On the Role of Partition Inequalities in Classical Algorithms for Steiner Problems in Graphs

by

Kunlun Tan

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, 2006

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

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

Abstract

The Steiner tree problem is a classical, well-studied, $\mathcal{NP}$-hard optimization problem. Here we are given an undirected graph $G = (V, E)$, a subset R of V of terminals, and non-negative costs c_e for all edges e in E. A feasible Steiner tree for a given instance is a tree T in G that spans all terminals in R. The goal is to compute a feasible Steiner tree of smallest cost. In this thesis we will focus on approximation algorithms for this problem: a c-approximation algorithm is an algorithm that returns a tree of cost at most c times that of an optimum solution for any given input instance.

In a series of papers throughout the last decade, the approximation guarantee c for the Steiner tree problem has been improved to the currently best known value of 1.55 [39]. Robins' and Zelikovsky's algorithm as well as most of its predecessors are greedy algorithms.

Apart from algorithmic improvements, there also has been substantial work on obtaining tight linear-programming relaxations for the Steiner tree problem. Many undirected and directed formulations have been proposed in the course of the last 25 years; their use, however, is to this point mostly restricted to the field of exact optimization. There are few examples of algorithms for the Steiner tree problem that make use of these LP relaxations. The best known such algorithm for general graphs is a 2-approximation (for the more general Steiner forest problem) due to Agrawal, Klein and Ravi [2]. Their analysis is tight as the LP-relaxation used in their work is known to be weak: it has an IP/LP gap of approximately 2.

Most recent efforts to obtain algorithms for the Steiner tree problem that are based on LP-relaxations has focused on directed relaxations. In this thesis we present an undirected relaxation and show that the algorithm of Robins and Zelikovsky returns a Steiner tree whose cost is at most 1.55 times its optimum solution value. In fact, we show that this algorithm can be viewed as a primal-dual algorithm.

The Steiner forest problem is a generalization of the Steiner tree problem. In the problem, instead of only one set of terminals, we are given more than one terminal set. An feasible Steiner forest is a forest that connects all terminals in the same terminal set for each terminal set. The goal is to find a minimum cost feasible Steiner forest. In this thesis, a new set of facet defining inequalities for the polyhedra of the Steiner forest is introduced.

Acknowledgements

Thanks to my wife Lixin Zhang, my supervisor Jochen Könemann, my thesis readers, and all professors and friends who support my research.

Contents

List of Figures

Chapter 1

Introduction

The Steiner tree and Steiner forest problems are $\mathcal{NP}$-hard combinatorial optimization problems with important applications in network design and very-large-scale integration (VLSI) layout design.

In the Steiner Tree problem, we are given an undirected connected graph $G = (V, E)$, non-negative costs c_e for all edges $e \in E$, and a set of vertices $R \subseteq V$ that are called *terminals*. We refer to the vertices that are not terminals as *Steiner vertices*, and let S denote the set of such vertices. We are interested in finding a minimum cost tree that spans all terminals and may or may not contain some Steiner vertices; such a tree is called a *Steiner tree*.

The Steiner forest problem is a generalization of the Steiner tree problem. In this problem, we are given a collection of disjoint subsets of V, $R_1, R_2, ..., R_k$, which are called terminal sets. A Steiner forest is a forest such that each pair of terminals in the same set R_i is connected in the forest. The goal of the Steiner forest problem is to find a minimum cost Steiner forest. By definition, the Steiner tree problem is a special case of the Steiner forest problem, in which the class of terminal sets contains only one terminal set. An equivalent formulation of the the Steiner forest problem is the following. We are given an undirected connected graph $G = (V, E)$ with non-negative costs c_e for all edges, and and set $\mathcal{R}$ of terminal pairs (s_i, t_i) for some $i > 0$. We want to find a forest that connects each pair $(s_i, t_i) \in \mathcal{R}$, such that the total edge costs of the forest is minimum.

For examples of Steiner tree and Steiner forest instances, please refer to *Fig. 1.1.* In the

figure, the graph on the left is an example for Steiner tree problem. The solid vertices are terminals, the square vertices are Steiner vertices. The solid edges are edges in the Steiner tree. The graph on the right is an example for Steiner forest problem. The solid round and square vertices are two sets of terminals, the hollow square vertices are the Steiner vertices. The solid edges are edges in the Steiner forest.

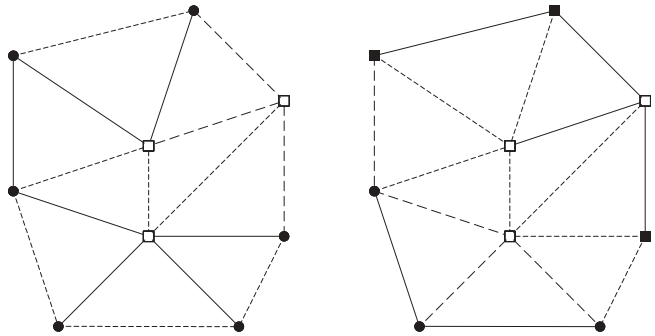


Figure 1.1: Examples for Steiner tree and Steiner forest

Karp [26] proved that the Steiner tree problem is $\mathcal{NP}$-hard. Therefore, we cannot find a polynomial-time exact algorithm for the Steiner tree problem, unless $\mathcal{P} = \mathcal{NP}$. Since the Steiner forest problem is a generalization of the Steiner tree problem, the Steiner forest problem is also $\mathcal{NP}$-hard. Many heuristics and exactly solutions for the problems have been introduced during past decades, but their running time are exponential long in terms of the problems size for general graphs. Since we cannot hope to find exact algorithms that can solve the problems efficiently, in the thesis, we resort to finding polynomial-time algorithms that efficiently solve the problems approximately, i.e., approximation algorithms. An *approximation algorithm* is a heuristic algorithm, which does not solve the given optimization problem exactly, but guarantees an upper bound on the worst case ratio between the value of any approximate solution and that of an optimum solution.

In this chapter, we first give the definitions and the notations of the terms used in this thesis; then we will give a survey on known algorithms of Steiner tree and Steiner forest. We will end the chapter with a summary of results obtained by this thesis.

1.1 Definitions and notations

In this section, we will introduce some definitions and their notations of Steiner tree and Steiner forest problems. Then, we will state the main assumption in this thesis, and prove it is without loss of generality.

In an instance of the Steiner tree or Steiner forest problem, we are given an undirected graph $G = (V, E)$, and a cost function $c(\cdot) : E \mapsto \mathbb{R}_+$. In this thesis, we refer to Steiner tree and Steiner forest problems as Steiner problems, if we do not differentiate them.

- A *terminal set* R_i is a subset of V, in which all elements of R_i are required to be located in a same connected component in a feasible solution. In this thesis, we let R represent the union of all R_i's, i.e.,

$$R = \bigcup_{\forall i} R_i.$$

  Each element v of R is called a *terminal*. In a Steiner forest problem, we usually have more than one disjoint terminal set, while in a Steiner tree problem, we have only one terminal set. The vertices, that are not in any terminal set, are called the *Steiner vertices* of G. The set of Steiner vertices is denoted by S, i.e., $S = V \setminus R$.

- A *Steiner subgraph* H of G is a subgraph of G, such that for each terminal set R_i, all elements of R_i are located in some connected component of H. A Steiner subgraph is a feasible solution for a Steiner problem. If a Steiner subgraph is a tree in the Steiner tree problem, it is called a *Steiner tree*; and if it is a forest in the Steiner forest problem, it is called a *Steiner forest*. Since the cost function $c(\cdot)$ is nonnegative, without loss of generality, we may assume that an optimal solution for a Steiner problem to be a does not contain Steiner leaves, and that it contains no cycles.

- A *minimum spanning tree* of G is a spanning tree of G whose total edge cost is minimum. We denote it by **MST**, and its cost by **mst**. Similarly, we denote the optimal solution for a Steiner problem by **OPT**, and its cost by **opt**.

- A *metric completion* G' of G is a complete graph on V, such that for any $uv \in E(G')$, the cost of uv is the cost of shortest uv-path in G.

- A *quasi-bipartite* graph is a graph G such that its Steiner vertices form a independent set of G.

1.2 The metric assumption

For a Steiner problem, if the triangle inequality holds for the cost function $c(\cdot)$, i.e.,

$$c_{uv} \leq c_{uw} + c_{wv}, \quad \forall uv, uw, wv \in E(G), \tag{1.1}$$

we call the problem a metric Steiner problem. The following theorem is a folklore.

Theorem 1. *There is an approximation factor preserving reduction from a general Steiner problem to a metric Steiner problem.*

Proof. Suppose we have an instance I of a general Steiner problem on graph $G = (V, E)$. Construct the metric completion graph G' of G. Clearly, the cost function $c'(\cdot)$ of G' satisfies the triangle inequality. Then we get an instance I' of a metric Steiner problem.

By the definition of G', $c'_e \leq c_e$ for all $e \in E(G)$, since each $e = uv \in E$ is a uv-path in G. Therefore, the cost of an optimal solution in I' is less than or equal to the cost of an optimal solution in I.

On the other hand, let F' be a feasible solution in I'. Now we show how to construct a feasible solution F in I with cost no more than the cost of F'. For each edge $uv \in E(F')$, since the cost of uv in G' is the cost of a uv-path P_{uv} in G. We include all edge in P_{uv} in F for all edge $uv \in E(F')$. If F contain cycles, we may remove some edges to eliminate cycles without affecting feasibility. Clearly, F is a feasible solution in I with cost no more than the cost of F'; and the reduction is in polynomial time. Therefore this is an approximation factor preserving reduction. □

By the above theorem, we may assume that $c(\cdot)$ satisfies the triangle inequality for G. This assumption is without loss of generality.

1.3 Results obtained in the thesis

In a series of papers throughout the last decade, the approximation guarantee for the Steiner tree problem has been continuously improved to the currently best known value of 1.55 [39].

Apart from algorithmic improvements, there also has been substantial work on obtaining tight linear-programming relaxations for the Steiner tree problem. Many undirected and directed formulations have been proposed in the course of the last 25 years; their use, however, is to this point mostly restricted to the field of exact optimization. There are few examples of algorithms for the Steiner tree problem that make use of these LP relaxations. The best known such algorithm for general graphs is a 2-approximation (for the more general Steiner forest problem) due to Agrawal, Klein and Ravi [2]. Their analysis is tight as the LP-relaxation used in their work is known to be weak: it has an IP/LP gap of approximately 2.

Most recent efforts to obtain algorithms for the Steiner tree problem that are based on LP-relaxations has focused on directed relaxations. In this thesis we unify the approximation algorithm approach and the undirected relaxation to show that the iterated 1-Steiner heuristic for quasi-bipartite graphs, and the algorithm of Robins and Zelikovsky can be viewed as a primal-dual algorithm. Also we provide the transition performance of Robins and Zelikovsky's algorithm between quasi-bipartite graphs and general graphs. At the end of the thesis, we provide a set of facet defining inequality for the Steiner forest polyhedra, which is stronger than the current known inequality.

Chapter 2

A survey of Steiner tree and Steiner forest problems

In this chapter, we give a survey for Steiner tree and Steiner forest problems.

2.1 History and variants of the Steiner tree problem

2.1.1 History of Steiner tree problem

The Steiner tree problem is a variant of the Euclidean Steiner tree problem, which is much older, and it asks for a Steiner tree for a given set of points in the plane. A special three-point case of this problem was discussed by Fermat in 1640. Jacob Steiner considered a generalization of this problem. The general case of Euclidean Steiner trees was proposed by Janník and Kössler in 1934. The combinatorial version of the Steiner tree problem was proposed independently by Hakimi and Levin in 1971. Since then, many papers were published about different variants of the Steiner tree problem. A more detailed history of the Steiner tree problem is given in [24].

2.1.2 Variants of the Steiner tree problem

- **Euclidean Steiner tree problem**
  In the Euclidean Steiner tree problem, we are give a set R of points in the plane (or n-dimensional Euclidean space). The goal is to find a set S of points along with a spanning tree T on $R \cup S$, such that T has minimum Euclidean length. Comparing to the Steiner tree problem in general graph, the Euclidean Steiner tree problem has an infinite number of Steiner vertices, and all vertices correspond to points in a Euclidean space. The Euclidean Steiner tree problem can be approximately reduced to an instance of general Steiner tree problem by defining a complete graph whose vertex set is the set of grid-line intersection points on the given Euclidean space. [36].

- **Rectilinear Steiner tree problem**
  In the Rectilinear Steiner tree problem, we are give a set R of points in the plane. The goal is to find a set S of points in the plane along with a spanning tree T on $R \cup S$, such that T has minimum L_1-norm length. It can be proved that rectilinear Steiner tree problem is a special case of the Euclidean Steiner tree problem. [23].

- **Steiner tree in directed graphs**
  In this variant, we work on directed graphs instead of undirected graphs. The best known approximation algorithm for acyclic directed graph was introduced by Zelikovsky [44], and the approximation ratio is $O(k^{\epsilon})$ for any $\epsilon > 0$, where k is the number of terminals in the graph.

2.2 Integer programming formulations of the Steiner tree problem

In the section, we will introduce two different types of integer programming formulations of the Steiner tree problem. There are many other different formulations for the Steiner tree problem, such as flow formulations and different types of tree formulations. For complete reference, please refer to [16, 34, 33].

2.2.1 Undirected formulations

In [3], Aneja gave an integer programming formulation for the Steiner tree problem. This formulation is called the cut formulation.

For all edge $e \in E$, let x_e be a binary variable associate with e. A solution x will correspond to a feasible Steiner tree, i.e., $x_e = 1$ if e is part of the corresponding Steiner tree. A *Steiner cut* is a cut $\delta(W)$, such that W is a nonempty proper subset of V, and $W \cap R \neq R, W \cap R \neq \varnothing$, where a proper subset of V is a subset of V but not equal to V. Therefore, a Steiner cut is a cut that separates the terminal set R into two non-empty parts. Please refer to *Fig. 2.1.*

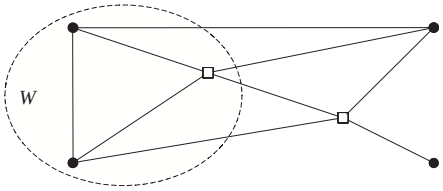


Figure 2.1: An example for Steiner cut

Let $\mathcal{C}_S$ be set of Steiner cuts. The cut formulation is as follows:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & & & (\mathrm{IP_{ST}}) \\ \text{s.t.} \quad & \sum_{e \in \delta(W)} x_e \geq 1 & & \forall \delta(W) \in \mathcal{C}_S & (2.1) \\ & x_e \in \{0,1\} & & \forall e \in E \end{aligned}$$

Now we discuss the spanning tree polyhedron. Chopra [8] defined the spanning tree polyhedron as:

$$P_T = \mathrm{conv}\left\{x \middle| x \text{ is an incidence vector of a spanning tree of } G\right\} + \mathbb{R}_+^E.$$

Chopra introduced a notion of *feasible partition* of vertex set V. A partition $\pi = \{V_1, V_2, ..., V_p\}$ of V is a class of subsets of V, such that $\bigcup_{i=1}^{p} V_i = V$ and $V_i \cap V_j = \varnothing$, for all $i \neq j$. A feasible partition π of V is a partition of V, such that the induced graph $G[V_i]$

of V_i is connected for all $1 \leq i \leq p$. The *rank* $r(\pi)$ of π is defined as the number of parts of π. We obtain a multi-graph G_π from G by identifying the vertices of V_i to a vertex v_i for all $1 \leq i \leq p$. We define

$$E_\pi = \left\{ uv \in E \middle| \exists 1 \leq i < j \leq p, u \in V_i, v \in V_j \right\}$$

to be the set of edges whose endpoints are in different part of π, i.e., the edges of G_π except the loops. Chopra showed that

$$P_T = \left\{ x \in \mathbb{R}_+^E \middle| \sum_{e \in E_\pi} x_e \geq r(\pi) - 1, \ \forall \pi \in \Pi \right\}, \tag{2.2}$$

where Π is the set of all feasible partitions of V. Also,

$$\sum_{e \in E_\pi} x_e \geq r(\pi) - 1, \ \pi \in \Pi$$

is facet defining for P_T if and only if G_π is 2-connected.

In [9], Chopra and Rao extended Chopra's idea for the spanning tree polyhedron to the Steiner tree polyhedron:

$$P_{ST} = \text{conv}\{x | x \text{ is the incidence vector of a Steiner tree}\} + \mathbb{R}_+^E.$$

Chopra and Rao proved that P_{ST} equals to the feasible region of IP_{ST}, and (2.1) is facet defining for P_{ST}. They also defined two classes of valid inequalities for P_{ST}, the Steiner partition inequalities and odd-hole inequalities.

A *Steiner partition* $\pi = \{V_1, V_2, ..., V_p\}$ of V is a partition of V, such that $V_i \cap R \neq \varnothing$, for all i. Recall that E_π is the set of edges whose endpoints are in different parts of π. The *Steiner partition inequality* is defined as

$$\sum_{e \in E_\pi} x_e \geq p - 1 \quad \forall \pi \in \Pi_S \tag{2.3}$$

where Π_S is the set of Steiner partitions of G. Note that (2.1) is just a special case of (2.3)

for $p = 2$.

Lemma 1 (Chopra-Rao)**.** *The Steiner partition inequality (2.3) is valid for P_{ST}. Furthermore, (2.3) is facet defining if and only if G_π is 2-connected and $G[V_i]$ is connected for all $1 \leq i \leq p$.*

Another valid inequality that is provided by Chopra and Rao is called the odd-hole inequality. Consider the graph $G_k = (V_k, E_k)$ with vertex set $V_k = R_k \cup S_k$. The terminal set is given by $R_k = \{u_1, u_2, ..., u_k\}$, and the Steiner vertices are $S_k = \{v_1, v_2, ..., v_k\}$. The edges are $E_k = \{u_i v_i : 1 \leq i \leq k\} \cup \{u_i v_{i-1} : 1 \leq i \leq k\}$. We define $v_0 = v_k$. Please refer to *Fig. 2.2* for graph G_k, $k = 3$.

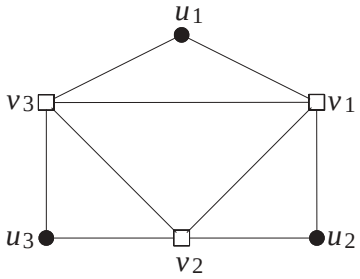


Figure 2.2: G_3

Notice that S_k forms a chordless cycle, and all terminals are of degree two. A *chordless cycle* is a cycle C, such that for any two vertices u and v in C, if uv is not an edge of C, then uv is not an edge of the whole graph either, i.e., $uv \notin E$. The odd-hole inequality is defined as

$$\sum_{e \in E_k} x_e \geq 2(k-1).$$

Lemma 2 (Chopra-Rao)**.** *The odd-hole inequality (2.4) is valid for $P_{ST}(G_k)$ for all $k \geq 3$. Furthermore, (2.4) is facet defining for $P_{ST}(G_k)$ if k is odd, and $k \geq 3$.*

In [6], Biha, Kerivin and Mahjoub generalized the Steiner partition inequality and odd-hole inequality, and gave a procedure of constructing facets for P_{ST}.

2.2.2 Directed formulations

The bidirected cut formulation of the spanning tree problem was introduced by Edmonds [13]. This formulation can be easily extended to the Steiner tree problem, which we will discuss in this section. It is believed that the integrality gap of the bidirected formulation for Steiner tree problem is very small, and the worst known example, where the gap is $\frac{8}{7}$, was proposed by Goemans, and it appears first in a recent paper by Agarwal and Charikar [1].

In the formulation, first we construct a new directed graph $\overrightarrow{G}$ by replacing every edge of E by a pair of anti-parallel directed edges, and the cost of each of these edges is the same as the cost of the original edge. Let $\overrightarrow{E}$ be the directed edge set of $\overrightarrow{G}$. Let r be an arbitrary terminal. In the formulation, we want to find an in-tree rooted at r that spans all terminals. Formally, for every nonempty subset C of V that contains at least one terminal but does not contain r, let $\delta_+(C)$ be the set of arcs whose tail is in C, and head is not in C. If $\delta_+(C)$ is not empty, we call C a *valid set*. Let $\mathcal{F}$ be the collection of all valid sets, i.e. $\mathcal{F} = \{C \subset V | r \notin C, C \cap R \neq \varnothing\}$.

For every directed edge $e \in \overrightarrow{E}$, we define a corresponding decision variable x_e, such chat $x_e = 1$ if e is included in our solution, and $x_e = 0$ otherwise. The Bidirected cut formulation is as follows:

$$
\begin{aligned}
\min \quad & \sum_{e \in \overrightarrow{E}} c_e x_e && && (\mathrm{IP}_{\mathrm{BD}}) \\
\text{s.t.} \quad & \sum_{e \in \delta_+(C)} x_e \geq 1 && \forall C \in \mathcal{F} && (2.4) \\
& x_e \in \{0,1\} && \forall e \in \overrightarrow{E}
\end{aligned}
$$

An optimal solution of $\mathrm{IP}_{\mathrm{BD}}$ corresponds to an in-tree $\overrightarrow{T}$ rooted at r, the underlying undirected graph T of $\overrightarrow{T}$ is a Steiner tree, and the cost of T and $\overrightarrow{T}$ is the same. It is easy to check that T is an optimal Steiner tree. It is also true that the choice of r does not matter, since T can be oriented to get a feasible solution of $\mathrm{IP}_{\mathrm{BD}}$ independent of the choice of r.

2.3 Exact algorithms for the Steiner tree problem

Though the Steiner tree problem is $\mathcal{NP}$-hard, it is polynomial time solvable in some special cases. For example, when $R = V$, Steiner tree problem becomes the minimum spanning tree problem, which can be solved efficiently by applying Kruskal's algorithm. On the other hand, if $|R| = 2$, the problem becomes the shortest path problem, which also can be solved efficiently. In this section, we will review two exact algorithms for solving the generalizations of these two cases in polynomial time. Precisely, we consider the cases where $|S|$ is bounded by a constant or $|R|$ is bounded by a constant.

2.3.1 The enumeration algorithm

Hakimi [22] provided an enumeration algorithm for instances of the Steiner tree problem where the number of Steiner vertices is bounded by a constant. Lawler [32] provided a modification of Hakimi's algorithm that makes use of the fact that one may assume that the given graph is complete and that the cost on its edges satisfy the triangle inequality.

The following lemma shows that the number of Steiner vertices of degree at least 3 in a T is bounded.

Lemma 3. *A Steiner tree T without Steiner leaves contains at most $|R|-2$ Steiner vertices of degree at least 3.*

Proof. Let S_2 denote the set of Steiner vertices of degree 2, let S_3 denote the set of Steiner vertices of degree at least 3, and let s_2 and s_3 be the size of these two sets, respectively. Then $|V(T)| = |R| + s_2 + s_3$. Summing up the degrees of all vertices in T, we have

$$\sum_{v \in V(T)} \deg_T(v) = 2|E(T)| = 2(|V(T)| - 1) = 2(|R| + s_2 + s_3 - 1).$$

Since

$$\sum_{v \in V(T)} \deg_T(v) = \sum_{v \in R} \deg_T v + \sum_{v \in S_2} \deg_T(v) + \sum_{v \in S_3} \deg_T(v) \geq |R| + 2s_2 + 3s_3,$$

we have

$$s_3 \leq |R| - 2.$$

□

By our metric assumption, we may assume that a minimum Steiner tree on G' does not contain Steiner vertices of degree less than 3. Therefore, in our algorithm, we can enumerate all subset of S of size at most $|R| - 2$, and find the minimum Steiner tree. The enumeration algorithm is listed as Algorithm 1.

Algorithm 1 Enumeration algorithm

1: Compute metric completion G' of G.
2: For all $S' \subseteq S$, such that $|S'| \leq |R| - 2$, find the minimum spanning tree of $G'[R \cup S']$. Let T be the minimum one.
3: Transform T into a Steiner tree of G.

Step 1 can be compute by running Dijkstra's algorithm for each vertex in V. Since Dijkstra's algorithm can be implemented in $O(n \log n + n)$ [14], where $n = |V|$ and $m = |E|$, step 1 also can be done in polynomial time. In Step 2, we need to compute at most

$$\sum_{i=0}^{|R|-2} \binom{|S|}{i} \leq \min\left\{|V|^{|R|-2}, 2^{|S|}\right\}$$

trees. But $|S|$ is bounded by a constant, therefore, we need to compute $O(1)$ many trees, hence step 2 can be done in polynomial time too. The transformation in Step 3 can also be done in polynomial time. Therefore, the enumeration algorithm runs in polynomial time for instances in which $|S|$ is bounded by a constant.

2.3.2 The algorithm by Dreyfus and Wagner

Dreyfus and Wagner [12] introduced a dynamic programming algorithm for instances of the Steiner tree problem with bounded number of terminals. The algorithm calculates Steiner trees for all subset of the terminal set.

First we will introduce some notation. Let $X \subseteq R$ be a proper subset of the terminals, and let $v \in V \setminus X$; we use $s(X \cup \{v\})$ to denote the cost of a minimum Steiner tree of

G with terminal set $X \cup \{v\}$; and let $s_v(X \cup \{v\})$ denote the cost of a minimum Steiner tree of G with terminal set $X \cup \{v\}$ such that v is an internal vertex in the tree. Clearly, $s(X \cup \{v\}) \leq s_v(X \cup \{v\})$. Dreyfus and Wagner's algorithm is bases on the following lemma.

Lemma 4. *For any $X \subset R$, $X \neq \varnothing$, and $v \in V$,*

$$s_v(X \cup \{v\}) = \min_{\varnothing \neq Y \subsetneq X} \{s(Y \cup \{v\}) + s(X \setminus Y \cup \{v\})\} \tag{2.5}$$

$$s(X \cup \{v\}) = \min \left\{ \min_{w \in X} \{d(v,w) + s(X)\}, \min_{w \in V \setminus X} \{d(v,w) + s_w(X \cup \{w\})\} \right\} \tag{2.6}$$

where $d(v,w)$ is the cost the shortest vw-path.

Proof. To prove (2.5), let T be a minimum Steiner tree for $X \cup \{v\}$, and $\deg_T(v) \geq 2$. Then T can be decomposed into two subtrees T_1 and T_2, such that T_1 is a Steiner tree of $Y \cup \{v\}$ and T_2 is a Steiner tree of $X \setminus Y \cup \{v\}$, where Y is a proper nonempty subset of X. Therefore, for all proper nonempty subset Y of X, we choose the one that gives the minimum value of 2.5, we obtain a minimum spanning tree for $X \cup \{v\}$, this proves (2.5).

To prove (2.6), we let T be a minimum Steiner tree for $X \cup \{v\}$. For a leaf u in T, we let P_u be the longest path in T starting at u, in which all internal vertices have degree 2 in T are not contained in X. There are three cases with respect to v.

Case 1. $\deg_T(v) \geq 2$, then $s(X \cup \{v\}) = s_v(X \cup \{v\})$, (2.6) is attained for $v = w$.

Case 2. $\deg_T(v) = 1$, and the last vertex w of P_v is a terminal in X. Clearly, P_u is the shortest vw-path, and $T \setminus P_u$ is a minimum Steiner tree for X, therefore, $s(X \cup \{v\}) = s(X) + d(v,w)$.

Case 3. $\deg_T(v) = 1$, and the last vertex w of P_v is a vertex in $V \setminus X$ of degree at least 3. Then P_u is the shortest vw-path, and $T \setminus P_u$ is a minimum Steiner tree for $X \cup \{w\}$. Therefore, $s(X \cup \{v\}) = s_w(X) \cup \{w\} + d(v,w)$.

These three cases prove that (2.6) is true. □

The Dreyfus-Wagner algorithm starts with computing Steiner trees for all 2-subsets of R, and in the following iterations, it computes Steiner tree for 3-subsets of R, and continue this way until a Steiner tree for R is found. The algorithm is listed as Algorithm 2.

Algorithm 2 Dreyfus-Wagner algorithm

1: Compute $d(u, v)$ for each pair $u, v \in V$.
2: For each pair $u, v \in R$, $s(\{u, v\}) \leftarrow d(u, v)$
3: **for** $i = 2$ to $|R| - 1$ **do**
4: **for** $\forall X \subset R, |X| = i$ and $v \in V \setminus X$ **do**
5: $s_v(X \cup \{v\}) \leftarrow \min_{\varnothing \neq Y \subsetneq X} \{s(Y \cup \{v\}) + s(X \setminus Y \cup \{v\})\}$
6: $s(X \cup \{v\}) \leftarrow \min \left\{ \min_{w \in X} \{d(v, w) + s(X)\}, \min_{w \in V \setminus X} \{d(v, w) + s_w(X \cup \{w\})\} \right\}$
7: **end for**
8: **end for**

Similar to Algorithm 1, Step 1 can be computed in polynomial time. For the first recursion, since every terminal appears in exactly one of Y, $X \setminus Y$ and $R \setminus X$, the total number of comparisons and additions of the first recursion is $O(3^{|R|} \cdot |V|)$. For the second recursion, the total number of comparisons and additions is $O(2^{|R|} \cdot |V|)$. Recall that $|R|$ is bounded by a constant, therefore the total running time of Algorithm 2 is polynomial.

2.4 The minimum spanning tree heuristic for the Steiner tree problem

Since a spanning tree of G satisfies the connection requirement of the Steiner tree problem, a spanning tree is a Steiner tree. Therefore, we have a first approximation algorithm the Steiner tree problem. Many researchers rediscovered the approximation ratio for this algorithm, but it is first mentioned in [15], and attributed to Moore.

In the minimum spanning tree heuristic, we first compute the metric completion G' on R and find a minimum spanning tree T' for G'. Then we replace the edges in T' by the corresponding paths in G to get a Steiner subgraph F in the original graph, and find the minimum spanning tree T of F, which is the output of the algorithm.

Theorem 2. *The minimum spanning tree heuristic is a 2-approximation algorithm for the Steiner tree problem.*

Proof. Let T^* be an optimal Steiner tree of G. By Duplicating the edges of T^*, we get an

Eulerian graph. Find an Euler tour for the Eulerian graph, then the cost of the Euler tour is $2c(T^*)$. Please refer to *Fig. 2.3*. (In the figure and the entire thesis, the dots represent terminals, and the squares represent Steiner vertices.)

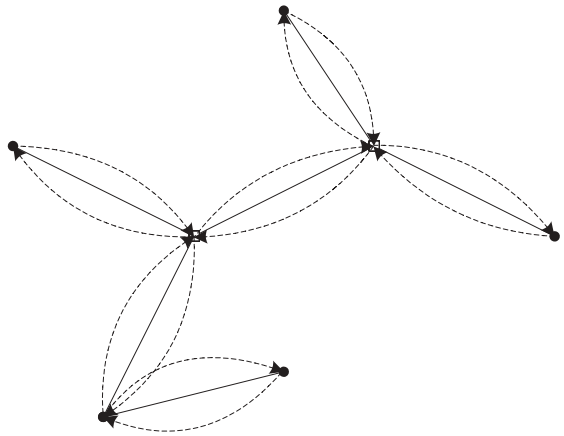


Figure 2.3: A Euler tour obtained from T^*

Now we can obtain a Hamiltonian cycle on R by traversing the Euler tour and short-cutting the Steiner vertices and visited terminals. By the triangle inequality, short-cutting does not increase the cost, i.e., the cost of the Hamiltonian cycle is at most $2c(T^*)$. Please refer to *Fig. 2.4*.

Now discard one edge in the Hamiltonian cycle, we have a spanning tree T' on R, whose cost is also at most $2c(T^*)$. Since the T is the minimum spanning tree on R, we have

$$c(T) \leq c(T') \leq 2c(T^*).$$

□

The approximation ratio 2 is tight. Please refer to *Fig. 2.5*. The graph is an n-wheel, whose center is the only Steiner vertex. The Minimum Spanning Tree Heuristic computes a path that contains only terminals with cost $(n-1)(2-\varepsilon)$, while the optimal Steiner tree

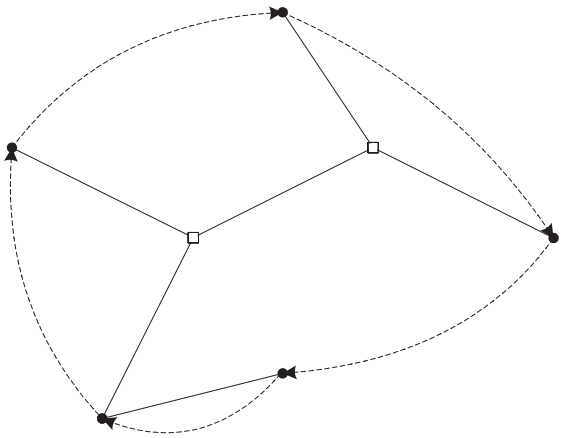


Figure 2.4: A Hamiltonian cycle obtained by short-cutting.

is of cost n. Therefore, the approximation ratio

$$\frac{(n-1)(2-\varepsilon)}{n} \to 2,$$

as $n \to \infty$ and $\varepsilon \to 0$.

2.5 The approximation algorithms better than 2

A Steiner tree without Steiner leaves can be decomposed into subtrees, such that all terminals are leaves of these subtrees. Formally, we define that a *full component* K of G as a subgraph of G, such that K is a tree, whose terminals coincide with its leaves. Note that any Steiner tree without Steiner leaves is the disjoint union of a set of full components. Please refer to *Fig. 2.6*.

A full component is *k-restricted* if the number of terminals is less or equal to k. A Steiner tree is *k-restricted* if all its full components are k-restricted. A k-restricted Steiner tree problem is to find a minimum cost k-restricted Steiner tree. In a k-restricted Steiner tree, we allow Steiner vertices and edges to overlap between k-restricted full components, since some graphs do not have feasible solution if overlapping is not allowed. For example,

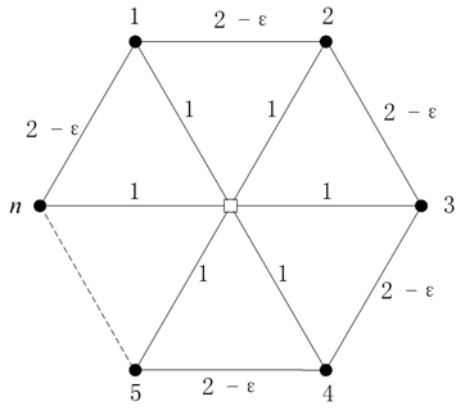


Figure 2.5: An example shows that the ratio 2 is tight.

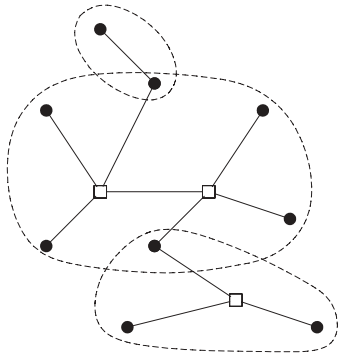


Figure 2.6: A Steiner tree decomposed into full components

consider the star with 6 tips (see *Fig. 2.7*) in which the center is the only Steiner vertex. This instance does not have a non-overlapping 4-restricted Steiner tree.

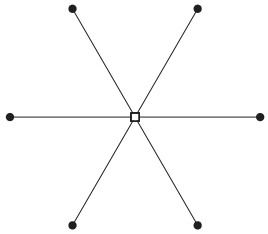


Figure 2.7: A 6-Star, which does not have non-overlapping 4-restricted Steiner tree.

The approximation ratio obtained by the minimum spanning tree heuristic remained the best known until very recently when Zelikovsky started to make use of k-restricted Steiner trees to analyze approximation algorithms for Steiner trees in 1990. After that, all known newly developed approximation algorithms make use of k-restricted Steiner trees.

The following table lists the authors and the proved ratio of the corresponding algorithm for the Steiner tree problem.

Authors	year	ratio
Moore	1968	2.0
Zelikovsky	1990	1.834
Berman, Ramaiyer	1991	1.734
Zelikovsky	1995	1.694
Prömel, Steger	1996	1.667
Karpinski, Zelikovsky	1996	1.644
Hougardy, Prömel	1999	1.598
Robins, Zelikovsky	2000	1.550

2.5.1 The k-Steiner ratio

We denote the optimal k-restricted Steiner tree by $\mathbf{OPT}_k$, and its cost by $\mathbf{opt}_k$. It is clear that if k is large enough, say $k \geq |R|$, an optimal k-restricted Steiner tree is an optimal Steiner tree, i.e., $\mathbf{opt}_k = \mathbf{opt}$. But in general, the cost of an optimal k-restricted Steiner

tree may exceed that of an optimal Steiner tree. We define the *k-Steiner ratio* as

$$\rho_k = \sup_G \left\{ \frac{\mathbf{opt}_k}{\mathbf{opt}} \right\}.$$

Theorem 2 implies that $\rho_2 \leq 2$. In [45], Zelikovsky obtained an upper bound for ρ_3. Borchers and Du generalized Zelikovsky's result and proved the following theorem in [7].

Theorem 3 (Borchers-Du)**.** *For $k = 2^r + s$, where $0 \leq s < 2^r$, the k-Steiner ratio is*

$$\rho_k = \frac{(r+1)2^r + s}{r2^r + s}.$$

This theorem implies that

$$\lim_{k \to \infty} \rho_k = 1.$$

Therefore, when k is large enough, finding an optimal k-restricted Steiner tree is a good approximation for the Steiner tree problem. However, the k-restricted Steiner tree problem is also $\mathcal{NP}$-hard, for $k \geq 4$ [5].

Lemma 5. *For $k \geq 4$, the k-restricted Steiner tree problem is $\mathcal{NP}$-hard.*

Proof. We prove this lemma by reducing the vertex cover problem for graphs with maximum vertex degree $k-1$ to the k-restricted Steiner tree problem. Let $G = (V, E)$ be a graph whose maximum vertex degree is $k-1$. Now we construct a graph H from G. Let V be the set of Steiner vertices of H, and for each $e \in E$, construct a terminal t_e in H, and connect it to the endpoints of e. Also add one extra terminal t to H, and connect it to all $v \in V$. Formally,

$$\begin{aligned} V(H) &= R \cup S \\ &= V \cup (\{t_e : e \in E\} \cup \{t\}) \\ E(H) &= \{t_e v : v \text{ is an endpoint of } e\} \cup \{tv : v \in V\} \end{aligned}$$

Let the cost of every edge of H be 1. Let C^* be the minimum vertex cover of G, and let its size be c^*. Let $\mathbf{OPT}_k$ be the minimum cost k-restricted Steiner tree of H, and denote

its cost by $\mathbf{opt}_k$. We claim that

$$c^* = \mathbf{opt}_k - |E|.$$

First we show how to obtain a k-restricted Steiner tree T of H from C^*. To form T, we connect terminal t to all vertex $v \in C^*$. Since C^* is a vertex cover, for all $e \in E$, e is covered by some $v \in C^*$. Therefore, for each terminal t_e, we pick exactly one of the edges $t_e v$, such that e is covered by v. If e is covered by more than one vertex, we pick only one of them. These selected edges form a k-restricted Steiner tree of H, and

$$\mathbf{opt}_k \leq c(T) = c^* + |E|. \tag{2.7}$$

Now we show how to obtain a vertex cover from $\mathbf{OPT}_k$. If there is a degree 2 terminal t_e in $\mathbf{OPT}_k$, then at least one of its neighbours, which is a Steiner vertex v, is not connected to t. Then add edge tv, and remove edge $t_e v$, we have another k-restricted Steiner tree, which is also optimal. Therefore, we may assume that $\mathbf{OPT}_k$ has no degree 2 terminal corresponding to and edge $e \in E$. Since $\mathbf{OPT}_k$ is a Steiner tree, all t_e are spanned, i.e., all $e \in E$ are cover by one of the endpoints. Therefore, the set C of Steiner vertices in $\mathbf{OPT}_k$ is a vertex cover of G., and

$$c^* \leq |C| = \mathbf{opt}_k - |E|. \tag{2.8}$$

Equations (4.3) and (4.4) prove our claim. Also the running time of our reduction is polynomial in the size of the vertex cover problem. Since the vertex cover problem is $\mathcal{NP}$-hard for graphs with maximum vertex degree at least 3, the k-restricted Steiner tree problem is $\mathcal{NP}$-hard, for $k \geq 4$. □

Furthermore, since the vertex cover problem is $\mathcal{APX}$-hard, the k-restricted Steiner tree problem is also $\mathcal{APX}$-hard for $k \geq 4$. Thus, there is no fully polynomial-time approximation scheme for the k-restricted Steiner tree problem for $k \geq 4$.

For $k = 3$, it is not known whether the problem is in $\mathcal{P}$ or not. Prömel and Steger [35] showed that there is a randomized polynomial time $1 + \varepsilon$ approximation algorithm for 3-restricted Steiner tree problem.

2.5.2 A framework of greedy algorithms for Steiner tree problem

Almost all approximation algorithms for the Steiner tree problem are greedy algorithms and fit in the framework shown in **Algorithm 3**.

Algorithm 3 A greedy algorithm framework

1: $T \leftarrow$ minimum spanning tree on R.
2: $\mathcal{K} \leftarrow$ the set of k-restricted full components.
3: $i \leftarrow 1$
4: **while** There is an improving full component **do**
5: Select $K_i \in \mathcal{K}$ that minimizes a selection function $f(K)$.
6: $i \leftarrow i + 1$.
7: **end while**
8: **return** A Steiner tree using $K_1, K_2, ..., K_{i-1}$

A *terminal-spanning tree* is defined as a Steiner tree that does not contain any Steiner vertices. All greedy algorithms, that fit in the framework, start with a minimum spanning tree on R, which is terminal spanning tree. Then the algorithms try to find a k-restricted full component, such that including it into the solution will yield a better result. If there are more than one such k-restricted full component, the algorithm greedily chooses the one that minimizes a specific selection function. Therefore, by specifying a selection function $f(K)$, we define an algorithm that fits in this framework.

Let k be a fixed integer. By the triangle inequality, we may assume that full components with degree 2 Steiner vertices will never be selected in our greedy algorithm. Then, a k-restricted full component has at most $k - 2$ Steiner vertices. Hence, there are at most $|R|^{k-2}|S|^k$ possible choices for each iteration in our greedy algorithms. This proves that any approximation algorithm fits into the framework is a polynomial time algorithm.

Suppose our algorithm computes a k-restricted Steiner tree within a constant factor α_k of the optimal k-restricted Steiner tree, then it is an approximation algorithm for the Steiner tree with approximation ratio $\rho_k\alpha_k$. If α_k is independent of k, i.e., $\alpha_k = \alpha$, by Theorem 3, as $k \to \infty$, we have a sequence of algorithms, and the approximation ratio converge to α.

In 1990, Zelikovsky introduced a $\frac{11}{6}$-approximation algorithm [45], which considers only 3-restricted full components that improve the cost of the Steiner tree. Later, Borchers

and Du extended Zelikovsky's idea to k-restricted Steiner tree, in [7]. In [43], Zelikovsky considered the "relative" cost improvement in the selection function $f_n(K)$, which means the cost improvement relative to the cost of the selected full component, instead of the absolute cost improvement, and get a better approximation ratio of 1.694.

The notion *loss* was introduced by Karpinski and Zelikovsky, which means the the cost of connecting the Steiner vertices of a full component to its terminals. The idea behind loss is that we would like to choose Steiner vertices that are also in an optimal Steiner tree. Therefore in the algorithms, we try to choose full components with small losses. Karpinski and Zelikovsky's algorithm achieves an approximation ratio of 1.644. In 1995, Robins and Zelikovsky [39] introduced the k-restricted loss contraction algorithm (k-LCA), using a new selection function with the loss concept, and showed that k-LCA is a 1.550-approximation algorithm for the k-restricted Steiner tree problem, which is the best known approximation ratio to date. We will discuss k-LCA in more detail later in this thesis.

2.6 Approximation algorithms for Steiner tree on quasi-bipartite graphs

The Steiner tree problem in quasi-bipartite graphs is a special case of the general Steiner tree problem. In this special case, there are no edges connecting Steiner vertices. Therefore, any full component that contains a Steiner vertex is a star. Hence, in this special case, we can consider the Steiner vertices one by one. However, the Steiner tree problem is still $\mathcal{NP}$-hard in quasi-bipartite graphs [37]. This special case has been studied by many researchers. Robins and Zelikovsky's k-LCA gave a much better result on quasi-bipartite graphs than on general graphs. The approximation ratio of k-LCA is 1.279 [39].

In [37], Rajagopalan and Vazirani introduced the notion of quasi-bipartite graphs, and gave a $\left(\frac{3}{2}+\varepsilon\right)$-approximation algorithm for the Steiner tree problem in these graphs. Kahng and Robins [25] presented an algorithm (the iterated 1-Steiner heuristic), which always produces a Steiner tree whose cost is at most $\frac{3}{2}$ of the cost of the optimal Steiner tree.

The iterated 1-Steiner heuristic is a local search algorithm; Rizzi [38] proved that it can be implemented in polynomial time of the input by using the *bit scaling* technique [38].

The bit scaling technique creates a sequence of Steiner tree problem instances $I_0, I_1, ..., I_k$. I_0 is the original instance, and for all $0 < i \leq k$, I_i is obtained by dividing the edge costs of I_{i-1} by 2, i.e., $c_i(e) = \left\lfloor \frac{1}{2} c_{i-1}(e) \right\rfloor$, for all $e \in E$. I_k is the instance in which the edges costs are 0 or 1. Rizzi's implementation starts from I_k, for each iteration, the input is the output tree of the last iteration. This implementation is proved to run in polynomial time. We will discuss the iterated 1-Steiner heuristic in more detail in the next chapter.

For a much more special case, the Steiner tree in uniform quasi-bipartite graphs, i.e., on quasi-bipartite graphs with the same cost on all edges, Gröpl, Hougardy, Nierhoff and Prömel introduced an algorithm, which produces a Steiner tree that costs at most 1.217 times the cost of an optimal solution.

2.7 Integer programming formulations for the Steiner forest problem

Aneja gave a formulation similar to the cut formulation of Steiner tree problem [3]. For all edge $e \in E$, let x_e be a binary variable associate with e. In a solution, $x_e = 1$ represents we include edge e into our Steiner forest. We call a cut $\delta(U)$ a *Steiner cut*, if for some i, the terminal pairs $(s_i, t_i) \in \mathcal{R}$, and $s_i \in U, t_i \in \overline{U}$. The formulation is the following.

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e & & & (\mathrm{IP_{SF}}) \\
\text{s.t.} \quad & \sum_{e \in \delta(U)} x_e \geq 1 & & \forall U \in \mathcal{U} & (2.9) \\
& x_e \in \{0, 1\} & & \forall e \in E &
\end{aligned}$$

In the formulation, $\mathcal{U}$ is the collection of subset U of V, such that $\delta(U)$ is a Steiner cut. The dual of $\mathrm{IP}_{\mathrm{SF}}$ is

$$\begin{aligned} \max \quad & \sum_{U \in \mathcal{U}} y_U && && (\mathrm{D}_{\mathrm{SF}}) \\ \text{s.t.} \quad & \sum_{U \in \mathcal{U}: e \in \delta(U)} y_U \leq c_e && \forall e \in E && (2.10) \\ & y \geq 0 \end{aligned}$$

In [30] [29], Könemann et al. gave a new integer programming formulation for the Steiner forest problem. Their undirected formulation is strictly stronger than that of Aneja.

2.8 A primal-dual based strict algorithm for Steiner forests

In this section we review a $\left(2 - \frac{1}{k}\right)$-approximate primal-dual algorithm for Steiner forest problem. The algorithms for Steiner forest presented in [2] and [17] differ only slightly. In this thesis, we focus on the viewpoint taken in [2]. We use AKR to refer to this algorithm.

The primal-dual algorithm AKR constructs both a feasible primal and a feasible dual solution for a linear programming formulation of the Steiner forest problem and its dual, respectively. Algorithm AKR constructs a primal solution for (LP) and a dual solution for (D_{SF}). The algorithm has two goals:

Compute a feasible solution for the given Steiner forest instance. The algorithm reduces the degree of infeasibility as it progresses.

Create a dual feasible packing of sets of largest possible total value. The algorithm raises dual variables of certain subsets of nodes at all times. The final dual solution is maximal in the sense that no single set can be raised without violating a constraint of type (2.10).

Consider the execution of algorithm AKR as a process over time and let x^τ and y^τ be the primal incidence vector and feasible dual solution at time τ. Note that in any optimal solution to (IP_{SF}), $x_e \in \{0,1\}$. Let F^τ denote the forest corresponding to the set of edges with $x_e^\tau = 1$. Initially, let $x_e^0 = 0$ for all $e \in E$ and $y_U^0 = 0$ for all $U \in \mathcal{U}$. The algorithm maintains the invariant $x_e^\tau \leq x_e^{\tau'}$ and $y_e^\tau \leq y_e^{\tau'}$ for all $\tau < \tau'$.

An edge $e \in E$ is *tight* if the corresponding constraint (2.10) holds with equality; and a path is *tight* if every edge in the path is tight. Assume that the forest F^τ at time τ is infeasible. A terminal node $v \in R$ is *active* at time τ if v and its *mate* $\bar{v}$, i.e., $(v, \bar{v}) \in R$, are in different trees in the forest F^τ; v is *inactive* otherwise. Let $\bar{F}^\tau$ denote the subgraph of G that is induced by the tight edges for dual y^τ. To avoid confusion between connected components in F^τ and those in $\bar{F}^\tau$, the term *moat* refers to a connected component in $\bar{F}^\tau$. The algorithm maintains that if $C \in F^\tau$ then $C \subseteq U \in \bar{F}^\tau$. A moat U of $\bar{F}^\tau$ is active at time τ if U contains an active terminal; U is inactive otherwise. Let A^τ be the set of all active moats in $\bar{F}^\tau$ at time τ. AKR raises the dual variables for all sets in A^τ uniformly at all times $\tau \geq 0$, so that if U is active from time τ' until time τ'', then $y^U = \tau'' - \tau'$.

Two disjoint moats *collide* at time τ in the execution of AKR if there is a path in G from one moat to the other that becomes tight at time τ. In order for this to happen, at least one of the two moats must be active. Suppose that a path P connecting two *active* terminals u and u' becomes tight at time τ in the execution of AKR. Then u is contained in some active moat U and u' is in a disjoint active moat U'. When this happens, AKR adds the edges of P not already in F^τ to F^τ: that is, for all $e \in P$, the algorithm sets $x_e^\tau = 1$. For $\tau' > \tau$, sets U and U' are part of the same moat of $\bar{F}^{\tau'}$.

Subsequently, we use $U^\tau(v)$ to refer to the moat in $\bar{F}^\tau$ that contains node $v \in V$ at time τ. Similarly, we let $U^\tau(C)$ denote the moat in $\bar{F}^\tau$ that contains the connected component $C \in F^\tau$ at time τ. We define the age of a connected component C as the first time τ at which $C \in F^\tau$, and denote this by (C). Let F be the final forest.

The following is the main theorem of [2]:

Theorem 4. *Let F be the forest computed by AKR on terminal set R. We then have $c(F) \leq \left(2 - \frac{1}{k}\right) \cdot \mathbf{opt}_R$, where $\mathbf{opt}_R$ is the minimum cost of a Steiner forest for the given input instance with terminal set R.*

Chapter 3

Iterated 1-Steiner heuristic for the Steiner tree problem on quasi-bipartite graphs

In this chapter, we will prove that iterated 1-Steiner heuristic is a $\frac{3}{2}$-approximation algorithm for the Steiner tree problem on quasi-bipartite graphs by using duality theory of linear programming, and give a polynomial implementation of the heuristic which was introduced by Rizzi in [38].

We start by reviewing Kruskal's minimum-cost spanning tree algorithm [31] and show how it can be interpreted as a primal-dual algorithm.

3.1 A primal-dual interpretation of Kruskal's algorithm

The integer programming formulation for the minimum spanning tree problem based on (2.2) is:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e && && \text{(IP)} \\ \text{s.t.} \quad & \sum_{e \in E_\pi} x_e \geq r(\pi) - 1 && \forall \pi \in \Pi \\ & x_e \in \{0, 1\} && \forall e \in E \end{aligned}$$

The dual of the linear relaxation (LP) of (IP) is:

$$\begin{aligned} \max \quad & \sum_{\pi \in \Pi} y_\pi (r(\pi) - 1) && && \text{(D)} \\ \text{s.t.} \quad & \sum_{\pi : e \in E_\pi} y_\pi \leq c_e && \forall e \in E \\ & y_\pi \geq 0 && \forall \pi \in \Pi \end{aligned}$$

Recall that Π is the set of all feasible partition of V.

Kruskal's algorithm starts with an empty subgraph T of G. At each iteration, we add one edge e of G with minimum cost to T, such that no cycle is created by adding e to T. The algorithm terminates when T become a spanning tree of G. The algorithm is presented below.

In the algorithm, an edge e being tight means the dual constraint

$$\sum_{\pi : e \in E_\pi} y_\pi \leq c_e,$$

holds with equality.

Theorem 5. *T is a minimum spanning tree of G.*

Algorithm 4 Kruskal's minimum spanning tree algorithm

$y_\pi \leftarrow 0,\ \forall \pi \in \Pi$.
$x_e \leftarrow 0,\ \forall e \in E$.
$T_1 \leftarrow (V, \varnothing)$.
$\pi_1 \leftarrow$ the partition of V, such that each part of the partition is a singleton vertex.
$i \leftarrow 1$
while x is not feasible **do**
    increase y_{π_i} until some edge $e_i \in E_\pi$ is tight.
    $x_{e_i} \leftarrow 1$.
    $T_{i+1} \leftarrow T_i + e_i$.
    Merge the parts of π_i connected by e_i to get π_{i+1}.
    $i \leftarrow i + 1$.
end while
$T \leftarrow T_i$.
return $T,\ x,\ y$.

Proof. By the algorithm, we have

$$y_{\pi_i} = c_{e_i} - c_{e_{i-1}} \tag{3.1}$$

where we define $c_{e_0} = 0$. Notice that $r(\pi_i) = n - i + 1$. We also have

$$E_{\pi_i} \cap T = \{e_i, e_{i+1}, ..., e_{n-1}\}.$$

Then,

$$c(T) = \sum_{i=1}^{n-1} c_{e_i} = \sum_{i=1}^{n-1} \sum_{j=1}^{i} (c_{e_j} - c_{e_{j-1}}) = \sum_{i=1}^{n-1} \sum_{j=1}^{i} y_{\pi_i}$$

Exchanging the order of summation on the right-hand-side of the above equation, we get

$$c(T) = \sum_{i=1}^{n-1} y_{\pi_i}(n - i) = \sum_{i=1}^{n-1} y_{\pi_i}(r(\pi_i) - 1).$$

Now we prove that y is feasible for (D). For any $e \in E$, let $1 \leq i \leq n - 1$ be the largest

index such that $e \in E_{\pi_i}$. Then

$$\sum_{\pi: e \in E_\pi} y_\pi = \sum_{j=1}^{i} y_{\pi_j} = \sum_{j=1}^{i} (c_{e_j} - c_{e_{j-1}}) = c_{e_i}.$$

Since $e \in E_{\pi_i}$, e connects two connected components of T_i, it is a candidate for Kruskal's algorithm for the i-th iteration, therefore,

$$c_{e_i} \leq c_e.$$

This proved that y is dual feasible. Using week duality, it follows that T is optimal. □

3.2 Iterated 1-Steiner heuristic

The iterated 1-Steiner heuristic keeps a set of useful Steiner vertices S' (initially, S' is the empty set) and always maintains a minimum spanning tree T in $G[R \cup S']$. Basically, the following three operations are applied as long as possible:

1. If there is a Steiner vertex $v \in V \setminus (R \cup S')$ such that $\mathbf{mst}(G[R \cup S' \cup \{v\}]) < c(T)$ then add v to S'.
2. If T has a Steiner leaf $v \in S' \setminus R$ then remove v from S'.
3. If T has a degree 2 Steiner vertex $v \in S' \setminus R$ that is incident to edges (u, v) and (v, w) from T then replace these edges by edge (u, w) in T. Delete v from S' (this is called a *short-cutting* operation).

Theorem 6. *Algorithm 5 is $\frac{3}{2}$-approximation for Steiner tree problem for quasi-bipartite graphs.*

In the following, we denote the set of all Steiner partitions in the graph $G[R \cup S']$ by $\tilde{\Pi}$. Recall that Π_S is the set of all Steiner partitions in G.

Let $X = S \setminus S'$ be the set of Steiner vertices not in T. For a vertex $v \in X$, let $\mathcal{U}_v$ be the set of all stars centered at v, i.e., $\mathcal{U}_v$ is the collection of all subset of $\delta(v)$. We define $\mathcal{U}$ as the union of all $\mathcal{U}_v$'s.

Algorithm 5 Iterated 1-Steiner heuristic.

Given: Undirected quasi-bipartite graph $G = (V, E)$, cost function $c : E \to \mathbb{R}^+$, and terminals R
$S' \leftarrow \varnothing, T \leftarrow \mathbf{MST}(G[R \cup S'])$
while $\exists v \in V \setminus (R \cup S')$ s.t. $\mathbf{mst}(G[R \cup S' \cup \{v\}]) < c(T)$ **do**
  $T \leftarrow \mathbf{MST}(G[R \cup S' \cup \{v\}])$
  Remove Steiner leaves together with their incident edges from T
  Short-cut degree 2 Steiner vertices
  Let S' be the set of Steiner vertices in T
end while
return T

Consider a Steiner partition $\pi \in \tilde{\Pi}$, for a star $U \in \mathcal{U}_v$, we define the *rank contribution* of U with respect to π as

$$\mathbf{rc}_U^\pi = |\{S \in \pi \big| \exists u \in S, uv \in U\}| - 1.$$

Now we present a new integer programming formulation for the Steiner tree problem. For every edge $e \in E_{S'}$, there is a binary variable x_e corresponding to e, where $E_{S'}$ is the the set of edges whose both endpoints are in $R \cup S'$. We let $x_e = 1$ if e is included in the corresponding solution, and $x_e = 0$ otherwise. There is a variable x_U for every $U \in \mathcal{U}$, and we let $x_U = 1$ if and only if U is a part of the corresponding solution.

$$\min \quad \sum_{e \in E_{S'}} c_e x_e + \sum_{U \in \mathcal{U}} c(U) x_U \tag{IP$_S$}$$

$$\text{s.t} \quad \sum_{e \in E_\pi} x_e + \sum_{U \in \mathcal{U}} \mathbf{rc}_U^\pi x_U \geq r(\pi) - 1 \qquad \forall \pi \in \tilde{\Pi} \tag{3.2}$$

$$x_e, x_U \in \{0, 1\} \qquad \forall e \in E_{S'}, U \in \mathcal{U} \tag{3.3}$$

In [21], Grötschel, Monma and Stoer proved that the separating problem for partition inequalities is $\mathcal{NP}$-hard. Since (3.2) is a generalization of the partition inequalitis, separating (3.2) is also $\mathcal{NP}$-hard. Hence we cannot solve the linear progamming relaxation of (IP$_S$) in polynomial time, unless $\mathcal{P} = \mathcal{NP}$.

Lemma 6. *An optimum solution to (IP$_S$) is a minimum-cost Steiner tree in G.*

Proof. Let T^* be a minimum cost Steiner tree in G. Let S^* be the set of Steiner vertices in T^*. For a vertex $v \in S^* \cap X$, let

$$U_v^* = \{e \in \delta(v) \big| e \in E[T^*]\}$$

be the set of edges in T^* that are incident to v. Let $x^*_{U_v^*} = 1$ for all $v \in S^* \cap X$, and $x^*_U = 0$ for all other $U \in \mathcal{U}$. Let $x^*_e = 1$ if $e \in E(T^*)$ is not incident to a vertex in X, and $x^*_e = 0$ otherwise. Clearly,

$$c(T^*) = \sum_{e \in E_{S'}} x^*_e + \sum_{U \in \mathcal{U}} c(U) x^*_U.$$

Now we wants to show that x^* is feasible for (IP$_S$).

Let $\pi \in \tilde{\Pi}$ be a partition of the vertices in $R \cup S'$. We will now create a new partition $\overline{\pi}$ from π by contracting the stars in T^*. For each vertex $v \in S^* \cap X$, merge the parts of π that are connected to v by U_v^*, and include v into the newly created part. Then we have

$$r(\overline{\pi}) \geq r(\pi) - \sum_{v \in S^* \cap X} \mathbf{rc}^{\pi}_{U_v^*}.$$

Clearly, $\overline{\pi}$ is a Steiner partition of graph G. By partition inequality (2.3), the number of edges in $E(T^*) \cap E_{\overline{\pi}}$ is at leat

$$r(\overline{\pi}) - 1 \geq r(\pi) - \sum_{v \in S^* \cap X} \mathbf{rc}^{\pi}_{U_v^*} - 1.$$

By the construction of $\overline{\pi}$, none of the edges in $E_{\overline{\pi}}$ has an endpoint in X, therefore

$$\sum_{e \in E_{S'}} x^*_e \geq r(\pi) - \sum_{v \in S^* \cap X} \mathbf{rc}^{\pi}_{U_v^*} - 1$$

or

$$\sum_{e \in E_{S'}} x^*_e + \sum_{v \in S^* \cap X} \mathbf{rc}^{\pi}_{U_v^*} x^*_{U_v^*} = \sum_{e \in E_{S'}} x^*_e + \sum_{v \in S^* \cap X} \mathbf{rc}^{\pi}_{U_v^*} \geq r(\pi) - 1.$$

This proves that x^* is feasible for (IP$_S$).

Now assume that x^* is a optimal solution to (IP_S), we construct T^* by letting

$$T^* = \bigcup_{v \in X} \bigcup_{\substack{U \in \mathcal{U}_v \\ x^*_U = 1}} U \cup \{e \in E_{S'} \,\big|\, x^*_e = 1\}.$$

We prove that T^* is a feasible Steiner subgraph for G. Assume for the sake of contradiction that T^* is infeasible. Then there must exist a Steiner cut $\delta(C)$ such that $E[T^*] \cap \delta(C) = \varnothing$. Let π be the partition with parts C and $V \setminus C$. Since x^* is feasible for (IP_S), we must have

$$\sum_{e \in E_\pi} x^*_e + \sum_{U \in \mathcal{U}} \mathbf{rc}^\pi_U x^*_U \geq r(\pi) - 1 = 1.$$

But $E[T^*] \cap \delta(C) = \varnothing$, we must have $\mathbf{rc}^\pi_U = 0$ for all $U \in \mathcal{U}$ with $x^*_U = 1$ and therefore

$$\sum_{e \in E_\pi} x^*_e \geq 1$$

contradicting the fact that T^* has no edges crossing the cut C. □

Replacing the integrality constraints (3.3) by non-negativity constraints, we obtain the standard linear programming relaxation (LP_S) of (IP_S). The dual of (LP_S) is as follows:

$$\begin{aligned}
\max \quad & \sum_{\pi \in \tilde{\Pi}} (r(\pi) - 1) y_\pi && && (\mathrm{D}_S) \\
\text{s.t} \quad & \sum_{\pi : e \in E_\pi} y_\pi \leq c_e && \forall e \in E_{S'} && (3.4) \\
& \sum_{\pi \in \tilde{\Pi}} \mathbf{rc}^\pi_U y_\pi \leq c(U) && \forall U \in \mathcal{U} && (3.5) \\
& y_\pi \geq 0 && \forall \pi \in \tilde{\Pi}
\end{aligned}$$

We now show that the dual $\overline{y}$ computed by Algorithm 5 can be converted into a feasible dual solution y for (D_S) whose objective function value is at least $\frac{2}{3}$ that of $\overline{y}$. The conversion is done in two steps. First we show how we can convert each of the partitions

$\overline{\pi}_i$ of $G[R \cup S']$ into a Steiner partition π_i of not much smaller rank. We then let

$$y_{\pi_i} = \overline{y}_{\overline{\pi}_i}$$

for $1 \leq i \leq p-1$ and show that y is feasible for (D_S), where p is the number of vertices in $R \cup S'$.

First consider an arbitrary $1 \leq i \leq p-1$ and assume that $\overline{\pi}_i = (V_1, \ldots, V_q)$. This partition may contain singleton Steiner vertices, i.e. $V_j = \{v\}$ for some $1 \leq j \leq q$ and for some $v \in S'$. We obtain π_i from $\overline{\pi}_i$ by merging each such singleton Steiner vertex v with one of the parts of $\overline{\pi}_i$ containing a terminal neighbor of v. Let s_i be the number of singleton Steiner vertices in $\overline{\pi}_i$. The rank of π_i is

$$r(\pi_i) = r(\overline{\pi}_i) - s_i.$$

Obtain tree T_i from T by identifying the vertices in each part of $\overline{\pi}_i$. The rank of $\overline{\pi}_i$ is equal to the number of vertices in T_i. By assumption, each Steiner vertex $v \in S'$ has degree at least 3 in T. Therefore, each singleton Steiner vertex has degree at least 3 in T_i and the number of vertices in T_i is at least

$$s_i + 3s_i - (s_i - 1) = 3s_i + 1.$$

In the above equation, since T_i is a tree, there are least $s_i - 1$ vertices are double counted when we add up the number of the vertices for all full components. (To see this, consider the graph H, whose vertex set is the set of all full components of T, and the edge set is the set of adjacent pairs of full components. Since T is a tree, then H is also a tree, and $|V(H)| \geq s_i$. Therefore, $|E(H)| \geq s_i - 1$, i.e., at least $s_i - 1$ vertices in T_i is double counted.) Therefore, we need to subtract the double counted number $s_i - 1$.

Equivalently, we obtain

$$s_i \leq \frac{1}{3} r(\overline{\pi}_i) - \frac{1}{3} \tag{3.6}$$

and hence

$$(r(\pi_i) - 1) = (r(\overline{\pi}_i) - s_i - 1) \geq \left(\frac{2}{3} r(\overline{\pi}_i) - \frac{2}{3}\right) = \frac{2}{3}(r(\overline{\pi}_i) - 1)$$

where the inequality uses (3.6).

Let $y_{\pi_i} = \overline{y}_{\overline{\pi}_i}$ for all $1 \leq i \leq p-1$. We claim that y is feasible for (D) in graph $G[R \cup S']$. In order to see this, notice that for any $e \in E_{S'}$, $e \in E_{\pi_i}$ implies that $e \in E_{\overline{\pi}_i}$ as well and hence

$$\sum_{i:e \in E_{\pi_i}} y_\pi \leq \sum_{i:e \in E_{\overline{\pi}_i}} \overline{y}_{\overline{\pi}_i} \leq c_e \tag{3.7}$$

where the last inequality uses the feasibility of $\overline{y}$ for (D) in graph $G[R \cup S']$.

Lemma 7. *There is a feasible dual $\{y_{\pi_i}\}_{i=1}^{p-1}$ for (D) in graph $G[R \cup S']$ such that*

$$\sum_{i=1}^{p-1} (r(\pi_i) - 1) y_{\pi_i} \geq \frac{2}{3} \cdot \sum_{i=1}^{p-1} (r(\overline{\pi}_i) - 1) \overline{y}_{\overline{\pi}_i}$$

and $\pi_1, \ldots, \pi_{p-1}$ are Steiner partitions in $G[R \cup S']$.

We now show that dual y is feasible for (D_S). Notice that y satisfies constraints (3.4) as it is feasible for (D) in graph $G[R \cup S']$ (see also (3.7)). Now consider a star $U \in \mathcal{U}_v$ for some $v \in X$. Assume for the sake of contradiction that (3.5) is violated for U and hence

$$\sum_{i=1}^{p-1} \mathbf{rc}_U^{\pi_i} y_{\pi_i} > c(U). \tag{3.8}$$

Notice first that $\mathbf{rc}_U^{\pi_i} = \mathbf{rc}_U^{\overline{\pi}_i}$ for all $1 \leq i \leq p-1$ as π_i is obtained from $\overline{\pi}_i$ by merging a singleton Steiner vertex with an adjacent terminal. Let $q = |U|$ be the number of tips in the star defined by U and observe that $\mathbf{rc}_U^{\pi_1} = q - 1$ and $\mathbf{rc}_U^{\pi_p} = 0$, recall that p is the number of vertices in T. Then let

$$0 \leq i_1 < i_2 < \ldots < i_q \leq p - 1$$

s.t. $\mathbf{rc}_U^{\pi_{i_j+1}} = \mathbf{rc}_U^{\pi_{i_j}} - 1$ for all $j \in \{1, \ldots, q\}$. This immediately implies that

$$e_{i_1}, e_{i_2}, \ldots, e_{i_q}$$

lie on pairwise disjoint cycles in $T+U$ (the graph obtained from T by adding the edges of U). In other words, removing $e_{i_1}, \ldots, e_{i_q}$ from $T+U$ yields a tree spanning T' of $R \cup S' \cup \{v\}$.

The cost of edge e_{i_j} for $1 \leq j \leq q$ is $\sum_{i=1}^{i_j} y_{\pi_i}$ and thus

$$\sum_{j=1}^{q} c(e_{i_j}) = \sum_{j=1}^{q} \sum_{i=1}^{i_j} y_{\pi_i} = \sum_{i=1}^{p-1} \mathbf{rc}_U^{\pi_i} y_{\pi_i}.$$

Inequality (3.8) implies that

$$c(T') = c(T) + c(U) - \sum_{j=1}^{q} c_{e_{i_j}} < c(T)$$

and this contradicts the termination condition of Algorithm 5.

Lemma 8. *Dual $\{y_{\pi_i}\}_{i=1}^{p-1}$ is feasible for (D_S).*

Since

$$c(T) = \sum_{i=1}^{p-1} c_{e_i} = \sum_{i=0}^{p-2} (r(\overline{\pi}_i) - 1),$$

Lemma 7 and Lemma 8 imply Theorem 6.

3.3 Polynomial time implementation of iterated 1-Steiner heuristic

In this section, we will show how to use Rizzi's [38] bit scaling technique to develop a polynomial time implementation of Algorithm 5.

Let the original instance of the Steiner tree problem be I_0, and define the cost function of I_0 as $c_0(\cdot) = c(\cdot)$. We define a sequence of instances $I_1, I_2, \ldots, I_k$ of Steiner tree problems on the same graph, where $c_i = \lfloor \frac{1}{2} c_{i-1} \rfloor$, and k be the smallest index such that $c_k(e) \leq 1$, for all $e \in E$. Hence, $k = \log_2 \max_{e \in E} c(e)$. We denote the output of instance I_i as T_i, and the optimal Steiner tree of instance I_i as T_i^*.

Consider the instance I_k. Since all edge costs are 0 or 1, the while-loop steps 3-8 in Algorithm 5 will terminate after at most n iterations, and we have $c_k(T_k) \leq \frac{3}{2} c_k(T_k^*)$.

Now, for $0 \leq i \leq k$, Algorithm 5 is a $\frac{3}{2}$-approximation algorithm, $c_i(T_i) \leq \frac{3}{2}c_i(T_i^*)$. Then the difference of the costs between T_i and T_{i-1}^* using cost function c_i is

$$c_i(T_i) - c_i(T_{i-1}^*) \leq c_i(T_i) - c_i(T_i^*) \leq \frac{1}{2}c_i(T_i^*), \tag{3.9}$$

where the first inequality is true because T_i^* is an optimal Steiner tree using cost function $c_i(\cdot)$.

Since every tree has at most $n-1$ edges, $c_{i-1}(T_i^*) \leq 2c_i(T_i^*) + n$. By the definition of c_i, we also have $c_{i-1}(T_{i-1}^*) \geq 2c_i(T_{i-1}^*)$. Therefore, we have

$$c_{i-1}(T_i^*) - c_{i-1}(T_{i-1}^*) \leq 2c_i(T_i^*) + n - 2c_i(T_{i-1}^*) \leq n. \tag{3.10}$$

Now we want to know the difference between $c_{i-1}(T_i)$ and $c_{i-1}(T_{i-1}^*)$. Similar to the discussion of inequality (3.10), we have

$$c_{i-1}(T_i) - c_{i-1}(T_{i-1}^*) \leq 2c_i(T_i) + n - 2c_i(T_{i-1}^*) \leq n + 2(c_i(T_i) - c_i(T_{i-1}^*)). \tag{3.11}$$

By inequality (3.9), (3.11) becomes

$$c_{i-1}(T_i) - c_{i-1}(T_{i-1}^*) \leq n + c_i(T_i^*) \leq n + \frac{1}{2}c_{i-1}(T_i^*). \tag{3.12}$$

By inequality (3.10), (3.12) becomes

$$c_{i-1}(T_i) - c_{i-1}(T_{i-1}^*) \leq n + \frac{1}{2}(n + c_{i-1}(T_{i-1}^*)) = \frac{3}{2}n + \frac{1}{2}c_{i-1}(T_{i-1}^*);$$

i.e.,

$$c_{i-1}(T_i) \leq \frac{3}{2}n + \frac{3}{2}c_{i-1}(T_{i-1}^*). \tag{3.13}$$

Since c_{i-1} is integral, after iterating the while-loop step 3-8 in Algorithm 5 at most $\frac{3}{2}n$ times, we have an output tree T_{i-1} of cost at most $\frac{3}{2}$ times of T_{i-1}^* using cost function c_{i-1}. Notice that in the implementation, we change the stopping condition of the algorithm, since it is guaranteed that, after $\frac{3}{2}n$ iterations, we obtain a Steiner tree of cost at most $\frac{3}{2}$

times of the optimal cost. Therefore, we can stop and proceed to the next instance.

Therefore, we can implement algorithm 5 in polynomial time. This result is stated in the following theorem.

Theorem 7 (Rizzi, [38]). *The iterated 1-Steiner heuristic for Steiner tree problem on quasi-bipartite graphs can be implemented in polynomial time, and the running time is* $O(knrT(G))$*, where* $k = \log_2 \max_{e \in E} c(e)$ *is the number of digits required to represent the maximum value of the cost of edges,* $n = |V|$ *is the number of vertices,* $r = |R|$ *is the number of terminals, and* $T(G)$ *is the running time for computing the minimum spanning tree of* G*.*

Chapter 4

The greedy algorithm of Robins and Zelikovsky

The currently best known approximation algorithm for the Steiner tree problem is due to Robins and Zelikovsky [39]. Their algorithm has an approximation ratio of $1 + \frac{\ln 3}{2} \approx 1.55$. The algorithm performs better in quasi-bipartite graphs where it obtains an approximation ratio of ≈ 1.28. In the chapter, we will revise Robins and Zeikovsky's proof of the approximation ratio; then we will extend the ideas presented for the quasi-bipartite special case to general graphs and thereby provide a new proof of this result.

4.1 Definitions basic properties

Before introducing the algorithm, we define some terms needed in the algorithm.

- A *terminal-spanning tree* is a Steiner tree that does not contain any Steiner vertex.
- The *gain* of a subgraph H of G with respect to a terminal-spanning tree T is the cost reduction obtained by including H in T. Formally, we define the gain as following:

  $$g_T(H) = \max\{0, c(T) - c(T[H])\},$$

  where $T[H]$ is a minimum cost subgraph in $H \cup T$ that contains H and spans all

terminals of T. Note that if H is a Steiner tree, then $T[H]$ is just H itself.

- The *loss* $L(K)$ of a full component K represents the cost increase by including K in our solution, while the optimal solution does not contain K. Formally, loss of H is defined as the minimum cost forest that connects each Steiner vertex of K to some terminal of K. The cost of $L(K)$ is denoted by $l(K)$. Let K be a full component, $C[K]$ denotes the terminal-spanning tree over the terminals of K obtained by contracting $L(K)$. Please refer to *Fig. 4.1.* In the figure, the numbers on the edges of K are the costs of the edges. For a Steiner tree H, $C[H]$ denotes the union of $C[K]$ for all full component K of H, $L(H)$ denotes the union of $L(K)$, and $l(H)$ is the cost of $L(H)$.

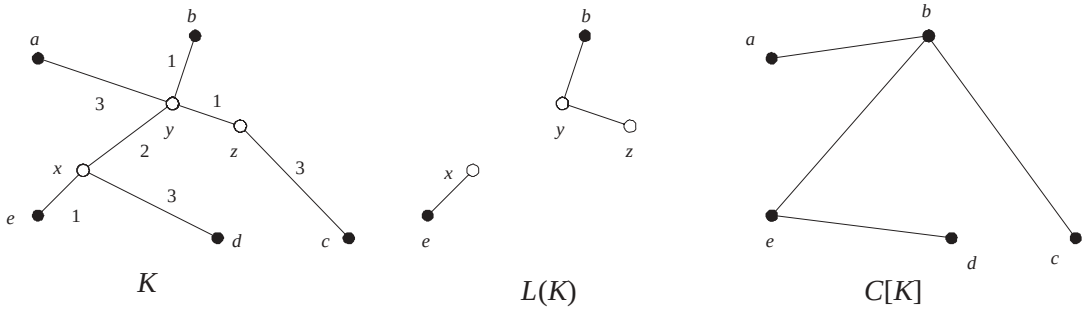


Figure 4.1: Example of a full component K, the Loss of the full component and $C[k]$ on general graphs.

- For a terminal-spanning tree T and a subgraph H, we define the *drop set* of H to be the set of edges of T that do not appear in $T[H]$, i.e., $D_T(H) = E(T) \setminus E(T[H])$. To simplify notation, we use $D(H)$ instead of $D_T(H)$ to refer to the drop set of H if no ambiguity exists. Please remark that the gain of H with respect to T can also be expressed as $g_T(H) = \max\{0, c(D(H)) - c(H)\}$.

If our algorithm accepts one full component K, the cost of our output will decrease by $g_T(K)$. But if the full component is not part of the optimal solution, our solution needs to connect the Steiner vertices in K while **OPT** does not. That is, we may pay up to $l(K)$ more than **OPT**.

Lemma 9. *Let T be a terminal-spanning tree, and let H and H' be two subgraphs of G, such that the sets of Steiner vertices of H and H' are disjoint, then*

$$g_T(H \cup H') \leq g_T(H) + g_T(H').$$

Proof. Clearly, if $g_T(H \cup H') = 0$, by the definition of gain, the lemma is true. Now we suppose that $g_T(H \cup H') > 0$.

First we prove that

$$c(D(H \cup H')) = c(D(H)) + c(D(H')). \tag{4.1}$$

Without loss of generality, we may assume the cost of any edge is distinct, if ties occur, we may increase the cost of one of the tie edges by a small amount to break ties. Hence, it is sufficient to prove that $D(H \cup H') = D(H) \cup D(H')$.

To prove that $D(H) \cup D(H') \subseteq D(H \cup H')$, we consider an edge $e \in D(H) \cup D(H')$. W.l.o.g., we may assume $e \in D(H)$. Then, e is a longest edge on a cycle C of $T \cup H$. Clearly C is also a cycle of $T \cup H \cup H'$. By our edge cost uniqueness assumption, C has only one longest edge, which is e. Therefore, e must also be dropped when we construct $T[H \cup H']$, i.e., $e \in D(H \cup H')$.

Conversely, suppose that there exist an edge $e \in D(H \cup H') \setminus (D(H) \cup D(H'))$. Let C be a cycle (edge set) in $T \cup H \cup H'$ such that $e \in C$, and C is edge-minimal, i.e., there does not exist a subset P' of $C \setminus E(T)$, such that $P' \cup T$ contains a cycle. Let $P = C \setminus E(T)$. Since C is minimum, P must be a path in $H \cup H'$ connecting two terminals without internal terminals. Please refer to *Fig. 4.2.* Because the sets of Steiner vertices of H and H' are disjoint, $P \subseteq H$ or $P \subseteq H'$. Therefore, $e \in D(H)$ or $e \in D(H')$. This shows that $D(H \cup H') = D(H) \cup D(H')$, and $c(D(H \cup H')) = c(D(H)) + c(D(H'))$ follows.

Now we prove the lemma. By the remark after the definition of drop set, we have

$$g_T(H \cup H') = c(D(H \cup H')) - c(H \cup H').$$

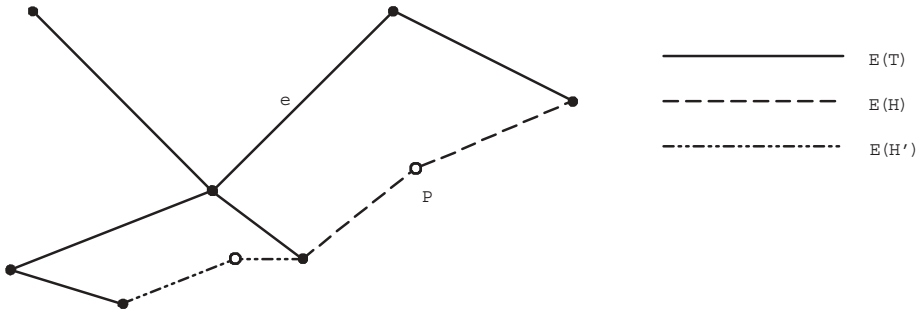


Figure 4.2: Lemma 9.

Substituting $c(H \cup H') = c(H) + c(H')$ and (4.1) into above equation, we get

$$\begin{aligned} g_T(H \cup H') &\leq c(D(H)) - c(H) + c(D(H')) - c(H') \\ &\leq g_T(H) + g_T(H') \end{aligned}$$

□

Lemma 10. *Let H be a Steiner tree. If $g_{C[H]}(K) = 0$ for all k-restricted full components K, then*

$$c(H) - l(H) = c(C[H]) \leq \mathbf{opt}_k.$$

Proof. By the definition of $C[H]$ and $L(H)$, we have $c(H) - l(H) = c(C[H])$. Let $K_1, K_2, ..., K_p$ be the full components of $\mathbf{OPT}_k$. By definition of gain,

$$c(C[H]) - \mathbf{opt}_k = g_{C[H]}(\mathbf{OPT}_k).$$

However, $\mathbf{OPT}_k$ is the disjoint union of $K_1, K_2, ..., K_p$, thus

$$g_{C[H]}(\mathbf{OPT}_k) = g_{C[H]}\left(\bigcup_{i=1}^{p} K_i\right).$$

By Lemma 9,

$$g_{C[H]}\left(\bigcup_{i=1}^{p} K_i\right) \leq \sum_{i=1}^{p} g_{C[H]}(K_i) = 0;$$

where the last inequality is true by the assumptions made in the lemma. □

4.2 The k-LCA algorithm

In the algorithm, we want to maximize the gain and minimize the loss when we include a full component, therefore, we would like to greedily pick the k-restricted full component with largest gain-to-loss ratio at each iteration. By the metric completion assumption, there are terminal-spanning trees of G, let T_0 be the one with smallest cost. We will start our algorithm with T_0 and include appropriate full components into our solution. Let $r_T(K)$ be the ratio of $g_T(K)$ and $l(K)$. The k-restricted Loss-Contracting Algorithm (k-LCA) is presented below. (Please note that $G[R]$ is the subgraph of G induced by all terminals R.)

Algorithm 6 k-LCA

1: $G_0 \leftarrow G[R]$, $H_0 \leftarrow G[R]$
2: $T_0 \leftarrow \text{MST}(G[R])$
3: $L_0 \leftarrow \varnothing$
4: $i \leftarrow 1$
5: **while** $\exists$ full component K such that $r_{T_{i-1}}(K) > 0$ **do**
6: $\quad r \leftarrow \max\{r_{T_{i-1}}(K) \big| K \text{ is a full component}\}$.
7: $\quad K_i \leftarrow$ a full component K such that $r_{T_{i-1}}(K) = r$ and with minimum number of edges
8: $\quad G_i \leftarrow G_i \cup C[K]$
9: $\quad T_i \leftarrow \text{MST}(G_i)$
10: $\quad L_i \leftarrow L_{i-1} \cup L(K)$
11: $\quad H_i \leftarrow G_i \cup L_i$
12: $\quad i \leftarrow i + 1$
13: **end while**
14: $T \leftarrow \text{MST}(G[H_{i-1}])$
15: **return** T

This algorithm is a generalization of the 1-Steiner heuristic in some sense. The 1-Steiner heuristic deals with quasi-bipartite graphs, it maintains a set S' of Steiner vertices and a Steiner tree $T_S = MST(G[R \cup S])$. We include a Steiner vertex $v \in V \setminus (R \cup S')$ only if its gain is positive, i.e., only if it improves our solution. In the algorithm k-LCA, we include a

full component, i.e., a set of connected Steiner vertices, only if the full component reduces the cost of the solution. However, in k-LCA, we only consider k-restricted full components, while the 1-Steiner heuristic does not restrict the number of neighbours of a Steiner vertex. This restriction is essential to have a polynomial time algorithm, otherwise, we may have $2^{|S|}$ full components to choose in worst case.

4.3 Robins and Zelikovsky's proofs

Lemma 11. $g_{T_{i-1}}(K_i) = c(T_{i-1}) - \text{mst}(T_{i-1} \cup K_i)$.

Proof. Similarly with Lemma 9, we assume that the costs of all edges are distinct. By the definition of $g_{T_{i-1}}(K_i)$, we only need to show that $T_{i-1}[K_i] = \text{MST}(T_{i-1} \cup K_i)$. Suppose, for the sake of contradiction, that there exists $e \in T_{i-1}[K_i] \setminus \text{MST}(T_{i-1} \cup K_i)$. Since e is in $T_{i-1}[K_i]$ but not in $\text{MST}(T_{i-1} \cup K_i)$, e must be the longest edge of a cycle C of $T_{i-1} \cup K_i$. By the definition of $T_{i-1}[K_i]$, if e is an edge of K_i, e would have been eliminated when we construct $T_{i-1}[K_i]$. Therefore, e must be in K_i. Let A and B be the two connected components of $K_i \setminus e$. We will first show that A and B are full components. Then we will show that one of A and B has a better gain-to-loss ratio, and this contradicts the choice made by the algorithm.

It can be seen that if at least one A and B is not a full component, than K_i must contain a Steiner vertex of degree 2. Therefore, we only need to prove K_i does not contain a vertex of degree 2. Suppose K_i does contain a Steiner vertex v of degree 2. Let u and w be the neighbours of v. By the metric completion assumption, we many replace the path uvw by an edge uw. Thus we have a new full component K_i' whose gain equals that of K_i.

Now we consider the loss of the two full components, $L(K_i)$ and $L(K_i')$. If the path uvw is contained in $L(K_i)$, then uv is contained in $L(K_i')$. Then $l(K_i) = l(K_i')$, but K_i' has fewer edges. Thus K_i' should have been chosen instead of K_i in the i-th iteration. If the path uvw is not contained in $L(K_i)$, then exactly one of uv and vw is not in $L(K_i)$, w.l.o.g., assume that $uv \notin L(K_i)$. Since v is not a Steiner vertex of K_i', $L(K_i) = L(K_i') \cup \{vw\}$. Hence $l(K_i) > l(K_i')$. Thus K_i' should also have been chosen instead of K_i in the i-th iteration. Therefore, K_i does not contain a Steiner vertex of degree 2.

Since e is not in $\text{MST}(T_{i-1} \cup K_i)$, and we assume that the edge costs are distinct, it

must be the case that $c(T_{i-1}[A \cup B]) < c(T_{i-1}[K_i])$, and hence $g_{T_{i-1}}(K_i) < g_{T_{i-1}}(A \cup B)$. Since e is not in $\text{MST}(T_{i-1} \cup K_i)$, $\text{MST}(T_{i-1} \cup K_i) + e$ must has a unique cycle C, and e is the longest edge of C. Thus, e is the longest edge on a K_i path (which is part of C) between two terminals. A K_i path is a path whose edge set is a subset of the edges of K_i. By definition of $L(K_i)$, e is not in $L(K_i)$, and hence $L(K_i) = L(A) \cup L(B)$.

Since K_i is a tree, A and B are vertex disjoint, hence they do not share any Steiner vertices. by Lemma 1, we have $g_{T_{i-1}} \leq g_{T_{i-1}}(A) + g_{T_{i-1}}(B)$.

Hence, by the above discussion, we have

$$\begin{aligned}
\frac{g_{T_{i-1}}(K)}{l(K_i)} &\leq \frac{g_{T_{i-1}}(A \cup B)}{l(K_i)} \\
&\leq \frac{g_{T_{i-1}}(A) + g_{T_{i-1}}(B)}{l(A) + l(B)} \\
&\leq \max\left\{\frac{g_{T_{i-1}}(A)}{l(A)}, \frac{g_{T_{i-1}}(B)}{l(B)}\right\}
\end{aligned}$$

This inequality contradicts the choice of full components made by Algorithm 6. The algorithm always chooses a full component with maximum gain-to-loss ratio and with minimum number of edges. □

Now we consider the cost difference δ_i between T_i and T_{i-1}.

Lemma 12. $\delta_i = g_{T_{i-1}}(K_i) + l(K_i)$.

Proof. By the definition of gain and loss, we have

$$\begin{aligned}
g_{T_{i-1}}(K_i) &= c(T_{i-1}) - \text{mst}(T_{i-1} \cup K_i) \\
l(K_i) &= \text{mst}(T_{i-1} \cup K_i) - c(T_i)
\end{aligned}$$

Summing the above equations, we get $\delta_i = g_{T_{i-1}}(K_i) + l(K_i)$. □

Define $\Delta_i = g_{T_i}(\mathbf{OPT}_k) + l(\mathbf{OPT}_k)$, recall that $L(\mathbf{OPT}_k)$ connects Steiner vertices used in $\mathbf{OPT}_k$ to terminals. Intuitively Δ_i is the change in cost by integrating $\mathbf{OPT}_k$ into tree T_i. To simplify notations, we denote $l(\mathbf{OPT}_k)$ by l^*_k. Observe that

$$\Delta_{i-1} - \Delta_i = (c(T_{i-1}) - \mathbf{opt}_k + l^*_k) - (c(T_i) - \mathbf{opt}_k + l^*_k) = c(T_{i-1}) - c(T_i) = \delta_i; \quad (4.2)$$

where the last equation uses the definition of δ_i.

Lemma 13. *let $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ be positive real numbers. Then*

$$\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \leq \max_i \frac{a_i}{b_i}.$$

Proof. Let $j = \arg\max_i \frac{a_i}{b_i}$. Then,

$$\frac{a_i}{b_i} \leq \frac{a_j}{b_j},$$

for all $1 \leq i \leq n$. Thus

$$a_i b_j \leq a_j b_i,$$

hence,

$$\sum_{i=1}^{n} a_i b_j \leq \sum_{i=1}^{n} a_j b_i.$$

Therefore,

$$\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \leq \frac{a_j}{b_j}.$$

□

Lemma 14. *Let $l(j)$ be the sum of the losses of the first j full components added by Algorithm 6. Then*

$$\frac{l(j)}{l_k^*} \leq \ln \frac{\Delta_0}{\Delta_n}.$$

Proof. Let $\mathcal{K}(\mathbf{OPT}_k)$ be the set of maximal full components in $\mathbf{OPT}_k$, then $l_k^* = \sum_{K \in \mathcal{K}(\mathbf{OPT}_k)} l(K)$. Observe that we have

$$\Delta_{i-1} = g_{T_{i-1}}(\mathbf{OPT}_k) + l_k^* \leq \sum_{K \in \mathcal{K}(\mathbf{OPT}_k)} g_{T_{i-1}}(K) + l_k^*,$$

where the equality follows from the definition of Δ_i and the inequality uses Lemma 9.

Therefore,

$$\begin{aligned}\frac{\Delta_{i-1}}{l_k^*} &\leq \frac{\sum_{K\in\mathcal{K}(\mathbf{OPT}_k)} g_{T_{i-1}}(K) + l_k^*}{l_k^*} \\ &= 1 + \frac{\sum_{K\in\mathcal{K}(\mathbf{OPT}_k)} g_{T_{i-1}}(K)}{\sum_{K\in\mathcal{K}(\mathbf{OPT}_k)} l(K)} \\ &\leq 1 + \max_{K\in\mathcal{K}(\mathbf{OPT}_k)} \left\{ \frac{g_{T_{i-1}}(K)}{l(K)} \right\};\end{aligned}$$

where the last inequality follows from Lemma 13. The algorithm chooses the maximum gain-to-loss full component in each step, and hence

$$\frac{\Delta_{i-1}}{l_k^*} \leq 1 + \frac{g_{T_{i-1}}(K_i)}{l(K_i)} = \frac{\delta_i}{l(K_i)}.$$

Hence,

$$\delta_i \geq \frac{l(K_i)\Delta_{i-1}}{l_k^*}. \tag{4.3}$$

Then by (4.2) and (4.3), $\Delta_i = \Delta_{i-1} - \delta_i \leq \Delta_{i-1}(1 - \frac{l(K_i)}{l_k^*})$. By definition, $\Delta_i > 0$, then we have

$$\frac{\Delta_j}{\Delta_0} \leq \prod_{i=1}^{j} \left(1 - \frac{l(K_i)}{l_k^*}\right).$$

By taking logarithms on both sides, we have

$$\ln \frac{\Delta_0}{\Delta_j} \geq -\sum_{i=1}^{j} \ln \left(1 - \frac{l(K_i)}{l_k^*}\right)$$

Notice that $\ln(1+x) \leq x$ implies that $-\ln(1-x) \geq x$ for $0 \leq x < 1$, thus

$$\ln \frac{\Delta_0}{\Delta_j} \geq \sum_{i=1}^{j} \frac{l(K_i)}{l_k^*} = \frac{l(j)}{l_k^*}$$

□

Suppose that k-LCA stops after adding t k-restricted full components. Notice that the stopping condition of k-LCA implies that $g_{T_t}(K) = 0$, for all k-restricted full components K. Lemma 10 therefore implies that $c(T_t) \leq \mathbf{opt}_k$. The definition of Δ_j also implies that $\Delta_t = l_k^*$.

Let T be the output of k-LCA, by the above lemma, we have

$$\begin{aligned} c(T) &\leq c(T_t) + l(t) \\ &\leq \mathbf{opt}_k + l_k^* \ln \frac{c(T_0) - \mathbf{opt}_k + l_k^*}{l_k^*} \end{aligned}$$

Recall that T_0 is a minimum cost Steiner tree in $G[R]$, we obtain the following theorem.

Theorem 8. *Algorithm k-LCA outputs a Steiner tree with cost at most*

$$\mathbf{opt}_k + l_k^* \ln \left(1 + \frac{c(T_0) - \mathbf{opt}_k}{l_k^*}\right). \tag{4.4}$$

4.3.1 Approximation ratio for general graphs

Recall that we use ρ_k for the ratio of $\mathbf{opt}_k$ and $\mathbf{opt}$. Borchers and Du [7] proved that $\rho_k \leq 1 + (\lfloor \log_2 k \rfloor + 1)^{-1}$. We provide an upper bound for the approximation ratio of k-LCA.

Theorem 9. *The approximation ration of k-LCA algorithm is at most*

$$\rho_k \left(1 + \frac{1}{2} \ln \left(\frac{4}{\rho_k} - 1\right)\right).$$

Proof. In [40], Takahashi and Matsuyama proved that $c(T_0) \leq 2\mathbf{opt}$, thus by (4.4),

$$c(T) \leq \mathbf{opt}_k + l_k^* \ln \left(1 + \frac{2\mathbf{opt} - \mathbf{opt}_k}{l_k^*}\right) \leq \mathbf{opt}_k \left(1 + \frac{l_k^*}{\mathbf{opt}_k} \ln \left(1 + \frac{\mathbf{opt}_k}{l_k^*}\right)\right).$$

Let $x = \frac{l_k^*}{\mathbf{opt}_k}$, then,

$$c(T) \leq \mathbf{opt}_k \left(1 + x \ln \left(1 + \frac{1}{x}\right)\right).$$

By Karpinski and Zelikovisky [27], $l(T) \leq \frac{1}{2}c(T)$, then $l_k^* \leq \frac{1}{2}\mathbf{opt}_k$, or $x \leq \frac{1}{2}$. Since the function $f(x) = \mathbf{opt}_k \left(1 + x \ln\left(1 + \frac{1}{x}\right)\right)$ is increasing, it attains its maximum value at the largest possible value of x, i.e., at $x = \frac{1}{2}$. Therefore, the approximation ratio is

$$\frac{c(T)}{\mathbf{opt}} \leq \frac{\mathbf{opt}_k}{\mathbf{opt}} \left(1 + \frac{\ln 3}{2}\right) = \rho_k \left(1 + \frac{\ln 3}{2}\right)$$

□

When $k \to \infty$, the ratio converges to $1 + \frac{\ln 3}{2} \approx 1.55$.

4.3.2 The approximation ratio of k-LCA for quasi-bipartite graphs

For quasi-bipartite graphs, we can find a new bound for $c(T_0)$, and by applying the new bound to (4.4), we obtain a better approximation ratio for quasi-bipartite graphs.

Lemma 15. *For quasi-bipartite graphs,*

$$c(T_0) \leq 2(\mathbf{opt}_k - l_k^*) \tag{4.5}$$

Proof. Let K be a full component of Steiner tree T with p terminals and edges $e_1, \ldots, e_p$, i.e., K is a p-star. Let T^K be the minimum terminal spanning tree of K. Then $l(K)$ is the cost of the shortest edge, say e_1, connecting a terminal of K to the Steiner vertex of K. Then by our metric assumption,

$$c(T^K) \leq \sum_{i=2}^{p} (c(e_1) + c(e_i)) = pc(e_1) + c(K) - 2c(e_1) \leq 2(c(K) - l(K)).$$

Therefore,

$$c(T_0) \leq \sum_{K \in \mathbf{opt}_k} c(T^K) \leq 2(\mathbf{opt}_k - l_k^*).$$

□

Substituting (4.5) in to (4.4), we have

$$c(T) \leq l_k^* \ln\left(\frac{\mathbf{opt}_k}{l_k^*} - 1\right) + \mathbf{opt}_k.$$

Let $x = \frac{l_k^*}{\mathbf{opt}_k - l_k^*}$, then

$$c(T) \leq \mathbf{opt}_k \left(1 + \frac{x}{1+x} \ln \frac{1}{x}\right).$$

By taking the partial derivative of the right hand side of the above inequality with respect to x, we can find that its maximum is attained when x is the root of the equation $1+\ln x+x = 0$. Solve for x, we have $x \approx 0.279$. Then

$$c(T) \leq \mathbf{opt}_k \left(1 + \frac{x}{1+x} \ln \frac{1}{x}\right) = (1+x)\mathbf{opt}_k \approx 1.279 \cdot \mathbf{opt}_k.$$

Therefore, as $k \to +\infty$, the approximation ratio of k-LCA in quasi-bipartite graphs converges to 1.279.

The running time of k-LCA for quasi-bipartite graphs is discussed by Robins and Zelikovsky in [39]. Let r be the number of terminals in G, and s be the number of Steiner vertices in G. There are at most s iterations. Since for a quasi-bipartite graph, for a Steiner vertex v, the maximum gain with respenct to a terminal spanning tree T among all full components contains v, is the gain of the full component whose tips are neigbours of v in $\mathbf{MST}(T+v)$. Therefore, for each iteration, there are at most $O(s)$ gain calculation, and each gain calculation requires $O(r)$ time. Therefore, the running time of k-LCA for quasi-bipartite graphs is $O(s^2 r)$.

4.4 The transition of approximation ratio from quasi-bipartite graphs to general graphs

In the previous section, we discussed the performance of k-LCA for quasi-bipartite graphs and general graphs. There is an apparent gap in the performance guarantee obtained by k-LCA on quasi-bipartite graphs and general graphs. Can we find a graph parameter that

allows us to quantify k-LCA's performance as we transit from quasi-bipartite graphs to general graphs?

By our metric assumption, we may assume that any two Steiner vertices that are connected by a path whose vertices are all Steiner vertices are joint by an edge. Therefore, after removing all terminals of G, we get a collection of Steiner cliques. We call a graph G is *b-quasi-bipartite* if the maximum Steiner clique of G is isomorphic to K_b. By this definition, a quasi-bipartite graph is a 1-quasi-bipartite graph, and as $b \to +\infty$, a b-quasi-bipartite graph becomes a general graph. For the b-quasi-bipartite graphs, we have the following lemma.

Lemma 16. *For b-quasi-bipartite graphs,*

$$c(T_0) \leq 2\mathbf{opt}_k - \frac{2}{b} l_k^*.$$

Proof. Let K be a full component of Steiner tree T, then K contains at most b Steiner vertices. Let T^K be the minimum terminal spanning tree of K. Let u and v be the two terminals of K such that the distance between u and v is maximum, i.e., the length of the path uv on K is the diameter d of K. By performing depth first search on K starting from u and ending at v, we obtain a uv-walk w, and

$$c(w) = 2c(K) - d.$$

By our metric assumption, we have $c(T^K) \leq c(w)$, i.e.,

$$c(T^K) \leq 2c(K) - d. \tag{4.6}$$

Now we want to prove that

$$d \geq \frac{2}{b} l(K). \tag{4.7}$$

For each Steiner vertex $v \in V(K)$, the distance to the closes terminal is at most half the diameter of K. In other words, the lass of K is at most $b \cdot \frac{d}{2}$. Therefore,

$$c(T^K) \leq 2c(K) - \frac{2}{b} l(K).$$

Then,

$$c(T_0) \leq \sum_{K \in \mathcal{K}(\mathbf{OPT}_k)} c(T^K) \leq 2\mathbf{opt}_k - \frac{2}{b} l_k^*, \tag{4.8}$$

where $\mathcal{K}(\mathbf{OPT}_k)$ is the set of maximal full components in $\mathbf{OPT}_k$. □

Substituting (4.8) into (4.4), we have

$$c(T) \leq l_k^* \ln \left(\frac{\mathbf{opt}_k}{l_k^*} + 1 - \frac{2}{b} \right) + \mathbf{opt}_k. \tag{4.9}$$

When $b = 1$, we have the quasi-bipartite case. When $b = 2$, (4.9) becomes

$$c(T) \leq l_k^* \ln \left(\frac{\mathbf{opt}_k}{l_k^*} \right) + \mathbf{opt}_k. \tag{4.10}$$

By taking partial derivative of the right hand side of (4.10) with respect to l_k^*, we find the single maximum of the right hand side is attained at $l_k^* = \frac{1}{e}\mathbf{opt}_k$. Then (4.10) becomes

$$c(T) \leq \frac{1}{e}\mathbf{opt}_k \ln \left(\frac{\mathbf{opt}_k}{\frac{1}{e}\mathbf{opt}_k} \right) + \mathbf{opt}_k = \left(1 + \frac{1}{e} \right) \mathbf{opt}_k.$$

Therefore, in the case $b = 2$, the approximation ratio of k-LCA is $\left(1 + \frac{1}{e}\right) \approx 1.368$.

When $b \geq 3$, $f(x) = x \ln \left(\frac{\mathbf{opt}_k}{x} + 1 - \frac{2}{b} \right) + \mathbf{opt}_k$ is increasing. Similar with the proof in Theorem 9, $l_k^* \leq \frac{1}{2}\mathbf{opt}_k$, then the maximum of $f(l_k^*)$ is attained at $l_k^* = \frac{1}{2}\mathbf{opt}_k$, and we obtain

$$c(T) \leq \left(1 + \frac{1}{2} \ln \left(3 - \frac{2}{b} \right) \right) \mathbf{opt}_k.$$

Now we look into two special cases, $b = 3$ and $b = 4$. We will prove that

$$c(T^K) \leq 2c(K) - l(K) \tag{4.11}$$

for these cases. This implies that the approximation ratio of k-LCA for these two cases is the same as in the case where $b = 2$, i.e, it is 1.368.

Let K be a full component with three Steiner vertices. Notice that the Steiner vertices of K lie on a path. Let x and y be the ends of the Steiner path, let u and v be the furthest

terminal neighbours of x and y respectively. Please refer to *Fig. 4.3*.

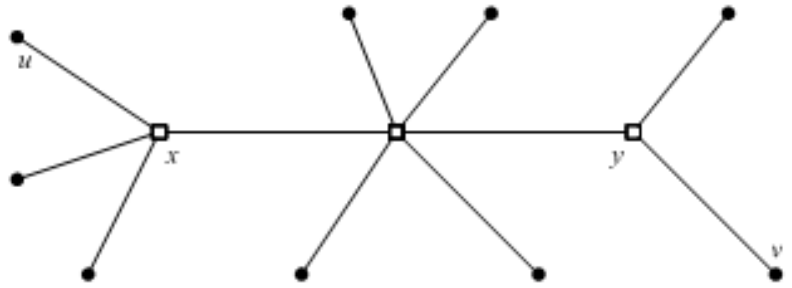


Figure 4.3: A full component with 3 Steiner vertices

Now we find a uv-walk by performing depth first search starting from u and ending at v. Let P be the uv-path on K, and T^K be the minimum terminal spanning tree of K. Then we have

$$c(T^K) \leq 2c(K) - c(P).$$

Observe that $P \setminus \{ux\}$ is a candidate for $L(K)$, hence,

$$l(K) \leq c(P \setminus \{ux\}) \leq c(P).$$

Therefore, we have a new bound for $c(T^K)$,

$$c(T^K) \leq 2c(K) - l(K).$$

Now we consider a full component with four Steiner vertices. There are two subcases. One is that we have a Steiner path that is formed by the four Steiner vertices. The proof for this subcase is similar with the proof of the case where $b = 3$. The other subcase is that in which we have a Steiner vertex that is joined to all other three Steiner vertices, i.e., there is a *Steiner star* in K. Please refer to *Fig. 4.4*.

For subcase 2, let x, y, z be the three tips of the Steiner star, and t, u, v be terminal neighbours of x, y, z respectively. Without loss of generality, we may assume that $c(xt) \leq$

$c(yu) \leq c(zv)$. Like the previous proofs, we find a uv-walk by performing depth first search starting from u and ending at v. Let P be the uv-path on K, and T^K be the minimum terminal spanning tree K. Similarly, we have

$$c(T^K) \leq 2c(K) - c(P).$$

Observe that $P \setminus \{yu\} \cup \{xt\}$ is a candidate for $L(K)$, hence

$$l(K) \leq c(P \setminus \{yu\} \cup (xt)) \leq c(P),$$

where the last inequality is true because of our assumption that $c(xt) \leq c(yu)$. Therefore,

$$c(T^K) \leq 2c(K) - l(K)$$

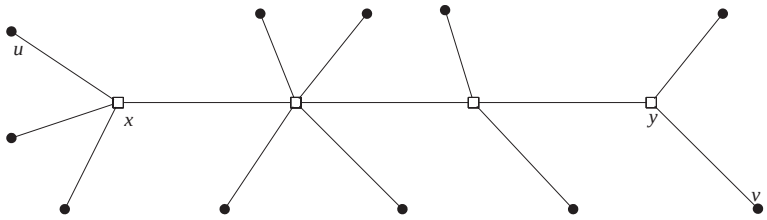
is also true for the case where $b = 4$.

Since (4.11) is true for $b = 3$ and $b = 4$, and it can also be obtained by substituting $b = 2$ into (4.8); the approximation ratio of k-LCA for the cases $b = 3$ and $b = 4$ is the same as the one for case $b = 2$. Thus we obtain the following theorem.

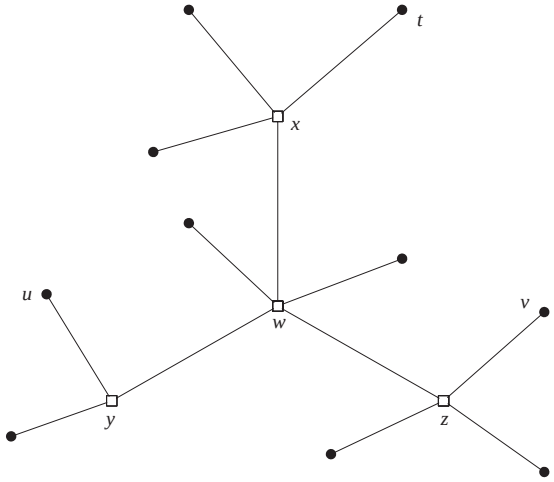
Theorem 10. *For a b-quasi-bipartite graph, when $b \in \{2, 3, 4\}$, the approximation ratio of k-LCA is 1.368; when $b \geq 5$, the approximation ratio of $k - LCA$ is $\left(1 + \frac{1}{2} \ln \left(3 - \frac{2}{b}\right)\right)$.*

Notice that (4.7) is tight only if K is a star and all the edges are of the same weight. As b getting larger, the inequality is getting more loose. This also affects that (4.8) is not tight either. Therefore, there must be a better bound for the approximation ratio of k-LCA.

We conjecture that for b fixed, the worst case of $c(T^K)$ occurs at the following example. In a graph G , consider a full component K, where the Steiner vertices form a full binary tree (a binary tree whose leaves are on the same level) T_S. Then we attach terminals to the root and the leaves of the binary tree. In the resulting tree, all Stiener vertices have degree 3. Please refer to *Fig 4.5*. Let the costs of all edges in K be 1. Suppose that, for any two vertices $u, v \in V(K)$, the cost c_{uv} of edge uv in G is the cost of the uv-path of in K.



Subcase 1, Steiner vertices form a path



Subcase 2, There is a Steiner vertex is joint to all other three Steiner vertices

Figure 4.4: Full components with 4 Steiner vertices

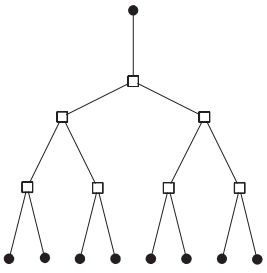


Figure 4.5: An example of the conjectured worse case

Clearly, $l(K) = 2^h - 1$ and $c(K) = 2^{h+1} - 1$, where $h = \log_2(b+1)$ is the height of T_S. We run Kruskal's minimum spanning tree algorithm on the graph induced by the terminals to get T^K. The first chosen edges are the edges joining the terminals that are the neighbours of the same leaves of T_S. The costs of each such edges is 2, and there are 2^{h-1} such edges. And then we will choose the edges of cost 4 joining the terminals that are the neighbours of the leaves with the same parent in T_S. There are 2^{h-2} such edges are chosen. We will keep doing this $2^{\lfloor \frac{h}{2} \rfloor}$ times. At last, we will choose 2 edges of cost $h+1$, that join the the terminal of the root and the terminals of the leaves. Therefore, the cost of T^K is

$$\begin{aligned} c(T^K) &= \sum_{i=1}^{\lfloor \frac{h}{2} \rfloor} 2i \cdot 2^{h-i} + 2(h+1) \\ &= 2^{h+1} \cdot \sum_{i=1}^{\lfloor \frac{h}{2} \rfloor} i2^{-i} + 2(h+1) \end{aligned}$$

Now we want to calculate $\sum_{i=1}^{n} i2^{-i}$.

$$\begin{aligned} \sum_{i=0}^{n} ix^i &= x \sum_{i=0}^{n} ix^{i-1} \\ &= x \left(\sum_{i=0}^{n} x^i \right)' \\ &= x \left(\frac{1-x^{n+1}}{1-x} \right)' \\ &= \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2} \end{aligned}$$

Substituting $x = \frac{1}{2}$ and $n = \lfloor \frac{h}{2} \rfloor$, we have

$$c(T^K) = 2^{h+1} \left(2 - \left(\left\lfloor \frac{h}{2} \right\rfloor + 2 \right) 2^{-\lfloor \frac{h}{2} \rfloor} \right) + 2(h+1)$$

Therefore,

$$c(T^K) = 2c(K) - \frac{2^{\lfloor \frac{h}{2} \rfloor + 1}\left(\lfloor \frac{h}{2} \rfloor + 2\right) - 2h - 4}{2^h - 1} \cdot l(K)$$

Therefore, by the conjecture,

$$c(T) \leq \mathbf{opt}_k + l_k^* \ln\left(\frac{\mathbf{opt}_k}{l_k^*} + 1 - \frac{2^{\lfloor \frac{h}{2} \rfloor + 1}\left(\lfloor \frac{h}{2} \rfloor + 2\right) - 2h - 4}{2^h - 1}\right).$$

For $h \geq 3$, the maximum of the right hand side of the previous inequality is attained at $l_k^* = \frac{1}{2}\mathbf{opt}_k$, and we obtain

$$c(T) \leq \left(1 + \frac{1}{2}\ln\left(3 - \frac{2^{\lfloor \frac{h}{2} \rfloor + 1}\left(\lfloor \frac{h}{2} \rfloor + 2\right) - 2h - 4}{2^h - 1}\right)\right) \cdot \mathbf{opt}_k.$$

In the previous discussion, we assumed $h = \log_2(b+1)$. If h is not integer, we can simply choose $h = \lceil \log_2(b+1) \rceil$, and the discussion is also valid. Then,

$$\begin{aligned} c(T) &\leq \left(1 + \frac{1}{2}\ln\left(3 - \frac{2^{\lfloor \frac{h}{2} \rfloor + 1}\left(\lfloor \frac{h}{2} \rfloor + 2\right) - 2h - 4}{2^h - 1}\right)\right) \cdot \mathbf{opt}_k \\ &\leq \left(1 + \frac{1}{2}\ln\left(3 - \frac{(b+1)^{\frac{1}{2}}\left(\log_2(b+1)^{\frac{1}{2}} + 1\right) - 2\log_2(b+1) - 4}{b}\right)\right) \cdot \mathbf{opt}_k \end{aligned}$$

4.5 A new proof of the approximation ratio

In this section, we will present a primal-dual proof for Robins and Zelikovsky's approximation ratio for k-LCA.

We denote the set of all Steiner partitions of G by $\tilde{\Pi}$, and the set of Steiner partitions of the subgraph H of G by $\tilde{\Pi}_H$. Let $\mathcal{K}$ be the set of k-restricted full components in G. Let $\pi = (V_1, V_2, \ldots, V_p) \in \tilde{\Pi}_H$, for a full component $K \in \mathcal{K}$, if we merge the parts of π that is joined by K, the rank of the new partition drops. The rank difference is exactly one less than the number of these parts. We call the difference *rank-contribution* of K. Formally,

we define the rank-contribution of K with respect to π as

$$\mathbf{rc}_K^\pi = |\{S \in \pi : S \cap V(K) \neq \varnothing\}| - 1.$$

Now we present an integer programming formulation (IP_H) with respect to H_i in algorithm k-LCA. To simplify notation, we omit the subscript i. In (IP_H), there is a binary variable x_K for every $K \in \mathcal{K}$. If K is included in our solution, we let $x_K = 1$; otherwise, we let $x_K = 0$. There is a binary variable x_e for each edge $e \in E(H)$, and one for each full component $K \in \mathcal{K}$. Given a k-restricted Steiner tree T in G, let $\mathcal{K}'(T)$ be the set of full components of T that contain edges that are not in $E(H)$. We obtain a solution to our IP by letting $x_K = 1$, for all $K \in \mathcal{K}'(T)$, and by letting $x_e = 1$, for all $e \in E(T) \setminus \bigcup_{K \in \mathcal{K}'(T)} E(K)$. Let $c(K)$ be the sum of the costs of all edges of K. (IP_H) is as follows:

$$\begin{aligned}
\min \quad & \sum_{e \in E(H)} c_e x_e + \sum_{K \in \mathcal{K}} c(K) x_K && && (\mathrm{IP}_H)\\
\text{s.t.} \quad & \sum_{e \in H_\pi} x_e + \sum_{K \in \mathcal{K}} \mathbf{rc}_K^\pi x_K \geq r(\pi) - 1 && \forall \pi \in \tilde{\Pi}_H\\
& x_e, x_K \in \{0,1\} && \forall e \in E(H), K \in \mathcal{K}
\end{aligned}$$

where $H_\pi = E_\pi \cap E(H)$.

Lemma 17. *Let T^* be a minimum cost k-restricted Steiner tree in G. Then there is a feasible solution to (IP_H) of cost at most $c(T^*)$.*

Proof. Since T^* is minimum, and all costs are non-negative, we may assume T^* does not have Steiner leaves. By the remark of the definition of full component, T^* is the union of a set of edge disjoint full components $\mathcal{K}^* = \{K_1, \ldots, K_s\}$. For any $1 \leq i \leq s$, we let $x^*_{K_i} = 1$, and $x^*_K = 0$, for any other $K \in \mathcal{K}$; also we let x_e be zero, for all $e \in E(H)$. Then we have

$$c(T^*) = \sum_{K \in \mathcal{K}^*} c(K) = \sum_{e \in E(H)} c_e x^*_e + \sum_{K \in \mathcal{K}} c(K) x^*_K.$$

Consider $\pi \in \tilde{\Pi}_H$, for each $K \in \mathcal{K}^*$, let $\pi_1^K, \ldots, \pi_s^K$ be the parts of π that have a non-empty

intersection with K. Include all vertices in $V(K)$ that are not in $G[H]$ into π_1^K. Call the result partition π'. Then we have a Steiner partition π' of G, such that $r(\pi') = r(\pi)$. Now obtain $T^*_{\pi'}$ by identifying the vertices in each of the parts of π'. Delete all duplicate edges and loops. Notice that for all parts of π' that are adjacent to K, their corresponding supernodes in $T^*_{\pi'}$ is connected by at least $\mathbf{rc}^\pi_K$ edges. Since T^* is connected, $T^*_{\pi'}$ is also connected. Therefore, $T^*_{\pi'}$ has at least $r(\pi') - 1 = r(\pi) - 1$ edges. This proves that

$$\sum_{K \in \mathcal{K}^*} \mathbf{rc}^\pi_K \geq r(\pi) - 1,$$

and our lemma follows. □

By replacing the integrality constraints of (IP_H) by nonnegativity requirements, we obtain the linear programming relaxation (LP_H) of (IP_H). In the dual (D_H) of (LP_H), we have a variable y_π for each $\pi \in \tilde{\Pi}_H$, and constraints for all edges $e \in E(G[H])$, and all k-restricted full components. (D_H) is the following.

$$\begin{aligned}
\max \quad & \sum_{\pi \in \tilde{\Pi}_H} (r(\pi) - 1) y_\pi && && (\mathrm{D}_H)\\
\text{s.t.} \quad & \sum_{\pi : e \in E_\pi} y_\pi \leq c_e && \forall e \in E(G[H]) && (4.12)\\
& \sum_{\pi \in \tilde{\Pi}_H} \mathbf{rc}^\pi_K y_\pi \leq c(K) && \forall K \in \mathcal{K} &&\\
& y_\pi \geq 0 && \forall \pi \in \tilde{(\Pi)}_H &&
\end{aligned}$$

Now we show that the dual $\bar{y}$ computed by the final call to Kruskal's algorithm in the last step of revised k-LCA for $G[H]$, can be converted into a feasible dual solution y for (D_H), and the objective value of y is at least a constant factor of the objective value of $\bar{y}$.

Notice that G_t is obtained by contracting the losses of all included full components from H_n, i.e., $G_t = H/L_t$. Let $\bar{\Pi} = \{\bar{\pi}_1, \bar{\pi}_2, \ldots, \bar{\pi}_{|R|}\}$ be the sequence of partitions induced

by running Kruskal's MST algorithm. Then by Theorem 5, we have

$$\sum_{i=1}^{|R|}(r(\bar{\pi}_i)-1)\bar{y}_{\bar{\pi}_i} = c(T_t).$$

For a terminal v, let E_t^v are the edge set of connected components of $G[L_t]$ that contains v. Without loss of generality, we may assume that each of such connected component contains exactly one terminal, overwise, we can discard one edge in the component and have a new candidate for loss with lower or same cost. That is, $E_t^v \cap E_t^u = \varnothing$, for all $u \neq v$. Therefore, we can decompose L_t as

$$L_n = \bigcup_{v\in R} E_t^v.$$

We denote the vertex set of $G[E_t^v]$ as $V[E_t^v]$. Now consider a partition $\bar{\pi}_i = (\bar{V}_1^i, \ldots, \bar{V}_p^i) \in \bar{\Pi}$. For $\forall 1 \leq j \leq p$, let

$$V_j^i = \bigcup_{v\in\bar{V}_j^i} V[E_t^v],$$

and define $\pi_i = (V_1^i, V_2^i, \ldots, V_p^i)$. Clearly, $\pi_i \in \tilde{\Pi}_H$, and

$$e \in H_{\pi_i} \Longleftrightarrow e \in E[G_t]_{\bar{\pi}_i}. \tag{4.13}$$

Let $y_{\pi_i} = \bar{y}_{\bar{\pi}_i}$ for all $1 \leq i \leq |R|$. We will show y_{π_i} is feasible for (D_H). Since $\bar{y}$ is feasible for (D) in G_t, by (4.13), y satisfies (4.12).

Now we prove that y also satisfies the second constraint of (D_H). Assume that for the sake of contradiction, the second constraint of (D_H) is violated for a k-restricted full component K, i.e.,

$$\sum_{i=1}^{|R|} \mathbf{rc}_K^{\pi_i} y_{\pi_i} > c(K).$$

Let q be the number of terminal leaves of K, then $\mathbf{rc}_K^{\pi_{|R|}} = 0$, and $\mathbf{rc}_K^{\pi_1} = q-1$. Let

$$0 \leq i_1 < i_2 < \cdots < i_q \leq |R|$$

such that $\mathbf{rc}_K^{\pi_{i_{j+1}}} = \mathbf{rc}_K^{\pi_{i_j}} - 1$ for all $1 \leq j \leq q$. Then we have a sequence of edges

$$e_{i_1}, e_{i_2}, \ldots, e_{i_q} \in E[T_t]$$

lie on pairwise disjoint cycles in $T_t \cup K$. Hence, if we remove all these e_{i_j} edges for all $1 \leq j \leq q$, we have a spanning tree T' of $G_t \cup K$.

Since $c(e_{i_j}) = \sum_{i=1}^{i_j} y_{\pi_i}$,

$$\sum_{j=1}^{q} c(e_{i_j}) = \sum_{j=1}^{q} \sum_{i=1}^{i_j} y_{\pi_i} = \sum_{i=1}^{|R|} \mathbf{rc}_K^{\pi_i} y_{\pi_i}.$$

By our assumption, $c(T') = c(T_t) + c(K) - \sum_{j=1}^{q} c(e_{i_j}) < c(T_t)$, then $g_{T_t}(K) > 0$. This contradicts the stopping condition of k-LCA. Hence y is feasible for (D_H).

Then we have

$$c(T) \leq \sum_{i=1}^{|R|} (r(\pi_i) - 1)\bar{y}_{\bar{\pi}_i} + l(t) \leq \mathbf{opt}_k + l(t).$$

This result is the same as Theorem 8.

Chapter 5

New facet defining inequality for the Steiner forest polyhedra

The best known approximation algorithms for the Steiner forest problem are based on (IP_{SF}), such as AKR [2] and Goemans' and Williamson's [17]. But the approximation ratio of the these two approximation algorithms is 2, while the best know approximation ratio for the Steiner tree problem is 1.55.

A tight worst case of AKR is the wheel example. Please refer to *Fig 2.5*. This example is an instance of the Steiner tree problem, but the Steiner tree problem is a special case of the Steiner forest problem. We can run AKR on this instance. The output of AKR is the same as the output of minimum spanning tree heuristic, i.e., a path contain only the terminals. Therefore the approximation ratio of AKR for this instance converge to 2 as $n \to +\infty$, and $\varepsilon \to 0$.

The example shows that the Steiner cut formulation is not good enough. Can we find stronger valid inequalities for the Steiner forest polyhedra and apply them in developing better approximation algorithms?

In this chapter, we give a new set of valid inequalities for the Steiner forest polyhedra, and prove that they are facet defining.

First we will give some definitions.

Definition 1. *For any vertex partition $\pi = \{V_1, ..., V_p\}$ of G, we construct an auxiliary graph H from G by*

- *letting* $V(H) = \{v_1, ..., v_p\}$*, where* v_i *corresponds to set* V_i*, for all* $1 \leq i \leq p$*;*
- *joining* $v_i, v_j \in V(H)$ *if there exists* $(s,t) \in \mathcal{R}$*, such that* $s \in V_i, t \in V_j$*.*

Define the rank *of* π *to be*

$$r(\pi) = p - c + 1,$$

where c *is the number of connected components of* H*.*

Definition 2. *Let* $\pi \in \Pi$ *be a partition of* G*. The* partition inequality *with respect to* π *is defined as*

$$\sum_{e \in E_\pi} x_e \geq r(\pi) - 1, \tag{5.1}$$

where E_π *is the set of edges whose endpoints lie in different sets of* π*.*

Lemma 18. *For any partition* π*, the partition inequality is valid for the convex hull of Steiner forests.*

Proof. We prove the theorem by contradiction. Suppose that there exits a Steiner Forest F, such that $|E_\pi \cap F| < p - c$ for some partition $\pi = \{V_1, ..., V_p\}$. We construct the auxiliary graph H_F for F by identifying the vertices in V_i for all i, and deleting the parallel edges and loops. Therefore $|E(H_F)| < p - c$. Then H_F has at least $c + 1$ connected components, and there must be at least one pair of terminals that is not connected by F. This contradicts the fact that F is a Steiner forest. □

Recall that, (2.9) is a valid inequality for the convex hull of Steiner forest. The following theorem shows that (2.9) is a special case of (5.1).

Lemma 19. *Steiner cut inequality (2.9) for Steiner forest problem is a special case of the partition inequality for Steiner forest problem.*

Proof. For any $S \subseteq V$ induces a Steiner cut, the partition $\pi^S = \{S, V \setminus S\}$ has rank 1. By (5.1), we have

$$\sum_{e \in \delta(S)} x_e \geq 1.$$

□

Definition 3. *A vertex partition* $\pi = \{V_1, ..., V_p\}$ *is* feasible *if* $G[\pi_i]$ *is connected for* $\forall 1 \leq i \leq p$.

Let G_π be the graph obtain by identifying all nodes in π_i into a node v_i, and deleting all loops.

Lemma 20. *Let* $\pi = \{V_1, ..., V_p\}$ *be a feasible vertex partition of* G. *If* G_π *is 2-connected, and* $r(\pi) = p$, *then the partition inequality with respect to* π *defines a facet of*

$$P_{SF} = \text{conv}\{x | x \text{ is the incidence vector of a Steiner forrest}\} + \mathbb{R}^m_+.$$

Furthermore, any partition π' *obtained by combining some partition sets of* π *is also facet inducing.*

Proof. Clearly, the partition inequality with respect to π is valid for P_{SF}.

Let $ax \geq b$ be a valid inequality of P_{SF}, such that

$$\left\{ x \in P_{SF} \Big| \sum_{e \in E_\pi} x_e = p - 1 \right\} \subseteq \left\{ x \in P_{SF} \Big| ax = b \right\}$$

First, we will prove that $a_e \geq 0, \forall e \in E$, and $b \geq 0$. Suppose that $a_e < 0$ for some $e \in E$. Let u_e be the incidence vector of e, i.e., u_e is a vector whose all entries are 0 except that the e-th entry is 1. Let $\hat{x}$ be a vector in P_{SF}. Then $a\hat{x} \geq b$. Now consider vector $\hat{x} + ku_e$, $k \geq 0$. By the definition of P_{SF}, $\hat{x} + ku_e \in P_{SF}$. Thus

$$a(\hat{x} + ku_e) \geq b$$

However, as $k \to +\infty$, $a(\hat{x} + ku_e) \to -\infty$. This contradiction proves that $a_e \geq 0, \forall e \in E$. Now suppose that $b < 0$, then $\{x \in P_{SF} \big| ax = b\} = \varnothing$. This contradicts that the partition inequality is valid.

Now we prove $a_e = 0, \forall e \notin E_\pi$. Let x be an equality solution for the partition inequality. Obtain x' from x by adding a positive number c to x_e for some edge $e \notin E_\pi$. Then $\sum_{e \in E_\pi} x'_e = p - 1$ as well. Thus $ax' - ax = 0$, or $a_e c = 0$, i.e., $a_e = 0, \forall e \notin E_\pi$.

Now consider any two edges $e_1, e_2 \in E_\pi$, we want to prove that $a_{e_1} = a_{e_2}$. Since G_π is 2-connected, there exist two spanning trees T^1_π and T^2_π of G_π, such that the symmetric

difference $E(T_\pi^1) \triangle E(T_\pi^2) = \{e_1, e_2\}$, where the *symmetric difference* is defined as the set of elements in exactly one of $E(T_\pi^1)$ and $E(T_\pi^2)$. Let T_i be the spanning tree of $G[\pi_i]$ (since π is feasible, these T_i's are well defined). Then

$$\begin{aligned} T_1 &= T_\pi^1 \cup \left(\bigcup_{i=1}^{p} T_i \right) \\ T_2 &= T_\pi^2 \cup \left(\bigcup_{i=1}^{p} T_i \right) \end{aligned}$$

are two spanning trees of G, and two Steiner forests. Let x^1 and x^2 be the incidence vectors of T_1 and T_2, respectively. Then, x^1 and x^2 satisfy the partition inequality with equality. Then $ax^1 = ax^2$, and $a_{e_1} = a_{e_2}$ follows. Let $a_e = 1, \forall e \in E_\pi$, we have $b = p - 1$. This shows that the partition inequality defines a facet of P_{SF}.

For any π' obtained from π by combining some of π's partition sets, there is a sequence of partitions

$$\pi = \pi^0, \pi^1, ..., \pi^q = \pi',$$

such that π^i is obtain by combining two adjacent vertices of $G_{\pi^{i-1}}$, for any $1 \leq i \leq m \leq q$; and π^i is obtained by combining two non-adjacent vertices of $G_{\pi^{i-1}}$, for any $m+1 \leq i \leq n$.

Clearly, for $i \leq m$, π^i is feasible and full rank, and G_{π^i} is 2-connected, then by the above result, π^i is facet inducing. For $i > m$, $E_{\pi^i} = E_{\pi^m}$, hence the partition inequality with respect to π^i is the same as the partition inequality with respect to π^m, and is facet inducing. □

Lemma 21. *Let $\pi = \{V_1, ..., V_p\}$ be a vertex partition of G. If G_π is not 2-connected, then the partition inequality with respect to π does not define a facet of P_{SF}.*

Proof. If G_π is not connected, it is easy to see that the partition inequality is not facet inducing, since it can be obtained by adding up the partition inequalities for each connected component.

Now, suppose G_π is connected. Let $v \in V(G_\pi)$ be a cut vertex of G_π. Let $G_\pi - v$ be the graph obtained from G_π by removing v and all its incident edges. Let C_1 be a connected component of $G_\pi - v$, and C_2 be the union of all the other connected components

of $G_\pi - v$. Without loss of generality, suppose $v_p = v$, $v_i \in V(C_1)$, $1 \leq i \leq r$; and $v_i \in V(C_2)$, $r+1 \leq i \leq p-1$. Let

$$\begin{aligned} \pi^1 &= \left\{ V_1, V_2, ..., V_r, \bigcup_{i=r+1}^{p-1} V_i, V_p \right\}, \\ \pi^2 &= \left\{ \bigcup_{i=1}^{r} V_i, V_{r+1}, V_{r+2}, ..., V_{p-1}, V_p \right\}. \end{aligned}$$

Then $r(\pi^1) + r(\pi^2) = r(\pi) + 1$, And the partition inequalities with respect to π^1 and π^2 are

$$\sum_{e \in E_{\pi^1}} x_e \geq r(\pi^1) - 1 \tag{5.2}$$

$$\sum_{e \in E_{\pi^2}} x_e \geq r(\pi^2) - 1 \tag{5.3}$$

Adding up (5.2) and (5.3), we have

$$\begin{aligned} \sum_{e \in E_\pi} x_e &\geq r(\pi^1) + r(\pi^2) - 2 \\ &= r(\pi) - 1. \end{aligned}$$

This is the partition inequality with respect to π, and the lemma follows. □

Lemma 22. *Let $\pi = \{V_1, ..., V_p\}$ be a vertex partition of G. If $r(\pi) < p$, then the partition inequality with respect to π does not define a facet of P_{SF}.*

Proof. Since $r(\pi) < p$, then the auxiliary graph H from G with respect to π is not connected. Let C_1 be one of the connected component of H, and C_2 be the union of other connected components of H. Without loss of generality, we may assume that $V(C_1) = \bigcup_{i=1}^{r} V_i$, and $V(C_2) = \bigcup_{i=r+1}^{p} V_i$. Let

$$\begin{aligned} \pi^1 &= \{V_1, V_2, ..., V_r, V(C_2)\} \\ \pi^2 &= \{V_{r+1}, V_{r+2}, ..., V_p, V(C_1)\} \end{aligned}$$

Then, $r(\pi^1) + r(\pi^1) = r(\pi) + 1$, hence,

$$\begin{aligned} \sum_{e \in E_\pi} x_e &= \sum_{e \in E_{\pi 1}} x_e + \sum_{e \in E_{\pi 2}} x_e \\ &\geq r(\pi^1) + r(\pi^2) - 2 \\ &= r(\pi) - 1. \end{aligned}$$

This is the partition inequality with respect to π, and the lemma follows. □

By Lemmas 20, 21 and 22, we have a following theorem.

Theorem 11. *Let $\pi = \{V_1, ..., V_p\}$ be a feasible vertex partition of G or obtained from a feasible partition by combining some of its partition sets. The partition inequality with respect to π defines a facet of P_{SF} if and only if G_π is 2-connected and $r(\pi) = p$.*

Chapter 6

Conclusions

The partition inequalities play an important role in the approximation algorithm design for the Steiner problems. In the thesis, we try to use linear programming relaxations of the integer programming formulations to interpret local search and greedy algorithms as primal-dual algorithms. We also give an in depth view of the Robins and Zelikovsky's greedy algorithm, and prove the performance of the algorithm for the graphs with different size Steiner blocks. For the Steiner forest problem, a set of new facet defining partition inequality is defined.

There are still many open problems for future research.

1. Can we find a better selection function in the greedy algorithm framwork?

2. We believe that the analysis of k-LCA is slack. Therefore, we may find a better approximation ratio for the algorithm. In particular, if our conjecture in section 4.4 is true, the approximation ratio can be improved for graphs with different size Steiner blocks.

3. The new Steiner forest parition inequalities have not been applied to a new approximation alogorithm. The inequalities are stronger than the cut inequalities. Can we use them algorithmically?

Bibliography

[1] A. Agarwal and M. Charikar. On the advantage of network coding for improving network throughput. In *Proceedings of the IEEE Information Theory Workshop*, 2004.

[2] A. Agrawal, P. Klein, and R.Ravi. When tree collide: An approximation algorithm for the generalized Steiner problem in networks. *SIAM J. Comput.*, 24:440–456, 1995.

[3] Y.P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980.

[4] P. Berman and V. Ramaiyer. Improved approximations for the steiner tree problem. *J. Algorithms*, 17:381–408, 1994.

[5] M. Bern and P. Plassmann. The steiner problem with edge lengths 1 and 2. *Inf. Process. Lett.*, 32(4):171–176, 1989.

[6] M. D. Biha, H. Kerivin, and A. R. Mahjoub. Steiner trees and polyhedra. *Discrete Applied Mathematics*, 112(1-3):101–120, 2001.

[7] A. Borchers and D.-Z. Du. The k-Steiner ratio in graphs. *SIAM J. Computing*, 26(3):857–869, 1997.

[8] S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8:25–29, 1989.

[9] S. Chopra and M. R. Rao. The Steiner tree problem 1: Formulations, compositions, and extension of facets. *Mathemaitcal Programming*, 64:209–229, 1994.

[10] S. Chopra and M. R. Rao. The Steiner tree problem 2: Properties and classes of facets. *Mathemaitcal Programming*, 64:231–246, 1994.

[11] R. Cole, R. Hariharan, M. Lewenstein, and E. Porat. A faster implementation of the Goemans-Williamson clustering algorithm. In *Proceedings, ACM-SIAM symposium on Discrete algorithms*, pages 17–25, 2001.

[12] S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1:195–207, 1971.

[13] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Standards*, B71:233 – 240, 1967.

[14] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. In *Proceedings of 25th Annual Symposium on Foundations of Computer Science*, pages 338–346, 1984.

[15] E. N. Gilbert and H. O. Pollak. Steiner minimum trees. *SIAM J. Appl. Math.*, 16:1–29, 1968.

[16] M. X. Goemans and Y. Myung. A catalog of Steiner tree formulations. *Networks*, 23:19–28, 1993.

[17] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1995.

[18] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel. Approximation algorithms for the Steiner tree problem in graphs. In X. Cheng and D.Z. Du, editors, *Steiner trees in industry*, pages 235–279. Kluwer, 2001.

[19] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel. Lower bounds for approximation algorithms for the Steiner tree problem. In *WG: Graph-Theoretic Concepts in Computer Science, International workshop WG*, 2001.

[20] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel. Steiner tree in uniformly qusi-bipartite graphs. *Inform. Process. Lett.*, 83(4):195–200, 2002.

[21] M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing comunication networks with low-connectivity constraints. *Operations Research*, 40:309–330, 1992.

[22] S. L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–133, 1971.

[23] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14:255–265, 1966.

[24] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*, volume 53 of *Annals of Discrete Mathematics*. North-Holland, Amsterdam, Netherlands, 1992.

[25] A. B. Kahng and G. Robins. A new class of iterative Steiner tree heuristics with good performance. *IEEE Transactions on Computer-Aided Design*, 11:893–902, 1992.

[26] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, Plenum, New York, 1972. R.E. Miller and J.W. Thatcher.

[27] M. Karpinski and A. Zelikovisky. New approximation algorithm for the Steiner tree problem. *Journal of Combinatorial Optimization*, 1:47–65, 1997.

[28] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.

[29] J. Könemann, S. Leonardi, and G. Schäfer. A group-strategyproof mechanism for Steiner forests. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[30] J. Könemann, S. Leonardi, G. Schäfer, and S. van Zwam. From primal-dual to cost shares and back: A stronger LP relaxation for the Steiner forest problem. In *Proceedings, International Colloquium on Automata, Languages and Processing*, 2005.

[31] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings, American Mathematical Society*, volume 7, pages 48–50, 1956.

[32] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York, 1976.

[33] T. Polzin. *Algorithms for the Steiner problem in networks*. PhD thesis, Universität des Saarlandes, 2003.

[34] T. Polzin and S. V. Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112(1-3):241–262, 2001.

[35] H. J. Prömel and A. Steger. A new approximation algorithm for the Steiner tree problem with performance ratio $\frac{5}{3}$. *J. Algorithms*, 36:89–101, 2000.

[36] J. S. Provan. Convexity and the Steiner tree problem. *Netw.*, 18(1):55–72, 1988.

[37] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 742–751, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[38] R. Rizzi. On Rajagopalan and Vazirani's 3/2-aproximation bound for the Iterated 1-Steiner heuristic. *Information Processing Letters*, 86(6):335–338, 2003.

[39] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proccedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.

[40] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Jap.*, 24:573–577, 1980.

[41] V. V. Vazirani. *The Steiner tree problem and its generalizations*, volume 1444 of *Lecture Notes in Computer Science*. 1998.

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

[43] A. Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. Technical report, University of Virginia, Charlottesville, VA, USA, 1996.

[44] A. Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18(1):99 – 110, 1997.

[45] A. Z. Zelikovsky. The 11/6-approximation algorithm for the Steiner problem on networks. *Algorithmica*, 9:463–470, 1993.