (D'). Note that Q is the set selected by algorithm 2 when algorithm 1 terminates unsuccessfully and γ is the dual solution computed by algorithm 1. Then in each iteration we consider outermost loaded dual sets, as in definition 4.5, such that these sets cross at Q' for some $Q' \subseteq Q$. If there are more than one loaded sets crossing at Q', say $K_1, ..., K_\ell$, then an amount of dual λ is subtracted from these sets. This dual is then added to their intersection. In addition, if $R \cup I \cup Q \nsubseteq K_1 \cup ... \cup K_\ell$ then we also add dual to the union of these sets. Note that this cannot increase the dual felt by an arc and so we maintain feasibility in (D'). We repeat this step until there are no loaded dual sets crossing at Q. As output we obtain a dual solution γ' feasible for (D') for $I' := I \cup Q$. Moreover, this solution is $(R \cup I')$-laminar. In this chapter we will prove important properties about this algorithm and will then show how to implement algorithm 2 in polynomial time. Combining this with Theorem 5.5 leads to a $\frac{5}{3} + \epsilon$ approximation factor. We start by showing that the pruning algorithm is well defined.

Observation 6.1. *Let γ, I, and Q be defined as above, and $I' = I \cup Q$. If γ is not $(R \cup I')$-laminar then we can find outermost loaded dual sets $K_1, ..., K_\ell$ crossing at some vertex of Q such that for each $u \in Q$ either:*

a. $u \in K_i$ *for each* $i \in \{1, ..., \ell\}$

b. $u \notin K_i$ *for each* $i \in \{1, ..., \ell\}$

c. $u \in K_i$ *for exactly one* $i \in \{1, ..., \ell\}$

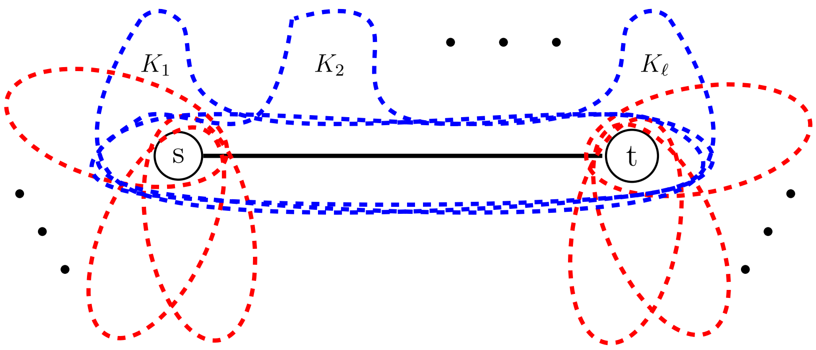
Proof. If $|Q| = 1$ then the result clearly holds. So suppose instead that $Q = \{s, t\}$ and consider an arbitrary iteration of the pruning algorithm. First suppose there are loaded sets $K_1, ..., K_\ell$ crossing, that contain both s and t. This is shown in figure

Algorithm 3: Pruning Algorithm

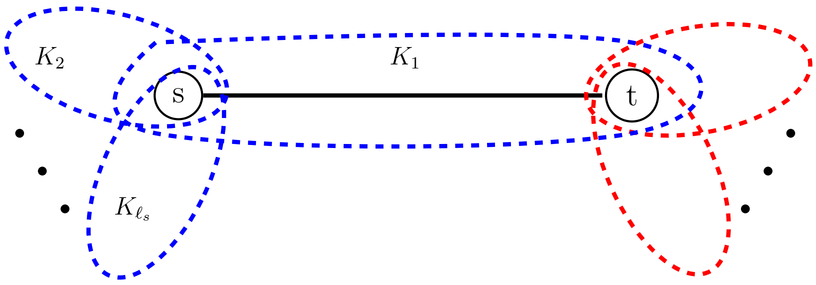
```
• Input: A dual solution $\gamma$ computed by algorithm 1 which terminated
  unsuccessfully for some sets $R$ and $I$, and tight arc $ab$ with $a \in R \cup I$ and
  $b \in S \setminus I$
• Let $Q \subseteq S \setminus I$ be the vertices $v$ for which there is a tight $va$-path in $G[S \setminus I]$
• Set $I' = I \cup Q$
• while $\exists$ loaded dual sets crossing at some vertex of $Q$
    1. Let $K_1, ..., K_\ell$ be outermost loaded dual sets crossing at some $Q' \subseteq Q$
       such that for each $u \in Q$ either:
         (a) $u \in K_i$ for each $i \in \{1, ..., \ell\}$
         (b) $u \in K_i$ for exactly one $i \in \{1, ..., \ell\}$
         (c) $u \notin K_i$ for each $i \in \{1, ..., \ell\}$
    2. Set $\lambda = \min\{\gamma_{K_i} \mid i \in \{1, ..., \ell\}\}$
    3. Set $\gamma_{K_i} = \gamma_{K_i} - \lambda$ for each $i \in \{1, ..., \ell\}$
    4. Set $\gamma_{\bigcap_{i=1}^{\ell} K_i} = \gamma_{\bigcap_{i=1}^{\ell} K_i} + (\ell - 1)\lambda$
    5. If $R \cup I' \subseteq K_1 \cup ... \cup K_\ell$ set$\gamma_{\bigcup_{i=1}^{\ell} K_i} = \gamma_{\bigcup_{i=1}^{\ell} K_i} + \lambda$
```

6.1(a) where the dual sets are outermost loaded sets. By selecting all outermost loaded dual sets containing both s and t gives dual sets satisfying the desired properties.
Otherwise, suppose there are outermost loaded sets $K_1, ..., K_{\ell_s}$ crossing at s. Because there are no crossing loaded sets that contain both s and t, at most one of $K_1, ..., K_{\ell_s}$ contain t. This is shown in figure 6.1(b) where the dual sets are the corresponding outermost sets. Thus, if we consider all outermost loaded dual sets that cross at s then these sets satisfy the conditions. Similarly, if there are outermost loaded sets $K_1, ..., K_{\ell_t}$ crossing at t, they also satisfy the conditions.
Thus, in each iteration, we can find the desired dual sets. □

Although we do not actually apply this algorithm when finding our approximate Steiner tree, it is worthwhile to show that the algorithm terminates.
Consider an arbitrary iteration of the pruning algorithm where we pick loaded dual sets $K_1, ..., K_\ell$ crossing at some $Q' \subseteq Q$. In this iteration, γ_{K_i} is decreased to 0 for at least one $i \in \{1, ..., \ell\}$. So the total number of sets crossing at Q' decreases. Also, dual is added to the union and intersection of $K_1, ..., K_\ell$. Thus, by conditions a, b, and c of the pruning algorithm, for any vertex $u \in Q$, the sum of dual values for sets crossing at u does not increase. In addition, note that $K_1 \cup ... \cup K_\ell$ could cross at some nodes of $Q \setminus Q'$. But in this case, at least one of $K_1, ..., K_\ell$ cross at such a node. Thus for each vertex of $Q \setminus Q'$, the total number of outermost loaded dual sets crossing at this vertex does not increase. Also, as we decrease γ_{K_i} by λ for each $i \in \{1, ..., \ell\}$ and then increase $\gamma_{\bigcup_{i=1}^{\ell} K_i}$ by λ, we have that the sum of dual values crossing at such nodes has not increased.
Moreover, loaded dual sets form a $R \cup I$-laminar family. Thus there are at most



(a) $K_1, ..., K_\ell$ cross at s and t



(b) $K_1, ..., K_{\ell_s}$ cross at s

Figure 6.1: Possible Crossings During The Pruning Algorithm

$2|R \cup I| \leq 2|V|$ loaded dual sets (see for example [6]). Thus, the pruning algorithm does in fact terminate and can be implemented in $O(n)$ time.
We now turn to comparing the dual felt by an arc $e \in \vec{E}$ before and after the pruning algorithm.

Lemma 6.2. *For any $e = ab \in \vec{E}(G[R \cup I'])$, after the pruning algorithm, the dual felt by e is at most the dual felt by e before the pruning algorithm.*

Proof. Note that in each iteration of the pruning algorithm we consider outermost loaded dual sets $U_1, ..., U_h$ crossing at some vertex of Q. We set $\lambda = \min\{\gamma_{U_i} \mid i \in \{1, ..., h\}\}$ and update dual values. By lemma 3.4, these sets are $R \cup I$-disjoint. So for $v \in R \cup I$, either $v \notin U_i$ for each $i \in \{1, ..., h\}$ or $v \in U_i$ for exactly one $i \in \{1, ..., h\}$. Also, by observation 6.1, for any $u \in Q$, either u is contained in all of $U_1, ..., U_h$, none of $U_1, ..., U_h$, or exactly one of $U_1, ..., U_h$.
So first suppose that $a \in U_i$ for each $i \in \{1, ..., h\}$, so $a \in Q$.
If $b \in U_i$ for each $i \in \{1, ..., h\}$ then e does not feel dual from any of the sets updated in this iteration.
If b is contained in one of $U_1, ..., U_h$, say U_1 then e feels a decrease of $(h-1)\lambda$ from decreasing $\gamma_{U_2}, ..., \gamma_{U_h}$. It feels an increase of $(h-1)\lambda$ from increasing $\gamma_{\bigcap_{i=1}^h U_i}$ and e does not feel any dual from $\gamma_{\bigcup_{i=1}^h U_i}$.
If $b \notin U_i$ for each $i \in \{1, ..., h\}$ then e feels the decrease of $h\lambda$ from decreasing $\gamma_{U_1}, ..., \gamma_{U_h}$. It also feels the increase of λ from increasing $\gamma_{\bigcup_{i=1}^h U_i}$ and $(h-1)\lambda$ from increasing $\gamma_{\bigcap_{i=1}^h U_i}$.
So the total dual felt by e is unchanged in this case.
Next suppose that a is contained in exactly one of $U_1, ..., U_h$, say U_1.

If $b \in U_i$ for each $i \in \{1, ..., h\}$ then e does not feel dual from any sets updated in this iteration. If b is in exactly one of $U_1, ..., U_h$, then if $b \in U_1$, e does not feel dual from any sets updated in this iteration. So suppose instead that $b \in U_2$. Then e feels a decrease of λ from decreasing γ_{U_1}. e does not feel dual from $\gamma_{\bigcup_{i=1}^h U_i}$ or from $\gamma_{\bigcap_{i=1}^h U_i}$. If $b \notin U_i$ for each $i \in \{1, ..., h\}$, then e feels a decrease of λ from decreasing γ_{U_1}. It feels an increase of λ from increasing $\gamma_{\bigcup_{i=1}^h U_i}$ and e does not feel any dual from $\gamma_{\bigcap_{i=1}^h U_i}$.
Thus in this case the dual felt by e does not increase.
Lastly suppose $a \notin U_i$ for each $i \in \{1, ..., h\}$. Then e does not feel any dual from dual sets updated in this iteration. Thus, the dual felt by e is unchanged.
So in all cases, the dual felt by e does not increase during the pruning algorithm. □

Note that in the proof of lemma 6.2, the only case in which the dual felt by the arc ab decreases is if both a and b are contained in exactly one of the loaded dual sets that cross. Moreover, a and b must be in different loaded dual sets. However, note that $\emptyset \neq K_1 \cap ... \cap K_\ell \subseteq Q$. Moreover, G is 2-Steiner, so for any such arc $|\{a, b\} \cap Q| \leq 1$. Thus any arcs with both ends in Q remain tight during the pruning algorithm. By theorem 4.8, Q is strongly connected, with respect to tight edges, when the pruning algorithm begins and so now we have that this property holds when the pruning algorithm terminates. We can now show the following.

Observation 6.3. *Let $w \in R \cup I$. Then there is at least one vertex $u \in Q$ such that whenever w is contained in a loaded dual set K crossing at Q, $u \in K$.*

Proof. Note that $|Q| \in \{1, 2\}$. Clearly, the result holds if $|Q| = 1$. Similarly, if there is no loaded dual set containing w and a node of Q then the result also holds. So consider the case that $Q = \{s, t\}$. Then note that by lemma 4.2, the loaded dual sets form a $(R \cup I)$-laminar family and this property is maintained throughout the pruning algorithm. So we cannot have two loaded dual sets, K_1 and K_2, such that $s, w \in K_1$, $t \notin K_1$ and $t, w \in K_2$, $s \notin K_2$ because such sets cross at w. So without loss of generality, suppose there is a loaded dual set containing s and w but not t. Thus for any loaded dual set containing w and t, this set also contains s. The result follows. □

We will also use the following observation.

Observation 6.4. *Suppose $Q = \{s, t\}$. Then there is a vertex $w \in R \cup I$ such that ws became tight at the same time or before there was a tight path from w to t during algorithm 1. Similarly, there is a vertex $v \in R \cup I$ such that vt became tight at the same time or before there was a tight path from v to s during algorithm 1.*

Proof. Suppose as a contradiction that for each $w' \in R \cup I$ such that $w's$ is tight, there was a tight path from w' to t before the arc $w's$ became tight. Note that $s, t \in S \setminus I$, so that if K is a loaded dual set containing s then $t \in K$. But then there are no loaded dual sets K such that $st \in \delta^{out}(K)$ so that the arc st never becomes tight. So now if we consider the graph of tight edges, Q is not strongly connected. This contradicts theorem 4.8.
A similar argument shows that the result also holds for t. □

The next result shows that dual symmetry holds when the pruning algorithm completes.

Lemma 6.5. *Throughout the pruning algorithm, dual symmetry holds for $R \cup I'$.*

Proof. Let $u \in R \cup I'$. First suppose that $u \in R \cup I$. Note that in each iteration of the pruning algorithm we consider dual sets $K_1, ..., K_\ell$ that cross at some vertex of Q, with $\ell \geq 2$. We set $\lambda = \min\{\gamma_{K_i} \mid i \in \{1, ..., \ell\}\}$ and then we update dual values. As $u \in R \cup I$, by lemma 4.3 we have that u is contained in at most one of $K_1, ..., K_\ell$. If u is not contained in any of $K_1, ..., K_\ell$, then the total dual around u is unchanged in this iteration. Otherwise, suppose $u \in K_1$. Then $u \notin K_2 \cup ... \cup K_\ell$. So the dual around u decreases by λ when we subtract λ from γ_{K_1}. Also, by assumption, $u \notin \bigcap_{i=1}^{\ell} K_i$ and so the total dual around u is unaffected when updating dual variable $\gamma_{\bigcap_{i=1}^{\ell} K_i}$. Lastly, if $K_1 \cup ... \cup K_\ell \subset R \cup I'$ then the dual around u increases by λ when we add λ to $\gamma_{\bigcup_{i=1}^{\ell} K_i}$. Otherwise, $R \cup I' \subseteq K_1 \cup ... \cup K_\ell$. But in this case the total dual around each node of $R \cup I$ decreases by λ. Thus the total dual around u is unchanged in this iteration.
It thus follows that in all iterations of the pruning algorithm, the total dual around u is either unchanged or is changed by the same amount for all nodes of $R \cup I$. This holds for all vertices of $R \cup I$. However, we already showed that dual symmetry holds for $R \cup I$ before the pruning algorithm begins. Thus, dual symmetry holds for $R \cup I$ throughout the pruning algorithm.
So now suppose that $u \in Q$. Now choose the vertex $w \in R \cup I$ according to observation 6.4. That is, the arc wu was the first tight path from w to Q. Thus, whenever w is contained in a loaded dual set crossing at Q, u is also contained in this dual set. But now observe that the dual around w can be expressed as

$$\sum_{K: w \in K} \gamma_K = \sum_{K: w \in K, u \notin K} \gamma_K + \sum_{K: u, w \in K} \gamma_K$$

But the arc wu is tight when the pruning algorithm begins so that

$$\sum_{K: w \in K, u \notin K} \gamma_K = c_{wu} \tag{6.1}$$

Also, from the proof of lemma 6.2, this arc remains tight throughout the pruning algorithm so that this quantity stays constant.
Next note that the dual around u can be expressed as

$$\sum_{K: u \in K} \gamma_K = \sum_{K: u \in K, w \notin K} \gamma_K + \sum_{K: u, w \in K} \gamma_K$$

However, by theorem 4.8, the arc uw is tight when the pruning algorithm begins. Also, by lemma 6.2, this arc remains tight throughout the pruning algorithm. So consider an arbitrary point during the pruning algorithm, then we have

$$\sum_{K: u \in K, w \notin K} \gamma_K = c_{uw} \tag{6.2}$$

Thus at this point in the pruning algorithm we have

$$\begin{aligned}\sum_{K:w\in K}\gamma_K &= \sum_{K:w\in K, u\notin K}\gamma_K + \sum_{K:u,w\in K}\gamma_K\\ &= c_{uw} + \sum_{K:u,w\in K}\gamma_K\\ &= \sum_{K:u\in K, w\notin K}\gamma_K + \sum_{K:u,w\in K}\gamma_K\\ &= \sum_{K:u\in K}\gamma_K\end{aligned}$$

Where the second line follows from (6.1) and the third line follows from (6.2). Thus, the dual around w is the same as the dual around u. But w was a vertex of $R\cup I$ and u was an arbitrary vertex of Q. Moreover, we showed that dual symmetry held for $R\cup I$. Thus, dual symmetry holds for $R\cup I'$ throughout the pruning algorithm. $\square$

Now using this result, we can show the importance of the pruning algorithm. Specifically, our next result shows that Q is used to join multiple strongly connected components.

Lemma 6.6. *Let K be a loaded dual set during the pruning algorithm. Then for any $u, v \in K\cap(R\cup I')$ there is a tight uv-path using vertices of K. In particular, this property holds when the pruning algorithm terminates*

Proof. Consider an arbitrary point during the pruning algorithm. First consider the case that $K\cap Q=\emptyset$. Then K was a loaded dual set prior to the pruning algorithm. Moreover, by lemma 4.3, the loaded dual sets form a $(R\cup I)$-laminar family. Now notice that during the pruning algorithm we only consider loaded dual sets crossing at some vertex of Q. Thus if ab is an arc with $a, b\in K\cap(R\cup I)$ then the dual felt by ab is unchanged throughout the pruning algorithm. But now by lemma 4.2, there was a tight uv-path in K before the pruning algorithm. Moreover, as $K\cap Q=\emptyset$ this path does not use any vertex of $S\setminus I$, by the termination condition of algorithm 1. Thus, this path remains tight throughout the pruning algorithm.
So suppose instead that $K\cap Q\neq\emptyset$. Note that by the proof of theorem 4.8, for $a, b\in Q$, there is a tight ab-path in Q when the pruning algorithm initializes. Moreover, by the proof of lemma 6.2, this path remains tight throughout the pruning algorithm. Thus Q is strongly connected, with respect to tight arc, throughout the pruning algorithm. Thus, it suffices to show that there is a tight path in K from u to a vertex of $K\cap Q$ and a tight path in K from a vertex of $K\cap Q$ to v.
So we start by showing there is a tight path from u to a vertex of $K\cap Q$. If $u\in Q$ then we are done. Otherwise, by observation 6.3, there is a vertex $w\in Q$ such that whenever u is contained in a loaded set K' crossing at Q, $w\in K'$. Also, by lemma 4.2, there is a tight path from u to w contained in K'. By lemma 6.2, this path remains tight throughout the pruning algorithm.
Now we want to show there is a tight path from a vertex of $K\cap Q$ to v. But again, $v\in K$ and so there is a tight path P' from v to some vertex $t\in K\cap Q$. As shown above, we can choose this path so that P' remains tight throughout the pruning algorithm. So now by lemma 6.5 we have that dual symmetry holds for $R\cup I'$ throughout the pruning algorithm. Thus, by corollary 2.10, path symmetry holds

for P'. That is $\overline{P}'$ is a tight path.
So now we have a tight path from u to w and a tight path from t to v for $t, w \in C \cap Q$. But $|Q| \leq 2$. So either $t = w$ or $C \cap Q = Q$. But Q is strongly connected with respect to tight arcs. So in either case we have a tight path from w to t. Thus, we have a tight path from u to v. □

Corollary 6.7. *The set Q joins multiple strongly connected components of $R \cup I$ in the graph of tight arcs when the pruning algorithm terminates.*

Proof. Consider an iteration of the pruning algorithm in which we pick outermost loaded dual sets $K_1, ..., K_\ell$ crossing at Q with $\ell \geq 2$. By lemma 6.6, $K_1, ..., K_\ell$ are strongly connected on $R \cup I'$ when restricted to tight arcs. But then in particular, for each node $u \in K_i \cap (R \cup I')$ for $i \in \{1, ..., \ell\}$, there is a tight path from u to $K_1 \cap ... \cap K_\ell \subseteq Q$ using vertices of $R \cup I'$. By lemma 6.5 and lemma 4.4, the reverse path is also tight. Thus $K_1 \cup ... \cup K_\ell$ is strongly connected through Q.
Now notice that by observation 4.9, there are multiple outermost loaded dual sets, $K'_1, ..., K'_{\ell'}$, crossing at Q when algorithm 1 terminates unsuccessfully, with $\ell' \geq 3$. Moreover, by lemma 4.2, these loaded dual sets are strongly connected on $R \cup I$ when restricted to tight arcs. By the above analysis and the description of the pruning algorithm, we have that when the pruning algorithm terminates $K'_1 \cup ... \cup K'_\ell \subseteq K \subseteq V$ such that K is strongly connected on $R \cup I'$ when restricted to tight edges. □

6.2 Termination of Algorithm 2

Given the previous results, we now have the tools to show that algorithm 2 terminates. The organization of this section mirrors the approach taken in [25] and all of our results are generalizations.
Suppose that when algorithm 1 terminates unsuccessfully, the current dual solution is γ. We will create two new dual solutions: γ^1 and γ^2.
To generate γ^1 we do the following:

- We project γ to $R \cup I$ as follows:
  - $\widetilde{\gamma}_C = \sum_{C':\ C' \cap (R \cup I) = C} \gamma_{C'}$ $\quad \forall$ proper sets C
  - $\widetilde{\gamma}_X = \gamma_X$ $\quad \forall$ residual X
- Run Algorithm 1 on G$[R \cup I]$ starting from $\widetilde{\gamma}$ to obtain γ^1

We now generate a second dual, γ^2, with the following procedure:

- Apply the pruning algorithm to obtain dual γ' from γ
- Project γ' to $R \cup I'$ as follows:
  - $\bar{\gamma}_C = \sum_{C':\ C' \cap (R \cup I') = C} \gamma'_{C'}$ $\quad \forall$ proper sets C
  - $\bar{\gamma}_X = \gamma'_X$ $\quad \forall$ residual X
- Run Algorithm 1 on G$[R \cup I']$ starting from $\bar{\gamma}$ to obtain γ^2

Definition 6.8. *For a graph H let $MST(H)$ denote the cost of a minimum spanning tree of H.*

The next two results describe the cost of these dual solutions.

Lemma 6.9. $\sum_{\textit{valid } C} \gamma^1_C + \sum_{\textit{residual } X} \gamma^1_X = MST(G[R \cup I])$

Proof. Consider an arc $e = ab$. We will compare the dual felt by e with respect to γ and the dual felt by e with respect to $\widetilde{\gamma}$. There are two cases:

Case 1: $a \in R$.
Then e only feels dual from proper sets and so the total dual felt by e with respect to $\widetilde{\gamma}$ is

$$\sum_{C: e \in \delta^{out}(C)} \widetilde{\gamma}_C = \sum_{C: e \in \delta^{out}(C)} \left[\sum_{C': C' \cap (R \cup I) = C} \gamma_{C'} \right] = \sum_{C': e \in \delta^{out}(C')} \gamma_{C'}$$

Where the last equality follows from lemma 4.3, namely that loaded dual sets form a $(R \cup I)$-laminar family.

Case 2: $a \in I$.
Then e feels dual from proper sets and from residual sets X with $a \in X$, $b \notin X$. So the total dual felt by e with respect to $\widetilde{\gamma}$ is

$$\begin{aligned}\sum_{C: e \in \delta^{out}(C)} \widetilde{\gamma}_C + \sum_{X: e \in \delta^{out}(X)} \widetilde{\gamma}_X &= \sum_{C: e \in \delta^{out}(C)} [\sum_{C': C' \cap (R \cup I) = C} \gamma_{C'}] + \sum_{X: e \in \delta^{out}(X)} \gamma_X \\ &= \sum_{C': e \in C'} \gamma_{C'} + \sum_{X: e \in \delta^{out}(X)} \gamma_X\end{aligned}$$

Where again the last equality follows from lemma 4.3.

Thus in both cases e feels the same amount of dual with respect to γ as it feels with respect to $\widetilde{\gamma}$. Thus after projecting γ to $R \cup I$, all edges that were tight before the projection remain tight and no new edge becomes tight. Thus because path symmetry held with respect to γ, path symmetry holds with respect to $\widetilde{\gamma}$. So by lemma 2.9, dual symmetry holds for $\widetilde{\gamma}$ and our minimally unsatisfied sets are unchanged. Thus we can continue running algorithm 1 with $\widetilde{\gamma}$. Moreover, as we have restricted to the graph $G[R \cup I]$ algorithm 1 will terminate successfully with dual solution γ^1 and tree T^1. Note that γ was feasible for (D') so that $\widetilde{\gamma}$ is feasible for (D'). Now by lemma 5.2, $\sum_{\text{valid } C} \gamma^1_C + \sum_{\text{residual } X} \gamma^1_X = \sum_{e \in E(T^1)} c_e = MST(G[R \cup I])$. □

Lemma 6.10. $\sum_{\textit{valid } C} \gamma^2_C + \sum_{\textit{residual } X} \gamma^2_X = MST(G[R \cup I'])$

Proof. The pruning algorithm was applied to dual solution γ because arc uv became tight with $u \in S \setminus I$ and $v \in R \cup I$. After the pruning algorithm, γ' was the new dual solution. By lemma 6.5, dual symmetry holds for $R \cup I'$ when the pruning algorithm completes. Thus by corollary 2.10, path symmetry holds for any path connecting two vertices of $R \cup I'$.

Once the pruning algorithm completes we project γ' to a new dual, $\bar{\gamma}$. So consider an arc $e = ab \in \vec{E}(G[R \cup I'])$. We will compare the dual felt by e with respect to γ' and the dual felt by e with respect to $\bar{\gamma}$. There are two cases:

<u>Case 1:</u> $a \in R$.
Then e only feels dual from proper sets. So the total dual felt by e with respect to $\bar{\gamma}$ is

$$\sum_{C: e \in \delta^{out}(C)} \bar{\gamma}_C = \sum_{C: e \in \delta^{out}(C)} \Big[\sum_{C': C' \cap (R \cup I') = C} \gamma'_{C'} \Big] = \sum_{C': e \in \delta^{out}(C')} \gamma'_{C'}$$

Where the last equality follows from lemma 4.3.

<u>Case 2:</u> $a \in I$. Then e feels dual from proper sets and from residual sets X with $a \in X$, $b \notin X$. So the total dual felt by e with respect to $\bar{\gamma}$ is

$$\begin{aligned}\sum_{C: e \in \delta^{out}(C)} \bar{\gamma}_C + \sum_{X: e \in \delta^{out}(X)} \bar{\gamma}_X &= \sum_{C: e \in \delta^{out}(C)} \Big[\sum_{C': C' \cap (R \cup I') = C} \gamma'_{C'} \Big] + \sum_{X: e \in \delta^{out}(X)} \gamma'_X \\ &= \sum_{C': e \in \delta^{out}(C')} \gamma'_{C'} + \sum_{X: e \in \delta^{out}(X)} \gamma'_X\end{aligned}$$

Where the last equality follows from lemma 4.3.

Thus in both cases e feels the same amount of dual with respect to γ' as it feels with respect to $\bar{\gamma}$. So the minimally unsatisfied sets with respect to γ' and with respect to $\bar{\gamma}$ are the same. Moreover, for any path connecting two vertices of $R \cup I'$, path symmetry held for γ' and so path symmetry holds for $\bar{\gamma}$. So by corollary 2.10, the dual $\bar{\gamma}$ is symmetric. Thus we can continue running algorithm 1 with $\bar{\gamma}$. Also, because γ' was feasible for (D'), $\bar{\gamma}$ is feasible for (D'). Moreover, as we have restricted to the graph $G[R \cup I']$, algorithm 1 will terminate successfully with dual solution γ^2 and tree T^2. Now by lemma 5.2,

$$\sum_{\text{valid } C} \gamma^2_C + \sum_{\text{residual } X} \gamma^2_X = \sum_{e \in E(T^2)} c_e = MST(G[R \cup I \cup Q])$$

□

With these two results we can now show that algorithm 2 does terminate.

Theorem 6.11. $MST(G[R \cup I']) < MST(G[R \cup I])$

Proof. When algorithm 1 terminates unsuccessfully, some arc from a vertex of $S \setminus I$ to $R \cup I$ became tight. Let γ be the dual values at this point. The pruning algorithm identifies a set of vertices Q to add to I and then we create two new duals, γ^1 and γ^2 as discussed above. Now we define two quantities β^1 and β^2 as follows:

$$\beta^1 = \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma^1_C + \sum_{\text{residual } X} \gamma^1_X - \sum_{\substack{\text{proper } C: \\ r \notin C}} \tilde{\gamma}_C - \sum_{\text{residual } X} \tilde{\gamma}_X$$

$$\beta^2 = \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma^2_C + \sum_{\text{residual } X} \gamma^2_X - \sum_{\substack{\text{proper } C: \\ r \notin C}} \bar{\gamma}_C - \sum_{\text{residual } X} \bar{\gamma}_X$$

That is, β^1 is the total increase in dual sets (for sets not containing r) from running algorithm 1 on $\widetilde{\gamma}$ to obtain γ^1. Similarly, β^2 is the total increase in dual sets (for sets not containing r) from running algorithm 1 on $\bar{\gamma}$ to obtain γ^2. However, note that during the projection steps if $C' \subseteq V$ is a loaded set then C' is mapped to a unique set in $R \cup I$ and a unique set in $R \cup I'$. Thus the projection step does not change the sum of the dual variables. So we have

$$\sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma^1_C + \sum_{\text{residual } X} \gamma^1_X = \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma_C + \sum_{\text{residual } X} \gamma_X + \beta^1$$
$$\sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma^2_C + \sum_{\text{residual } X} \gamma^2_X = \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma'_C + \sum_{\text{residual } X} \gamma'_X + \beta^2$$

So now consider the quantity

$$\sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma_C + \sum_{\text{residual } X} \gamma_X \tag{6.3}$$

and consider an iteration of the pruning algorithm. We select outermost loaded dual sets $K_1, ..., K_\ell$ that cross at some $Q' \subseteq Q$. We compute $\lambda = \min\{\gamma_{K_i} \mid i \in \{1, ..., \ell\}\}$ and then we update dual values.
First suppose that $r \notin K_1 \cup ... \cup K_\ell$. Then (6.3) is decreased by $\ell\lambda$ from decreasing γ_{K_i} for $i \in \{1, ..., \ell\}$. It is also increased by $(\ell - 1)\lambda$ when $\gamma_{\bigcap_{i=1}^{\ell} K_i}$ is increased, and is increased by λ when $\gamma_{\bigcup_{i=1}^{\ell} K_i}$ is increased.
Otherwise $r \in K_1 \cup ... \cup K_\ell$. But $K_1, ..., K_\ell$ are outermost loaded sets so by lemma 4.3, $r \in K_j$ for exactly one $j \in \{1, ..., \ell\}$ (assume $j = 1$). Then (6.3) is decreased by $(\ell - 1)\lambda$ from decreasing γ_{K_i} for $i \in \{2, ..., \ell\}$. It is also increased by $(\ell - 1)\lambda$ when $\gamma_{\bigcap_{i=1}^{\ell} K_i}$ is increased. Moreover, as $r \in K_1 \cup ... \cup K_\ell$, (6.3) is unchanged from increasing $\gamma_{\bigcup_{i=1}^{\ell} K_i}$.
Thus (6.3) is invariant throughout the pruning algorithm and so

$$\sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma_C + \sum_{\text{residual } X} \gamma_X = \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma'_C + \sum_{\text{residual } X} \gamma'_X$$

Now note that by observation 4.9 and corollary 6.7, by applying the pruning algorithm we have connected at least three components via Q. Thus when algorithm 1 starts on $\bar{\gamma}$ there are fewer components than when algorithm 1 starts on $\widetilde{\gamma}$. Moreover, from the proof of lemma 6.2, any arc leaving a minimally unsatisfied set is feeling the same dual after the pruning algorithm. Thus at all times, when we run algorithm 1 on $\bar{\gamma}$ there are at most as many components as when we run algorithm 1 on $\widetilde{\gamma}$ and initially there are fewer when we start algorithm 1 on $\bar{\gamma}$. By corollary 6.7, Q joins multiple components of $R \cup I$ and by our priority rule in algorithm 1, there are no tight arcs in the cut induced by these components. Thus $\beta^1 > \beta^2$ in both cases. Also note that if $C \subseteq V$ is a proper set with $r \notin C$ then C is a valid set, by definitions 2.1 and 2.3. So,

$$\begin{aligned}
MST(G[R \cup I]) = \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma_C^1 + \sum_{\text{residual } X} \gamma_X^1 &= \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma_C + \sum_{\text{residual } X} \gamma_X + \beta^1 \\
&= \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma'_C + \sum_{\text{residual } X} \gamma'_X + \beta^1 \\
&> \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma'_C + \sum_{\text{residual } X} \gamma'_X + \beta^2 \\
&= \sum_{\substack{\text{proper } C: \\ r \notin C}} \gamma_C^2 + \sum_{\text{residual } X} \gamma_X^2 \\
&= MST(G[R \cup I'])
\end{aligned}$$

Where the first equality follows from lemma 6.9 and the last follows from lemma 6.10. □

Note that if algorithm 1 terminates successfully with set $I \subseteq S$ then by lemma 5.2, T is a minimum spanning tree of $R \cup I$. So by theorem 6.11, by adding Q to I we have decreased the cost of our final Steiner tree. As all edge costs are finite, algorithm 2 will terminate. While this is not necessarily a polytime algorithm, we have proved the following.

Theorem 6.12. *The integrality gap of (BCR) is at most $\frac{5}{3}$ for 2-Steiner graphs.*

6.3 A Polynomial Time Implementation

To obtain a polytime algorithm we will round c_e for each edge $e \in E$. This will cause us to lose a factor of ϵ but will guarantee that algorithm 2 runs in polynomial time. We first need the following well known result, which can be found in [15]. We present the equally well known proof for completeness.

Lemma 6.13. $MST(G[R]) \leq 2OPT$

Proof. Let T be an optimal Steiner tree of G so that $c(T) = OPT$.
Double each edge of T to obtain an Eulerian multigraph H. As T spans R, H spans R. Moreover, $c(H) = 2c(T)$.
Now we can shortcut H to obtain a cycle C with $V(C) = R$ and $c(C) \leq c(H)$. By deleting any edge of C we obtain a tree $\overline{T}$ that spans R and so we have

$$MST(G[R]) \leq c(\overline{T}) \leq c(C) \leq c(H) = 2c(T) = 2OPT$$

□

Now with this lemma we are ready to discuss the polytime implementation of algorithm 2.
Note that if $OPT = 0$ then by lemma 6.13, $MST(G[R]) = 0$ and so a minimum

spanning tree on R gives an optimal solution. So we may assume $MST(G[R]) > 0$ and thus $OPT > 0$ Pick a small $\epsilon > 0$ and define

$$\beta = \frac{(\epsilon)MST(G[R])}{2(|V|-1)}$$

Now define $c' \in \mathbb{R}_+^E$ for each $e \in E$ as $c'_e = k\beta$ for the smallest $k \in \mathbb{Z}$ such that $k\beta \geq c_e$
That is, c' is obtained from c by rounding up each coordinate to the closest integer multiple of β.
Now for $u, v, w \in V$ we have $c'_{uv} = k_{uv}\beta$, $c'_{uw} = k_{uw}\beta$, and $c'_{wv} = k_{wv}\beta$. But the triangle inequality holds for c so that $c_{uv} \leq c_{uw} + c_{wv}$.
By definition we have $k_{uv} = \left\lceil \frac{c_{uv}}{\beta} \right\rceil$, $k_{uw} = \left\lceil \frac{c_{uw}}{\beta} \right\rceil$, and $k_{wv} = \left\lceil \frac{c_{wv}}{\beta} \right\rceil$
Thus $k_{uv} \leq k_{uw} + k_{wv}$ and so $c'_{uv} \leq c'_{uw} + c'_{wv}$. That is, c' is also a metric. Now we can apply algorithm 2 to G with metric costs c'. The following appears in [25] but we present the proof here for completeness.

Lemma 6.14. *If T is an optimal Steiner tree of G then $c'(T) \leq (1+\epsilon)c(T)$*

Proof. Observe that T has at most $|V|-1$ edges. Also for $e \in E$, $c'_e \leq c_e + \beta$. In addition, because T is optimal, $c(T) = OPT$. So,

$$\begin{aligned} c'(T) \leq c(T) + (|V|-1)\beta &= c(T) + (|V|-1)\frac{(\epsilon)MST(G[R])}{2(|V|-1)} \\ &\leq OPT + \epsilon\frac{2OPT}{2} \\ &= (1+\epsilon)OPT \\ &= (1+\epsilon)c(T) \end{aligned}$$

where the second line follows from lemma 6.13 □

Note that we are assuming $OPT > 0$ and so the tree returned by algorithm 2 has cost at least $\beta > 0$. Now as shown above, in each iteration of algorithm 2 we decrease the cost of our Steiner tree. But this decrease is at least β. Moreover, we start with $I = \emptyset$ so that our tree has cost $MST(G[R]) \leq 2OPT$. Also, clearly our final tree has cost at least OPT so there are at most $\frac{MST(G[R])}{\beta} = \frac{2(|V|-1)}{\epsilon}$ iterations. We can now prove the following.

Theorem 6.15. *Given a 2-Steiner graph $H = (V, E_H)$ with cost function $c_H : E_H \to \mathbb{R}_+$, there exists a $(\frac{5}{3}+\epsilon)$-approximation algorithm for the Steiner tree problem for any $\epsilon > 0$.*

Proof. Set $\epsilon' = \frac{3}{5}\epsilon$. Let G be the metric completion of H with edge costs c. Now define $\beta = \frac{(\epsilon')MST(G[R])}{2(|V|-1)}$ and let c' be the edge costs obtained from c by rounding c_e up to the closest integer multiple of β for each $e \in E$. Now we run algorithm 2 on G with edge costs c' to obtain a Steiner tree T. By Theorem 5.5 and lemma 6.14 we have,

$$\sum_{e \in E(T)} c_e \leq (\frac{5}{3})(1+\epsilon')OPT = (\frac{5}{3}+\epsilon)OPT$$

Now consider an iteration of algorithm 2. Let $n = |V|$ and $m = |E|$. Note that if algorithm 1 terminates unsuccessfully then Q can be computed in $O(m)$ time. If instead, algorithm 1 terminates successfully then the minimum degree vertex of I can also be found in $O(m)$ time. So we turn to the analysis of algorithm 1.
Both the initialization and deletion phases can be implemented in polynomial time. Moreover, by corollary 3.5, each vertex of $R \cup I$ is contained in exactly one minimally unsatisfied set. Thus we can store components in a union-find structure, see [21], which can be implemented in time $O(nlogn)$.
To determine tight edges in the symmetric phase, we implement a priority queue of edges, using a notion of time. This involves sorting the edge weights once in time $O(mlogm)$ and keeping track of how many minimally unsatisfied sets each arc intersects. This can be implemented in $O(mlogn)$ time.
Thus, each iteration of algorithm 2 requires $O(mlogn)$ time. Moreover, by our above analysis, there are $O(\frac{n}{\epsilon})$ iterations. So after accounting for the edge sorting, we have a total runtime of $O(\frac{1}{\epsilon}nmlogn + mlogm)$. Combining this with the above approximation factor gives the desired result. $\square$

Chapter 7

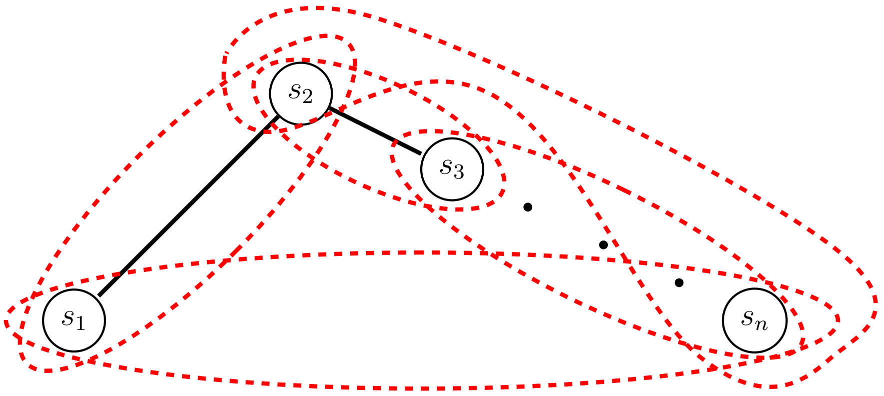
Concluding Remarks

The Steiner tree problem is a fundamental network design problem, which has received much attention in the literature. A major open question regarding this problem is proving that the integrality gap of the bidirected cut relaxation (BCR) is strictly smaller than 2.

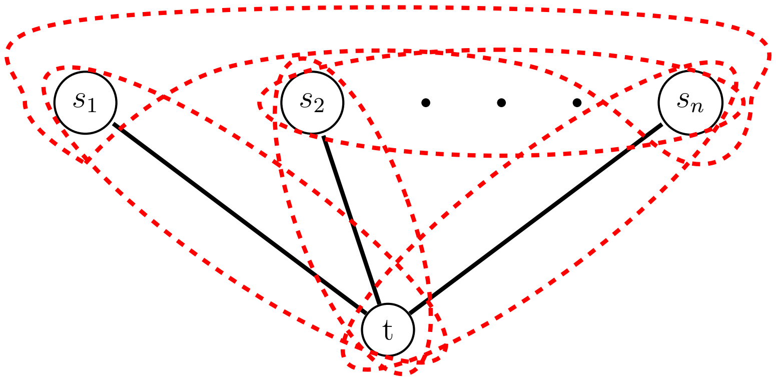
In this thesis, we have proven that (BCR) has integrality gap at most $\frac{5}{3}$ for a subset of instances, which we denote as 2-Steiner instances. We do so via a primal-dual approach. Our algorithm mimics the approach given in [25] for quasi-biartite instances, but its proof of correctness requires new ingredients in the form of lemmas 4.7, 6.2, 6.5, and 6.6, as well as observation 6.1 and observation 4.8. Although our ratio is strictly worse than the one achieved in [9], our result uses a primal-dual algorithm which is directly applied on (BCR), rather than mapping solutions to a different (hypergraphic) relaxation, as is done in [9]. This is nice both from a combinatorial and a practical perspective. Primal-dual algorithms exploit the structure of an LP and of the underlying problem. In particular, we believe that our method can be further generalized to tackle instances containing Steiner claws. Specifically, it may be possible to apply our techniques to instances where the set of Steiner vertices induces a graph in which each component is a star. In the appendix we provide the generalization of some of our results to this case.

We now end with some suggestions for areas of future work. As already mentioned, a first direction of future work is to generalize our primal-dual algorithm to instances where Steiner vertices induce stars (and in particular, claws). In order to tackle more general instances, a second direction is to find generalizations for observation 6.1 as well as lemmas 6.5 and 6.6. In the current proofs we have developed an uncrossing algorithm that ensures any loaded dual set is strongly connected when the pruning algorithm terminates. As G was 2-Steiner we could guarantee that $|Q| \leq 2$ and so we could use a case analysis to determine how to uncross dual sets. However, in general, these crossing could be arbitrary as two Steiner nodes in the same component could be connected via a path in $R \cup I$. Figure 7.1 offers two examples of crossing patterns that could occur.

However, it is possible that alternative proofs of these lemmas could be more easily generalized to arbitrary graphs. This would then give researchers another method to analyze the integrality gap of (BCR) in other instances.



(a) Crossing Sets For A Path



(b) Crossing Sets For A Star

Figure 7.1: Crossings In Other Steiner Components

Bibliography

[1] J. Byrka, F. Grandoni, T. Rothvoss, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, Feb. 2013.

[2] D. Chakrabarty, J. Könemann, and D. Pritchard. Hypergraphic lp relaxations for steiner trees. In *IPCO*, pages 383–396. Springer, 2010.

[3] M. Chlebık and J. Chlebıková. Approximation hardness of the steiner tree problem on graphs. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 170–179. Springer, 2002.

[4] S. Cook. The p versus np problem. In *Clay Mathematical Institute; The Millennium Prize Problem*, 2000.

[5] D.-Z. Du and P. M. Pardalos. *Network optimization problems: algorithms, applications and complexity*, volume 2. World Scientific, 1993.

[6] P. J. Dukes. Generalized laminar families and certain forbidden matrices. *Order*, 32(3):401–408, 2015.

[7] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.

[8] J. Edmonds. Optimum branchings. *Mathematics and the Decision Sciences, Part*, 1:335–345, 1968.

[9] A. E. Feldmann, J. Könemann, N. Olver, and L. Sanità. On the equivalence of the bidirected and hypergraphic relaxations for steiner tree. *Math. Program.*, 160(1-2):379–406, 2016.

[10] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[11] L. Fortnow. *The golden ticket: P, NP, and the search for the impossible*. Princeton University Press, 2013.

[12] D. R. Fulkerson. Packing rooted directed cuts in a weighted directed graph. *Mathematical Programming*, 6(1):1–13, 1974.

[13] I. Fung, K. Georgiou, J. Koenemann, and M. Sharpe. Efficient algorithms for solving hypergraphic steiner tree relaxations in quasi-bipartite instances. *arXiv preprint arXiv:1202.5049*, 2012.

[14] M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[15] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[16] M. X. Goemans. Arborescence polytopes for series-parallel graphs. *Discrete Applied Mathematics*, 51(3):277–289, 1994.

[17] M. X. Goemans, N. Olver, T. Rothvoß, and R. Zenklusen. Matroids and integrality gaps for hypergraphic steiner tree relaxations. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1161–1176. ACM, 2012.

[18] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner tree problem*, volume 53. Elsevier, 1992.

[19] A. B. Kahng and G. Robins. A new class of iterative steiner tree heuristics with good performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(7):893–902, 1992.

[20] M. Karpinski and A. Zelikovsky. New approximation algorithms for the steiner tree problems. *Journal of Combinatorial Optimization*, 1(1):47–65, 1997.

[21] J. Kleinberg and E. Tardos. *Algorithm Design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[22] T. Polzin and S. V. Daneshmand. On steiner trees and minimum spanning trees in hypergraphs. *Operations Research Letters*, 31(1):12–20, 2003.

[23] A. Prodon, T. Liebling, and H. Gröflin. Steiner's problem on 2-trees, 1985.

[24] H. J. Prömel and A. Steger. A new approximation algorithm for the steiner tree problem with performance ratio 5/3. *J. Algorithms*, 36(1):89–101, July 2000.

[25] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric steiner tree problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 742–751, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[26] R. Rizzi. On the steiner tree 3/2-approximation for quasi-bipartite graphs. 6, 12 1999.

[27] G. Robins and A. Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM J. Discret. Math.*, 19(1):122–134, May 2005.

[28] M. Schaffers. Network flow design iii. polyhedral characterization of the single source fixed costs problem on series-parallel graphs. Technical report, CORE Discussion Paper, Université Catholique de Louvain (Louvain-la-Neuve, 1991), 1991.

[29] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, New York, NY, USA, 1st edition, 2011.

[30] R. T. Wong. A dual ascent approach for steiner tree problems on a directed graph. *Mathematical programming*, 28(3):271–287, 1984.

[31] A. Z. Zelikovsky. An 11/6-approximation algorithm for the network steiner problem. *Algorithmica*, 9(5):463–470, May 1993.

Appendix

In this section we generalize some of the results from chapters 4 and 5 to another instance of the Steiner Tree problem. Namely, we are given a graph $H = (V, E_H)$ with partition $V = R \cup I$ such that every component of $H[S]$ is a star with at most θ nodes. We will call such graphs θ-star-Steiner. Again, we can consider the metric completion.
We can still run algorithm 2 on such instances. Note that in this setting the proofs of all lemmas in chapter 4 are unchanged, only the proof of theorem 4.8 changes.

A.1 Theorem 4.8 For θ-Star-Steiner Graphs

In this section, we generalize theorem 4.8 to θ-Star-Steiner instances. Consider an iteration of algorithm 2 and suppose that algorithm 1 terminates unsuccessfully because the arc ab became tight where $a \in S \setminus I$ and $b \in R \cup I$. Then we define $Q \subseteq S \setminus I$ to be the set of all vertices $u \in S \setminus I$ such that there is a tight ua-path in $G[S \setminus I]$. As H is θ-Star-Steiner we have that Q is also a star with at most θ vertices. Using the results of chapter 4 we can now show the following,

Theorem A.1. *Suppose algorithm 1 terminates unsuccessfully when arc ab becomes tight for $a \in S \setminus I$, $b \in R \cup I$, and let $Q \subseteq S \setminus I$ be the set determined in algorithm 2. Then for any $c \in R \cup I$, if there is a tight path from c to a vertex of Q that only uses vertices of $R \cup I \cup Q$ then there is a tight path from Q to c that only uses vertices of $R \cup I \cup Q$ when algorithm 1 terminates. Moreover, Q is strongly connected with respect to tight edges.*

Proof. By lemma 4.7, we have that ba is also tight. Note that ab went tight and caused algorithm 1 to terminate unsucessfully. However, $a \in S \setminus I$, so there must be loaded sets crossing at a. Let $v_1, ..., v_n$ be the nodes of $R \cup I$ such that the arc $v_i a$ is tight for each $i \in \{1, ..., n\}$. As the arc ba is tight we have that $b \in \{v_1, ..., v_n\}$ and so $n \geq 1$. After relabelling, we can assume that $b = v_1$.
Now note that for any other loaded set crossing at a that is not identified by one of $v_1, ..., v_n$, such a set must contain a vertex $u \in S \setminus I$ such that $\{a, u\} \in E_H$. Thus either $|Q| = 1$ or $|Q| \geq 2$. So we have two cases:

<u>Case 1:</u> $|Q| = 1$.
Then $Q = \{a\}$ and is clearly strongly connected. In addition, any loaded dual set crossing at a contains one of $v_1, ..., v_n$. But now $b = v_1$ so that the arc av_1 is tight. Thus, there must be additional dual sets crossing at a that load dual for the arc av_1. That is, $n \geq 2$.
We will start by showing that the arc av_i is tight for each $i \in \{2, ..., n\}$. But now observe that arc $v_i a$ is tight, by definition of v_i. So together with the arc av_1 this gives

a tight path from v_i to v_1 through a, call this path P_i. Moreover, $v_1, v_i \in R \cup I$. Thus, by lemma 4.4, $\bar{P}_i$ is also a tight path. Also, as $v_i a \in \vec{E}(P)$, we have that $av_i \in \vec{E}(\bar{P})$ so that the arc av_i is tight.
Thus, for each $i \in \{1, ..., n\}$, the arc av_i is tight. So consider any other vertex $c \in R \cup I$ such that there is a tight path from c to Q using nodes of $R \cup I \cup Q$, let P be such a path. We know that $Q = \{a\}$ and so $V(P) \setminus \{a\} \subseteq R \cup I$. Moreover, $v_1, ..., v_n$ are the only vertices of $R \cup I$ that are joined to a by a tight arc. Thus the last vertex of P before a is v_i for some $i \in \{1, ..., n\}$. So let P' be the subpath of P from c to v_i. Then P' is a tight between two nodes of $R \cup I$ and so by lemma 4.4, $\bar{P}'$ is also tight. Now the arc av_i followed by $\bar{P}'$ gives a tight path from a to c using nodes of $R \cup I \cup Q$. In particular, we have a tight path from Q to c using nodes of $R \cup I \cup Q$.

Case 2: $|Q| \geq 2$
Then $Q = \{a, t_1, ..., t_k\}$ for some $t_1, ..., t_k \in S \setminus I$ with $1 \leq k \leq \theta - 1$. Moreover, we have chosen $v_1, ..., v_n$ to be all nodes of $R \cup I$ that are joined to a by a tight arc. Thus, if K is any other dual set crossing at a then K must contain t_j for some $j \in \{1, ..., k\}$. In addition, by the description of algorithm 2, we have included t_j in Q because there is a tight path from t_j to a in $G[S \setminus I]$. By the choice of Q, there is a tight path from t_j to a in Q, say $P_{t_j a}$. So let $w_1^j, ..., w_{\ell_j}^j \in R \cup I$ be the vertices joined to t_j by a tight arc. We can choose $v_1, ..., v_n, w_1^1, ..., w_{\ell_1}^1, ..., w_1^k, ..., w_{\ell_k}^k$ to all be distinct. Note that $a \in S \setminus I$ and the arc ab caused algorithm 1 to terminate unsuccessfully. As we give priority to arcs with their tail in $R \cup I$, we must have that

$$n + \sum_{j=1}^{k} \ell_j \geq 2$$

We have that $b = v_1$, so now consider v_i for $i \in \{2, ..., n\}$. Then as the arc $v_i a$ is tight we have that $v_i a v_1$ gives a tight $v_i v_1$-path. As in case 1, we now have that $v_1 a v_i$ is also a tight path so that in particular the arc av_i is tight.
Next, consider w_i^j for $i \in \{1, ..., \ell_j\}$ and $j \in \{1, ..., k\}$. By the choice of w_i^j the arc $w_i^j t_j$ is tight. Moreover, $P_{t_j a}$ is a tight path and av_1 is also tight. So let P_i^j be the path defined by arc $w_i^j t$ followed by $P_{t_j a}$ followed by av_1. Then P_i^j is a tight path from w_i^j to v_1. However, $v_1, w_i^j \in R \cup I$. So by lemma 4.4, the path $\bar{P}_i^j$ is also tight. As $w_i^j t \in \vec{E}(P_i^j)$ we have that $tw_i^j \in \vec{E}(\bar{P}_i)$ so that the arc tw_i^j is tight as desired.
In addition, $\bar{P}_{t_j a}$ is a subpath of $\bar{P}_i^j$ so that this is also a tight path. So, for each $w \in Q \setminus \{a\}$, we have a tight path from w to a and a tight path from a to w. Moreover, both paths only use nodes of Q. Thus, Q is strongly connected when restricted to tight edges.
So now consider $c \in R \cup I$ such that there is a tight path P from c to a node in Q with $V(P) \subseteq R \cup I \cup Q$. Traverse P starting from c and let q be the first vertex of P in Q. Set u to be the vertex before q on P. That is, the arc uq is an arc of P. Now let P' be the subpath of P from c to u. By the choice of u we have that $V(P') \subseteq R \cup I$.
Now by definition, $v_1, ..., v_n$ are all vertices joined to a by a tight arc and $w_1^j, ..., w_{\ell_j}^j$ are all vertices joined to t_j by a tight arc for $j \in \{1, ..., k\}$. So either $u = v_i$ for some $i \in \{1, ..., n\}$ or $u = w_i^j$ for some $j \in \{1, ..., k\}$ and $i \in \{1, ..., \ell_j\}$. Thus, as shown above we have that there is a tight arc from a vertex of Q to u. Moreover,

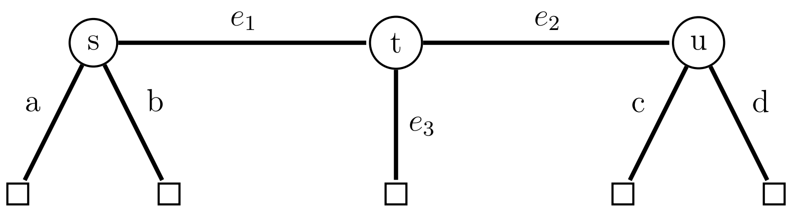


Figure A.1: A 7 edge 3-star

$c, u \in R \cup I$ and P' is a tight cu-path with $V(P') \subseteq R \cup I$. By lemma 4.4, $\bar{P}'$ is also tight. Thus we have a tight path from Q to c contained in $R \cup I \cup Q$.

Thus in both cases we obtain the desired result so that this completes the proof. □

A.2 Approximation Ratio For θ-Star Steiner

In this section, we generalize the results of chapter 5 for θ-Star-Steiner graphs. Note that the proofs of lemmas 5.1 and 5.2 are still valid for these instances. So we are left with generalizing lemma 5.4 and theorem 5.5. However, notice that these results are only valid if algorithm 2 terminates. Thus, these results alone are not sufficient to bound the integrality gap for (BCR).
We will actually derive two separate approximation factors. The first applies to the case when $\theta \leq 3$ and allows us to achieve a slightly better bound in this case. The second case is when $\theta \geq 4$.

Lemma A.2. *If algorithm 2 terminates with tree T and set $I \subseteq S$ then if $\theta \leq 3$ we have*

$$\sum_{\textit{residual } X} \gamma_X \leq \frac{\theta}{2\theta + 1} \sum_{e \in E(T)} c_e$$

Proof. Note that because $\theta \leq 3$ every component of $H[I]$ is a star with at most 3 nodes, and so every component of $H[I]$ is a path with at most 3 nodes. As was shown in lemma 5.4, we now have that every component of $T[I]$ is a path with at most 3 nodes. So let K be a component of $T[I]$ and suppose K has $\phi \leq \theta$ nodes.
First suppose that $\phi \leq 2$. Let $\bar{T}_K$ be the subtree of T consisting of all edges adjacent to a vertex of K. Then by lemma 5.4 we have that

$$\sum_{X : X \cap V(K) \neq \emptyset} \gamma_X \leq \frac{\phi}{2\phi + 1} \Big(\sum_{f \in \bar{T}_K} c_f \Big) \leq \frac{\theta}{2\theta + 1} \Big(\sum_{f \in \bar{T}_K} c_f \Big)$$

Otherwise, $\phi = 3 = \theta$. So K is a 2 edge path with edges $e_1 = \{s, t\}$, and $e_2 = \{t, u\}$. However, $\deg_T(s) \geq 3$ and $\deg_T(u) \geq 3$ so that both s and u are adjacent to at least two vertices of R. Similarly, $\deg_T(t) \geq 3$ so that t is adjacent to at least one vertex of R. So s, t, and u induce a 3-star $\bar{T}_K$. By possibly deleting edges, we can obtain a 3-star $\widetilde{T}_K$ in which all of s, t, and u have degree 3. Moreover, $\delta_{\widetilde{T}_K}(\{s\}) = \{a, b, e_1\}$, $\delta_{\widetilde{T}_K}(\{t\}) = \{e_1, e_2, e_3\}$, and $\delta_{\widetilde{T}_K}(\{u\}) = \{c, d, e_2\}$. $\widetilde{T}_K$ is depicted in figure A.1. Then notice that $\sum_{f \in E(\widetilde{T}_K)} c_f \leq \sum_{f \in E(\bar{T}_K)} c_f$. So first suppose that the maximum cost edge is e_j for $j \in \{1, 2, 3\}$. By lemma 5.1, no edge is overtightened during the execution of algorithm 2. The edges a and b both feel dual from all residual sets X with $s \in X$. So we have $\sum\limits_{X:s\in X} \gamma_X \leq c_a$ and $\sum\limits_{X:s\in X} \gamma_X \leq c_b$. Similarly, the edges c and d feel dual from all residual sets X with $u \in X$ and $s \notin X$. So we have $\sum\limits_{X:u\in X,s\notin X} \gamma_X \leq c_c$ and $\sum\limits_{X:u\in X,s\notin X} \gamma_X \leq c_d$. Lastly, the edges e_1, e_2 and e_3 feel dual from all residual sets X with $t \in X$ and $s, u \notin X$. So in particular, $\sum\limits_{X:t\in X,s,u\notin X} \gamma_X \leq c_{e_i}$ for each $i \in \{1,2,3\} \setminus \{j\}$. Thus we have,

$$\begin{aligned} 2\Big(\sum_{X:\{s,t,u\}\cap X\neq\emptyset} \gamma_X\Big) &= 2\Big(\sum_{X:s\in X} \gamma_X\Big) + 2\Big(\sum_{X:u\in X,s\notin X} \gamma_X\Big) + 2\Big(\sum_{X:t\in X,s,u\notin X} \gamma_X\Big) \\ &\leq c_a + c_b + c_c + c_d + \sum_{\substack{i=1 \\ (i\neq j)}}^{3} c_{e_i} \\ &\leq \frac{6}{7} c(\widetilde{T}_K) \end{aligned}$$

where the last line follows from the fact that $\widetilde{T}_K$ has 7 edges and e_j is the most expensive.

Otherwise, one of a, b, c, or d is the most expensive edge. By symmetry, we can assume that a is the most expensive edge. The edges b and e_1 feel dual from all residual sets X with $s \in X$. By lemma 5.1, no edge is overtightened during algorithm 2 and so $\sum\limits_{X:s\in X} \gamma_X \leq c_b$ and $\sum\limits_{X:s\in X} \gamma_X \leq c_{e_1}$. Similarly, the edges c and d feel dual from all residual sets X with $u \in X$ and $s \notin X$. So we have $\sum\limits_{X:u\in X,s\notin X} \gamma_X \leq c_c$ and $\sum\limits_{X:u\in X,s\notin X} \gamma_X \leq c_d$. Also, the edges e_2 and e_3 feel dual from all residual sets X with $t \in X$ and $s, u \notin X$. So we have $\sum\limits_{X:t\in X,s,u\notin X} \gamma_X \leq c_{e_i}$ for $i \in \{2,3\}$. Thus we have,

$$\begin{aligned} 2\Big(\sum_{X:\{s,t,u\}\cap X\neq\emptyset} \gamma_X\Big) &= 2\Big(\sum_{X:s\in X} \gamma_X\Big) + 2\Big(\sum_{X:u\in X,s\notin X} \gamma_X\Big) + 2\Big(\sum_{X:t\in X,u,s\notin X} \gamma_X\Big) \\ &\leq c_b + c_c + c_d + c_{e_1} + c_{e_2} + c_{e_3} \\ &\leq \frac{6}{7} c(\widetilde{T}_K) \end{aligned}$$

where the last line follows from the fact that $\widetilde{T}_K$ has 7 edges and a is the most expensive edge.

Thus, in both cases we have that $2\Big(\sum\limits_{X:\{s,t,u\}\cap X\neq\emptyset} \gamma_X\Big) \leq \frac{6}{7}$. So we have

$$\sum_{X:\{s,t,u\}\cap X\neq\emptyset} \gamma_X \leq \frac{3}{7} c(\widetilde{T}_K) \leq \frac{3}{7} c(\bar{T}_K) = \frac{\theta}{2\theta+1} c(\bar{T}_K)$$

where the last line follows from the fact that $\phi = \theta = 3$.

Thus for each component K of $T[I]$ we have $\sum\limits_{X:X\cap V(K)\neq\emptyset} \gamma_X \leq \frac{\theta}{2\theta+1}\Big(\sum_{f\in E(\bar{T}_K)} c_f\Big)$. However each component of $T[I]$ induces a ϕ-star with $\phi \leq 3$. Moreover, these graphs are all edge disjoint. In addition, clearly for each $s \in I$, s is contained in some component of $T[I]$. So let $\mathcal{K}_I$ be the set of components of $T[I]$. Then, we have,

$$\sum_{\text{residual } X} \gamma_X \leq \sum_{K \in \mathcal{K}_I} \Big(\sum_{X : X \cap V(K) \neq \emptyset} \gamma_X \Big) \leq \frac{\theta}{2\theta+1} \Big[\sum_{K \in \mathcal{K}_I} \Big(\sum_{f \in E(\bar{T}_K)} c_f \Big) \Big] \leq \frac{\theta}{2\theta+1} \Big(\sum_{f \in E(T)} c_f \Big)$$

□

With this result, we can now prove the approximation factor for $\theta \leq 3$ if algorithm 2 terminates.

Theorem A.3. *Let $G = (V, E)$ be a θ-Star-Steiner graph satisfying the triangle inequality and assume $\theta \leq 3$. If algorithm 2 terminates successfully with tree T then*

$$\sum_{e \in E(T)} c_e \leq \frac{2\theta+1}{\theta+1} \sum_{\textit{valid } K} \gamma_K$$

Proof. Let T' be an optimal Steiner Tree for the given instance. Also, let r be the root chosen in the last iteration of algorithm 2 and direct all edges of T' towards r. Let the resulting arcs be $\vec{E}(T')$. Now define x' for each $e \in \vec{E}(G)$ as

$$x'_e = \begin{cases} 1 & \text{if } e \in \vec{E}(T') \\ 0 & \text{otherwise} \end{cases}$$

Then, by definition of x', $\sum\limits_{e \in \vec{E}(G)} c_e x'_e = \sum\limits_{e \in E(T')} c_e$. Also, because the directed version of T' is an arborescence directed into r and spanning R, x' is feasible for (P). But now if x^* is the optimal solution for (P) and Y^* is the optimal solution for (D) then by strong duality
$\sum\limits_{e \in \vec{E}(G)} c_e x^*_e = \sum\limits_{\text{valid } C} Y^*_C$.
However, if γ is the dual solution returned by algorithm 2 then by lemma 5.1 γ is feasible for (D^*). So define $Y_C = \gamma_C$ for each valid set C. Then Y is feasible for D. Thus,

$$\sum_{e \in E(T')} c_e = \sum_{e \in \vec{E}(G)} c_e x'_e \geq \sum_{e \in \vec{E}(G)} c_e x^*_e = \sum_{\text{valid } C} Y^*_C \geq \sum_{\text{valid } C} Y_C$$

Moreover, by lemma 5.2, $\sum\limits_{\text{valid } C} \gamma_C + \sum\limits_{\text{residual } X} \gamma_X = \sum_{e \in E(T)} c_e$.
and by definition $\sum\limits_{\text{valid } C} \gamma_C = \sum\limits_{\text{valid } C} Y_C$
Also, by lemma A.2, $\sum\limits_{\text{residual } X} \gamma_X \leq \frac{\theta}{2\theta+1} \Big(\sum\limits_{e \in E(T)} c_e \Big)$.
Thus we have,

$$\sum_{e \in E(T)} c_e = \sum_{\text{valid } C} \gamma_C + \sum_{\text{residual } X} \gamma_X \leq \sum_{\text{valid } K} \gamma_K + \frac{\theta}{2\theta+1} \Big(\sum_{e \in E(T)} c_e \Big)$$

$$\implies \sum_{e \in E(T)} c_e \leq \frac{2\theta+1}{\theta+1} \Big(\sum_{\text{valid } K} \gamma_K \Big)$$

□

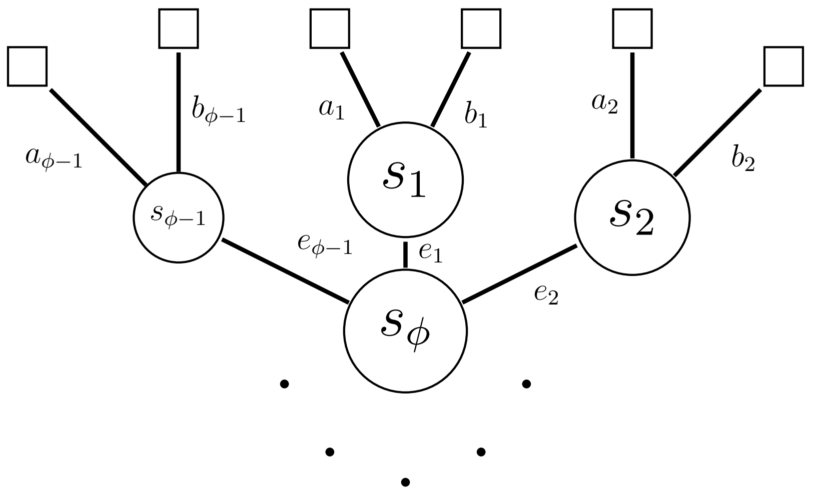


Figure A.2: $\widetilde{T}_K$ for lemma A.4

We now turn to the case where $\theta \geq 4$.

Lemma A.4. *Suppose algorithm 2 terminates with tree T and set $I \subseteq S$. Then if $\theta \geq 4$ we have*

$$\sum_{\textit{residual } X} \gamma_X \leq \frac{3\theta - 4}{6\theta - 6} \sum_{e \in E(T)} c_e$$

Proof. Note that every component of $H[I]$ is a star with at most θ nodes. By the same argument used in lemma 5.4, every component of $T[I]$ is a star with at most θ nodes. So let K be a component of $T[I]$ and suppose that K has $\phi \leq \theta$ vertices. If $\phi \leq 3$, then K is a path with at most 2 edges. Let $\bar{T}_K$ be the ϕ-star induced by the vertices of K. Then by lemma A.2 we have

$$\sum_{X : X \cap V(K) \neq \emptyset} \gamma_X \leq \frac{\phi}{2\phi + 1} \sum_{f \in E(\bar{T}_K)} \leq \frac{\theta}{2\theta + 1} \sum_{f \in E(\bar{T}_K)} \leq \frac{3\theta - 4}{6\theta - 6} \sum_{f \in E(\bar{T}_K)}$$

where the second inequality follows from $\phi \leq \theta$ and the third inequality follows from the fact that when $\theta \geq 4$ we have $\frac{\theta}{2\theta+1} \leq \frac{3\theta-4}{6\theta-6}$ with equality only if $\theta = 4$.
So instead suppose that $\phi \geq 4$. Then K is a star with $\phi - 1$ leaf nodes. Denote the center of K by s_ϕ and let $s_1, s_2, ..., s_{\phi-1}$ be the leaves of K. Then K has edges $e_i = \{s_i, s_\phi\}$ for each $i \in \{1, ..., \phi - 1\}$. Moreover, we know that $\deg_T(s_i) \geq 3$ for each $i \in \{1, ..., \phi - 1\}$. Thus s_i is incident to at least two vertices of R for each $i \in \{1, ..., \phi - 1\}$. Note that s_ϕ may also be incident to some vertices of R. Let $\bar{T}_K$ be the subtree corresponding to edges incident to at least one node of K. By possibly deleting edges, we obtain a subtree $\widetilde{T}_K$ of $\bar{T}_K$ in which s_i is incident to exactly two vertices of R. Specifically, $\delta_{\widetilde{T}_K}(\{s_i\}) = \{a_i, b_i, e_i\}$ for each $i \in \{1, ..., \phi - 1\}$ and $\delta_{\widetilde{T}_K}(\{s_\phi\}) = \{e_1, ..., e_{\phi-1}\}$. $\widetilde{T}_K$ is depicted in figure A.2. Notice that $\widetilde{T}_K$ has $3\phi - 3$ edges.
So first assume that e_j is the most expensive edge of $\widetilde{T}_K$ for some $j \in \{1, ..., \phi - 1\}$. Then by lemma 5.2, no edge is overtightened during the execution of algorithm 2. The edges a_1 and b_1 feel dual from all residual sets X with $s_1 \in X$. Thus we have $\sum\limits_{X : s_1 \in X} \gamma_X \leq c_{a_1}$ and $\sum\limits_{X : s_1 \in X} \gamma_X \leq c_{b_1}$. Also, for $i \in \{2, 3, ..., \phi - 1\}$, the edges a_i and

b_i feel dual from all residual sets X with $s_i \in X$ and $s_1, ..., s_{i-1} \notin X$. So we have $\sum\limits_{X:s_i\in X, s_1,...,s_{i-1}\notin X} \gamma_X \leq c_{a_i}$ and $\sum\limits_{X:s_i\in X, s_1,...,s_{i-1}\notin X} \gamma_X \leq c_{b_i}$ for each $i \in \{2, 3, ..., \phi - 1\}$. Lastly, for each $i \in \{1, ..., \phi - 1\}$ the edge e_i feels dual from all residual sets X with $s_\phi \in X$ and $s_1, ..., s_{\phi-1} \notin X$. So in particular, $\sum\limits_{X:s_\phi\in X, s_1,...,s_{\phi-1}\notin X} \gamma_X \leq c_{e_i}$ for each $i \in \{1, ..., \phi - 1\} \setminus \{j\}$ By combining these results we have

$$\begin{aligned}
2\Big(\sum_{X:X\cap V(K)\neq\emptyset} \gamma_X\Big) &= 2\Big(\sum_{X:1\in X} \gamma_X\Big) + 2\sum_{i=2}^{\phi-1}\Big(\sum_{X:s_i\in X:s_1,...,s_{i-1}\notin X} \gamma_X\Big) + 2\Big(\sum_{X:s_\phi\in X, s_1,...,s_{\phi-1}\notin X} \gamma_X\Big)\\
&\leq \sum_{i=1}^{\phi-1}\Big(c_{a_i} + c_{b_i}\Big) + \sum_{\substack{i=1\\(i\neq j)}}^{\phi-1} c_{e_i}\\
&\leq \frac{3\phi-4}{3\phi-3}c(\widetilde{T}_K)
\end{aligned}$$

where the second line follows from the fact that $\phi \geq 4$ so that $\phi - 2 \geq 2$. The last line follows from the fact that $\widetilde{T}_K$ has $3\phi - 3$ edges and e_j is the most expensive. Otherwise, a_i or b_i is the most expensive edge for some $i \in \{1, ..., \phi - 1\}$. Without loss of generality, we can assume that a_1 is the most expensive edge. The edges a_2 and b_2 feel dual from all residual sets X with $s_2 \in X$. As no edge is overtightened, we have $\sum\limits_{X:s_2\in X} \gamma_X \leq c_{a_2}$ and $\sum\limits_{X:s_2\in X} \gamma_X \leq c_{b_2}$. Similarly, for each $i \in \{3, ..., \phi - 1\}$, the edges a_i and b_i feel dual from all residual sets X with $s_i \in X$ and $s_2, ..., s_{i-1} \notin X$. So we have $\sum\limits_{X:s_i\in X, s_2,...,s_{i-1}\notin X} \gamma_X \leq c_{a_i}$ and $\sum\limits_{X:s_i\in X, s_2,...,s_{i-1}\notin X} \gamma_X \leq c_{b_i}$. In addition, for each $i \in \{2, ..., \phi-1\}$, the edge e_i feels dual from all residual sets X with $s_\phi \in X$ and $s_2, ..., s_{\phi-1} \notin X$. So we have $\sum\limits_{X:s_\phi\in X, s_2,...,s_{\phi-1}\notin X} \gamma_X \leq c_{e_i}$ for each $i \in \{2, ..., \phi - 1\}$. Lastly, the edges b_1 and e_1 feel dual from all residual sets X with $s_1 \in X$ and $s_2, ..., s_\phi \notin X$. So we have $\sum\limits_{X:s_1\in X, s_2,...,s_\phi\notin X} \gamma_X \leq c_{b_1}$ and $\sum\limits_{X:s_1\in X, s_2,...,s_\phi\notin X} \gamma_X \leq c_{e_1}$. By combining these inequalities we obtain

$$\begin{aligned}
2\Big(\sum_{X:X\cap V(K)\neq\emptyset} \gamma_X\Big) &= 2\Big(\sum_{X:s_2\in X} \gamma_X\Big) + 2\sum_{i=3}^{\phi}\Big(\sum_{X:s_i\in X, s_2,...,s_{i-1}\notin X} \gamma_X\Big) + 2\Big(\sum_{X:s_1\in X, s_2,...,s_\phi\notin X} \gamma_X\Big)\\
&\leq c_{b_1} + \sum_{i=2}^{\phi}\Big(c_{a_i} + c_{b_i}\Big) + \sum_{i=1}^{\phi-1} c_{e_i}\\
&\leq \frac{3\phi-4}{3\phi-3}c(\widetilde{T}_K)
\end{aligned}$$

where the second line follows from the fact that $\phi \geq 4$ and so $\phi - 2 \geq 2$. The last line follows from the fact that $\widetilde{T}_K$ has $3\phi - 3$ edges and a_1 is the most expensive. Thus in both cases we have that $2\Big(\sum\limits_{X:X\cap V(K)\neq\emptyset} \gamma_X\Big) \leq \frac{3\phi-4}{3\phi-3}c(\widetilde{T}_K)$. Thus we have,

$$\sum_{X:X\cap V(K)\neq\emptyset} \gamma_X \leq \frac{3\phi-4}{6\phi-6}c(\widetilde{T}_K) \leq \frac{3\phi-4}{6\phi-6}c(\bar{T}_K) \leq \frac{3\theta-4}{6\theta-6}c(\widetilde{T}_K)$$

where the second inequality follows from $c(\widetilde{T}_K) \leq c(\bar{T}_K)$ and the third inequality follows from $\phi \leq \theta$.

Thus for each component K of $T[I]$ we have $\sum\limits_{X:X\cap V(K)\neq\emptyset} \gamma_X \;\leq\; \frac{3\theta-4}{6\theta-6}\Big(\sum_{f\in E(\bar{T}_K)} c_f\Big)$. However, for each component K of $T[I]$, K together with all edges having an end in K induces a subtree of T, and all of these trees are edge disjoint. Moreover, clearly for each $s \in I$, s is contained in some component of $T[I]$. So let $\mathcal{K}_I$ be the set of components of $T[I]$. Then, we have,

$$\sum_{\text{residual } X} \gamma_X \;\leq\; \sum_{K\in\mathcal{K}_I}\Big(\sum_{X:X\cap V(K)\neq\emptyset} \gamma_X\Big) \;\leq\; \frac{3\theta-4}{6\theta-6}\Big[\sum_{K\in\mathcal{K}_I}\Big(\sum_{f\in E(\bar{T}_K)} c_f\Big)\Big] \;\leq\; \frac{3\theta-4}{6\theta-6}\Big(\sum_{f\in E(T)} c_f\Big)$$

□

Now we can show the second approximation factor for $\theta \geq 4$ if algorithm 2 terminates.

Theorem A.5. *Let $G = (V, E)$ be a θ-Star-Steiner graph satisfying the triangle inequality and assume $\theta \geq 4$. If algorithm 2 terminates successfully with tree T then*

$$\sum_{e\in E(T)} c_e \leq \frac{6\theta-6}{3\theta-2} \sum_{\text{valid } K} \gamma_K$$

Proof. Let T' be an optimal Steiner Tree for the given instance. Also, let r be the root chosen in the last iteration of algorithm 2 and direct all edges of T' towards r. Let the resulting arcs be $\vec{E}(T')$. Now define x' for each $e \in \vec{E}(G)$ as

$$x'_e = \begin{cases} 1 & \text{if } e \in \vec{E}(T') \\ 0 & \text{otherwise} \end{cases}$$

Then, by definition of x', $\sum\limits_{e\in\vec{E}(G)} c_e x'_e \;=\; \sum\limits_{e\in E(T')} c_e$. Also, because the directed version of T' is an arborescence directed into r and spanning R, x' is feasible for (P). But now if x^* is the optimal solution for (P) and Y^* is the optimal solution for (D) then by strong duality
$\sum\limits_{e\in\vec{E}(G)} c_e x^*_e \;=\; \sum\limits_{\text{valid } C} Y^*_C$.
However, if γ is the dual solution returned by algorithm 2 then by lemma 5.1 γ is feasible for (D^*). So define $Y_C = \gamma_C$ for each valid set C. Then Y is feasible for D. Thus,

$$\sum_{e\in E(T')} c_e \;=\; \sum_{e\in\vec{E}(G)} c_e x'_e \;\geq\; \sum_{e\in\vec{E}(G)} c_e x^*_e \;=\; \sum_{\text{valid } C} Y^*_C \;\geq\; \sum_{\text{valid } C} Y_C$$

Moreover, by lemma 5.2, $\sum\limits_{\text{valid } C} \gamma_C \;+\; \sum\limits_{\text{residual } X} \gamma_X \;=\; \sum_{e\in E(T)} c_e$.
and by definition $\sum\limits_{\text{valid } C} \gamma_C \;=\; \sum\limits_{\text{valid } C} Y_C$
Also, by lemma A.4, $\sum\limits_{\text{residual } X} \gamma_X \;\leq\; \frac{3\theta-4}{6\theta-6}\Big(\sum\limits_{e\in E(T)} c_e\Big)$.

Thus we have,

$$\begin{aligned}\sum_{e\in E(T)} c_e \;&=\; \sum_{\text{valid } C}\gamma_C \;+\; \sum_{\text{residual } X}\gamma_X \;\le\; \sum_{\text{valid } K}\gamma_K \;+\frac{3\theta-4}{6\theta-6}\Big(\sum_{e\in E(T)} c_e\Big)\\ \Longrightarrow \sum_{e\in E(T)} c_e \;&\le\; \frac{6\theta-6}{3\theta-2}\Big(\sum_{\text{valid } K}\gamma_K\Big)\end{aligned}$$

□