Linear Programming Tools and Approximation Algorithms for Combinatorial Optimization

by

David Alexander Griffith Pritchard

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

Waterloo, Ontario, Canada, 2009

© David Alexander Griffith Pritchard 2009

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

We study techniques, approximation algorithms, structural properties and lower bounds related to applications of linear programs in combinatorial optimization. The following *Steiner tree problem* is central: given a graph with a distinguished subset of required vertices, and costs for each edge, find a minimum-cost subgraph that connects the required vertices. We also investigate the areas of network design, multicommodity flows, and packing/covering integer programs. All of these problems are NP-complete so it is natural to seek approximation algorithms with the best provable approximation ratio. Overall, we show some new techniques that enhance the already-substantial corpus of LP-based approximation methods, and we also look for limitations of these techniques.

The first half of the thesis deals with linear programming relaxations for the Steiner tree problem. The crux of our work deals with hypergraphic relaxations obtained via the well-known full component decomposition of Steiner trees; explicitly, in this view the fundamental building blocks are not edges, but hyperedges containing two or more required vertices. We introduce a new *hypergraphic LP based on partitions*. We show the new LP has the same value as several previously-studied hypergraphic ones; when no Steiner nodes are adjacent, we show that the value of the well-known bidirected cut relaxation is also the same. A new *partition uncrossing* technique is used to demonstrate these equivalences, and to show that extreme points of the new LP are well-structured. We improve the best known integrality gap on these LPs in some special cases. We show that several approximation algorithms from the literature on Steiner trees can be re-interpreted through linear programs, in particular our hypergraphic relaxation yields a new view of the Robins-Zelikovsky [178] 1.55-approximation algorithm for the Steiner tree problem.

The second half of the thesis deals with a variety of fundamental problems in combinatorial optimization. We show how to apply the iterated LP relaxation framework to the problem of multicommodity integral flow in a tree, to get an approximation ratio that is asymptotically optimal in terms of the minimum capacity. Iterated relaxation gives an infeasible solution, so we need to finesse it back to feasibility without losing too much value. Iterated LP relaxation similarly gives an $O(k^2)$-approximation algorithm for packing integer programs with at most k occurrences of each variable; new LP rounding techniques give a k-approximation algorithm for covering integer programs with at most k variable per constraint. We study extreme points of the standard LP relaxation for the traveling salesperson problem and show that they can be much more complex than was previously known. The k-edge-connected spanning multi-subgraph problem has the same LP and we prove a lower bound and conjecture an upper bound on the approximability of variants of this problem. Finally, we show that for packing/covering integer programs with a bounded number of constraints, for any $\epsilon > 0$, there is an LP with integrality gap at most $1 + \epsilon$.

Acknowledgements

I warmly thank my family in Scarborough — Karen, Bradley and Hilary — for their support throughout my studies, especially their cheerful hospitality during random unannounced visits.

I thank my friends, my colleagues, UW staff, and my co-authors for making my PhD studies a fun and enriching experience. This especially includes my advisor Jochen and people whose paths crossed with mine in the Graduate Student Association, Graduate Studies Endowment Fund, and Grad House. I thank my thesis examiners for their helpful comments on content and presentation.

I thank the people who made the following very useful free software available: GeoGebra [117], the Maple `convex` package [75], and LaTeX.

Finally, I thank my teachers throughout my education, especially those who helped illuminate the elegance and usefulness of mathematics and computer science!

- The part on Steiner trees, Chapter 2 to Chapter 5, includes joint work with Jochen Könemann & Kunlun Tan (including the series [187, 136, 139]) and Jochen Könemann & Deeparnab Chakrabraty (including the series [138, 36]). In addition Section 4.8.3 is joint work with Jochen Könemann and Yehua Wei.
- Chapter 6 is an extended version of the conference paper [137]; it is joint work with Jochen Könemann and Ojas Parekh.
- Chapter 7 is an extended version of the conference paper [172] and includes subsequent joint work with Deeparnab Chakrabarty (Section 7.3).

Contents

Chapter 1

Introduction

What is the power of computers? How can mathematics help us understand the power of computers, and how can computers help us understand mathematics? These are the sort of theoretical questions that motivate this thesis, at a high level; we return to the interplay between computers and mathematics later in the introduction.

The concrete problem which motivates most of this thesis is the *Steiner tree problem*, which is as follows. You are given some required vertices (points) and some optional vertices, and want to build a graph (network) to connect the required vertices. You can purchase an edge (direct link) between any two points u and v; the cost of this edge, which you are given as part of the problem statement, is some dollar value c_{uv} that depends on u and v. The Steiner tree problem is then, *what is the cheapest way to purchase edges so that between any two required vertices, there is a path of edges?* Optional points can be included or excluded from the graph as you prefer.

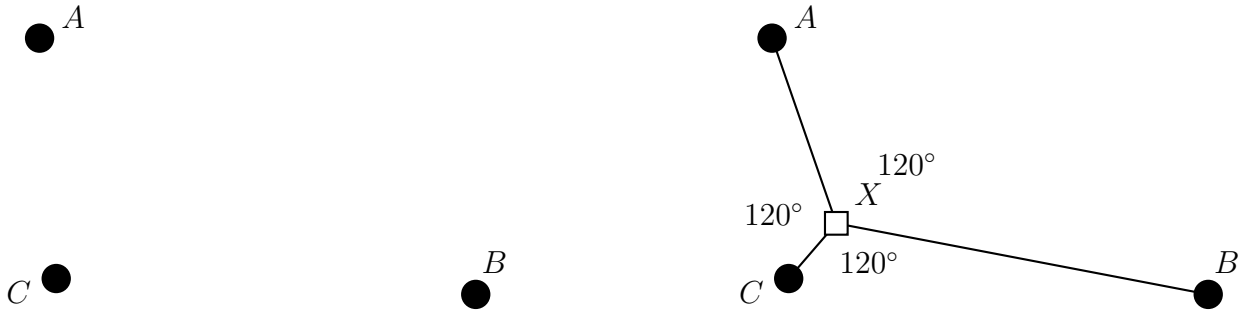


Figure 1.1: Left: an instance of the Steiner tree problem where there are three required vertices in the plane. Right: the solution for this instance uses one optional point.

In Figure 1.1 we give an example of how an optional point helps. There are three required points A, B, C in the two-dimensional plane, every other point in the plane is an

optional point, and the cost of connecting two points is the same as their distance; so the Steiner tree problem asks the shortest total length of line segments to connect A, B, C. This very special case was investigated by the classical mathematicians Fermat and Torricelli in the mid-1600s; they found that the optimal network consists of three edges AX, BX, CX where the point X satisfies $\angle AXB = \angle BXC = \angle CXA = 120^\circ$ (unless this X lies outside the triangle ABC, in which case the optimal solution is just the two shorter sides of the triangle.) We refer the reader to [174, §10.4] for these historical references.

We now skip forward a few centuries to the 1900s. Electronic computers were developed over the course of this century and so was a formal mathematical model of computation. Computers can be programmed to perform a variety of different tasks and can do basic mathematical operations much faster than a human. The Steiner tree problem has applications in industry [41] (for example, the network could be for telecommunication, transportation, or chip layout) and so one wonders: just how quickly can a computer can solve the Steiner tree problem? The mainstream notion in theoretical computer science is the following abstract mathematical expression of speed (i.e. one that is independent of whether you use a Mac or a PC): we seek an algorithm (abstract computer program) with smallest *time complexity* (number of basic operations performed) as a function of the input size n.

Another mainstream notion is that a *fast* running time is any running time of the form at most n^D for constant D, so-called *polynomial time complexity*. In the 1970s, the complexity-theoretic notion of NP-completeness was developed by Cook and Levin; then Karp [126] showed that Steiner tree is "NP-hard," so a fast algorithm for the Steiner tree problem is would imply fast algorithms for all "NP-complete" problems. Moreover, there are lots of well-known NP-complete problems and the best known algorithms for them have running time like C^n for some constant $C > 1$. Since $n^D < C^n$ for large enough n, we cannot use any known algorithms solve the Steiner tree problem this quickly — there is a $1,000,000 conjecture [122] that in fact no such algorithm exists.

It is therefore sensible to look at *approximation algorithms*: find a fast algorithm which outputs some valid answer that is *nearly* optimal, i.e. the output has cost within a factor α of the best possible, for some α. Such an algorithm is called an *α-approximation algorithm*. For the Steiner tree problem, a 2-approximation algorithm was known at least as early as 1968 [90] and in the 1990s other algorithms [202, 16, 203, 118, 127, 173] were discovered with better approximation ratios, with the current best ratio equal to $1 + \ln 3 \approx 1.55$ [178], by Robins & Zelikovksy.

The PCP theorem in the early 1990s showed that there are limits to approximability: e.g. this line established a $1 + \epsilon$ inapproximability result for the Steiner tree problem for a fixed tiny $\epsilon > 0$. More recently, Chlebík and Chlebíková [45] showed the best possible approximation ratio for Steiner tree is at least $\frac{96}{95} \approx 1.01$ (unless NP has fast algorithms). Where, between 1.01 and 1.55, is the best polynomial-time approximation ratio?

1.1 Linear Programs

Algorithmic techniques based on *linear programs* (LPs) have driven many modern developments in combinatorial optimization problems related to the Steiner tree problem. The linear programming approach is to relax a discrete problem (in Steiner tree there are two discrete choices per connection, purchase it or don't purchase it) into a continuous variant (we now allow each edge e to be "purchased" to any fractional extent $0 \leq x_e \leq 1$). Then one must overcome two inter-related technical challenges: first, modeling the problem by linear constraints; second, recovering an integral solution from the fractional relaxation without increasing the cost too much.

It has been 10 years since the last improvement to the best-known ratio for the Steiner tree problem. Moreover, the Steiner tree problem is an example where LP methods have *not* driven innovation: none of the long list of approximation algorithms [90, 202, 16, 203, 118, 127, 173, 178] were developed by LP methods. The best LP-based result known is that an alternative 2-approximation algorithm can be obtained using LP technology, e.g. [93, 2, 124].[1] Nonetheless, the overall breadth and depth of LP methods has developed over time, and there is no negative result suggesting that LP methods will remain forever ineffective in this setting. In addition, LP methods are useful in practice for solving large-scale instances of the Steiner tree problem, by using integer programming software [193, 3]. Therefore, a large part of this thesis is devoted to developing and understanding modern LP technology for use in the Steiner tree problem. We also successfully develop LP-based approximation algorithms, with better approximation ratios than were previously known, for several other problems in combinatorial optimization.

1.2 LP Preliminaries

While a substantial number of different successful techniques are known in the literature on LP methods, there is no hard-and-fast rule telling whether a given LP is useful or not. Some guidelines are known, including *small integrality gap* and *uncrossability*.

The *integrality gap* Γ of a linear program is a quantitative measure related to the suitability of an LP for use in designing approximation algorithms. We discuss it for minimization problems here, but analogous definitions hold for maximization problems. The integrality gap is defined as the worst-case cost ratio between the integral optimum $\texttt{cost}(x_I^*)$ (i.e., min Steiner tree cost) and the fractional optimum $\mathcal{L}^*$ (i.e., LP optimal value). The proof method of most LP-based approximation algorithms (e.g. *rounding* and

[1] In the special case of *quasibiparite instances* — where no two Steiner nodes are adjacent — a $\frac{4}{3}$-approximation [176, 177, 34] was developed using the *bidirected cut* LP, but the non-LP based algorithm of Robins & Zelikovsky [178] still performs even better on these instances, with approximation ratio 1.29.

primal-dual) is to find a feasible x and prove that $\texttt{cost}(x) \leq \alpha\mathcal{L}^*$, which guarantees an α-approximation since

$$\texttt{cost}(x) \leq \alpha\mathcal{L}^* \leq \alpha\texttt{cost}(x_I^*).$$

But if the integrality gap satisfies $\Gamma > \alpha$, we also have

$$\texttt{cost}(x) \geq \texttt{cost}(x_I^*) \geq \Gamma\mathcal{L}^* > \alpha\mathcal{L}^*,$$

and so $\texttt{cost}(x) \leq \alpha\mathcal{L}^*$ is impossible. Conversely, approximation algorithms of this type, which we will call *LP-relative* approximation algorithms[2], prove that $\Gamma \leq \alpha$. One reason to demarcate the property of being LP-relative is that it is specifically necessary in some settings; in Chapter 6 we make use of a 2-approximation algorithm of Jain [124], crucially, because it is LP-relative. The notion of an LP-relative approximation algorithm is already ubiquitous in the literature; but we are not aware of a name for it as such.

Even if an LP has a large integrality gap, one may still find it useful in designing an approximation algorithm. For example, Carr et al. [33] gave a $\frac{21}{10}$-approximation algorithm for *edge-dominating set* using the natural LP. They also showed an integrality gap of $\frac{21}{10}$ for that LP, which precluded any better LP-relative ratio for that LP. But by strengthening the natural LP with additional constraints, two groups [79, 166] obtained another LP with integrality gap of 2 and also obtained a 2-approximation algorithm for the problem. Another example is *multicommodity flow in a line*. The natural LP has integrality gap at least $n/2$, however by reducing to structured subproblems and applying the LP to these subproblems, Bansal et al. [9] obtained a much better $O(\log n)$-approximation algorithm.

It is convenient to introduce some common terms before discussing uncrossing. For a vector x, the *support* of x, denoted $\texttt{supp}(x)$, is the set $\{i \mid x_i \neq 0\}$. The *support size* of x is $|\texttt{supp}(x)|$. For a family $\mathcal{F}$ of subsets of a ground set X, we call $\mathcal{F}$ *laminar* if every $A, B \in \mathcal{F}$ satisfies either $A \subseteq B, B \subseteq A$, or $A \cap B = \varnothing$. One can show that a laminar family has at most $2|X|$ sets.

Uncrossability refers to several structural properties possessed by various LPs. We give a historical survey in Section 1.5 and examples later in the thesis. One typical consequence of uncrossing is to show that a large linear program has a small, well-structured optimal solution (e.g., one with laminar support). There is some sense in which an LP needs to be very "special" to be uncrossed, e.g. it typically requires that we appeal to super- or sub-modularity of certain values.

We remark that the two LP properties mentioned above — being uncrossable, and having a small integrality gap — are somewhat at odds with each other. On the one hand, strengthening an LP by adding extra valid constraints can only (weakly) decrease its integrality gap. On the other hand, uncrossability requires that all constraints relate

[2]For a specific LP $\mathcal{L}$, the term $\mathcal{L}$-*relative* means the output cost is at most $\alpha \cdot \text{OPT}(\mathcal{L})$.

to each other in a special way, which is destroyed once we add extra constraints[3]. We note that in the literature on Steiner trees, there are papers investigating successively stronger LPs [5, 47, 48, 54, 58, 95, 168, 170, 194, 200], and yet the weakest known LP (the undirected cut relaxation) has yielded remarkable results on bounded-degree versions of the spanning tree and Steiner tree problems [99, 148, 182, 10], because it remained uncrossable when adding simple degree constraints.

1.3 Contributions: Steiner Trees

The first half of the thesis, Chapters 2–5, is concerned with linear program relaxations of the Steiner tree problem. Our work utilizes the well-known *full component decomposition* of Steiner trees, which transforms the Steiner tree problem into a natural problem on *hypergraphs*, which are a generalization of graphs. The post-1990 Steiner tree algorithms [202, 16, 203, 118, 127, 173, 178] can all be seen as using this view in one way or another. The first instance of a Steiner tree LP based on hypergraphs is due to Warme in the late 1990s [194]; it uses *subtour constraints*.

We detail our results obtained for Steiner trees in Chapter 2, after introducing the LPs themselves, but give a sketch here. We introduce a new *hypergraphic LP based on partitions*. We show the new LP has the same value as several previously-studied hypergraphic ones (e.g. Warme's subtour LP); when no Steiner nodes are adjacent, we show, surprisingly, that the value of the well-known *bidirected cut relaxation* is also the same. A new *partition uncrossing* technique is used to demonstrate these equivalences, and to show that extreme points of the new LP are well-structured. We improve the best known integrality gap on these LPs in some special cases. We show that several approximation algorithms from the literature on Steiner trees can be re-interpreted through linear programs, in particular our hypergraphic relaxation yields a new view of the Robins-Zelikovsky [178] 1.55-approximation algorithm for the Steiner tree problem.

1.3.1 Discussion on Computational Techniques

We hope to use mathematical techniques to advance the state of the art in computer science; and it has turned out that computers were essential to obtain our mathematical results. The new partition uncrossing techniques depend on a surprisingly complicated intermediate

[3]There is some wiggle room here, e.g. if a pointed LP has support size at most s for each extreme point, adding any t constraints results in an LP all of whose extreme points have support size at most $s(t+1)$ — this follows from Carathéodory's theorem applied to the fact that every extreme point of the strengthened LP lies in a t-dimensional face of the original.

result (Lemma 3.11) which we only posited to be true after computing "tightness" properties of the new partition LP. Similar computations are used to refute otherwise-sensible seeming conjectures later in the paper — see Example 3.20 and Section 5.6.2. The proof a polyhedral result in Section 4.6 — in terms that will be introduced later, equivalence of hypergraphic and bidirected LPs on 5-restricted instances — depends on computational enumeration of extreme points of certain polyhedra. Computational experiments with the various LPs in this thesis were very valuable overall.

1.4 Contributions: Other Problems

Chapters 6–9 can be read independently of each other and of the first half of the thesis.

In Chapter 6 we consider the *integral multicommodity flow in a tree* problem. This problem is known to admit a 4-approximation algorithm. We show that in a variety of settings — edge-capacitated, arc-capacitated, and vertex-capacitated — the problem gets easier to approximate as the minimum capacity increases. Precisely, where μ denotes the minimum capacity, we get $1+O(1/\mu)$-approximation algorithms for all three problems, and the same result for their covering analogues. These results are obtained by proving that in all three settings, the recent technique of *iterated LP relaxation* [181] applies, and giving a framework to extend known consequences of iterated LP relaxation. We also determine the integer hull of all feasible solutions when the tree is a *spider*, i.e. has only one vertex has degree greater than 2.

In Chapter 7 we consider two closely-related problems. First, we give a k-approximation algorithm for *k-row-sparse covering integer programs*, correcting an erroneous claim of [32, 80] (they have a correct algorithm for unit upper bounds but erroneously claim unit upper bounds hold without loss of generality). To obtain this result we introduce a novel LP strengthening tool that extends well-known rounding methodology. Second, we give an $O(k^2)$-approximation algorithm for *k-column-sparse packing integer programs*, a new but natural problem. Our approach is based on iterated LP relaxation.

In Chapter 8 we give two results. The main result is a new complex extreme point for a well-known LP, the *subtour* formulation for the symmetric traveling salesperson problem. Specifically, we find an extreme point on n vertices with maximum degree $n/2$ and denominator exponential in n; the previous best was degree $\Theta(\sqrt{n})$ and denominator $\Theta(n)$. Our motivation for this problem is the k-edge-connected spanning multi-subgraph (k-ECSM) problem, whose natural LP is essentially the same; we give a new hardness result, showing that this problem has an inapproximability ratio bounded away from 1 even as k tends to infinity. We conjecture that k-ECSM admits a $1 + O(1/k)$-approximation algorithm.

In Chapter 9 we consider covering or packing integer programs with at most k constraints (plus box constraints), which are equivalent to the k-dimensional knapsack prob-

lem. A polynomial-time approximation scheme is known for this problem; we strengthen that result by, for any $\epsilon > 0$, giving a new LP admitting an LP-relative $(1+\epsilon)$-approximation algorithm. This also serves to put a recent result of Bienstock [18] in a much more general context.

1.5 A Partial History of Polyhedral Uncrossing

In this final section of the introduction we give a brief history of the technique known as (set) uncrossing for polyhedra, which is intended as a historical/literature review for readers who already have some familiarity with the field; we also do not attempt to give a formal definition of *uncrossed*. We distinguish between two main types of results. In *primal uncrossing* (e.g. for partition uncrossing, Theorem 3.1) one shows that every extreme point solution has an uncrossed basis; more specifically in *weak primal*, one shows that *some* uncrossed basis exists, and in *strong primal*, one shows that every maximal linearly uncrossed family of tight constraints is a basis (plus possibly some linearly dependent constraints). In *dual uncrossing* (e.g. for partition uncrossing, Theorem 3.3), one shows that the dual LP has an optimum with uncrossed support. We distinguish between strong and weak primal uncrossing because dual uncrossing implies weak primal uncrossing ([179, Thm. 5.35], [104, §8.4]) and strong primal uncrossing trivially implies weak primal uncrossing, but there seems to be no direct implication between dual uncrossing or strong primal uncrossing.

Dual uncrossing is very common in recent papers. In dual uncrossing for sets, one repeats the following operation: lower the dual variables for two sets and raise the dual variables for their union and intersection. This approach appears in the first published work on polyhedral uncrossing: the work of Edmonds and Giles [60, 91] on submodular flows, which was inspired jointly by Lovász' simplified proof [156] of the Lucchesi-Younger theorem [157, 158], and Edmonds' proof [59] that the polymatroid intersection LP is totally dual integral[4]. A year later, Hoffman & Schwartz used a very similar argument when they introduced the *lattice polytope* [116] model and proved it is totally dual integral.

Additionally, there are a number of earlier papers which use techniques and propositions related to pairwise uncrossing, e.g. Nash-Williams [162] (who showed that the function $|\delta(X)|$ counting edges leaving $X \subset V$ is submodular, and who also showed weak supermodularity of a function related to network design), Ford & Fulkerson [68, Ch. 1 Cor. 5.4] (min-cuts of a graph are closed under union and intersection), Younger [201] (reduction of a problem to the laminar case), and Lovász [153, 152, 154, 155] (coverage functions,

[4]Edmonds' proof in [59] uses a single step of *global uncrossing* applicable by virtue of the greedy algorithm for (poly)matroids. Note modern treatments (e.g., [179, §41.4]) often present a proof based on pairwise uncrossing instead.

Menger's theorem, orientations, and 2b-matchings in a hypergraph derived from T-joins). Frank [69] notes that Lovász used this technique on an undergraduate mathematics contest [151]. Aside from the developments in the mid-1970s [60, 157, 156] by Edmonds-Giles, Lucchesi-Younger and Lovász, most of these sources do not cite one another, so it is not clear how many times uncrossing techniques were independently discovered.

Explicit discussion of primal (i.e., basis) uncrossing results seems to have appeared first in the mid-1980s in the context of the subtour relaxation for the traveling salesperson problem: see the work of Boyd & Pulleyblank [26, 24] and Cornuéjols, Fonlupt & Naddef [50] (who also credit Bill Cunningham).

It appears that *strong primal* polyhedral uncrossing was first noted by Jain in 1998 [124] (even though the weak version suffices for the purposes of his approximation algorithm). This built upon earlier combinatorial uncrossing results in the 1990s [85, 94, 197, 100]; see also the note on polyhedral uncrossing for 0-1 proper functions in the survey [96]. We remark that a good fraction of the machinery used by Jain was also used by Frank [70], who solved a special case of Jain's problem exactly. As an aside, we also note non-polyhedral work on uncrossing covers of min-cuts, both in directed [71] and undirected [197, 28] graphs.

So far, all polyhedral uncrossing techniques that we are aware of (set uncrossing as outlined above, the lattice polyhedron [116] model and relatives [179, Ch. 60]) depend on replacing a pair with another pair. The main new technique in our partition uncrossing results (Chapter 3 and Section 3.6) is that we replace a pair of partitions with more than two partitions. (This technical step is encapsulated in Lemma 3.11; we give a reason in Example 3.8 why this extra complication is truly necessary and not just an artifact of our approach.) This general idea seems simple enough that it may have other applications.

Chapter 2

Steiner Tree LPs: Introduction & Summary

In the *Steiner tree* problem, we are given an undirected graph $G = (V, E)$, non-negative costs c_e for all edges $e \in E$, and a set of *terminal* vertices $R \subseteq V$. The goal is to find a minimum-cost tree T spanning R, and possibly some *Steiner vertices* from $V \setminus R$. The problem takes a central place in the theory of combinatorial optimization and has numerous practical applications (e.g., see [121, 167]). The problem is one of Karp's [126] original 21 NP-hard problems, and Chlebík and Chlebíková show that no $(96/95 - \epsilon)$-approximation algorithm can exist for any positive ϵ unless P=NP [45]. The same authors also show that it is NP-hard to obtain an approximation ratio better than $\frac{128}{127}$ for *quasi-bipartite* instances of the Steiner tree problem; these are instances in which no two Steiner vertices are adjacent in the underlying graph G.

One of the first approximation algorithms for the Steiner tree problem is the well-known *minimum-spanning tree heuristic* which is widely attributed to Moore [90]. Moore's algorithm has a performance ratio of 2 for the Steiner tree problem and this remained the best known until the 1990s, when Zelikovsky [202] suggested computing Steiner trees with a special structure, so called *r-restricted Steiner trees* based on the *full component decomposition*. Nearly all of the Steiner tree algorithms developed since then use r-restricted Steiner trees.

Given a Steiner tree T, a *full component* of T is a maximal subtree of T all of whose leaves are terminals and all of whose internal nodes are Steiner nodes. The edge set of any Steiner tree can be partitioned in a *unique* way into full components by splitting at internal terminals; see Figure 2.1 for an example.

The above observation leads to the following hypergraph view of the Steiner tree problem which was first made explicit by Warme [195] and Prömel and Steger [173]. Let $\mathcal{K}$ be

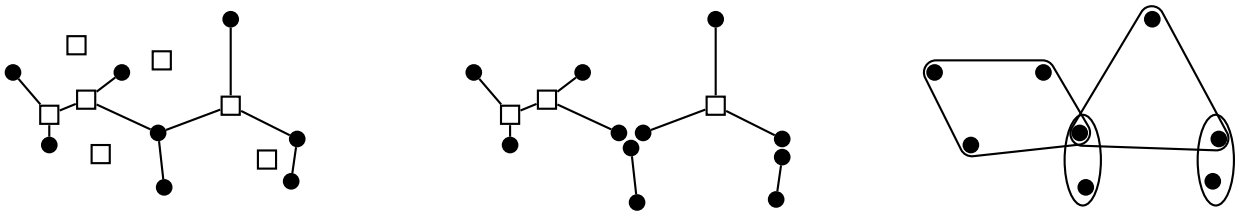


Figure 2.1: Black nodes are terminals and white nodes are Steiner nodes. Left: a Steiner tree for this instance. Middle: the Steiner tree's edges are partitioned into full components; there are four full components. Right: the hyperedges corresponding to these full components.

the set of all nonempty subsets of terminals (*hyperedges*). We associate with each $K \in \mathcal{K}$ a fixed min-cost full component spanning the terminals in K, and let C_K be its cost[1]. The problem of finding a minimum-cost Steiner tree spanning R now reduces to that of finding a minimum-cost hyper-spanning tree in the hypergraph $(R, \mathcal{K})$.

The basic terminology for hypergraphs is as follows. A hyperedge (called just an edge sometimes) is any nonempty subset of the vertex set (usually of size at least 2). Vertices v, v' are connected if there is a sequence $v = v_1 \in e_1 \ni v_2 \in e_2 \ni \cdots \ni v_\ell = v'$ of vertices and hyperedges in the hypergraph. A connected hypergraph is one in which all vertices are connected. A simple cycle is a sequence of distinct vertices and hyperedges with $r_1 \in K_1 \ni r_2 \in K_2 \cdots \ni r_\ell \in K_\ell \ni r_1$ and $\ell > 1$. A hyper-spanning tree may then be defined as a connected hypergraph with no simple cycles: see the right of Figure 2.1 for an example.

There are technical points in the now-standard full component approach which we must now address. It is not obvious how easy it is to compute C_K, but the dynamic programming algorithm of Dreyfus and Wagner [56] allows us to determine C_K (and an optimal full component, if one exists) for a given K in time $O(\exp(|K|)\text{poly}(|V|))$. Even though there exponentially many subsets of R, we will only compute C_K for those $K \subset R$ with size $|K| \leq r$ — such K are called *r-restricted full components*. For fixed r there are polynomially many r-restricted full components and each C_K can be computed in polynomial time. Therefore in polynomial time we can compute the set

$$\mathcal{K}_r := \{K \subseteq R : 2 \leq |K| \leq r \text{ and there exists a full component whose terminal set is } K\}$$

and C_K for all $K \in \mathcal{K}_r$. An instance in which all full components have size at most r is called *r-restricted*. From now on, we will stop writing the ubiquitous subscript r and just write $\mathcal{K}$.

[1]If there is no full component spanning K, we let C_K be infinity.

Notation. For ease of notation we use K to mean both a subset of R, and the min-cost full component (tree) with leaf set K. In Sections 4.4 and 4.6 we will need to be more precise; there we use K for the tree and $R(K)$ for its leaves (e.g., $R(K) \subset R$).

We now introduce the notion of *r-preprocessing*. We talked above about transforming a Steiner tree instance $\mathcal{I} = (V, R, E, c)$ into a hypergraph $(\mathcal{K}, C)$. We may also transform the hypergraph back into a Steiner tree instance $\mathcal{I}' = (V', R', E', c')$ that is equivalent to the hypergraph — specifically the new instance contains no full components of size greater than r, and the feasible hypergraph solutions are essentially isomorphic to the feasible Steiner tree solutions on the new instance. We construct $\mathcal{I}'$ starting with just the terminals R of the original instance. Then we delete all edges and Steiner nodes of the original instance. We set $R' = R$. Then, for each $K \in \mathcal{K}$, we take the min-cost full component with leaf set K, clone its *elements* (edges and Steiner nodes) and put the clones in V', E', and attach the clones to R' the same as they were connected to R. The edge costs are copied from the original instance. Reiterating, $\mathcal{I}'$ has just terminals and element-disjoint full components, one for each $K \in \mathcal{K}$. We say that *r-preprocessing* is the process which transforms $\mathcal{I}$ to $\mathcal{I}'$; note that $\mathcal{I}'$ is r-restricted. This transformation, although pedantic on first sight, is sometimes useful because it establishes that the "r-restricting full component approach" still results in a Steiner tree instance. (Above, *essentially* isomorphic was a weasel word due to the fact that a full component K in $\mathcal{I}'$ also has sub-full components (sub-trees) $K' \subset K$; but we can always replace such a sub-full component by the clone corresponding to K' without increasing costs.)

What is the cost of r-restricting? Consider for example a Steiner tree instance where the graph is a star with one Steiner centre, five terminal tips, unit edge costs; now take $r = 3$. The cheapest Steiner tree has cost 5 (take all edges), but the cheapest r-restricted hyper-spanning tree has cost 6 (two full components each of size 3 and cost 3). The *r-Steiner ratio*, denoted ρ_r, is the largest possible ratio of the min-cost r-restricted spanning hypergaph and the min-cost Steiner tree in the original instance (so we see $\rho_3 \geq 6/5$). Fortuitously, Borchers and Du [21] exactly determined the r-Steiner ratio for all r, showing $\rho_r = 1 + \Theta(1/\log r)$. (More precisely, for $r = 2^a + b$ with $0 \leq b < 2^a$, $\rho_r = 1 + (a + \frac{b}{2^a})^{-1}$.) Therefore for any fixed $\epsilon > 0$, for large enough r, focusing on r-restricted full components only affects the optimal value by a $1 + \epsilon$ factor.

Aside from the ρ_r factor loss, one should ask what is the difference by looking at the hypergraph view. Computing min-cost spanning r-restricted hypergraphs is not computationally easy. For $r = 2$ it is just the minimum spanning tree problem. The case $r = 3$ admits a polynomial-time algorithm (by Lovász) for unit weights; for general weights a PTAS is known [173] but the problem is neither known to be in P nor NP-complete. For $r \geq 4$ the problem is APX-hard with inapproximability ratio $\Omega(\ln r)$ — see Section 4.8. However, fortuitously again, the hypergraphs obtained from the Steiner tree problem have special properties which are exploited algorithmically to get approximation algorithms

better than what is possible for arbitrary hypergraphs.

In [202], Zelikovsky used 3-restricted full components to obtain an 11/6-approximation for the Steiner tree problem. Subsequently, a series of papers (e.g., [16, 203, 118, 127, 173]) improved upon this result. These efforts culminated in a $\left(1+\frac{\ln 3}{2}\right) \approx 1.55$-approximation algorithm of Robins and Zelikovsky [178] (subsequently referred to as RZ) for the Steiner tree problem for r-restricted instances. They hence obtain, for each fixed $r \geq 2$, a $1.55\rho_r$ approximation algorithm for the (unrestricted) Steiner tree problem, i.e. a $1.55+\epsilon$ approximation for every fixed $\epsilon > 0$. We refer the reader to two surveys in [101, 174].

2.1 LP Approaches & Background on MST

Development of combinatorial algorithms has been paralleled by extensive LP-based research to understand the Steiner tree problem. Such a polyhedral study often leads to better exact and approximate algorithms (although this has not yet actually happened in the particular case of Steiner trees). There is vast literature on various LP relaxations for the Steiner tree problem, e.g. [5, 47, 48, 54, 58, 95, 168, 194, 200]. One offshoot of better LP relaxations is to achieve vast improvements in the area of integer programming-based exact algorithms (e.g., see Warme [194] and Polzin [167, 169]) for the Steiner tree problem.

Despite their apparent strength, none of the above LP relaxations has been proven to exhibit an integrality gap smaller than 2 in general. In particular, the best LP-based approximation algorithm for the Steiner tree problem is due to Goemans and Bertsimas [93], and has a performance guarantee of $2-\frac{2}{|R|}$. This algorithm uses the weakest among the Steiner tree LP formulations cited above — the *undirected cut relaxation* [5] — whose integrality gap is precisely $2-\frac{2}{|R|}$. (Other LP-relative 2-approximation algorithms for the Steiner tree problem were later obtained by Agarwal et al. [2] and Jain [124].)

Improved LP-based approximation algorithms have so far only been obtained for structured instances of the Steiner tree problem. Notably, Chakrabarty, Devanur, and Vazirani [34] recently showed that the *bidirected cut relaxation* [58, 200] has an integrality gap of at most 4/3 for quasi-bipartite instances of the problem, improving upon an earlier bound of 3/2 [176, 177]. (Nonetheless, RZ achieves an approximation ratio better than $\frac{4}{3}$ for these graphs.) The worst known example is due to Goemans [1] and exhibits an integrality gap of only $\frac{8}{7}$ (see also a compact example due to Skutella reported in Section 4.8.4). The bidirected cut relaxation is widely conjectured (e.g. in [191]) to have an integrality gap smaller than 2, but a proof remains elusive.

Our work is strongly motivated by linear programming formulations for the spanning tree polyhedron due to Fulkerson [81] and Chopra [46]. We now introduce Chopra's partition-based formulation for the MST polytope, which is important for a few reasons.

$$\begin{aligned} \min \quad & \sum_{e\in E} c_e x_e && (\mathcal{M}) \\ \text{s.t.} \quad & \sum_{e\in E_\pi} x_e \geq r(\pi) - 1 \quad \forall \pi \in \Pi, \\ & x \geq 0. \end{aligned} \qquad \begin{aligned} \max \quad & \sum_{\pi\in\Pi} (r(\pi)-1)\cdot y_\pi && (\mathcal{M}_D) \\ \text{s.t.} \quad & \sum_{\pi: e\in E_\pi} y_\pi \leq c_e \quad \forall e \in E, && (\mathcal{M}_D.1) \\ & y \geq 0. && (\mathcal{M}_D.2) \end{aligned}$$

Figure 2.2: Chopra's MST formulation and its dual.

First, the formulation will be generalized to get LP relaxations for the Steiner tree problem (the Steiner partition formulation, and the hypergraph-partition formulation). Second, one may re-interpret Kruskal's spanning tree algorithm [147] (see Section 3.6.1) as a primal-dual algorithm using this LP, which will be useful.

A *partition of* X is a family of disjoint nonempty sets (*parts*) whose union is X. The *rank* $r(\pi)$ of a partition π is the number of parts of π. Here and later, E_π denotes the set of edges whose ends lie in different parts of π.

To formulate the minimum-cost spanning tree (MST) problem as an LP, we associate a variable x_e with every edge $e \in E$. Each spanning tree T corresponds to its *incidence vector* x^T, which is defined by $x^T_e = 1$ if T contains e and $x^T_e = 0$ otherwise. We show Chopra's MST formulation $(\mathcal{M})$ along with its dual $(\mathcal{M}_D)$ in Figure 2.2.

Chopra [46] showed that the feasible region of $(\mathcal{M})$ is the dominant[2] of the convex hull of all incidence vectors of spanning trees, and hence each basic optimal solution corresponds to a minimum-cost spanning tree.

We use $\texttt{MST}(G)$ to denote a minimum-spanning tree of the graph G and $\texttt{mst}(G)$ to denote its cost. Notation like $\texttt{MST}(G, c)$ specifies a cost function to use. Later (using Chopra's result) $(T, y) = \texttt{MST}(G)$ denotes a primal-dual pair where y is feasible for $(\mathcal{M}_D)$ and T, y have the same objective value.

2.2 A List of LPs for Steiner Trees

We now introduce several LP formulations known for the Steiner tree problem. This is not a comprehensive list — see [95, 168, 170] for more. A shared feature of the LPs below is that when $R = V$, they are integral.

[2]The dominant of polyhedron P is $\{x \mid \exists x' \in P : x' \leq x\}$.

$$\min \Big\{ \sum_{a \in A} c_a x_a : \quad x \in \mathbf{R}^A \tag{$\mathcal{B}$}$$

$$\sum_{a \in \delta^{\text{in}}(U)} x_a \geq 1, \quad \forall U \in \text{valid}(V) \tag{$\mathcal{B}$.1}$$

$$x_a \geq 0, \quad \forall a \in A \Big\} \tag{$\mathcal{B}$.2}$$

Figure 2.3: The bidirected cut relaxation.

2.2.1 Bidirected Cut Formulation

The bidirected cut formulation, initially mentioned by Wong [200], is a generalization of the arborescence polytope formulation of Edmonds [58]. In order to obtain this relaxation, we introduce two arcs (u, v) and (v, u) for each edge $uv \in E$, and let both of their costs be c_{uv}. We then fix an arbitrary terminal $r \in R$ as the root.

By orienting the edges of a Steiner tree T towards r we obtain the unique in-tree corresponding to T in the digraph (V, A). Call a subset $U \subseteq V$ *valid* if $r \in U$ and at least one terminal lies outside U (i.e., $R \backslash U \neq \varnothing$); let valid$(V)$ be the family of all valid subsets of V. Clearly, the in-tree corresponding to Steiner tree T must have at least one arc entering each such set $U \in \text{valid}(V)$. This motivates the bidirected cut relaxation which has a variable for each arc $a \in A$, and a constraint for every valid set U. Here and later, $\delta^{\text{in}}(U)$ denotes the set of arcs in A whose head is in U and whose tail lies in $V \setminus U$.

The bidirected cut relaxation is one of the most well-studied relaxations for the Steiner tree problem [95, 98, 47, 176, 191, 1, 34]. The largest lower-bound on its integrality gap is $8/7$ (due to Goemans [1]). Many conjecture the actual gap to be closer to this bound than to 2 (e.g. [176, 191]).

2.2.2 Bidirected Cut with Flow-Balance

Polzin & Vahdati Daneshmand [168] investigated (citing [132]) a family of constraints that can be added to the bidirected cut formulation: namely for every Steiner node, constrain the flow into that node to be less than or equal to the flow out of that node. Clearly this is valid for every in-Steiner-tree rooted at r. They showed that there are some Steiner tree instances in which this LP is strictly stronger than the normal bidirected cut formulation.

Chakrabarty et al. [34] introduced two *embedding formulations* for the Steiner tree problem, one "on the simplex" and another "above the simplex." They observed that the

"on the simplex" LP is exactly as strong as the bidirected cut formulation. We mention here (but omit the straightforward details) that the "above the simplex" LP is exactly as strong as bidirected cut with the flow-balance constraints.

2.2.3 Steiner Partition Formulation

In an instance of the Steiner tree problem, a partition π of V is defined to be a *Steiner partition* when each part of π contains at least one terminal [47, 106, 196].

Recall that E_π denotes the set of edges whose ends lie in different parts of π. Consider the following LP. When x is the incidence vector of a Steiner tree and π is a Steiner partition, the inequality

$$\sum_{e \in E_\pi} x_e \geq r(\pi) - 1. \tag{2.1}$$

holds. The *Steiner partition formulation* is defined to be the polytope in $\mathbf{R}_+^E$ satisfying all Steiner partition constraints (2.1).

White et al. [196] showed that the Steiner partition formulation has an integral optimal value for strongly chordal unit-weight graphs, but not for all series-parallel unit-weight graphs (explicitly, the so-called *odd hole* graph on 3 terminals and 4 Steiner nodes). Grötschel et al. [105] showed that it is NP-hard to separate the Steiner partition constraints.

2.2.4 Subtour Hypergraph Formulation

Warme [194] introduced the first hypergraph-based LP formulation for the Steiner problem. Let $\rho(X) := \max(0, |X| - 1)$ be the *rank* of a set X of vertices. It is not hard to prove that a collection $(R, \mathcal{K}')$ forms a hyper-spanning tree iff $\sum_{K \in \mathcal{K}'} \rho(K) = \rho(R)$ and $\sum_{K \in \mathcal{K}'} \rho(K \cap S) \leq \rho(S)$ for every subset S of R. We thereby obtain the LP relaxation of the Steiner tree problem shown in Figure 2.4, which we call the *subtour formulation*.

Although ($\mathcal{S}'$) has exponentially many constraints, one can separate over ($\mathcal{S}'$) using a flow subroutine [194, Section 4.1.2.1] attributed to Maurice Queyranne, which takes time polynomial in $|\mathcal{K}|$ (and the encoding length of C). (Another approach is to use a compact formulation for ($\mathcal{S}'$) presented in Section 4.4 based on strengthening the bidirected cut relaxation and preprocessing.)

As for the LP when we do not r-restrict but rather take all full components: it is not known whether ($\mathcal{S}'$) can be exactly solved in this case, but it is easy to argue that OPT($\mathcal{S}'$) increases by at most a factor ρ_r if we look at the r-restricted instance, so we can solve the LP within any $1 + \epsilon$ factor in polynomial time.

$$\min\Big\{\sum_{K\in\mathcal{K}} C_K x_K : \quad x\in\mathbf{R}^{\mathcal{K}} \tag{$\mathcal{S}'$}$$

$$\sum_{K\in\mathcal{K}} x_K\rho(K) = \rho(R), \tag{$\mathcal{S}'.1$}$$

$$\sum_{K\in\mathcal{K}} x_K\rho(K\cap S) \le \rho(S), \quad \forall S\subset R \tag{$\mathcal{S}'.2$}$$

$$x_K \ge 0, \quad \forall K\in\mathcal{K}\Big\} \tag{$\mathcal{S}'.3$}$$

Figure 2.4: The subtour relaxation.

2.2.5 Directed Hypergraph Formulations

Given a full component (tree) K and $i\in K$, let K^i denote the arborescence obtained by directing all the edges of K towards i. Think of this as directing the hyperedge K towards i to get the directed hyperedge K^i. Vertex i is called the *head* of K^i while the terminals in $K\setminus i$ are the *tails* of K. The cost of each directed hyperedge, K^i is the cost of the undirected hyperedge, K.

A subset U of R is *valid* if $r\in U$ and $U\neq R$; *valid*(R) denotes all valid subsets of R. For $U\subset R$ let $\Delta^{\text{in}}(U)$ be the set of directed full components entering U, that is all K^i such that $i\in U$ and $U\setminus K\neq\varnothing$; define also $\Delta^{\text{out}}(U)=\Delta^{\text{in}}(R\backslash U)$. Given a hyper-spanning tree, it is evident we can direct its hyperedges towards the root r (in the sense that the tree contains a unique path from every vertex to r). For every valid set X the resulting directed hyper-spanning tree has at least one edge in $\Delta^{\text{in}}(X)$, which leads to the directed hypergraphic formulation below, introduced by Polzin & Vahdati Daneshmand [170].

$$\min\Big\{\sum_{K\in\mathcal{K},i\in K} C_{K^i}Z_{K^i} : \tag{$\mathcal{D}$}$$

$$\sum_{K^i\in\Delta^{\text{in}}(U)} Z_{K^i} \ge 1, \quad \forall U\in \text{valid}(R) \tag{$\mathcal{D}.1$}$$

$$Z_{K^i}\ge 0, \quad \forall K\in\mathcal{K}, i\in K\Big\} \tag{$\mathcal{D}.2$}$$

The same authors also introduced the following *bounded* version of the formulation:

$$\min\Big\{\sum_{K^i} C_{K^i}x_{K^i} : \quad x\in\mathcal{D};\ x(\Delta^{\text{out}}(r))=0;\ x(\Delta^{\text{out}}(v))=1\ \forall v\in R\backslash\{r\}\Big\} \tag{$\mathcal{D}'$}$$

$$\min\Big\{\sum_{K\in\mathcal{K}} C_K x_K : \quad x\in\mathbf{R}^{\mathcal{K}} \qquad (\mathcal{P})$$
$$\sum_{K\in\mathcal{K}} x_K \mathtt{rc}_K^\pi \geq r(\pi)-1, \quad \forall \pi\in\Pi_R \quad (\mathcal{P}.1)$$
$$x_K\geq 0, \quad \forall K\in\mathcal{K}\Big\} \quad (\mathcal{P}.2)$$

$$\max\Big\{\sum_{\pi}(r(\pi)-1)y_\pi : \quad y\in\mathbf{R}^{\Pi_R} \qquad (\mathcal{P}_D)$$
$$\sum_{\pi\in\Pi_R} y_\pi \mathtt{rc}_K^\pi \leq C_K, \quad \forall K\in\mathcal{K} \quad (\mathcal{P}_D.1)$$
$$y_\pi\geq 0, \quad \forall \pi\subset\Pi_R\Big\} \quad (\mathcal{P}_D.2)$$

Figure 2.5: The unbounded partition relaxation and its dual.

2.2.6 Partition Hypergraph Formulations

In this section we give hypergrapic formulations introduced in [136, 138, 139] based on partitions. We let Π_R be the set of all partitions of R; the LP has a constraint for every partition in Π_R.

Given a partition $\pi=\{\pi_1,\ldots,\pi_q\}$ of R, any feasible hyper-spanning tree must connect the q parts of π. In order to express this fact, we define the *rank contribution* $\mathtt{rc}_K^\pi$ of hyperedge $K\in\mathcal{K}$ for π as the effective rank reduction of π obtained by merging the parts of π that are touched by K; i.e.,

$$\mathtt{rc}_K^\pi = |\{i \,:\, K\cap\pi_i\neq\varnothing\}|-1.$$

The hypergraphic LP and its dual are given in Figure 2.5.

The feasible region of the partition LP is *unbounded* in the geometric sense, for the following reason: if x is a feasible solution for ($\mathcal{P}$) then so is any $x'>x$. It is not hard to see (or see Warme [194]) that the following constraint is valid for all hyper-spanning trees.

$$\sum_{K\in\mathcal{K}} x_K(|K|-1)=|R|-1. \qquad (\mathcal{P}'.1)$$

Adding this constraint to ($\mathcal{P}$) we obtain the *bounded* partition relaxation ($\mathcal{P}'$).

$$\min\Big\{\sum_{K\in\mathcal{K}} C_K x_K : x\in\mathcal{P}, (\mathcal{P}'.1) \text{ holds for } x\Big\} \qquad (\mathcal{P}')$$

On 2-restricted instances (i.e., ordinary graphs), ($\mathcal{P}$) is Chopra's [46] integral partition-based LP for min-cost spanning tree. On 3-restricted instances, ($\mathcal{P}$) is essentially a special case of the *matroid matching* LP introduced by Vande Vate [189] (which is totally dual half-integral [89]).

2.2.7 Gainless Tree Formulations

The final formulations we introduce are not linear programs, but closely related. A *terminal spanning tree* is defined to be any possible (non-hypergraphic) spanning tree T of R; the edge costs of T can be arbitrary and do not need to depend on the input instance. Given a terminal spanning tree T with its costs denoted c' for clarity, and any hyperedge $K \subset R$, we define the *gain* of K in T to be the cost decrease when K is included in the spanning tree,

$$\texttt{gain}_T(K) := c'(T) - \texttt{mst}(T/K, c') - C_K,$$

where $\texttt{mst}(T/K, c')$ is the minimum cost of any spanning tree in the graph T after the terminals K are contracted into a single pseudonode. So to say that K has positive gain means that T can be replaced by a cheaper spanning hypertree if K is included. We say that a terminal spanning tree T is *gainless* if $\texttt{gain}_T(K) \leq 0$ for all $K \in \mathcal{K}$.

We now give the gainless tree formulation; they come originally from Karpinksi and Zelikovsky [127].

Definition 2.1. *The quantity $t_+^{\mathcal{K}}$ is the maximum cost of any gainless terminal spanning tree T with arbitrary nonnegative edge weights. The quantity $t^{\mathcal{K}}$ is the maximum cost of any gainless terminal spanning tree T with arbitrary edge weights.*

We will show these values are the same as those arising from the hypergraph LPs. Note that this equates a combinatorial value with an LP bound, analogous to the theorem of Held and Karp relating so-called *1-trees* to the optimum of an LP relaxation [112] for TSP.

Here is some intuition as to why gainless trees are significant, based on the well-known *contraction lemma* from the Steiner tree literature (Lemma 5.18). The contraction lemma implies that if we take a disjoint union of a Steiner tree instance with a gainless terminal spanning tree (T, c'), then T is an optimal Steiner tree for the union. Therefore every Steiner tree in the original instance has cost at least $c'(T)$. Now considering all possible T we deduce that $t_+^{\mathcal{K}}$ is a lower bound on the cost of an optimal Steiner tree.

2.3 Summary of Known and New Results

A main result of our work is that all the hypergraphic relaxations above have the same value. This extends work of Polzin & Vahdati Daneshmand [170] who showed equivalence of the directed and subtour formulations. We defer the details for a moment.

Trivially, the bidirected cut relaxation is strengthened by adding flow-balance constraints; Polzin & Vahdati Daneshmand [170] showed that the strengthening is sometimes strict. Goemans [98] proved that the bidirected cut formulation is stronger than the

Steiner partition formulation; it can also be shown (we will do so in Proposition 2.2) that this strengthening is sometimes strict. Furthermore, Polzin & Vahdati Daneshmand [170] showed that the hypergraphic formulation strengthens the flow-balance-plus-bidirected formulation, again sometimes strictly. Therefore we have

$$\text{hypergraphic} \gneq (\mathcal{B})\text{+flow-balance} \gneq (\mathcal{B}) \gneq \text{Steiner partition}$$

where $\mathcal{LP} \gneq \mathcal{LP}'$ denotes that the optimal value of $\mathcal{LP}$ is always at least as great as the value of $\mathcal{LP}'$, and that on some instances the values are unequal.

We show (again, deferring the details momentarily) that in all quasi-bipartite graphs, the bidirected cut formulation has equal value to the hypergraphic formulations. We also demonstrate that bidirected cut is not equal to the Steiner partition formulation on quasi-bipartite instances (Proposition 2.2); so in quasi-bipartite instances we have

$$\text{hypergraphic} = (\mathcal{B})\text{+flow-balance} = (\mathcal{B}) \gneq \text{Steiner partition}.$$

Here is the quasi-bipartite example separating the Steiner partition and bidirected formulations.

Proposition 2.2. *There is a quasi-bipartite instance of the Steiner tree problem in which the optimum value of the bidirected cut formulation is strictly greater than the optimum of the Steiner partition formulation.*

Proof. Consider the example in Figure 2.6. The bidirected cut relaxation has optimal value 7 on this instance — to quickly prove this, note that the graph is series-parallel, and it is known [98] that bidirected cut is integral on series-parallel graphs. However, with a small computation one may verify that the Steiner partition relaxation has optimal value 48/7 — we can assign fractional value 4/7 to every edge. □

2.3.1 Equivalence of Formulations

In Figure 2.7, we outline the various known and new equivalences between the hypergraphic relaxations.

- Theorem 4.1. We show that $(\mathcal{P}')$ and $(\mathcal{S}')$ are actually polyhedrally equal, i.e., that a solution is feasible for one iff it is feasible for the other.
- Theorem 4.5. We show that $\text{OPT}(\mathcal{P}) = \text{OPT}(\mathcal{D})$ by using hypergraph orientation results of Frank et al. [73, 72].

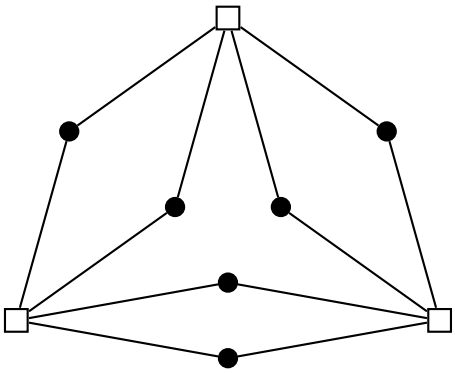


Figure 2.6: A quasi-bipartite unit-cost example upon which the hypergraph/bidirected cut relaxations are strictly stronger than the Steiner partition formulation. Dots are terminals and squares are Steiner vertices.

- Theorem 3.16. Note ($\mathcal{P}'$) trivially strengthens ($\mathcal{P}$). We show the nontrivial converse $\text{OPT}(\mathcal{P}') \leq \text{OPT}(\mathcal{P})$ by showing that some optimal solution to ($\mathcal{P}$) satisfies the constraint ($\mathcal{P}'.1$). This is done with a new technique, (primal) *partition uncrossing*. We also rely on a notion we call *shrinking*: replace a fraction of a full component K with another full component $K' \subset K$.
- Theorems 3.22 and 3.21. We equate the gainless tree formulation values with the partition formulation values using *dual* partition uncrossing.
- Theorem 4.4. This is an elaboration of a sketch given in [170], using shrinking.

Note that the "equivalence graph" in Figure 2.7 is cyclic, i.e. redundant, but we include all proofs since they are significantly different from one another.

2.3.2 Polyhedral Comparisons with Bidirected Cut

Polzin & Vahdati Daneshmand [170] observed that ($\mathcal{D}$) is at least as strong as the bidirected cut formulation. We are able to show (Section 4.5) that in quasi-bipartite graphs, the reverse inequality $\text{OPT}(\mathcal{D}) \leq \text{OPT}(\mathcal{B})$ is true, i.e. the bidirected formulation has value equal to the hypergraphic formulations. This is one of the most surprising results from our studies: first, since the LPs have both been studied heavily and this has gone un-noticed;

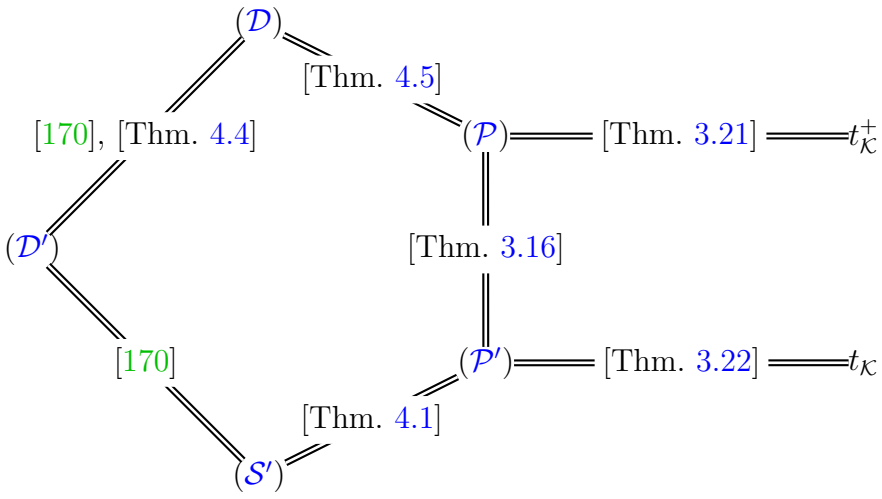


Figure 2.7: Results which show that various hypergraphic formulations have equal value.

second, because the LPs look quite different from one another; third, because they have different values in general instances. We prove $\mathrm{OPT}(\mathcal{D}) \leq \mathrm{OPT}(\mathcal{B})$ with a *lifting argument* which takes the optimal dual solution to $(\mathcal{D})$ and fractionally inserts Steiner nodes into various dual sets to get a dual solution to $(\mathcal{B})$ of the same value.

Lifting also shows (in Section 4.6) that in r-preprocessed graphs for $r \leq 5$, the hypergraphic and bidirected formulations have the same value. Completing this proof requires some brute-force computation and enumeration of all possible configurations of 5-restricted full components, which we do with Maple. For 6-preprocessed graphs, we show (in Section 4.6.4) that the hypergraphic and bidirected formulations are not always equal.

As a final result about the bidirected cut formulation, we give (in Section 4.4) a new explanation of the precise relation between the hypergraphic formulations and the bidirected cut formulation. The result applies to r-preprocessed instances for any r; recall such instances have disjoint full components except at terminals. We start with a subtour formulation for bidirected cut due to Goemans & Myung [95] which has variables for edges and nodes; then we strengthen it by forcing the values of all elements in each full component to be equal. We show that the resulting LP is polyhedrally equivalent to the subtour formulation $(\mathcal{S}')$.

2.3.3 Structural & Extreme Point Results

Chapter 3 is devoted to new *partition uncrossing* techniques. In addition to LP equivalences, we use them to show structural theorems for the partition-hypergraph formulation. This technique is an extension of well-known *set uncrossing* techniques from polyhedral combinatorics to the milieu of partitions. The first natural thing one might try is to replace two partitions by their *meet* and *join*, well-known operations obtained when viewing the partitions as a combinatorial lattice [186]. However, this turns not to work (as we make precise in Chapter 3). A more careful uncrossing, from pairs of partitions into larger families of partitions, does the trick.

As a result of primal and dual partition uncrossing, we find that every extreme primal solution to ($\mathcal{P}$) has at most $|R|-1$ nonzero coordinates, and that its dual ($\mathcal{P}_D$) always has an optimal solution where the partitions in the *support* (those with nonzero dual value) form a *chain*.

In Section 3.5, using these structural properties, we describe a computational result: we enumerated all vertices of ($\mathcal{P}$) when $|R| \leq 6$. We use this data to refute a natural conjecture about the fractionality of the extreme points of ($\mathcal{P}$).

Finally, we mention two other structural results proved in the thesis. Section 4.9 shows that, in contrast to the sparseness of the partition-hypergraph formulation, the bidirected cut formulation has extreme points with support size $\Omega(|V|^2)$. To give the other result some background, we mention a result of Chopra [46]: on ordinary (2-restricted) instances, ($\mathcal{P}$) is the dominant for ($\mathcal{P}'$). This turns out to be false in general hypergraphic instances; we show in Section 4.7 a true result that is similar but more complex.

2.3.4 LP-Relative Algorithms / Integrality Gap Bounds

We show in Section 4.8 that a greedy algorithm of Gröpl et al. [103] gives good ($\mathcal{P}$)-relative approximation guarantees, using a simple *dual fitting* argument.

- On uniformly quasi-bipartite instances — where all edges incident to every Steiner node have the same cost — we get a ($\mathcal{P}$)-relative ratio of 73/60. This matches the 73/60 (non-LP-relative) approximation ratio proven by the original analysis of Gröpl et al [103].
- On 3-restricted instances of the Steiner tree problem, we get a ($\mathcal{P}$)-relative ratio of 5/4. This is the best integrality gap bound known for such instances (although a non-LP-relative PTAS exists).

- On arbitrary hypergraphs (not derived from the Steiner tree problem) with maximum edge size t, we show that the algorithm is a $(\mathcal{P})$-relative $H(t-1)$-approximation algorithm, where $H(i)$ denotes the ith harmonic number. This is nearly best possible since there is a $\ln t - o(\ln t)$ hardness of approximation.

We also give an independent *filtering*-based argument (in Section 4.8.3) that in *unit-cost* quasi-bipartite instances the integrality gap is at most 73/60. Interestingly, the 73/60 value arises a different way in the analysis.

We give (in Section 4.8.2) a new small example showing that the algorithm of Gröpl et al. [103] does not attain approximation ratio better than 73/60.

The largest lower bound known on the integrality gap of the hypergraph formulations is 8/7, due to an example of Skutella, which we give in Section 4.8.4.

2.3.5 LP-Based Interpretations of Other Known Algorithms

Chapter 5, the final chapter on Steiner trees, contains LP-inspired analyses of two approximation algorithms from the literature on Steiner trees.

First, we give the details of one of the original results that inspired our investigation into hypergraph-based LPs: we show that the Robins-Zelikovsky algorithm can be interpreted using a generalization of $(\mathcal{P})$. We also deduce some tighter approximation/integrality gap results in a class of instances called *b-quasi-bipartite*, which means that $G \backslash R$ has connected components of size at most b. This part of the thesis includes a new matroid-based generalization of the *contraction lemma*, which features prominently in the analysis of mdoern approximation algorithms for Steiner tree.

Second, in Section 5.6 we give a new LP for the Steiner tree problem derived from the *submodular set cover* problem and the *relative greedy* algorithm for Steiner tree. The resulting LP has an integrality gap of at most $1+\ln 2$, and we give an algorithm to compute explicit certificates of this gap bound (a feasible Steiner tree and feasible dual whose costs are within a factor $1+\ln 2$ of each other) in polynomial time. However, the LP is somewhat awkward and we do not know how to solve it in polynomial time.

Chapter 3

Partition Uncrossing

In this chapter we investigate the hypergraphic partition LP formulations for Steiner trees and extend the technique of uncrossing, usually applied to set systems, to families of partitions. The technique comes in two flavours, *primal* and *dual*, which lead to characterizations of certain solutions to the partition-hypergraph LP ($\mathcal{P}$) and its dual ($\mathcal{P}_D$). Specifically, extreme point primal solutions have at most $|R|-1$ nonzero coordinates, and there is an optimal dual solution with a well-structured ("uncrossed") support. The first polyhedral consequence we show is that ($\mathcal{P}'$) has the same optimal value as ($\mathcal{P}$).

Next, we briefly discuss how these structural results yield a more efficient procedure to compute all extreme points of ($\mathcal{P}$). We have computed all of these extreme points for $|R| \leq 6$ and we give some of their properties. Finally, we discuss the *dual* version of partition uncrossing and show that the gainless spanning tree values $t_{\mathcal{K}}$ and $t^+_{\mathcal{K}}$ equal the values of the hypergraphic LPs ($\mathcal{P}'$) and ($\mathcal{P}$), respectively.

3.1 Introduction

Uncrossing is a fundamental and ubiquitous technique in combinatorial optimization; among its applications are the Lucchesi-Younger theorem [157], the Edmonds-Giles theorem [60], Jain's 2-approximation algorithm for survivable network design [124], and Singh & Lau's ± 1-degree approximation algorithm for min-cost bounded-degree spanning trees [182]. In all of the above applications, a family of sets is uncrossed by choosing a pair of crossing sets and uncrossing them into a pair of cross-free ones. In this chapter, we define an uncrossing operation for a family of partitions. As it turns out, the standard technique of taking pairs at a time doesn't work and requires replacing a pair of partitions by a a family of possibly more than two uncrossed partitions.

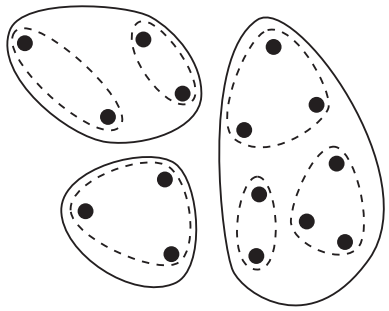


Figure 3.1: The dashed partition refines the solid one.

The family of partitions of R, denoted Π_R, forms a partially ordered set under the *refinement* relation. For any two partitions $\pi = \{\pi_1, \ldots, \pi_q\}$ and $\pi' = \{\pi'_1, \ldots, \pi'_p\}$ we say that π refines π' if for any part π_i of π there is a part π'_j of π' such that $\pi_i \subseteq \pi'_j$. The figure on the right shows partitions π (dashed lines) and π' (solid lines), where π refines π'. A set of partitions $\pi^1, \ldots, \pi^k \in \Pi_R$ is a *chain* if π^i refines π^{i-1} for all $2 \leq i \leq k$. For notational convenience, we let $\overline{\pi}$ be partition of R into singletons, and we define $\underline{\pi}$ be the trivial partition with the single part R.

As mentioned at the start of the chapter, we get structural results in both primal and dual form. Here is the primal structure theorem. (In terms of Section 1.5 it is *strong* primal uncrossing.)

Theorem 3.1 (Primal partition uncrossing)**.** *Let x^* be a basic feasible solution of ($\mathcal{P}$), and let*

$$\Pi_R^* = \{\pi \in \Pi_R \, : \, \sum_{K \in \mathcal{K}} \mathtt{rc}_K^\pi x_K^* = r(\pi) - 1\}$$

be the set of tight *partitions for x^*. Let $\mathcal{C}$ be an inclusion-wise maximal chain in Π_R^* not containing $\underline{\pi}$. Then x^* is uniquely defined by*

$$\sum_{K \in \mathcal{K}} \mathtt{rc}_K^\pi x_K^* = r(\pi) - 1 \quad \forall \pi \in \mathcal{C}. \tag{3.1}$$

Any chain of distinct partitions of R that does not contain $\underline{\pi}$ has size at most $|R|-1$, and this is an upper bound on the rank of the system in (3.1). Elementary linear programming theory immediately yields the following corollary.

Corollary 3.2. *Any basic solution x^* of ($\mathcal{P}$) has at most $|R| - 1$ non-zero coordinates.*

The dual uncrossing structural theorem is particularly simple to state.

Theorem 3.3 (Dual partition uncrossing)**.** *($\mathcal{P}_D$) always has an optimum y^* such that $\mathtt{supp}(y^*)$ is a chain.*

We remark that Theorem 3.3 also implies Corollary 3.2 (since the dual variable for $\underline{\pi}$ is vacuous, and then using standard techniques as in [179, Thm. 5.35] or [104, §8.4]).

3.2 Preliminaries

In order to prove these structural results, and the consequent LP equivalences, we start with some definitions from combinatorial lattice theory. The first was illustrated in Section 3.1.

Definition 3.4. *We say that a partition π'* refines *another partition π if each part of π' is contained in some part of π. We also say π* coarsens *π'. Two partitions* cross *if neither refines the other.*

Definition 3.5. *A family of partitions forms a* chain *if no pair of them cross. Alternatively, a chain is any family $\pi^1, \pi^2, \ldots, \pi^t$ such that π^i refines π^{i-1} for each $1 < i \leq t$.*

The family Π_R of all partitions of R forms a *combinatorial lattice* with a *meet operator* $\wedge : \Pi_R^2 \to \Pi_R$ and a *join operator* $\vee : \Pi_R^2 \to \Pi_R$ defined below. Abstractly, $\pi \wedge \pi'$ is the coarsest partition that refines both π and π', and $\pi \vee \pi'$ is the most refined partition that coarsens both π and π'. See Figure 3.2 for an illustration.

Definition 3.6 (Meet of partitions)**.** *Let the parts of π be $\pi_1, \ldots, \pi_t$ and let the parts of π' be $\pi'_1, \ldots, \pi'_u$. Then the parts of the meet $\pi \wedge \pi'$ are the nonempty intersections of parts of π with parts of π',*

$$\pi \wedge \pi' = \{\pi_i \cap \pi'_j \mid 1 \leq i \leq t, 1 \leq j \leq u \text{ and } \pi_i \cap \pi'_j \neq \varnothing\}.$$

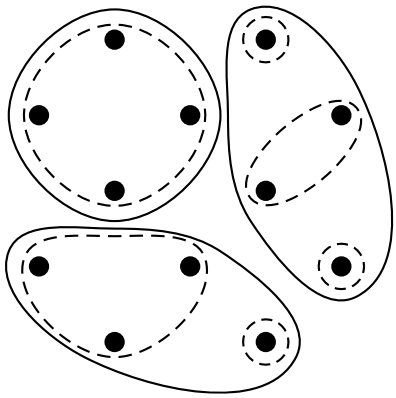
Given a graph G and a partition π of $V(G)$, we say that G *induces* π if the parts of π are the vertex sets of the connected components of G.

Definition 3.7 (Join of partitions)**.** *Let (R, E) be a graph that induces π, and let (R, E') be a graph that induces π'. Then the graph $(R, E \cup E')$ induces $\pi \vee \pi'$.*

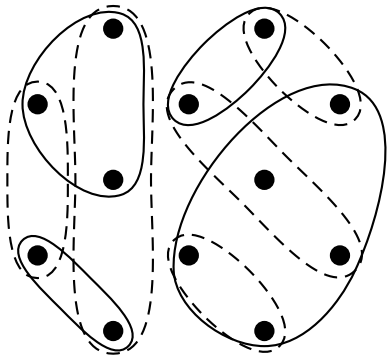
It is not hard to see that meet and join are both commutative and associative, see Stanley [186] for this and a good overview of combinatorial lattice theory in general.

Given a feasible solution x to ($\mathcal{P}$), we are interested in *uncrossing* the following set of *tight* partitions

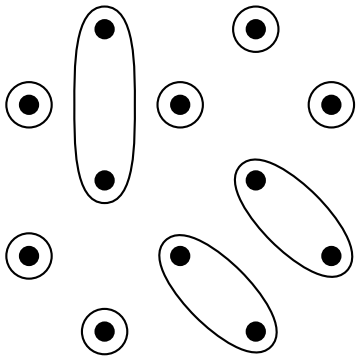
$$\Pi_R^* = \{\pi \in \Pi_R : \sum_{K \in \mathcal{K}} \mathtt{rc}_K^\pi x_K^* = r(\pi) - 1\}$$



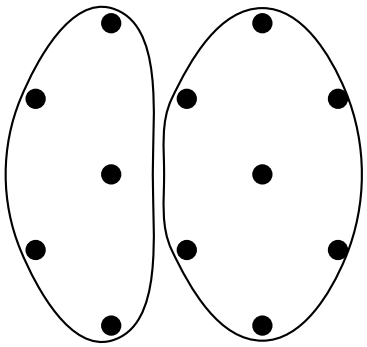
(a): The dashed partition refines the solid one.



(b): Two partitions that cross.



(c): The meet of the partitions from (b).



(d): The join of the partitions from (b).

Figure 3.2: Illustrations of some partitions. The black dots are the terminal set R.

Informally, we would like to prove

If two crossing partitions π and π' are in Π_R^*, then so are $\pi \wedge \pi'$ and $\pi \vee \pi'$. (3.2)

Almost all results based on set-uncrossing ([60, 50, 124, 182]) prove such a theorem (with partitions replaced by sets and meets and joins replaced by unions and intersections), and the standard approach is the following. One considers the equalities in ($\mathcal{P}$) corresponding to π and π' and uses the "supermodularity" of the RHS and the "submodularity" of the coefficients in the LHS. In particular, if the following two inequalities are true,

$$\forall \pi, \pi' : \; r(\pi \vee \pi') + r(\pi \wedge \pi') \quad \geq \quad r(\pi) + r(\pi') \tag{3.3}$$

$$\forall K, \pi, \pi' : \; \mathtt{rc}_K^{\pi} + \mathtt{rc}_K^{\pi'} \quad \geq \quad \mathtt{rc}_K^{\pi \vee \pi'} + \mathtt{rc}_K^{\pi \wedge \pi'} \tag{3.4}$$

then (3.2) can be proved easily by writing a string of inequalities.[1]

Unfortunately, although inequality (3.3) is true (see, for example, [186]), inequality (3.4) isn't always true as the following example shows.

Example 3.8. *Let* $R = \{1,2,3,4\}$, $\pi = \{\{1,2\},\{3,4\}\}$ *and* $\pi' = \{\{1,3\},\{2,4\}\}$. *Let* K *denote the full component* $\{1,2,3,4\}$. *Then* $\mathtt{rc}^{\pi}_K + \mathtt{rc}^{\pi'}_K = 1+1 < 0+3 = \mathtt{rc}^{\pi\vee\pi'}_K + \mathtt{rc}^{\pi\wedge\pi'}_K$.

Nevertheless, the statement (3.2) is correct and we prove it in Section 3.3. The crux is not to consider *pairs* of equalities of ($\mathcal{P}$), but rather consider (multi)-sets of equalities and use them instead. We give details in the next subsection. We close this section with some remarks.

Remark 3.9. *The naïve approach just described* does *work if all edges of the hypergraph have size at most 3. This can be viewed in the framework of* lattice polyhedra *[116] (e.g., see [189]), but for larger hyper-edges this is no longer the case (e.g., due to Example 3.8).*

If we replace $\mathtt{rc}^{\pi}_K$ *in* ($\mathcal{P}$) *by* $\min\{1, \mathtt{rc}^{\pi}_K\}$ *we again are in a situation where the naïve approach works. Then* ($\mathcal{P}$) *is an integral formulation for all* partition-connected *hypergraphs (see [74, 73]), and may be viewed as a contrapolymatroid as well as a lattice polyhedron.*

A sort of partition uncrossing operation, from pairs to pairs, is given by Schrijver [179, Thms. 48.2 & 49.4], based on set uncrossing. This can be developed into alternate proofs of some of our partition uncrossing results.

3.3 Partition Uncrossing Inequalities

In this section we develop the technology which avoids the problems with the naive approach sketched earlier. We start with the following definition.

Definition 3.10. *Let* $\pi \in \Pi_R$ *be a partition and let* $S \subset R$. *Define the* merged partition $m(\pi, S)$ *to be the most refined partition that coarsens* π *and contains all of* S *in a single part. See Figure 3.3 for an example. Informally,* $m(\pi, S)$ *is obtained by merging all parts of* π *which intersect* S. *Formally, the parts of* $m(\pi, S)$ *are* $\{\{\pi_j\}_{j:\pi_j\cap S=\emptyset}, \bigcup_{j:\pi_j\cap S\neq\emptyset}\pi_j\}$.

We will use the following straightforward fact later:

$$\mathtt{rc}^{\pi}_K = r(\pi) - r(m(\pi, K)). \tag{3.5}$$

We now state the (true) inequalities which replace the false inequality (3.4). Later, we show how one uses these to obtain partition uncrossing, e.g. to prove (3.2).

[1] We get $r(\pi)+r(\pi')-2 = \sum_K x_K(\mathtt{rc}^{\pi}_K + \mathtt{rc}^{\pi'}_K) \geq \sum_K x_K(\mathtt{rc}^{\pi\wedge\pi'}_K + \mathtt{rc}^{\pi\vee\pi'}_K) \geq r(\pi\wedge\pi')+r(\pi\vee\pi')-2 \geq r(\pi)+r(\pi')-2$; thus the inequalities hold with equality, and the middle one shows $\pi\wedge\pi'$ and $\pi\vee\pi'$ are tight.

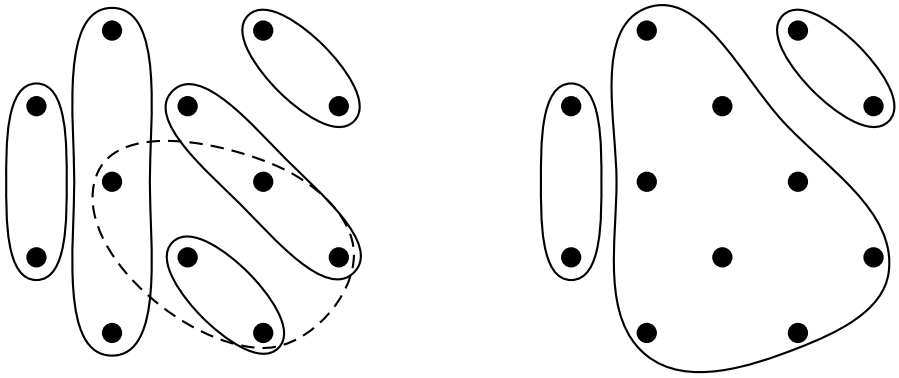


Figure 3.3: Illustration of merging. The left figure shows a (solid) partition π along with a (dashed) set S. The right figure shows the merged partition $m(\pi, S)$.

Lemma 3.11 (Partition Uncrossing Inequalities)**.** *Let $\pi, \pi' \in \Pi_R$ and let the parts of π be $\pi_1, \pi_2, \ldots, \pi_{r(\pi)}$.*

$$r(\pi)\,[r(\pi')-1] + [r(\pi)-1] \;=\; [r(\pi\wedge\pi')-1] + \sum_{i=1}^{r(\pi)} [r(m(\pi',\pi_i))-1] \tag{3.6}$$

$$\forall K \in \mathcal{K}: \quad r(\pi)\left[\mathtt{rc}_K^{\pi'}\right] + \left[\mathtt{rc}_K^{\pi}\right] \;\geq\; \left[\mathtt{rc}_K^{\pi\wedge\pi'}\right] + \sum_{i=1}^{r(\pi)} \left[\mathtt{rc}_K^{m(\pi',\pi_i)}\right] \tag{3.7}$$

Before giving the proof of the above lemma, let us first show how it can be used to prove the statement (3.2).

Corollary 3.12. *Given a feasible solution x to ($\mathcal{P}'$), if π and π' are tight partitions, then so are $\pi \wedge \pi'$, each $m(\pi', \pi_i)$, and $\pi \vee \pi'$.*

Proof. Since π and π' are tight,

$$\begin{aligned} r(\pi)[r(\pi')-1] + [r(\pi)-1] &= r(\pi)\left[\sum_K x_K \mathtt{rc}_K^{\pi'}\right] + \left[\sum_K x_K \mathtt{rc}_K^{\pi}\right] = \sum_K x_K \left(r(\pi)\left[\mathtt{rc}_K^{\pi'}\right] + \left[\mathtt{rc}_K^{\pi}\right]\right) \\ &\geq \sum_K x_K \left(\left[\mathtt{rc}_K^{\pi\wedge\pi'}\right] + \sum_{i=1}^{r(\pi)} \left[\mathtt{rc}_K^{m(\pi',\pi_i)}\right]\right) = \sum_K x_K \left[\mathtt{rc}_K^{\pi\wedge\pi'}\right] + \sum_{i=1}^{r(\pi)} \sum_K x_K \left[\mathtt{rc}_K^{m(\pi',\pi_i)}\right] \\ &\geq [r(\pi\wedge\pi')-1] + \sum_{i=1}^{r(\pi)} [r(m(\pi',\pi_i))-1] = r(\pi)\,[r(\pi')-1] + [r(\pi)-1] \end{aligned}$$

where the first inequality follows from (3.7) and the second from ($\mathcal{P}$.1) (as x is feasible); the last equality is (3.6). Since the first and last terms are equal, all the inequalities are equalities, in particular our application of ($\mathcal{P}$.1) shows that $\pi \wedge \pi'$ and each $m(\pi', \pi_i)$ is tight. Iterating the latter fact, we see that $m(\cdots m(m(\pi', \pi_1), \pi_2), \cdots) = \pi \vee \pi'$ is also tight. □

To give the proof of Lemma 3.11 we need the following lemma that relates the rank of sets and the rank contribution of partitions. Recall $\rho(X) := \max(0, |X| - 1)$.

Lemma 3.13. *For a partition $\pi = \{\pi_1, \ldots, \pi_t\}$ of R, where $t = r(\pi)$, and for any $K \subseteq R$, we have*

$$\rho(K) = \mathtt{rc}^{\pi}_K + \sum_{i=1}^{t} \rho(K \cap \pi_i).$$

Proof. By definition, $K \cap \pi_i \neq \varnothing$ for exactly $1 + \mathtt{rc}^{\pi}_K$ values of i. Also, $\rho(K \cap \pi_i) = 0$ for all other i. Hence

$$\sum_{i=1}^{t} \rho(K \cap \pi_i) = \sum_{i: K \cap \pi_i \neq \varnothing} (|K \cap \pi_i| - 1) = \left(\sum_{i: K \cap \pi_i \neq \varnothing} |K \cap \pi_i| \right) - (\mathtt{rc}^{\pi}_K + 1). \tag{3.8}$$

Observe that $\sum_{i: K \cap \pi_i \neq \varnothing} |K \cap \pi_i| = |K| = \rho(K) + 1$; using this fact together with Equation (3.8) we obtain

$$\sum_{i=1}^{t} \rho(K \cap \pi_i) = \left(\sum_{i: K \cap \pi_i \neq \varnothing} |K \cap \pi_i| \right) - (\mathtt{rc}^{\pi}_K + 1) = \rho(K) - 1 + (\mathtt{rc}^{\pi}_K + 1).$$

Rearranging, the proof of Lemma 3.13 is complete. □

Proof of Lemma 3.11

First, we argue that $\pi \wedge \pi' = \overline{\pi}$ holds without loss of generality. In the general case, for each part p of $\pi \wedge \pi'$ with $|p| \geq 2$, contract p into one pseudo-vertex and define the new K to include the pseudo-vertex corresponding to p if and only if $K \cap p \neq \varnothing$. This contraction does not affect the value of any of the terms in Equations (3.7) and (3.6), so is without loss of generality. After contraction, for any part π_i of π and part π'_j of π', we have $|\pi_i \cap \pi'_j| \leq 1$, so indeed $\pi \wedge \pi' = \overline{\pi}$.

Proof of Equation (3.6). Fix i. Since $|\pi_i \cap \pi'_j| \leq 1$ for all j, the rank contribution $\mathtt{rc}^{\pi'}_{\pi_i}$ is equal to $|\pi_i| - 1$. Then using Equation (3.5) we know that $r(m(\pi', \pi_i)) = r(\pi') - |\pi_i| + 1$. Thus adding over all i, the right-hand side of Equation (3.6) is equal to

$$|R| - 1 + \sum_{i=1}^{r(\pi)} (r(\pi') - |\pi_i|) = |R| - 1 + r(\pi) r(\pi') - |R|$$

and this is precisely the left-hand side of Equation (3.6). □

Proof of Equation (3.7). Fix i. Since $|\pi_i \cap \pi'_j| \leq 1$ for all j, we have

$$\mathtt{rc}_K^{\pi'} - \mathtt{rc}_K^{m(\pi',\pi_i)} \geq \rho(\pi_i \cap K) \tag{3.9}$$

because, when we merge the parts of π' intersecting π_i, we make K span at least $\rho(\pi_i \cap K)$ fewer parts. Note that the inequality could be strict if both π_i and K intersect a part of π' without having a common vertex in that part.

Adding the right-hand side of Equation (3.9) over all i gives

$$\sum_{i=1}^{r(\pi)} (\mathtt{rc}_K^{\pi'} - \mathtt{rc}_K^{m(\pi',\pi_i)}) \geq \sum_{i=1}^{r(\pi)} \rho(\pi_i \cap K) = \rho(K) - \mathtt{rc}_K^{\pi}. \tag{3.10}$$

where the last equality follows from Lemma 3.13. To finish the proof we observe $\rho(K) = \mathtt{rc}_K^{\pi \wedge \pi'}$, since $\pi \wedge \pi' = \overline{\pi}$. □

3.3.1 Primal Partition Uncrossing Structure Theorem

We restate the theorem first.

Theorem 3.1. *Let x^* be a basic feasible solution of ($\mathcal{P}$), and let*

$$\Pi_R^* = \{\pi \in \Pi_R : \sum_{K \in \mathcal{K}} \mathtt{rc}_K^{\pi} x_K^* = r(\pi) - 1\}$$

be the set of tight *partitions for x^*. Let $\mathcal{C}$ be an inclusion-wise maximal chain in Π_R^* not containing $\underline{\pi}$. Then x^* is uniquely defined by*

$$\sum_{K \in \mathcal{K}} \mathtt{rc}_K^{\pi} x_K^* = r(\pi) - 1 \quad \forall \pi \in \mathcal{C}.$$

Here and later, for a vector x, we use $\mathtt{supp}(x) := \{i \mid x_i \neq 0\}$ to denote the *support* of x, i.e. the set of indices upon which the vector is nonzero.

Proof. Let $\mathtt{supp}(x^*)$ be the full components K with $x_K^* > 0$. Consider the constraint submatrix with rows corresponding to the tight partitions and columns corresponding to the full components in $\mathtt{supp}(x^*)$. Since x^* is a basic feasible solution, any full-rank subset of rows uniquely defines x^*. We now show that any maximal chain $\mathcal{C}$ in Π_R^* corresponds to such a subset.

Let $\texttt{row}(\pi) \in \mathbf{R}^{\texttt{supp}(x^*)}$ denote the row corresponding to partition π of this matrix, i.e., $\texttt{row}(\pi)_K = \texttt{rc}^{\pi}_K$, and given a collection $\mathcal{R}$ of partitions (rows), let $\text{span}(\mathcal{R})$ denote the linear span of the rows in $\mathcal{R}$. We now prove that for any tight partition $\pi \notin \mathcal{C}$, we have $\texttt{row}(\pi) \in \text{span}(\mathcal{C})$; this will complete the proof of the theorem.

For sake of contradiction, suppose $\texttt{row}(\pi) \notin \text{span}(\mathcal{C})$. Choose π to be the counterexample partition with smallest rank $r(\pi)$. Firstly, since $\mathcal{C}$ is maximal, π must cross some partition σ in $\mathcal{C}$. Choose σ to be the most refined partition in $\mathcal{C}$ which crosses π. Let the parts of σ be $(\sigma_1, \ldots, \sigma_t)$. The following claim uses the partition uncrossing inequalities to derive a linear combination between the rows corresponding to σ, π and the partitions formed by merging parts of σ with π.

Claim 3.14. *We have* $\texttt{row}(\sigma) + |r(\sigma)| \cdot \texttt{row}(\pi) = \texttt{row}(\pi \wedge \sigma) + \sum_{i=1}^{t} \texttt{row}(m(\pi, \sigma_i))$.

Proof. Since σ and π are both tight partitions, the proof of Corollary 3.12 shows that the partition inequality (3.7) holds with equality for all $K \in \texttt{supp}(x^*)$, π and σ, implying the claim. □

Let $\texttt{cp}_\pi(\sigma)$ be the parts of σ which intersect at least two parts of π; i.e., merging the parts of π that intersect σ_i, for any $\sigma_i \in \texttt{cp}_\pi(\sigma)$, decreases the rank of π. Formally,

$$\texttt{cp}_\pi(\sigma) := \{\sigma_i \in \sigma : \quad m(\pi, \sigma_i) \neq \pi\}$$

Note that one can modify Claim 3.14 by subtracting $(r(\sigma) - |\texttt{cp}_\pi(\sigma)|)\texttt{row}(\pi)$ from both sides to get

$$\texttt{row}(\sigma) + |\texttt{cp}_\pi(\sigma)| \cdot \texttt{row}(\pi) = \texttt{row}(\pi \wedge \sigma) + \sum_{\sigma_i \in \texttt{cp}_\pi(\sigma)} \texttt{row}(m(\pi, \sigma_i)) \tag{3.11}$$

Now if $\texttt{row}(\pi) \notin \text{span}(\mathcal{C})$, we must have either $\texttt{row}(\pi \wedge \sigma)$ is not in $\text{span}(\mathcal{C})$ or $\texttt{row}(m(\pi, \sigma_i))$ is not in $\text{span}(\mathcal{C})$ for some i. We show that either case leads to the needed contradiction, which will prove the theorem.

Case 1: $\texttt{row}(\pi \wedge \sigma) \notin \text{span}(\mathcal{C})$. Note there is $\sigma' \in \mathcal{C}$ which crosses $\pi \wedge \sigma$, since $\pi \wedge \sigma$ is not in the maximal chain $\mathcal{C}$. Since $\sigma', \sigma \in \mathcal{C}$ and by considering the refinement order, it is easy to see that σ' (strictly) refines σ and σ' crosses π. This contradicts our choice of σ as the most refined partition in $\mathcal{C}$ crossing π, since σ' was also a candidate.

Case 2: $\texttt{row}(m(\pi, \sigma_i)) \notin \text{span}(\mathcal{C})$. Note $m(\pi, \sigma_i)$ is also tight. Since $\sigma_i \in \texttt{cp}_\pi(\sigma)$, $m(\pi, \sigma_i)$ has smaller rank than π. This contradicts our choice of π. □

3.4 Equivalence of ($\mathcal{P}$) and ($\mathcal{P}'$)

In this section, we show that for Steiner tree instances, the unbounded and bounded partition LP relaxations, ($\mathcal{P}$) and ($\mathcal{P}'$), have the same optimum value. In fact, we prove something stronger. We need the following definition.

Definition 3.15. *The collection $\mathcal{K}$ of hyperedges is* down-closed *if whenever $S \in \mathcal{K}$ and $\varnothing \neq T \subset S$, then $T \in \mathcal{K}$. For down-closed $\mathcal{K}$, the cost function $C : \mathcal{K} \to \mathbf{R}_+$ is* non-decreasing *if $C_S \leq C_T$ whenever $S \subset T$.*

Theorem 3.16. *If the set of hyperedges is down-closed and the cost function is non-decreasing, then ($\mathcal{P}'$) and ($\mathcal{P}$) have the same optimal value.*

We remark that there exist some cost functions C for which the LPs ($\mathcal{P}'$) and ($\mathcal{P}$) do not have the same optimal value (see [138, Ex. 3.1]). Thus, the feasible region of the two LPs is not the same. However, note that the hypergraph and cost function induced by the Steiner tree problem are down-closed and non-decreasing, respectively.

Our proof relies on the following operation which we call *shrinking*. Note that our formulation includes hyperedges of size 1, which serve only as placeholders; when treating the Steiner tree problem, they have cost 0.

Definition 3.17. *Given an assignment $x : \mathcal{K} \to \mathbf{R}_+$ to the full components, suppose $x_K > 0$ for some K. The operation* $\texttt{Shrink}(x, K, K', \delta)$ *where $K' \subseteq K$, $|K'| = |K| - 1$ and $0 < \delta \leq x_K$ changes x to x' by decreasing $x'_K := x_K - \delta$ and increasing $x'_{K'} := x_{K'} + \delta$. We call the parameter δ the* shrinkage *of the shrink operation.*

Note that shrinking is defined only for down-closed hypergraphs. Also note that on performing a shrinking operation, the cost of the solution cannot increase, if the cost function is non-decreasing. Now we are ready to prove the theorem.

Proof of Theorem 3.16. Since ($\mathcal{P}'$) is just ($\mathcal{P}$) with an extra constraint, it is immediate that $\mathrm{OPT}(\mathcal{P}') \geq \mathrm{OPT}(\mathcal{P})$. We now prove the other direction.

Let x be an optimal solution to ($\mathcal{P}$) which minimizes the sum $\sum_{K \in \mathcal{K}} x_K |K|$. We claim that x satisfies the equation ($\mathcal{P}'.1$) implying x is feasible for ($\mathcal{P}'$) as well. This completes the proof. To do so, we first make the following claim.

Claim 3.18. *For every K with $x_K > 0$ and for every $r \in K$, there exists a tight partition (w.r.t. x) π such that the part of π containing r contains no other vertex of K.*

Proof. Let $K' = K \setminus \{r\}$. If the above is not true, then this implies that for every tight partition π, we have $\mathtt{rc}^\pi_K = \mathtt{rc}^\pi_{K'}$. We now claim that there is a $\delta > 0$ such that we can perform $\mathtt{Shrink}(x, K, K', \delta)$. This is a contradiction since the shrink operation strictly reduces $\sum_K |K| x_K$ and doesn't increase cost. In fact, the δ is the following

$$\delta := \min\{x^{(i)}_K, \min_{\pi:\mathtt{rc}^\pi_{K'} \neq \mathtt{rc}^\pi_K} \sum_K \mathtt{rc}^\pi_K x^{(i)}_K - r(\pi) + 1\}. \tag{3.12}$$

and is positive since for tight partitions we have $\mathtt{rc}^\pi_K = \mathtt{rc}^\pi_{K'}$. □

Let Π^*_R be the set of tight partitions and $\pi^* := \bigwedge\{\pi \mid \pi \in \Pi^*_R\}$ the meet of all tight partitions. By Corollary 3.12, π^* is tight. We need to show that $\pi^* = \overline{\pi}$ since $(\mathcal{P}')$ is just $(\mathcal{P})$ with the additional constraint $(\mathcal{P}'.1)$ that $\overline{\pi}$ is tight. The above claim shows that for every K with $x_K > 0$, we have $\mathtt{rc}^{\pi^*}_K = |K| - 1$ since for every vertex r in K there is a partition with a part containing r and no other vertex in K and the meet preserves this property. We get

$$r(\pi^*) - 1 = \sum_{K \in \mathcal{K}} x_K \mathtt{rc}^{\pi^*}_K = \sum_{K \in \mathcal{K}} x_K (|K| - 1) \geq r(\overline{\pi}) - 1$$

where the last inequality follows from the feasibility of x, constraint $(\mathcal{P}.1)$. This implies $\pi^* = \overline{\pi}$, so $\overline{\pi}$ is indeed tight. □

3.5 Computing Extreme Points of $(\mathcal{P})$

In investigating the hypergraphic formulations $(\mathcal{P})$, $(\mathcal{P}')$ and $(\mathcal{S}')$, it was helpful to encode them via the Maple `convex` package [75] to get a sense for their properties. As a natural part of this investigation, we decided to enumerate their extreme points. For definiteness, in this section we discuss enumerating the vertices of the unbounded partition formulation $(\mathcal{P})$. (Since $(\mathcal{P}')$ is a face of $(\mathcal{P})$, its vertices are also enumerated.)

Enumerating the extreme points of $(\mathcal{P})$ is non-trivial since vertex enumeration of a polytope can take exponential time, and the number of variables and constraints in $(\mathcal{P})$ is already exponential in $|R|$. Nonetheless, the partition uncrossing technology of Theorem 3.1 gives us a more efficient way to find extreme points. Namely, we enumerate through every possible chain $\mathcal{C}$ of partitions, we enumerate all possible sets $\mathtt{supp} \subset \mathcal{K}$ for which $|\,\mathtt{supp}\,| = |\mathcal{C}|$, and when the square equality subsystem of $(\mathcal{P}.1)$ corresponding to $|\,\mathtt{supp}\,|$ and $|\mathcal{C}|$ is nonsingular, if its unique solution satisfies all other partition constraints, we add it to a running list. Our implementation is given in Appendix B; for efficiency, we also kept only a list of solutions which were unique up to isomorphism (i.e., up to all possible relabelings of the terminal set R).

The polytope ($\mathcal{P}$) has exactly six nonisomorphic extreme points when $|R| = 4$, which we list below; the notation is shorthand where, for example, $\frac{1}{2}134$ means that $x_{\{1,3,4\}} = \frac{1}{2}$, and sets not listed have value 0.

$$[234, 134], [23, 34, 13], [23, 34, 12], [\frac{1}{2}123, \frac{1}{2}124, \frac{1}{2}134], [234, 12], [1234]$$

For $|R| = 5$, there are 27 vertices up to isomorphism:

$$[\frac{1}{2}123, \frac{1}{2}1345, \frac{1}{2}2345], [\frac{1}{3}1234, \frac{1}{3}1245, \frac{1}{3}1345, \frac{1}{3}2345], [1345, 2345], [134, 2345], [145, 245, 345],$$
$$[\frac{1}{3}145, \frac{1}{3}134, \frac{2}{3}2345, \frac{1}{3}124], [34, \frac{1}{2}123, \frac{1}{2}145, \frac{1}{2}245], [\frac{1}{2}1345, \frac{1}{2}14, \frac{1}{2}234, \frac{1}{2}245], [2345, 12],$$
$$[34, 23, 45, 12], [\frac{1}{2}1345, \frac{1}{2}12, \frac{1}{2}234, \frac{1}{2}245], [12345], [\frac{1}{3}123, \frac{1}{3}145, \frac{1}{3}134, \frac{2}{3}2345],$$
$$[245, 12, 345], [23, 13, 345], [245, 14, 345], [34, 245, 12], [134, 235], [14, \frac{1}{2}234, \frac{1}{2}245, \frac{1}{2}345],$$
$$[\frac{1}{2}1245, \frac{1}{2}12, \frac{1}{2}234, \frac{1}{2}345], [\frac{1}{2}123, \frac{1}{2}235, 45, \frac{1}{2}134], [34, 45, 12, 24], [23, 12, 345],$$
$$[\frac{1}{2}123, \frac{1}{2}2345, \frac{1}{2}34, \frac{1}{2}145], [12, \frac{1}{2}234, \frac{1}{2}245, \frac{1}{2}345], [34, 45, 14, 24], [\frac{1}{2}1345, \frac{1}{2}2345, \frac{1}{2}1245]$$

For $|R| = 6$ there are 496 nonisomorphic vertices, and for $|R| \geq 7$ our code would take too long to run. We note one particular pattern of extreme point here for reference; we omit the proof since it is lengthy but more or less standard.

Example 3.19. *For any integer n, where $R = \{1, 2, \ldots, n\}$, there is an extreme point solution x^* which assigns value $1/(n-2)$ to every set of size $n-1$ containing the terminal 1, and 0 to all other sets.*

For example, the unique non-integral vertex when $|R| = 4$ is of this form.

One insight gained from our enumerative information is about the *denominators* of extreme points; for a given extreme point solution x^*, its denominator is the least integer d for which dx^* is integral. We know (see Remark 3.9) that the hypergraphic formulation is integral if every hyperedge has size at most 2, and half-integral if every hyperedge has size at most 3. Hence, one might conjecture that if all hyperedges have size at most r, then the maximum denominator is $r-1$ (and Example 3.19 would show this is tight). However, the following example shows that this pattern does not hold.

Example 3.20. *The extreme point*

$$\frac{2}{5}[1234] + \frac{2}{5}[1256] + \frac{3}{5}[3456] + \frac{1}{5}[246] + \frac{1}{5}[135].$$

shows that it is possible for the denominator to be greater than the maximum hyperedge size.

3.6 Dual Partition Uncrossing & Gainless Trees

Using dual partition uncrossing, we will prove the following theorems.

Theorem 3.21. *The quantity $t^{\mathcal{K}}_{+}$ is equal to the optimum value of $(\mathcal{P})$.*

Theorem 3.22. *The quantity $t^{\mathcal{K}}$ is equal to the optimum value of $(\mathcal{P}')$.*

Of note is the fact that $t^{\mathcal{K}}$ was used in [127] as an upper bound in one place and as a lower bound in another; this resembles an LP optimal value already.

We develop the proof of these theorems in several steps. We note that all the results hold for arbitrary cost functions C_K, even those not derived from instances of the Steiner tree problem. Call $y \in \mathbf{R}^{\Pi_R}$ *chain-supported* if $\texttt{supp}(y)$ is a chain. First, we show our dual uncrossing theorem for $(\mathcal{P}_D)$, which shows that there is always a dual optimum whose support is chain-supported. We then precisely establish a link between terminal spanning trees and chain-supported y's. Next, we give a dual interpretation of gain. We complete the proofs of Theorems 3.21 and 3.22 in Section 3.6.4 by arguing that feasible chain-supported solutions to $(\mathcal{P}_D)$ (resp. the dual of $(\mathcal{P}')$) correspond to gainless terminal spanning trees with nonnegative costs (resp. arbitrary costs), which completes the proof.

Theorem 3.3 (Dual partition uncrossing)**.** *$(\mathcal{P}_D)$ always has an optimum y^* such that y^* is chain-supported.*

Proof. Suppose y is any feasible solution to $(\mathcal{P}_D)$ such that two crossing partitions π, σ have $y_\pi, y_\sigma \neq 0$. Note that $\overline{\pi}$ does not cross any other partition, so we may assume that $y_\pi, y_\sigma > 0$. Let e^π denote the unit basis vector for partition π. Define

$$y' := y - t \cdot (r(\pi)e^\sigma + e^\pi) + t\left(e^{\pi\wedge\sigma} + \sum_{i=1}^{r(\pi)} e^{m(\sigma,\pi_i)}\right)$$

where $t \geq 0$ is a parameter. We would like to increase t until one of the terms y'_π or y'_σ becomes zero, i.e., we claim that putting

$$t = \min\left\{\frac{y_\sigma}{r(\pi)}, y_\pi\right\}$$

produces a feasible y' with the same objective value as y. From Equation (3.7) we deduce that this y' is feasible for $(\mathcal{P}_D)$; from Equation (3.6) we deduce that y' has the same objective value as y. By *uncrossing π and σ* we mean the map $y \mapsto y'$.

With the uncrossing operation formally defined, we can complete the proof. Note that ($\mathcal{P}$) is feasible and bounded, whence ($\mathcal{P}_D$) is too. For a feasible solution y of ($\mathcal{P}_D$) define

$$\mathtt{v}_i(y) = \sum_{\pi : r(\pi)=i} y_\pi.$$

Let y^* be a optimal solution ($\mathcal{P}_D$) that is maximal with respect to lexicographic ordering on the vector $(\mathtt{v}_n(y^*), \mathtt{v}_{n-1}(y^*), \dots)$; to see that such a y^* exists, note that it can be computed by solving a series of linear programs. Now if the support of y^* were not a chain, then it contains two crossed partitions π and σ. By uncrossing them in y^*, we increase $y^*_{\pi\wedge\sigma}$; furthermore since π and σ cross, it is not hard to see that

$$r(\pi \wedge \sigma) > \max\{r(\pi), r(\sigma)\}.$$

Hence by uncrossing π and σ in y^*, the lexicographic value of $(\mathtt{v}_n(y^*), \mathtt{v}_{n-1}(y^*), \dots)$ strictly increases. This contradicts the maximality of y^*. Hence no such π, σ exist, and y^* is a chain-supported optimum to ($\mathcal{P}_D$). □

It is an interesting open problem to determine whether one can take an arbitrary feasible y to a chain-supported one in a polynomial number of uncrossing operations (versus the implicit approach given above which requires a poly-time black box for LP solving). This is known to be possible for set uncrossing, e.g. see [120].

At this point it is prudent to look at the difference between ($\mathcal{P}'$) and ($\mathcal{P}$): the equality constraint in ($\mathcal{P}'$). This equality constraint just says that ($\mathcal{P}$.1) holds with equality for $\overline{\pi}$. Hence the dual of ($\mathcal{P}'$) is the same as the dual of ($\mathcal{P}$) except that the variable $y_{\overline{\pi}}$ is unconstrained rather than negative.

Observe that the analogue of Theorem 3.3 for the dual of ($\mathcal{P}'$) holds using the same method; this is needed to prove Theorem 3.22.

3.6.1 A Primal-Dual Interpretation of Kruskal's MST Algorithm

Kruskal's algorithm, which we will denote by `MST`, can be viewed as a continuous process over *time*: we start with an empty tree at time 0 and add edges as time increases. The algorithm terminates at time τ^* with a spanning tree of the input graph G. In this section we show that Kruskal's method can be interpreted as a primal-dual algorithm (see also [96]). At any time $0 \le \tau \le \tau^*$ we keep a pair (x^τ, y^τ), where x^τ is a (not necessarily feasible) 0-1 primal solution for ($\mathcal{M}$) and y^τ is a feasible dual solution for ($\mathcal{M}_D$).

The initial primal and dual values x^0 and y^0 are the all-zero vectors. Let $G^\tau = (V, E^\tau)$ denote the forest corresponding to x^τ, i.e., $E^\tau = \{e \in E \mid x^\tau_e = 1\}$. Let $\pi(\tau)$ denote the

partition induced by the connected components of G^τ. At time τ, the algorithm increases $y_{\pi(\tau)}$ until a constraint of type ($\mathcal{M}_D$.1) becomes tight for some edge $e \in E_{\pi(\tau)}$. (If more than one such constraint becomes tight simultaneously, we pick any such e arbitrarily.) Let $\tau' \geq \tau$ be the time at which this happens. The dual update is

$$y^{\tau'}_{\pi(\tau)} := \tau' - \tau.$$

We then include e in our solution, i.e., the primal update is $x^{\tau'}_e := 1$. We terminate at time τ^* such that G^{τ^*} is a spanning tree. Chopra [46] showed that the final primal and dual solutions have the same objective value (and are hence optimal), and we give a proof of this fact for completeness.

In what follows, let G^* be shorthand for G^{τ^*} and similarly for x^*, etc.

Theorem 3.23. *Algorithm* `MST` *finishes with a pair* (x^*, y^*) *of primal and dual feasible solutions to* ($\mathcal{M}$) *and* ($\mathcal{M}_D$)*, respectively, such that*

$$\sum_{e \in E} c_e x^*_e = \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y^*_\pi.$$

Proof. Checking feasibility is straightforward. For each edge $e \in E^*$, the constraint ($\mathcal{M}_D$.1) holds with equality. Hence, rearranging, we can express the cost of the final tree as follows:

$$\sum_{e \in E} c_e x^*_e = \sum_{e \in E^*} \sum_{\pi : e \in E_\pi} y^*_\pi = \sum_{\pi \in \Pi} |E^* \cap E_\pi| \cdot y^*_\pi. \tag{3.13}$$

Note that for each τ, the final tree G^* has exactly $|V| - r(\pi(\tau))$ edges not in $E_{\pi(\tau)}$; hence for all π with $y^*_\pi > 0$, we have $|E^* \cap E_\pi| = |V| - 1 - (|V| - r(\pi)) = r(\pi) - 1$. This fact, combined with Equation (3.13), completes the proof. □

Observe that the above primal-dual algorithm is indeed Kruskal's algorithm: if the algorithm adds an edge e at time τ, then e has cost exactly equal to τ, and e is a minimum-cost edge connecting two connected components of G^τ.

3.6.2 MST Duals

Building on the previous section, we now show that terminal spanning trees are essentially the same as chain-supported y's. For any $y \in \mathbf{R}^{\Pi_R}$, define $c(y) := \sum_\pi y_\pi(r(\pi) - 1)$. Note in ($\mathcal{M}_D$) the variable $y_{\underline{\pi}}$ is vacuous, i.e. it can have any value without affecting the feasibility or optimality of the solution. So from now on we assume $y_{\underline{\pi}} = 0$ for convenience.

In Algorithm 3.1 we give an alternate procedure `TreeDual` for constructing a dual y to a spanning tree T. It is introduced in make explicit the sort of dual we mean when T contains negative-cost edges. The output of `TreeDual` is otherwise the same as that of `MST`; the type of arguments in the previous section easily give the following.

Algorithm 3.1 The algorithm $\texttt{TreeDual}(T)$.

1: Let $W := \{c(e) \mid e \in T\}$ be the set of distinct edge costs on T
2: Sort W into the increasing sequence $W = (w_1, \ldots, w_t)$
3: For $i = 1$ to t let $\pi^{[i]}$ be the partition of R induced by the graph $(R, \{e \in T \mid c(e) < w_i\})$
4: Return $w_1 e^{\pi^{[1]}} + \sum_{i=2}^{t} (w_i - w_{i-1}) e^{\pi^{[i]}}$ //Note $\pi^{[1]} = \overline{\pi}$

Claim 3.24. *For a terminal spanning tree T, where $y := \texttt{TreeDual}(T)$, we have $c(y) = c(T)$ and y is chain-supported. Moreover $y_{\overline{\pi}} \geq 0$ iff all edge costs are nonnegative; and $y_\pi \geq 0$ for all other partitions.*

We also need to show that `TreeDual` is surjective. The exact technical requirement is encapsulated in the following lemma.

Lemma 3.25. *Suppose y is chain-supported with $y_\pi \geq 0$ for all $\pi \neq \overline{\pi}$. Then there exists a terminal spanning tree T for which $y = \texttt{TreeDual}(T)$. If $y_{\overline{\pi}} \geq 0$ then all edge costs of T are nonnegative.*

Proof. Denote the chain $\texttt{supp}(y) \cup \{\overline{\pi}\}$ by $\pi^{[1]}, \pi^{[2]}, \ldots, \pi^{[t]}$ where $\pi^{[i]}$ refines $\pi^{[i+1]}$ for $1 \leq i < t$. For convenience let $\pi^{[t+1]}$ denote $\underline{\pi}$, the coarsest partition. Denote the $\pi^{[i]}$-coordinate of y by $y_{[i]}$.

We now define a set $E^{[i]}$ of edges for each $1 \leq i \leq t$. We claim such sets can be chosen so that $(R, \cup_{j=1}^{i} E^{[j]})$ induces $\pi^{[i]}$ for each $0 \leq i \leq t$. The base case $i = 0$ clearly holds. Then in the induction step, since $\pi^{[i]}$ refines $\pi^{[i+1]}$, such a set $E^{[i]}$ can be chosen — informally, $E^{[i]}$ is a spanning forest of the parts of $\pi^{[i+1]}$ when $\pi^{[i]}$ is contracted.

Now let $T = \cup_{j=1}^{i} E^{[j]}$, where we assign cost $\sum_{j=1}^{i} y_{[j]}$ to each edge in $E^{[i]}$. When $\texttt{TreeDual}(T)$ runs, $\pi_*^{[i]} = \pi^{[i]}$ for all i, $w_1 = y_{\overline{\pi}}$, and $w_i - w_{i-1} = y_{[i]}$ for all i. Hence the output y_* is equal to y. □

3.6.3 Dual Interpretation of Gain

Lemma 3.26. *Let T be a terminal spanning tree with costs c. Let $y = \texttt{TreeDual}(T)$. Then*

$$c(T) - \texttt{mst}(T/K, c) = \sum_{\pi} y_\pi \texttt{rc}_K^\pi.$$

Proof. Let us adopt the notation from the proof of Theorem 3.23, and assume that $\texttt{MST}(T)$ finishes at time τ^*. Consider how the rank contribution of K changes with respect to

$\pi(\tau)$ over time. Clearly, $\mathtt{rc}_K^{\pi(0)} = |K| - 1$ and $\mathtt{rc}_K^{\pi(\tau^*)} = 0$. Whenever an edge is added to E^τ in MST, the value $\mathtt{rc}_K^{\pi(\tau)}$ either stays the same or drops by 1; hence there are edges $e_1, \ldots, e_{|K|-1} \in T$ such that, for $1 \le i \le |K| - 1$, $\mathtt{rc}_K^{\pi(\tau)}$ drops from $|K| - i$ to $|K| - i - 1$ when edge e_i is added. Let $\tau(i)$ denote the time at which edge e_i is added, then by the definition of the e_i,

$$\int_0^{\tau^*} \mathtt{rc}_K^{\pi(\tau)} d\tau = \sum_{i=1}^{|K|-1} \tau(i). \tag{3.14}$$

Notice that due to the definition of MST, the following two facts hold: first, $\tau(i) = c_{e_i}$ for each i; second, the left hand side of Equation (3.14) is $\sum_\pi \mathtt{rc}_K^\pi y_\pi$. Hence we obtain

$$\sum_\pi \mathtt{rc}_K^\pi y_\pi = \sum_{i=1}^{|K|-1} c_{e_i} \tag{3.15}$$

Let the partition maintained by MST on input G at time τ be denoted by $\pi_G(\tau)$. An easy inductive argument shows that for all τ, we obtain $\pi_{T/K}(\tau)$ from $\pi_T(\tau)$ by first merging all parts that meet K, and by subsequently identifying the vertices of K. It follows that $T \backslash \{e_1, \ldots, e_{|K|-1}\}$ is a minimum spanning tree of T/K. With Equation (3.15) this yields

$$\sum_\pi \mathtt{rc}_K^\pi y_\pi = c(T) - \mathtt{mst}(T/K)$$

as needed. □

Then from the definition of gain, we immediately get the following corollary.

Corollary 3.27 (Dual interpretation of gain)**.** *Let* $y = \mathtt{TreeDual}(T)$*. Then*

$$\mathtt{gain}_T(K) = \sum_\pi y_\pi \mathtt{rc}_K^\pi - C_K,$$

in particular K *has positive gain iff* ($\mathcal{P}_D$.1) *is violated for this* K*.*

Another interesting corollary (which will be used in Section 4.8.3) is the following; it is analogous to an earlier result in [34] for ($\mathcal{B}$) on quasi-bipartite instances. Here $G[R]$ is the subgraph of G induced by the terminals, so $\mathtt{MST}(G[R])$ is the same as Moore's spanning tree heuristic for the Steiner tree problem.

Corollary 3.28. *Let* $(T, y) = \mathtt{MST}(G[R])$ *and suppose that every* K *has nonpositive gain in* T*. Then* $\mathrm{OPT}(\mathcal{P}) = c(T)$ *and* T *is an optimal Steiner tree.*

Proof. By Corollary 3.27, y is feasible for $(\mathcal{P}_D)$, so $\text{OPT}(\mathcal{P}_D) \geq c(y)$. But T is a Steiner tree with this cost, hence $\text{OPT}(\mathcal{P}) \leq c(T)$. Now apply LP duality and the fact that $c(y) = c(T)$ to see $\text{OPT}(\mathcal{P}) = c(T)$. Finally, T is an optimal Steiner tree since every Steiner tree has cost at least $\text{OPT}(\mathcal{P}) = c(T)$. □

Another consequence of the dual interpretation of gain will appear in Lemma 5.5.

3.6.4 Gainless Tree Equivalence

We give the proof of Theorem 3.21; the proof of Theorem 3.22 is almost the same but marginally more complicated due to negative edges.

Proof. We know that every terminal spanning tree with nonnegative costs corresponds to a chain-supported $y \geq 0$, and vice-versa (not bijectively, but this is ok). By Corollary 3.27, a terminal spanning tree is gainless if $(\mathcal{P}_D.1)$ holds for its dual. Note that the objective function in $(\mathcal{P})$ is just $c(y)$. Therefore, any gainless terminal spanning tree T with nonnegative costs can be converted into a feasible $y = \texttt{TreeDual}(T)$ for $(\mathcal{P})$ with objective value equal to $c(y) = c(T)$, so $\text{OPT}(\mathcal{P}) \geq t^+_{\mathcal{K}}$.

Conversely, by dual partition uncrossing (Theorem 3.3) there is an optimal dual solution y to $(\mathcal{P})$ which is chain-supported. It therefore corresponds to a spanning tree T, which is gainless by Corollary 3.27. Hence $t^+_{\mathcal{K}} \geq \text{OPT}(\mathcal{P})$. □

Chapter 4

Collected Proofs for Steiner Tree LPs

In this chapter we give proofs of results mentioned in Chapter 2.

4.1 LPs ($\mathcal{S}'$) and ($\mathcal{P}'$) Define the Same Polyhedron

Theorem 4.1. *The set of all feasible solutions to ($\mathcal{P}'$) equals the set of all feasible solutions to ($\mathcal{S}'$).*

Note that since ($\mathcal{S}'$) and ($\mathcal{P}'$) both have the objective function $\min \sum_K x_K C_K$, this theorem implies $\mathrm{OPT}(\mathcal{P}') = \mathrm{OPT}(\mathcal{S}')$.

Proof. Let x be any feasible solution to the LP ($\mathcal{S}'$). Note that the constraint ($\mathcal{P}'.1$) of ($\mathcal{P}'$) is the same as the constraint ($\mathcal{S}'.1$) of ($\mathcal{S}'$). We now show that x satisfies ($\mathcal{P}.1$). Fix a partition $\pi = \{\pi_1, \ldots, \pi_t\}$, so $t = r(\pi)$. For each $1 \le i \le t$, subtract the constraint ($\mathcal{S}'.2$) with $S = \pi_i$, from constraint ($\mathcal{S}'.1$) to obtain

$$\sum_{K \in \mathcal{K}} x_K \Big(\rho(K) - \sum_{i=1}^{t} \rho(K \cap \pi_i) \Big) \ge \rho(R) - \sum_{i=1}^{t} \rho(\pi_i). \tag{4.1}$$

From Lemma 3.13, $\rho(K) - \sum_{i=1}^{t} \rho(K \cap \pi_i) = \mathtt{rc}_K^{\pi}$. We also have $\rho(R) - \sum_{i=1}^{t} \rho(\pi_i) = |R| - 1 - (|R| - r(\pi)) = r(\pi) - 1$. Thus x is a feasible solution to the LP ($\mathcal{P}'$).

Now, let x be a feasible solution to ($\mathcal{P}'$) and it suffices to show that it satisfies constraint ($\mathcal{S}'.2$). Fix a set $S \subset R$. Note when $S = \varnothing$ that constraint ($\mathcal{S}'.2$) is vacuously true so we may assume $S \neq \varnothing$. Let $R \backslash S = \{r_1, \ldots, r_u\}$. Consider the partition $\pi = \{\{r_1\}, \ldots, \{r_u\}, S\}$. Subtract ($\mathcal{P}.1$) for this π from ($\mathcal{P}'.1$), to obtain

$$\sum_{K \in \mathcal{K}} x_K(\rho(K) - \mathtt{rc}^{\pi}_K) \leq \rho(R) - r(\pi) + 1. \tag{4.2}$$

Using Lemma 3.13 and the fact that $\rho(K \cap \{r_j\}) = 0$ (the set is either empty or a singleton), we get $\rho(K) - \mathtt{rc}^{\pi}_K = \rho(K \cap S)$. Finally, as $\rho(R) - r(\pi) + 1 = |R| - 1 - (|R \setminus S| + 1) + 1 = \rho(S)$, the inequality (4.2) is the same as ($\mathcal{S}'$.2). Thus x is a feasible solution to ($\mathcal{S}'$), proving the theorem. □

4.2 Proof that OPT($\mathcal{D}'$) = OPT($\mathcal{D}$)

Polzin & Vahdati Daneshmand claim at the end of Section 2 in their 2003 paper [170] that OPT($\mathcal{D}'$) = OPT($\mathcal{D}$) and sketch the details.[1] We will make these details explicit below. As mentioned in [170], these claims are analogous to Lemmas 8 and 9 from [168], which they attribute as similar to earlier uncrossing methods of Goemans & Myung [95]. We recall the *shrinking* operation defined in Section 3.3.1, which will be needed. We start with a simple lemma about submodularity for directed hyperedges. (We remark that it would fail in a model where directed hyperedges can have an arbitrary number of heads and tails.)

Lemma 4.2. *For any $A, B \subset R$, and a vector $Z \geq 0$ indexed by directed hyperedges, we have $Z(\Delta^{\text{out}}(A)) + Z(\Delta^{\text{out}}(B)) \geq Z(\Delta^{\text{out}}(A \cup B)) + Z(\Delta^{\text{out}}(A \cap B))$.*

Proof. Let $\chi[\mathcal{E}]$ be the indicator function for event $\mathcal{E}$, i.e. 1 if the event is true, and 0 otherwise. It clearly suffices to show for each K^i that

$$\chi[K^i \in \Delta^{\text{out}}(A)] + \chi[K^i \in \Delta^{\text{out}}(B)] \geq \chi[K^i \in \Delta^{\text{out}}(A \cup B)] + \chi[K^i \in \Delta^{\text{out}}(A \cap B)].$$

If this inequality is violated, either the left-hand side is 0 or the right-hand side is 2. Recall that $K^i \in \Delta^{\text{out}}(S)$ means that $K \cap S \neq \varnothing$ and $i \notin S$. We will show in more detail that the inequality is never violated by considering cases.

Case 1: The right-hand side equals 2. Then $i \notin A \cup B$ and $K \cap A \cap B \neq \varnothing$. This clearly implies that the left-hand side equals 2.

Case 2: The left-hand side equals 0. We break into three subcases (we skip the fourth by symmetry).

a: $i \in A, i \in B$. Then the right-hand side is also 0, since $i \in A \cup B$ and $i \in A \cap B$.

[1] We further note that OPT($\mathcal{D}'$) = OPT($\mathcal{D}$) is reminiscent of the *parsimonious property* [93] except the setting here of directed hypergraphs differs from undirected graph setting of [93].

b: $i \in A, K \cap B = \varnothing$. Then the right-hand side is also 0, since $i \in A \cup B$ and $K \cap (A \cap B) = \varnothing$.

c: $K \cap A = K \cap B = \varnothing$. Then the right-hand side is also 0, since $K \cap (A \cup B) = K \cap (A \cap B) = \varnothing$. □

We need one more lemma before the main result.

Lemma 4.3. *There is an optimal solution Z to ($\mathcal{D}$) such that $Z(\Delta^{\text{out}}(\{t\})) \leq Z(\Delta^{\text{out}}(T))$ for all $T \subset R \backslash \{r\}$ with $t \in T$.*

Proof. Of all optimal solutions, take one for which $\sum_{K^i} |K| Z_{K^i}$ is minimal. Note that shrinking strictly decreases this sum, so any Z' obtained from shrinking Z is infeasible.

Suppose for the sake of contradiction that there is a t, T with $r \notin T, t \in T$ and $Z(\Delta^{\text{out}}(T)) < Z(\Delta^{\text{out}}(t))$. Take T minimal subject to this condition. Since $Z(\Delta^{\text{out}}(T)) < Z(\Delta^{\text{out}}(t))$, there must be a $K^i \in \Delta^{\text{out}}(t) \backslash \Delta^{\text{out}}(T)$; this implies $i \in T$.

Now try shrinking the directed hyperedge K^i to $(K \backslash \{t\})^i$ (i.e., decrease the Z-value of the former by a small ϵ and increase the value of the latter by ϵ). By hypothesis, the shrunk solution is infeasible for every $\epsilon > 0$, which implies there is a tight inequality ($\mathcal{D}$.1) for some set U with $t \notin U$, $r \in U$, $i \in U$. Let $\overline{U}$ denote $R \backslash U$, then the fact that U is tight can be restated as $Z(\Delta^{\text{out}}(\overline{U})) = 1$.

We know that $Z(\Delta^{\text{out}}(\overline{U} \cup T)) \geq 1$ since Z is feasible for ($\mathcal{D}$), and using Lemma 4.2,

$$\begin{aligned} Z(\Delta^{\text{out}}(\overline{U} \cap T)) &\leq -Z(\Delta^{\text{out}}(\overline{U} \cup T)) + Z(\Delta^{\text{out}}(\overline{U})) + Z(\Delta^{\text{out}}(T)) \\ &< -1 + 1 + Z(\Delta^{\text{out}}(t)) = Z(\Delta^{\text{out}}(t)). \end{aligned}$$

Now $\overline{U} \cap T$ is strictly smaller than T since it does not contain i. Also, $\overline{U} \cap T$ contains t. This contradicts our choice of T as the smallest set with $r \notin T, t \in T$ and $Z(\Delta^{\text{out}}(T)) < Z(\Delta^{\text{out}}(t))$. Therefore no such T exists and we are done. □

This leads to the main result.

Theorem 4.4. $\text{OPT}(\mathcal{D}') = \text{OPT}(\mathcal{D})$.

Proof. Clearly $\text{OPT}(\mathcal{D}') \geq \text{OPT}(\mathcal{D})$. Let Z be an optimal solution to ($\mathcal{D}$) with the properties guaranteed by Lemma 4.3; by the proof of that lemma we may assume that any Z' obtained from shrinking Z is infeasible. We may clearly further assume that $x(\Delta^{\text{out}}(r)) = 0$ since any hyperedge $K^i \in \Delta^{\text{out}}(r)$ can be shrunk to $(K \backslash \{r\})^i$ while contributing just as much to all covering constraints ($\mathcal{D}$.1).

Now suppose for the sake of contradiction that some $t \in R\backslash\{r\}$ has $Z(\Delta^{\text{out}}(t)) > 1$. Then by Lemma 4.3, every set $T \subset R\backslash\{r\}$ with $t \in T$ has $Z(\Delta^{\text{out}}(T)) > 1$. Take any $K^i \in \Delta^{\text{out}}(t)$. We claim we can shrink K^i it to $(K\backslash t)^i$ without losing feasibility, which will complete the proof by contradiction. The only obstacle to this shrinking would be a tight set S with $K^i \in \Delta^{\text{out}}(S), (K\backslash t)^i \notin \Delta^{\text{out}}(S)$. But this would imply $K \cap S = \{t\}$ and thus $Z(\Delta^{\text{out}}(S)) > 1$, contradicting the fact that S is tight. □

4.3 Proof that OPT($\mathcal{P}$) = OPT($\mathcal{D}$)

Theorem 4.5. *For any graph,* OPT($\mathcal{P}$) = OPT($\mathcal{D}$).

Proof. Two subsets U and W of R are called *crossing* if all four sets $U \setminus W$, $W \setminus U$, $U \cap W$, and $R \setminus (U \cup W)$ are non-empty. A set-function $p : 2^R \to \mathbb{Z}$ is a *crossing supermodular* function if

$$p(U) + p(W) \leq p(U \cap W) + p(U \cup W)$$

for all crossing sets U and W. An orientation of the edges of a hypergraph is said to *cover* a set-function p on V if $|\Delta^{\text{in}}(U) \geq p(U)|$ for all subsets U of V. We use the following theorem of [73, 72] written in our terms.

Theorem 4.6 (Theorem 2.23 of [72])**.** *Given a hypergraph $H = (R, \mathcal{L})$ and a crossing supermodular function p, there is an orientation of the hyperedges of $\mathcal{L}$ covering p if and only if the following holds: for every partition π of R,*
(a) $\sum_{K\in\mathcal{L}} \min\{1, \mathtt{rc}^\pi_K\} \geq \sum_{\pi_i\in\pi} p(\pi_i)$, *and,* (b) $\sum_{K\in\mathcal{L}} \mathtt{rc}^\pi_K \geq \sum_{\pi_i\in\pi} p(R \setminus \pi_i)$.

Now we show that given any feasible solution x to ($\mathcal{P}$), we can obtain a feasible solution to ($\mathcal{D}$) of the same value. Let M be the smallest integer such that the vector Mx is integral. Let $\mathcal{K}$ be a multi-set of hyperedges which contains Mx_K copies of each K. Define the function p as $p(U) = M$ if U is valid and 0 otherwise. It is easy check that this is a crossing supermodular function. We claim that $H = (R, \mathcal{L})$ satisfies conditions (a) and (b) of the above theorem.

Note $\sum_{\pi_i\in\pi} p(R \setminus \pi_i) = M(r(\pi) - 1)$ since $R\backslash\pi_i$ is valid for all parts π_i of π except the part containing the root. Thus condition (b), upon scaling by $\frac{1}{M}$, is a restatement of constraint ($\mathcal{P}$.1), which holds since x is feasible for ($\mathcal{P}$).

For this choice of p, condition (a) follows from (b) in the following sense. Fix a partition π, and let π_1 be the part of π containing r. If $\pi_1 = R$ then (a) is vacuously true, so assume $\pi_1 \neq R$. Let σ be the rank-2 partition $\{\pi_1, R \setminus \pi_1\}$. Then it is easy to check that $\min\{1, \mathtt{rc}^\pi_K\} \geq \mathtt{rc}^\sigma_K$ for all K, and consequently $\sum_{K\in\mathcal{L}} \min\{1, \mathtt{rc}^\pi_K\} \geq \sum_{K\in\mathcal{L}} \mathtt{rc}^\sigma_K$ and $\sum_{\pi_i\in\sigma} p(R \setminus \pi_i) = M = \sum_{\pi_i\in\pi} p(\pi_i)$. Thus, (a) for π follows from (b) for σ.

Therefore there is an orientation of the hyperedges of H such that $\Delta^{\text{in}}(U) \geq p(U)$ for all $U \subseteq R$. Pick a hyperedge K and look at the Mx_K copies of this edge in H. For $i \in K$, let n_{K^i} denote the number of these Mx_K copies directed towards i. So, $\sum_{i \in K} n_{K^i} = Mx_K$.

Define $Z_{K^i} := \frac{n_{K^i}}{M}$. Note that $\sum_{K \in \mathcal{L}, i \in K} C_{K^i} Z_{K^i} = \sum_{K \in \mathcal{L}} C_K \sum_{i \in K} Z_{K^i} = \sum_{K \in \mathcal{L}} C_K x_K$, thus the value of this solution is the same as the value of x in ($\mathcal{P}$). We now claim that Z is feasible for ($\mathcal{D}$). To see this fix a valid subset U. The proof follows from the fact that the orientation covers p, that is, $\sum_{K^i \in \Delta^{\text{in}}(U)} n_{K^i} \geq p(U) = M$. □

4.4 Strengthening ($\mathcal{B}$) to Get ($\mathcal{S}'$)

This proof makes precise a polyhedral relation between the bidirected cut formulation and the hypergraph-subtour formulation. Specifically, we show that the hypergraph-subtour formulation is equivalent to a strengthened version of the bidirected cut relaxation. It is a polyhedral equivalence, if one projects both polyhedra on to the natural common variable space.

First we introduce an alternate formulation of the bidirected cut relaxation, due to Goemans and Myung [95], which we denote ($\mathcal{GM}$) and show in Figure 4.1.[2] It has variables for nodes and edges, rather than variables for arcs like the more standard formulation ($\mathcal{B}$). To get some intuition for the LP, note that for any fixed 0-1 vector y, it is basically Fulkerson's [81] subtour formulation for the MST of $\texttt{supp}(y)$.

Goemans & Myung showed that OPT($\mathcal{GM}$) = OPT($\mathcal{B}$) and more strongly that both polyhedra are the same under the following projection into the edge space: map $w \in \mathbf{R}^A$ in the arc space to $x \in \mathbf{R}^E$ via $x_{\{u,v\}} = w_{(u,v)} + w_{(v,u)}$; map the node+edge space to the edge space by ignoring (projecting away) the node variables[3]. We use $\gamma(S)$ to denote all edges with both endpoints in S.

In this section we will show that ($\mathcal{S}'$) is polyhedrally equivalent to a strengthened version of ($\mathcal{GM}$) in *preprocessed* graphs (as defined in Chapter 2).

To avoid confusion, in this section we now take K only to mean the tree representing a full component, with $\mathcal{K}$ still the collection of all full components, but now $R(K)$ will be explicitly used to denote the set of terminals belonging to a full component K. We also use $N(K)$ to denote the set of Steiner vertices in K, $E(K)$ the set of edges in K, and $V(K)$ the set of vertices in K.

[2] This LP was called P'_{xy} in [95]. Actually, their formulation has variables only for Steiner nodes, but it is easy to verify that adding dummy variables $y_r = 1$ for $r \in R$ gives the formulation stated above. Also, their formulation doesn't force $y \geq 0$, but it is not hard to see this is implied by their constraints.

[3] Also, one must add $x_{uv} + x_{vu} \leq 1$ to ($\mathcal{B}$) for all $\{u, v\} \in E$.

$$\min \Big\{ \sum_{e \in E} x_e c_e : \quad x \in \mathbf{R}^E, y \in \mathbf{R}^V \tag{$\mathcal{GM}$}$$

$$\sum_{e \in E} x_e = \sum_{v \in V} y_v - 1 \tag{4.3}$$

$$\sum_{e \in \gamma(S)} x_e \leq \sum_{v \in S \setminus \{k\}} y_v, \quad \forall S \subset V, \forall k \in S \tag{4.4}$$

$$x_e \geq 0, \quad \forall e \in E \tag{4.5}$$

$$1 \geq y_v \geq 0, \quad \forall v \in V \tag{4.6}$$

$$y_r = 1, \quad \forall r \in R \Big\} \tag{4.7}$$

Figure 4.1: An alternate formulation for the bidirected cut relaxation.

Since each full component is a tree in preprocessed graphs, and since every tree has one more node than edges, we have the following equation, which will be needed later:

$$|R(K)| + |N(K)| - |E(K)| = 1. \tag{4.8}$$

Define the map $\ell : z \mapsto (x, y)$ by $x_e = z_K$ for all $e \in E(K)$ and all $K \in \mathcal{K}$, and $y_n = z_K$ for all $n \in N(K)$ and all $K \in \mathcal{K}$, and $y_r = 1$ for all $r \in R$. Informally, $\ell(z)$ represents a bidirected cut solution where we have "glued together" all elements of each full component in the sense that we are forcing their variables to be equal.

Theorem 4.7. *Suppose $\ell(z) = (x, y)$ with $z \in [0, 1]^{\mathcal{K}}$. Then z is feasible for ($\mathcal{S}'$) if and only if (x, y) is feasible for ($\mathcal{GM}$).*

The objective value of z in ($\mathcal{S}'$) equals the objective value of $\ell(z)$ in ($\mathcal{GM}$), so this polyhedral equivalence also shows that $\text{OPT}(\mathcal{S}') \geq \text{OPT}(\mathcal{GM})$. We now give the proof.

Proof. We break the proof into three claims.

Claim 4.8. *z satisfies ($\mathcal{S}'$.1) if and only if $(x, y) = \ell(z)$ satisfies (4.3).*

Proof. Note that (4.3) holds if and only if

$$\begin{aligned}
&\sum_{e\in E} x_e - \sum_{v\in N} y_v = \sum_{v\in R} y_v - 1 \\
\Leftrightarrow &\sum_{K\in\mathcal{K}} \Big(\sum_{e\in E(K)} x_e - \sum_{v\in N(K)} y_v \Big) = |R| - 1 \\
\Leftrightarrow &\sum_{K\in\mathcal{K}} z_K(|E(K)| - |N(K)|) = |R| - 1 \\
\Leftrightarrow &\sum_{K\in\mathcal{K}} z_K(|R(K)| - 1) = |R| - 1
\end{aligned}$$

where in the last step we used (4.8). □

Claim 4.9. *If $(x, y) = \ell(z)$ satisfies (4.4), then z satisfies ($\mathcal{S}'.2$).*

Proof. We recall and rewrite (4.4):

$$\sum_{e\in\gamma(S)} x_e \le \sum_{v\in S\setminus\{k\}} y_v \tag{4.9}$$

$$\Leftrightarrow \sum_{K\in\mathcal{K}} \Big(\sum_{e\in E(K)\cap\gamma(S)} x_e - \sum_{v\in N(K)\cap S} y_v \Big) \le |S\cap R| - y_k \tag{4.10}$$

$$\Leftrightarrow \sum_{K\in\mathcal{K}} z_K(|E(K)\cap\gamma(S)| - |N(K)\cap S|) \le |S\cap R| - y_k \tag{4.11}$$

$$\Leftrightarrow \sum_{K\in\mathcal{K}} z_K f(K,S) \le |S\cap R| - y_k \tag{4.12}$$

where in the last step we use $f(K, S)$ to denote $|E(K) \cap \gamma(S)| - |N(K) \cap S|$. Using a counting argument we actually can equivalently define

$$f(K,S) := |R(K)\cap S| - \kappa(K[S\cap V(K)]) \tag{4.13}$$

where κ denotes the number of connected components and $[\cdot]$ denotes an induced subgraph; the counting argument follows since any induced subgraph of the tree K is a forest, and $\kappa = |V| - |E|$ holds for forests.

Finally, recall we want to show ($\mathcal{S}'.2$) holds for a given $\varnothing \neq S_0 \subset R$. Choose the S_1 defined by

$$S_1 := S_0 \cup \{N(K) \mid K \cap S_0 \neq \varnothing\}$$

and note that $f(K, S_1) = 0$ whenever $R(K) \cap S_0 = \varnothing$ and $f(K, S_1) = |R(K) \cap S_0| - 1$ whenever $K \cap S_0 \neq \varnothing$, so $f(K, S_1) = \rho(K \cap S_0)$ in either case.[4] We know that (4.4), and

[4] $\rho(X) = \max\{0, |X| - 1\}$

equivalently (4.12), holds for this S_1 and any $k \in S_0 = S_1 \cap R$, hence

$$\sum_{K \in \mathcal{K}} z_K \rho(R(K) \cap S_0) \leq |S_1 \cap R| - y_k = |S_0 \cap R| - 1,$$

giving ($\mathcal{S}'.2$) for S_0 as needed. □

Claim 4.10. *If z satisfies ($\mathcal{S}'.2$), then $(x, y) = \ell(z)$ satisfies (4.4).*

Proof. Rearranging as in the previous proof, it suffices to show (4.12) holds for all subsets S of V and all $k \in S$, with f defined by Equation (4.13).

First we handle the special case that $S \cap R = \varnothing$. Then $f(K, S) = -\kappa(K[S \cap V(K)])$ for all K. Now $k \in N(K^*)$ for some full component K^*, and $\kappa(K^*[S \cap V(K^*)]) \geq 1$ so

$$\sum_{K \in \mathcal{K}} z_K f(K, S) \leq -z_{K^*} = |S \cap R| - y_k,$$

so (4.12) holds as needed.

To finish we show (4.12) also holds in the case that $S \cap R \neq \varnothing$. Define $T = S \cap R$. Then from Equation (4.13) we see that each K with $R(K) \cap T = \varnothing$ has $f(K, S) \leq 0$, and each K with $R(K) \cap T \neq \varnothing$ has $f(K, S) \leq |R(K) \cap T| - 1$. Therefore

$$\sum_{K \in \mathcal{K}} z_K f(K, S) \leq \sum_{K \in \mathcal{K}} z_K \rho(R(K) \cap T) \leq_{(\mathcal{S}'.2)} \rho(T) \leq |S \cap R| - y_k,$$

so (4.12) holds as needed. □

This completes the proof of Theorem 4.7. □

4.5 Bidirected/Hypergraphic Equality in Quasibipartite Instances

We will prove the following theorem:

Theorem 4.11. *On quasibipartite instances of the Steiner tree problem,* $\mathrm{OPT}(\mathcal{B}) \geq \mathrm{OPT}(\mathcal{D})$.

Since Polzin and Vahdati Daneshmand [170] proved that for any graph (not necessarily quasibipartite graphs), $\mathrm{OPT}(\mathcal{S}') \geq \mathrm{OPT}(\mathcal{B})$, Theorem 4.11 implies that on quasi-bipartite instances, the hypergraphic and bidirected formulations have the same value.

$$\max \Big\{ \sum_U z_U : \quad z \in \mathbf{R}^{\text{co-valid}(R)} \tag{$\mathcal{D}_D$}$$

$$\sum_{U: K \cap U \neq \varnothing, i \notin U} z_U \leq C_K, \quad \forall K \in \mathcal{K}, \forall i \in K \tag{$\mathcal{D}_D$.1}$$

$$z_U \geq 0, \quad \forall U \in \text{co-valid}(R) \Big\} \tag{$\mathcal{D}_D$.2}$$

Figure 4.2: The dual of ($\mathcal{D}$).

Our proof of Theorem 4.11 uses the dual of ($\mathcal{D}$). In order to smooth the exposition, we define a *co-valid set* to be the complement of a valid set. Explicitly, a co-valid set is one that contains at least one terminal, but not r (whether we are talking about subsets of R or V). Then ($\mathcal{D}$) can be expressed as requiring value at least 1 *out* of every co-valid set, and its dual is given by the LP shown in Figure 4.2.

Recall that the *support* of a solution to ($\mathcal{D}_D$) is the family of sets with positive z_U. A family of sets is called *laminar* if for any two of its sets A, B we have $A \subseteq B, B \subseteq A$, or $A \cap B = \varnothing$. We start with the following standard uncrossing argument.

Lemma 4.12. *There exists an optimal solution to ($\mathcal{D}_D$) whose support is a laminar family of sets.*

Proof. Choose an optimal solution z to ($\mathcal{D}_D$) which maximizes $\sum_U z_U |U|^2$ among all optimal solutions. We claim that the support of this solution is laminar. Suppose not and there exists U and U' with $U \cap U' \neq \varnothing$ and $z_U > 0$ and $z_{U'} > 0$. Construct a new solution z' which is the same as z except $z'_U = z_U - \delta$, $z'_{U'} = z_{U'} - \delta$, $z'_{U \cup U'} = z_{U \cup U'} + \delta$ and $z'_{U \cap U'} = z_{U \cap U'} + \delta$, for a suitably small $\delta > 0$. Note that $U \cap U'$ is not empty, $U \cup U'$ doesn't contain r and the objective value remains unchanged. Also note that for any K and $i \in K$, if $z_{U \cup U'}$ appears in the summand of a constraint, then at least one of z_U or $z_{U'}$ also appears. If both $z_{U \cup U'}$ and $z_{U \cap U'}$ appears, then both z_U and $z_{U'}$ appear. Thus z' is an optimal solution and $\sum_U z'_U |U|^2 > \sum_U z_U |U|^2$, contradicting the choice of z. □

We also need a dual formulation for the bidirected cut formulation. Again, it turns out to be simpler to work with co-valid sets. In the case of (non-hyper) graphs, requiring flow of value 1 into every valid set is essentially the same (with respect to LP value) as requiring flow of value 1 into every co-valid set, since we can just exchange the flow of every arc with that of its reverse. Therefore we use the dual formulation ($\mathcal{B}_D$) given in Figure 4.3.

Note that ($\mathcal{D}_D$) has variables for co-valid subsets of R, while ($\mathcal{B}_D$) has variables for co-valid subsets of V. Inserting Steiner nodes arbitrarily into co-valid subsets of R gives

$$\max \Big\{ \sum_{U} \beta_U : \quad \beta \in \mathbf{R}^{\text{co-valid}(V)} \tag{$\mathcal{B}_D$}$$

$$\sum_{U: a \in \delta^{\text{in}}(U)} \beta_U \leq c_a, \quad \forall a \in A \tag{$\mathcal{B}_D$.1}$$

$$\beta_U \geq 0, \quad \forall U \in \text{co-valid}(V) \Big\} \tag{$\mathcal{B}_D$.2}$$

Figure 4.3: The dual of ($\mathcal{B}$).

co-valid subsets of V; this observation leads us to an approach that we will call *lifting*. In fact we have found two proofs along the same lines, but slightly different. The proofs in this section use an explicit lifting procedure to transform duals. The alternate approach is more like the one that will be given in Section 4.6, which gives a somewhat more direct proof, but one where the lifting is not made explicit. The heart of the lifting proof in this section is the following.

Lemma 4.13. *For quasibipartite instances, given a solution of ($\mathcal{D}_D$) with laminar support, we can get a feasible solution to ($\mathcal{B}_D$) of the same value.*

From Lemmas 4.12 and 4.13, it is immediate that Theorem 4.11 holds. We now sketch the proof of Lemma 4.13.

A feasible solution to ($\mathcal{B}_D$) assigns a value, β_U, to *every* co-valid subset of V such that for any bidirected arc of an edge (u, v), the total β faced by either end point, that is $\sum_{U: u \in U, v \notin U} \beta_U$ and $\sum_{U: u \notin U, v \in U} \beta_U$, is at most $c_{u,v}$. A feasible solution, z, to ($\mathcal{D}_D$), on the other hand, assigns a value only to co-valid subsets of the *terminals* and has a constraint for every full component rather an edge.

We use z to construct a feasible β of the same total value. Note that for a bidirected arc of edge (u, v) where v is a Steiner vertex, say, the total z-value faced by u, $\sum_{U: u \in U, v \notin U} z_U$, can be much larger than c_{uv}, while the total z-value faced by v is 0. To construct a co-valid solution, β, for ($\mathcal{B}_D$), we lift the value of z_U to subsets $U \subset U' \subset U \cup (V \backslash R)$ which contain Steiner vertices. In fact, we give an efficient algorithm to do so in a manner that the total value doesn't drop and in the end we get feasibility. This proves the lemma.

Quasi-bipartiteness is crucial in that we can treat every Steiner vertex independently in the *lifting procedure* and do not have to bother about the feasibility of edges between two Steiner vertices.

4.5.1 Proof of Lemma 4.13

In ($\mathcal{B}_D$), we have two constraints for each bidirected arc of an edge (u, v) — one is $\sum_{U:u\in U,v\notin U} \beta_U \leq c_{uv}$ and the other $\sum_{U:u\notin U,v\in U} \beta_U \leq c_{uv}$. Given β's, we say an edge (u, v) is *u-satisfied* if the first inequality holds and *v-satisfied* if the second holds. We say an edge is *u-tight* or *v-tight* if the inequalities are satisfied as equalities. We say (u, v) is satisfied if it is both u-satisfied and v-satisfied. Note $\beta \geq 0$ is feasible if and only if every edge is satisfied.

Our approach starts with $\beta_U = z_U$ for $U \subset R$ and $\beta_U = 0$ otherwise. Observe that for every Steiner node v and every edge (u, v), (u, v) is v-satisfied since z is nonzero only on subsets of R, but (u, v) is not necessarily u-satisfied. Also observe that all edges (u, v) where u and v are both *terminals* are both u-satisfied and v-satisfied by this initial β; to see this, note that constraint ($\mathcal{D}_D$.1) for the full component (u, v) precisely says that the edge (u, v) is satisfied. In the rest of our proof, we iterate through each Steiner vertex v and *lift* β *at* v. After lifting at v, all edges incident to v become satisfied. Lifting consists of repeated application of the following transfer operation: reduce the value β_U for some set U not containing v, and increase $\beta_{U\cup\{v\}}$ by an equal amount. Our proof depends crucially on the laminarity of $\texttt{supp}(z)$. Although $\texttt{supp}(\beta)$ does not remain laminar during the algorithm, we maintain that its restriction to R, $\{U \cap R \mid U \in \texttt{supp}(\beta)\}$, remains laminar. Note this is clearly true in the beginning since z's support is laminar.

We now precisely define the lifting operation at a Steiner vertex v. Let $\Gamma(v)$ be the set of neighbors of v in G. We will represent $\{U \in \texttt{supp}(\beta) \mid U \cap \Gamma(v) \neq \varnothing\}$ by a forest of rooted trees. In more detail, we start with the standard representation of the laminar family $\{U \cap \Gamma(v) \mid U \in \texttt{supp}(\beta)\}\backslash\{\varnothing\}$: the root nodes of the trees in the forest are sets with no proper supersets, and the parent of any other set is its minimal proper superset. Note that multiple $U \in \texttt{supp}(\beta)$ could have the same intersection S with $\Gamma(v)$. We expand the node representing $S \subset \Gamma(v)$ into a directed path with one node representing each set $\{U \in \texttt{supp}(\beta) \mid U \cap \Gamma(v) = S\}$, and with the children of S made children of the node furthest from the root. Now we describe the procedure for transferring duals with respect to v.

Procedure Lift(v)

1. We maintain a family $\mathbf{A}$ of *active sets*. Initially $\mathbf{A}$ is the set of all root sets of the trees in the forest.
2. We decrease β_U and increase $\beta_{U\cup\{v\}}$ for all $U \in \mathbf{A}$ at a uniform rate until either
   (a) all edges of the form $\{(u, v) : u \in U \cap \Gamma(v)\}$ become u-satisfied for some $U \in \mathbf{A}$,
   (b) or β_U becomes 0 for some set $U \in \mathbf{A}$.

3. In Case (a), we remove U from $\mathbf{A}$ and repeat Step 2.

4. In Case (b), we remove U from $\mathbf{A}$ and put all children of U (with respect to the forest) into $\mathbf{A}$. Go back to Step 2.

We terminate when there are no active sets; the process must terminate since $\texttt{supp}(\beta)$ is finite. Call two sets S, T $\Gamma(v)$-*disjoint* if $S \cap T \cap \Gamma(v) = \varnothing$. Note that the active sets are always $\Gamma(v)$-disjoint, and if at some point some vertex $u \in \Gamma(v)$ has $u \notin \bigcup \mathbf{A}$, then $u \notin \bigcup \mathbf{A}$ will continue to hold until termination.

We first make the following observations. For any $S \subset R$, lifting preserves the invariant

$$\sum_{U \in \text{co-valid}(V): U \cap R = S} \beta_U = z_S. \tag{4.14}$$

This is because the drop in the value of a set U is compensated by a corresponding increase in $U \cup v$. Thus any edge between two terminals remains satisfied throughout the whole algorithm, since it was satisfied before the liftings.

We will show next in Claims 4.14 and 4.15 that after lifting at v, all edges incident to v become satisfied. Furthermore, when we lift at a Steiner vertex v, it does not affect $\sum_{U: u \in U, v' \notin U} \beta_U$ or $\sum_{U: u \notin U, v' \in U} \beta_U$ for any edge (u, v') where v' is another Steiner vertex. Hence once we show both claims, it will follow that all edges are satisfied at termination, and the proof of Lemma 4.13 will be complete.

Claim 4.14. *When* Lift(v) *terminates, all edges of the form* (u, v) *are* u*-satisfied.*

Proof. Consider how the active set containing u evolves over time (if any exists). If u is not a member of any initial active set (i.e. the roots) then clearly $\beta_U = 0$ for all sets U containing u, so (u, v) is u-satisfied. If u leaves $\bigcup \mathbf{A}$ due to Step 3, then (u, v) is u-satisfied. If u leaves $\bigcup \mathbf{A}$ due to Step 4, then we have reduced β_U to 0 for all sets U containing u, so (u, v) is u-satisfied. □

Claim 4.15. *When* Lift(v) *terminates, all edges of the form* (u, v) *are* v*-satisfied.*

Proof. We first sketch the proof idea. Fix an edge (u, v) and look at the total β-value faced by v, that is, $\sum_{U: u \notin U} \beta_{U \cup \{v\}}$. Since the increase in $\beta_{U \cup v}$ is precisely the decrease in β_U, we will be done if we show that the total drop in $\sum_{U: u \notin U} \beta_U$ is at most c_{uv}.

At a high level, in the Lift procedure, the "decrease" of β_U for a set U stops when some edge (u', v) with in $u' \in U$ becomes u'-tight, that is, the "new" β-value faced by u' is precisely $c(u', v)$. Therefore, using the laminarity of the support of β, we can charge the

"new" $\sum_{U:u\notin U}\beta_U$ to the costs of a series of edges of the form $(u_1,v),(u_2,v),\cdots,(u_k,v)$. So, the total *drop* is the "old" $\sum_{U:u\notin U}\beta_U$ minus $\sum_{i=1}^{k}c(u_i,v)$, which equals

$$\sum_{U:u\notin U} z_U - \sum_{i=1}^{k} c(u_i,v)$$

from (4.14) Now for the full component K induced by v and $(u_1,\cdots,u_k,u)$, one can show that the constraint ($\mathcal{D}_D$.1) with $u\in K$, implies that the first term above is *at most* $[c(u,v)+\sum_{i=1}^{k}c(u_i,v)]$, which completes the proof. We now give details of this rather technical proof.

We must show that for any vertex $u\in\Gamma(v)$, at termination, we have $\sum_{U:u\notin U}\beta_{U\cup\{v\}}\le c_{uv}$. Let β'_U be the value of U before v was lifted. Since $\beta_{U\cup\{v\}}$ is precisely the decrease in β_U, we have $\beta_{U\cup\{v\}}=\beta'_U-\beta_U$. Thus we need to show

$$\sum_{U:u\notin U}(\beta'_U-\beta_U)\le c_{u,v}. \tag{4.15}$$

Let $\mathcal{Z}$ denote the family of all sets U for which β_U was reduced to 0. Let $\mathcal{F}$ denote the family of all sets U for which β_U was reduced, but remained nonzero at termination; such sets must have left $\mathbf{A}$ due to Step 3. Sets in $\mathcal{F}$ are $\Gamma(v)$-disjoint since once Step 3 executes on a set U, none of the vertices in $U\cap\Gamma(v)$ will belong to any active sets in any further iterations. Furthermore, due to the condition in Step 2(a), each set $F\in\mathcal{F}$ contains a vertex $u_F\in F\cap\Gamma(v)$ such that the edge (u_F,v) is u_F-tight, i.e.

$$\sum_{U:u_F\in U}\beta_U = c_{u_F,v} \tag{4.16}$$

Let K be the set of all such u_F's.

Now look at the left hand side of the inequality (4.15). This can be rewritten as follows:

$$\sum_{U:u\notin U}(\beta'_U-\beta_U)=\sum_{U\in\mathcal{Z}:u\notin U}(\beta'_U-\beta_U)+\sum_{U\in\mathcal{F}:u\notin U}(\beta'_U-\beta_U)+\sum_{U\notin\mathcal{Z}\cup\mathcal{F}:u\notin U}(\beta'_U-\beta_U) \tag{4.17}$$

The summand in the third term of the right-hand side of Equation (4.17) is zero, but we preserve this term for useful manipulations. Next, define the set K' as follows:

- If $u\in F^*$ for some $F^*\in\mathcal{F}$, define $K':=\{u_F\}_{F\in\mathcal{F},F\neq F^*}$;
- otherwise, if $u\notin\bigcup\mathcal{F}$, define $K':=\{u_F\}_{F\in\mathcal{F}}$.

Note that in any case, each set in $\mathcal{Z}$ contains at least one element of K', each set $F \in \mathcal{F}$ with $F \neq F^*$ contains exactly one element of K', F^* (if it exists) contains no element of K', and each set in $\texttt{supp}(\beta) \setminus (\mathcal{Z} \cup \mathcal{F})$ contains at most one element of K'. Define $\mathcal{X} := \{U \in \texttt{supp}(\beta) \setminus (\mathcal{Z} \cup \mathcal{F}) \mid U \cap K' \neq \varnothing\}$. Since the sets in $\mathcal{F}$ are disjoint, and since each $X \in \mathcal{X}$ is a subset of some $F \in \mathcal{F}$, each $X \in \mathcal{X}$ is a subset of F which *doesn't* contain u, and $|X \cap K'| = 1$.

Recall that $\beta'_U - \beta_U = 0$ for all U not in $\mathcal{Z} \cup \mathcal{F}$, so Equation (4.17) yields

$$\begin{aligned}\sum_{U:u\notin U} (\beta'_U - \beta_U) &= \sum_{U\in\mathcal{Z}:u\notin U} (\beta'_U - \beta_U) + \sum_{U\in\mathcal{F}:u\notin U} (\beta'_U - \beta_U) + \sum_{U\in\mathcal{X}:u\notin U} (\beta'_U - \beta_U) \\ &= \sum_{U\in\mathcal{Z}\cup\mathcal{F}\cup\mathcal{X}:u\notin U} \beta'_U - \sum_{U\in\mathcal{Z}\cup\mathcal{F}\cup\mathcal{X}:u\notin U} \beta_U \end{aligned} \tag{4.18}$$

Using the fact that $\beta_U = 0$ for all $U \in \mathcal{Z}$, the right summand in the RHS of Equation (4.18) is

$$\sum_{U\in\mathcal{Z}\cup\mathcal{F}\cup\mathcal{X}:u\notin U} \beta_U = \sum_{U\in\mathcal{F}\cup\mathcal{X}:u\notin U} \beta_U = \sum_{u_F\in K'} \sum_{U:u_F\in U} \beta_U = \sum_{u_F\in K'} c_{u_F,v} \tag{4.19}$$

where the rightmost equality uses Equation (4.16).

To interpret the left summand in the RHS of Equation (4.18), take the set $K := K' \cup \{u\}$ and the full component/star formed by the edges $\{(v, w) \mid w \in K\}$. Then constraint ($\mathcal{D}_D$.1) for K and u implies

$$\sum_{U:K\cap U\neq\varnothing,u\notin U} z_U \leq C_K \leq \sum_{u_F\in K'} c_{u_F,v} + c_{u,v}. \tag{4.20}$$

Furthermore, since all sets U in $\mathcal{Z} \cup \mathcal{F} \cup \mathcal{X}$ with $u \notin U$ have non-empty intersection with K, and using Equation (4.14), we find

$$\sum_{U\subset R:K\cap U\neq\varnothing,u\notin U} z_U = \sum_{U\subset V:K\cap U\neq\varnothing,u\notin U} \beta'_U \geq \sum_{U\in\mathcal{Z}\cup\mathcal{F}\cup\mathcal{X},u\notin U} \beta'_U. \tag{4.21}$$

Combining Equations (4.18) to (4.21), we get Equation (4.15) as needed. □

4.6 Bidirected/Hypergraphic Equality in 5-Preprocessed Instances

In the previous section Section 4.5, we used a *lifting* approach to show that the hypergraph and the bidirected relaxations have the same optimal value on quasi-bipartite graphs. In

this section we use a computational variant of this approach to prove the same equality for another class of instances. Recall (from Chapter 2) that an *r-preprocessed* instance consists of element-disjoint full components on all possible sets of at most r terminals. The main result of this section is the following.

Theorem 4.16. *On all 5-preprocessed instances,* $\text{OPT}(\mathcal{P}) \leq \text{OPT}(\mathcal{B})$.

As before, since it is known that $\text{OPT}(\mathcal{P}) \geq \text{OPT}(\mathcal{B})$ in all graphs, this actually establishes equality. Moreover we show that Theorem 4.16 is tight in the sense that there is a 6-preprocessed instance (Section 4.6.4) with $\text{OPT}(\mathcal{P}) > \text{OPT}(\mathcal{B})$.

We now make a remark. The notion of r-preprocessing contrasts with the notion of *b-quasi-bipartiteness* introduced in Chapter 5 in the following way: the former bounds the number of terminals per full component, while the latter bounds the number of Steiner nodes per full component. We showed in Section 4.5 that $\text{OPT}(\mathcal{P}) = \text{OPT}(\mathcal{B})$ on 1-quasi-bipartite instances, and the example in Section 4.6.4 is a 4-quasi-bipartite instance upon which $\text{OPT}(\mathcal{P}) \neq \text{OPT}(\mathcal{B})$. It would be interesting to find out whether the LPs have the same value on all 2-quasi-bipartite and 3-quasi-bipartite instances.

4.6.1 Strategy

We work along the lines of the lifting approach from Section 4.5 but with a couple of differences. First, we use an LP ($\mathcal{C}$) that is slightly stronger than ($\mathcal{D}_D$). Second, in contrast to the very explicit lifting algorithm of Section 4.5, we use an indirect approach. We write a cost-polyhedron which expresses the lifting problem in an abstract setting, and argue that lifting can be performed for all cost functions iff lifting can be performed for extreme points of this cost-polyhedron. Using Maple, we exhaustively find that all extreme points of the cost-polyhedron are 0-1 for 5-preprocessed instances. Finally, we give an easy argument showing that 0-1 vectors can be lifted, which completes the proof.

Let $\mathcal{P}(R)$ denote the power set of R. The new auxiliary LP ($\mathcal{C}$), pictured in Figure 4.4, differs from ($\mathcal{D}_D$) in the following way. First, we have variables z_U for sets in $\mathcal{P}(R)\backslash\text{co-valid}(R)$, but they may be thought of as dummy variables included for the sake of symmetry, since they do not contribute to the objective function. Second, we have the added constraint ($\mathcal{C}.2$) which will be useful computationally for the purposes of normalization.

Lemma 4.17. *We have* $\text{OPT}(\mathcal{C}) \geq \text{OPT}(\mathcal{P}_D)$*, and* ($\mathcal{C}$) *has an optimal solution with laminar support.*

$$\max\Big\{ \sum_{U\in\text{co-valid}(R)} z_U : \quad z\in\mathbf{R}^{\mathcal{P}(R)} \tag{$\mathcal{C}$}$$

$$\sum_{U:K\cap U\neq\varnothing, i\notin U} z_U \le C_K, \quad \forall K\in\mathcal{K}, \forall i\in K \tag{$\mathcal{C}$.1}$$

$$\sum_{U:i\in U} z_U = \sum_{U:j\in U} z_U \quad \forall i,j\in R \tag{$\mathcal{C}$.2}$$

$$z_U \ge 0, \quad \forall U\in\mathcal{P}(R)\Big\} \tag{$\mathcal{C}$.3}$$

Figure 4.4: The auxiliary LP used in our new lifting approach.

Proof. Let y^* be an optimal solution to ($\mathcal{P}_D$). Define a solution z^* to ($\mathcal{C}$) as follows. For any $U\subset R$ with $r\notin U$, define

$$z^*_U := \sum_{\pi:\ U \text{ is a part of } \pi} y^*_\pi$$

We claim that this z^* is a feasible solution to ($\mathcal{C}$) of value $\sum_\pi (r(\pi)-1)y^*_\pi$. This will show that $\text{OPT}(\mathcal{D}_D)\ge\text{OPT}(\mathcal{P}_D)$.

First, note that $\sum_{U:i\in U} z^*_U = \|y^*\|_1$ for all i, so ($\mathcal{C}$.2) is satisfied. Second, to see that z^* satisfies ($\mathcal{C}$.1), note that for any full component K and any $i\in K$,

$$\sum_{U:U\cap K\neq\varnothing, i\notin U} z^*_U = \sum_{U:U\cap K\neq\varnothing, i\notin U}\ \sum_{\pi:U\in\pi} y^*_\pi = \sum_\pi \sum_{U\in\pi:U\cap K\neq\varnothing, i\notin U} y^*_\pi = \sum_\pi y^*_\pi \mathtt{rc}^\pi_K$$

where the last equality follows from the definition of $\mathtt{rc}^\pi_K$. The feasibility of z^* now follows from the feasibility of y^*.

To see that the objective values are the same, note $z^*_\varnothing = 0$ and so

$$\sum_{U\text{ co-valid}} z^*_U = \sum_{U:r\notin U} z^*_U = \sum_{U:r\notin U}\ \sum_{\pi:U\in\pi} y^*_\pi = \sum_\pi \sum_{U\in\pi, r\notin U} y^*_\pi = \sum_\pi (r(\pi)-1)y^*_\pi.$$

To get laminarity we proceed as in the proof of Lemma 4.12; note that this set uncrossing preserves constraint ($\mathcal{C}$.2). □

What we will show in the next section is the following.

Lemma 4.18. *For 5-preprocessed graphs, given a solution z to $(\mathcal{C})$ with laminar support, it can be lifted to a feasible solution β for $(\mathcal{B}_D)$ of the same value.*

Combining Lemmas 4.17 and 4.18 clearly gives a proof of Theorem 4.16, hence we proceed to prove the latter. To be totally precise, there is a technical point which we must address. Lifting z can produce sets not in co-valid(V), for example since $(\mathcal{C})$ has variables for sets including the root. Therefore, in Lemma 4.18, we think of $(\mathcal{B}_D)$ as having variables for *every* subset of V, but that the objective function is $\sum_{U \text{ co-valid}} \beta_U$, i.e. non-co-valid sets do not contribute to the objective function.

4.6.2 Local Lifting: Proof of Lemma 4.18

One of the key ingredients will be that in lifting, it is sufficient to deal with a single full component at a time. (This is analogous to how, in Section 4.5, we were able to process each Steiner vertex in isolation, without affecting any other Steiner vertices.) At a high level, to lift z (indexed by subsets of R) to β (indexed by subsets of V), we add the Steiner vertices from each full component one at a time.

To explain our approach, it is helpful to take a more formal and general view of lifting. (Note, our convention unless otherwise specified is that the containment symbol $B \supset A$ means weak containment, i.e. possibly $B = A$.)

Definition 4.19. *Let A, B be finite sets with $B \supset A$, with $b \in \mathbf{R}_+^{\mathcal{P}(B)}$ and $a \in \mathbf{R}_+^{\mathcal{P}(A)}$. We say b* is a lift (of a to B) *if, for each $S \subset A$, we have*

$$a_S = \sum_{T \subset B : T \cap A = S} b_S.$$

We now give a formal technical statement of how local lifts can be combined into a global lift, which is how we deal with full components one at a time.

Lemma 4.20 (Common lift lemma)**.** *Let B, C be finite sets with with $a \in \mathbf{R}_+^{\mathcal{P}(B \cap C)}, b \in \mathbf{R}_+^{\mathcal{P}(B)}, c \in \mathbf{R}_+^{\mathcal{P}(C)}$. If b is a lift of a to B and c is a lift of a to C, there exists $d \in \mathbf{R}_+^{\mathcal{P}(B \cup C)}$ so that d is a lift of both b and c.*

To avoid confusion, we now take K only to mean the tree representing a full component, with $\mathcal{K}$ still the collection of all full components, but now $R(K)$ will be explicitly used to denote the set of terminals belonging to a full component K. We also use $N(K)$ to denote the set of Steiner vertices in K, $E(K)$ the set of edges in K, and $A(K)$ the arcs obtained by orienting edges in $E(K)$ in both ways.

The use of the lifting approach will become clear when we prove Lemma 4.18. Enumerate all full components as $\mathcal{K} = \{K_1, K_2, \dots\}$.

Lemma 4.21 (Local lift lemma)**.** *Let z be a laminar solution to ($\mathcal{C}$) on a 5-preprocessed instance. For every full component K_i, there is a lift z_i of z from R to $R \cup N(K_i)$ so that z_i satisfies the bidirected cut dual constraints ($\mathcal{B}_D$.1) for arcs in $A(K_i)$.*

We will prove both the common lift lemma and the local lift lemma later on, but for now show how they together yield Lemma 4.18.

Proof of Lemma 4.18. We work in iterations. In the first iteration we apply Lemma 4.20 to the lifts z_1 and z_2 of z; thus there is a common lift ω of z_1 and z_2 to $R \cup N(K_1) \cup N(K_2)$. Crucially, since the constraints ($\mathcal{B}_D$.1) for $A(K_i)$ only depend on nodes in $R(K_i) \cup N(K_i)$, the solution ω will also satisfy ($\mathcal{B}_D$.1) for all arcs in $A(K_1) \cup A(K_2)$. We then find a common lift of ω and z_3, and so forth, until we obtain a common lift β of all the z_i, which likewise satisfies ($\mathcal{B}_D$.1) for all arcs in all full components.

The proof is completed by noting two technicalities. First, each terminal-terminal arc $a = uv$ also satisfies ($\mathcal{B}_D$.1) by directly considering constraint ($\mathcal{C}$.1) for $K = \{u, v\}$, hence β is feasible for ($\mathcal{B}_D$). Second, the objective value is preserved by lifting. □

We now prove the common lifting lemma (there are also several other comparably short proofs using alternate methods).

Proof of Lemma 4.20. For any set $S \subset B \cap C$, we define values $\{d_{S \cup T \cup U} \mid T \subset B \backslash C, U \subset C \backslash B\}$ as follows. If $a_S = 0$, we set each such $d_{S \cup T \cup U}$ to be 0. Otherwise, define $d_{S \cup T \cup U} = b_{S \cup T} c_{S \cup U} / a_S$. Then it is easy to verify that d is a lift of b and also a lift of c. For example, to check that d is a lift of b, note that for any subset $S \cup T$ of B (where $S \subset B \cap C$ and $T \subset B \backslash C$) if $a_S \neq 0$ we have

$$\sum_{X \subset (B \cup C): X \cap B = S \cup T} d_X = \sum_{U \subset C \backslash B} d_{S \cup T \cup U} = \frac{b_{S \cup T}}{a_S} \sum_{U \subset C \backslash B} c_{S \cup U} = \frac{b_{S \cup T}}{a_S} a_S = b_{S \cup T}$$

as needed, where in the second-last equality we used the fact that c is a lift of a. □

4.6.3 Computational Proof of Local Lifting Lemma

We now give a proof of Lemma 4.21, which we restate, omitting the unnecessary index i.

Lemma 4.21. *Let z be a laminar solution to ($\mathcal{C}$) on a 5-preprocessed instance. For every full component K, there is a lift z' of z from R to $R \cup N(K)$ so that z' satisfies the bidirected cut dual constraints ($\mathcal{B}_D$.1) for arcs in $A(K)$.*

Proof. First, using the common lift lemma (Lemma 4.20), we forget about all vertices not in K. Precisely, let $y \in \mathbf{R}_+^{\mathcal{P}(R(K))}$ be the *projection* of z to $R(K)$, defined for all $W \subset R(K)$ by

$$y_W = \sum_{S \subset R : S \cap R(K) = W} z_S. \tag{4.22}$$

It is immediate that z is a lift of y from $R(K)$ to R. The hardest part of the proof is that we will find a lift y' of y from $R(K)$ to $R(K) \cup N(K)$, such that y' satisfies constraints ($\mathcal{B}_D$.1) for arcs in $A(K)$. Once we do this, the common lift lemma shows there is a common lift z' of y' and z to $R \cup N(K)$; and it is easy to see that this z' also satisfies constraints ($\mathcal{B}_D$.1) for arcs in K.

It is not hard to argue that without loss of generality, K has no Steiner vertices of degree 2. Formally, one may show that the two edges incident to such a vertex can be merged and their costs added together, and that feasible dual bidirected solutions on the merged graph can be converted to feasible dual bidirected solutions on the original. Alternatively, one may use a metric assumption like that in Section 5.1.1 to argue K has no Steiner vertices of degree 2.

Let $\mathcal{L}$ denote the support of y, which is easily seen to be a laminar family on $R(K)$. Let c be short-hand for the vector $\{c_e \mid e \in E(K)\}$ of costs of edges of K. For a subset W of $R(K)$, let $K[W]$ denote the subtree of full component K with terminal set W; i.e., the edge set of $K[W]$ is $\bigcup_{i,j \in W} P_{ij}$ where P_{ij} denotes the unique i-j path in the tree K. We know by the definition of the cost function C on full components that $C_W \leq c(E(K[W]))$ for all $W \subset R(K)$.

Constraint ($\mathcal{C}$.2) shows that $\sum_{S \subset R(K) : i \in S} y_i$ is constant for all choices of $i \in R(K)$. In fact without loss of generality, $\sum_{S \subset R(K)} y_i = 1$ for all $i \in R(K)$, since we can scale all of z, y, C, and $\{c_e \mid e \in E(K)\}$ by the needed factor without affecting the lifting task at hand. This simplifies ($\mathcal{C}$) considerably and guarantees that the pair (c, y) is a member of the polyhedron

$$\mathcal{Q} := \Big\{ (c, y) \in \mathbf{R}_+^{E(K) \uplus \mathcal{L}} \mid \ \forall i \in R(K) : \sum_{U \in \mathcal{L} : i \in U} y_U = 1; \tag{4.23}$$

$$\forall W \subset R(K) : -1 + \sum_{U : W \cap U \neq \varnothing} y_U \leq c(E(K[W])) \Big\}. \tag{4.24}$$

Our goal in lifting is to prove that (c, y) admits a feasible lift y', by which we mean

y' is a lift of y to $R(K) \cup N(K)$, such that for each $uv \in A(K)$, $\sum_{U : u \notin U, v \in U} y'_U \leq c_{uv}$.

The following claim is easy to verify, since we are dealing with only linear constraints.

Claim 4.22. *If (c_i, y_i) admits the feasible lift y'_i for $i = 1, 2$ and $0 \leq \lambda \leq 1$, then $(\lambda c_1 + (1-\lambda)c_2, \lambda y_1 + (1-\lambda)y_2)$ admits the feasible lift $\lambda y'_1 + (1-\lambda)y'_2$.*

Therefore, it suffices to show that every extreme point (c, y) of $\mathcal{Q}$ admits a feasible lift. This is facilitated by the following convenient fact.

Proposition 4.23. *If K is a full component with at most 5 terminals and $\mathcal{L}$ is a laminar family on $R(K)$, then every extreme point of $\mathcal{Q}$ is 0-1.*

Proof. We argued earlier that every Steiner vertex in K has degree at least 3, which means that K has at most $|R(K)| - 2$ Steiner nodes. Thus there are only a finite number of possible shapes for K. A little more strongly, we may assume $|N(K)| = |R(K)| - 2$, by splitting Steiner nodes of degree ≥ 4 and inserting edges of cost 0.

We use the Maple package `convex` [75] to enumerate all vertices of the polytope $\mathcal{Q}$ for all possible full component shapes and all possible laminar families $\mathcal{L}$ on $R(K)$. The Maple code which completes the proof of Proposition 4.23 is given in Appendix A. □

Finally, to complete the proof of Lemma 4.21, it suffices to show that 0-1 points in $\mathcal{Q}$ lift.

Proposition 4.24. *If (c, y) is a 0-1 vector in $\mathcal{Q}$, then y admits a feasible lift y'.*

Proof. First, note constraint (4.23) implies that y is the indicator function for a partition π of $R(K)$. For each part π_i of π let $\ell_i \subset V(K)$ be π_i together with all Steiner vertices connected to terminals in π_i by zero-cost edges.

We claim that the sets ℓ_i are pairwise disjoint. Suppose otherwise for the sake of contradiction that $v \in \ell_i \cap \ell_j$. Then is a zero-cost path from v to $r_i \in \pi_i$ and from v to $r_j \in \pi_j$, and consequently a zero-cost path from r_i to r_j. But constraint (4.24) for $W = \{r_i, r_j\}$ ensures that the unique r_i-r_j path in K has cost at least 1, a contradiction.

Finally, set $y'_{\ell_i} = 1$ for all i, and each other y' value to zero. Now fix any arc $uv \in A(K)$, which falls into one of two cases.

$c_{uv} = 0$. Then every ℓ_i either contains both or neither of u, v, so $\sum_{U: u \notin U, v \in U} y'_U = 0 \leq c_{uv}$ as needed.

$c_{uv} = 1$. There is at most one i for which $v \in \ell_i$, by the disjointness of the ℓ_i. Hence $\sum_{U: u \notin U, v \in U} y'_U \leq 1 = c_{uv}$ as needed. □

This completes the proof of Lemma 4.21. □

We remark that the laminarity of $\texttt{supp}(y)$ is key to our approach: if $\mathcal{L}$ is an arbitrary family on $R(K)$, then $\mathcal{Q}$ has non-0-1 vertices even when $|R(K)| = 3$. (This is demonstrated in Appendix A.)

4.6.4 A 6-Preprocessed Instance with $\mathrm{OPT}(\mathcal{P}) \neq \mathrm{OPT}(\mathcal{B})$

Consider the graph in Figure 4.5. It is based on a well-known gadget [1] used by Goemans to get an integrality gap of $8/7 - \epsilon$ for the bidirected cut formulation. The minimal Steiner tree cost is 8. This instance has $\mathrm{OPT}(\mathcal{B}) = 15/2$, and upon 6-preprocessing, this value can only decrease since the original graph remains as a subgraph. We now argue that $\mathrm{OPT}(\mathcal{P}) = 8$. Let π denote the partition $\{\{t_1, t_2\}, \{t_3, t_4\}, \{t_5, t_6\}\}$. It is not hard to see that $C_K \geq 4\mathtt{rc}_K^\pi$ for every full component K. Thus any feasible solution x has $\sum_K x_K C_K \geq 4 \sum_K \mathtt{rc}_K^\pi \geq 4(r(\pi) - 1) = 8$, since x meets constraint ($\mathcal{P}$.2).

We remark that we previously conjectured [138] that $\mathrm{OPT}(\mathcal{P}) = \mathrm{OPT}(\mathcal{B})$ on all preprocessed graphs, but this example refutes that conjecture.

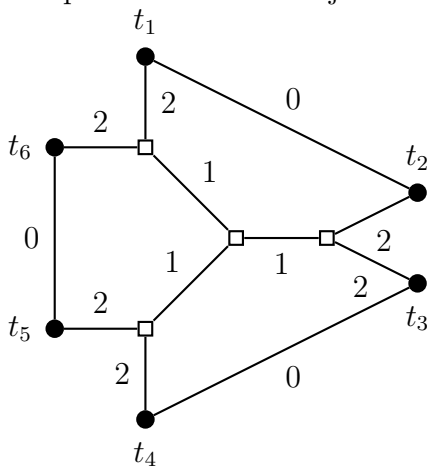


Figure 4.5: This graph shows that $\mathrm{OPT}(\mathcal{P})$ and $\mathrm{OPT}(\mathcal{B})$ can be different on 6-preprocessed instances. Dots are terminals and squares are Steiner vertices. Edge costs are shown.

4.7 Relating the Polyhedra Defined by $(\mathcal{P}')$ and $(\mathcal{P})$

For a polytope $\mathcal{X}$, its *dominant* is the polytope $\mathtt{dom}(\mathcal{X}) := \{y \mid \exists x \in \mathcal{X} : x \leq y\}$. In ordinary graphs (i.e., with all hyperedges of size at most 2), $(\mathcal{P})$ and $(\mathcal{P}')$ are related in a very simple way: the unbounded polytope $(\mathcal{P})$ is the the dominant of the bounded polytope $(\mathcal{P}')$. (See Chopra [46].) However, this polyhedral relation is no longer true in hypergraphs.

Example 4.25. *Consider $R = \{1, 2, 3, 4\}$ and the point $x \in \mathbf{R}^{\mathcal{K}}$ such that $x_{1,2,3} = x_{1,2,4} = 1$ and x is zero elsewhere. Then x lies in $(\mathcal{P})$, but there is no $y \leq x$ with $y \in (\mathcal{P}')$, i.e. $x \notin \mathtt{dom}(\mathcal{P}')$.*

Nonetheless, it is possible to change the false statement "($\mathcal{P}$) = $\mathtt{dom}(\mathcal{P}')$" into a true one by two relatively minor changes. Let $\mathtt{pos}(\mathcal{X}) := \{x \in \mathcal{X} \mid x \geq 0\}$ denote the nonnegative part of $\mathcal{X}$. Let ($\mathcal{EP}'$) denote an *extended* version of ($\mathcal{P}'$) where we remove the nonnegativity constraint:

$$(\mathcal{EP}') := \{x \in \mathbf{R}^{\mathcal{K}} : \sum_{K \in \mathcal{K}} x_K \mathtt{rc}_K^\pi \geq r(\pi) - 1, \quad \forall \pi \in \Pi_R; \quad \sum_{K \in \mathcal{K}} x_K(|K| - 1) = |R| - 1\}.$$

In the rest of this section, we prove the following result.

Theorem 4.26. $\mathcal{P} = \mathtt{pos}(\mathtt{dom}(\mathcal{EP}'))$.

Despite the fact that the theorem has such a succinct statement, it takes us quite a bit of technology to prove it. Namely, we look more carefully at the structure of the hypergraph-partition formulations. A shorter or more direct proof would be interesting.

4.7.1 Contraction

For any finite set X, we use $\mathcal{K}(X)$ to denote the family of all nonempty subsets of X, and we use ($\mathcal{P}$)$[X]$ to denote the version of ($\mathcal{P}$) on terminal set $R = X$.

Let S be a subset of R and let $R/S := (R \backslash S) \cup \{\langle S \rangle\}$ where $\langle S \rangle$ is a pseudonode corresponding to S being contracted. For a subset K of R (i.e., a hyperedge) we define the contraction K/S by

$$K/S := \begin{cases} (K \backslash S) \cup \{\langle S \rangle\}, & \text{if } K \cap S \neq \varnothing; \\ K, & \text{otherwise.} \end{cases}$$

Define *contraction* to be the linear map $x \mapsto x/S$ from $\mathbf{R}^{\mathcal{K}(R)}$ to $\mathbf{R}^{\mathcal{K}(R/S)}$ defined by

$$(x/S)_K = \sum_{J \in \mathcal{K}(R) : J/S = K} x_J.$$

By extension, we can contract a family $\mathcal{F}$ of sets via $\mathcal{F}/\cdot = \{f/\cdot \mid f \in \mathcal{F}\}$, and we can contract *by* a family $\mathcal{F} = \{f_1, f_2, \dots\}$ of disjoint sets via $\cdot/\mathcal{F} = \cdot/f_1/f_2/\cdots$. One example of why this is natural is that $\mathtt{rc}_K^\pi = \rho(K/\pi)$.

It turns out that ($\mathcal{P}$) behaves nicely under contraction. Note for $\pi, \sigma \in \Pi(R)$, that π/σ is a partition (of R/σ) if and only if σ refines π. The following claim is simple to prove.

Claim 4.27. *If σ refines π then $r(\pi/\sigma) = r(\pi)$, and $\mathtt{rc}_{K/\sigma}^{\pi/\sigma} = \mathtt{rc}_K^\pi$ for all K.*

Now we begin a series of structural observations about the partition formulations.

Claim 4.28. *If $x \in (\mathcal{P})[R]$ and $\sigma \in \Pi_R$ then $x/\sigma \in (\mathcal{P})[R/\sigma]$.*

Proof. For any partition π of R/σ, let $\widehat{\pi}$ denote the unique partition coarsening σ such that $\widehat{\pi}/\sigma = \pi$, i.e., $\widehat{\pi}$ is the partition of R obtained by uncontracting σ. Clearly x/σ is nonnegative and by Claim 4.27, for all $\pi \in \Pi_{R/\sigma}$,

$$\sum_{K \in \mathcal{K}(R/\sigma)} \mathtt{rc}^{\pi}_K(x/\sigma)_K = \sum_{K \in K(R)} \mathtt{rc}^{\widehat{\pi}}_K x_K \geq \rho(\widehat{\pi}) - 1 = \rho(\pi) - 1, \tag{4.25}$$

where the middle inequality holds because $x \in (\mathcal{P})[R]$ (specifically, constraint ($\mathcal{P}$.2) for $\widehat{\pi}$). □

Corollary 4.29. *If τ coarsens σ and τ is tight for $x \in (\mathcal{P})[R]$, then τ/σ is tight for x/σ.*

Proof. Clearly $\tau = \widehat{\pi}$ for some $\pi \in \Pi_{R/\sigma}$; then Corollary 4.29 follows from the fact that Equation (4.25) holds with equality for this choice of π. □

For the next corollary, the *most refined tight partition* for x^* is defined as the meet of all tight partitions for x^*, which by Corollary 3.12, is itself tight.

Corollary 4.30. *If x^* is a vertex of $(\mathcal{P})[R]$ and τ is the most refined tight partition for x^*, then x^*/τ is a vertex of $(\mathcal{P}')[R/\tau]$.*

Proof. By Claim 4.28, x^*/τ is feasible for $(\mathcal{P})[R/\tau]$. We now need to show it is a vertex.

By elementary LP theory, we know that there is a nonsingular square system indexed by $\mathcal{S} = \mathtt{supp}(x^*)$ and a family $\mathcal{C}$ of partitions,

$$x \in \mathbf{R}^{\mathcal{S}} : \forall \pi \in \mathcal{C}, \quad \sum_{K \in \mathcal{S}} x_K \mathtt{rc}^{\pi}_K = r(\pi) - 1 \tag{4.26}$$

for which the unique solution is x^*.

Now, consider the system

$$x \in \mathbf{R}^{\mathcal{S}/\tau} : \forall \pi \in \mathcal{C}, \quad \sum_{K \in \mathcal{S}} x_{K/\tau} \mathtt{rc}^{\pi/\tau}_{K/\tau} = r(\pi/\tau) - 1. \tag{4.27}$$

Both systems (4.26) and (4.27) use the same nonsingular square matrix of coefficients. Moreover, it is easy to see that x^*/τ is a solution to the latter, which implies that x^*/τ is a vertex of $(\mathcal{P})[R/\tau]$.

Finally, we know that τ is tight for x^* so Corollary 4.29 implies τ/τ is tight for x^*/τ, which implies that in fact x^* is a vertex of $(\mathcal{P}')[R/\tau]$. □

Note in the previous proof that $|\mathtt{supp}(x^*/\tau)| = |\mathtt{supp}(x^*)|$ because otherwise, if $K, K' \in \mathtt{supp}(x^*)$ are such that $K/\tau = K'/\tau$, then the columns corresponding to K and K' are identical, contradicting the fact that the square matrix is non-singular.

4.7.2 Expanding and Restricting

For succinctness, say that x is an *expansion of* x' *by* τ if $x' \in (\mathcal{P})[R/\tau]$, $x/\tau = x'$ and $|\,\mathtt{supp}(x)| = |\,\mathtt{supp}(x')|$. Then Corollary 4.30 says that every vertex of $(\mathcal{P})[R]$ is an expansion of a vertex of $(\mathcal{P}')[R/\tau]$ for some τ.

Remark 4.31. *It is possible that a fixed vertex of* $(\mathcal{P}')[R/\tau]$ *can be expanded into a vertex of* $(\mathcal{P})[R]$ *in more than one way. For example, let* $|R| = 4$, $\tau = \{\{1\},\{2\},\{3,4\}\}$, *and let* x^* *be the characteristic vector of* $K_{2,1}$ *where the pseudonode* $\langle\{3,4\}\rangle$ *has degree two. Then the characteristic vectors of* $\{\{1,3\},\{2,3,4\}\}, \{\{1,3,4\},\{2,3,4\}\}, \{\{1,3,4\},\{2,4\}\}$ *and others are extreme points of* $(\mathcal{P})[R]$ *which are expansions of* x^* *by* τ.

Given τ and a vertex x' of $\mathcal{P}(R/\tau)$, is there a good way to characterize all of the vertices into which x' can be expanded? On the one hand, for each $K \in \mathtt{supp}(x')$, writing $K = \{\tau_{i_1}, \ldots, \tau_{i_k}\}$, we need to "replace" K by a set of the form $T_{i_1} \cup \cdots \cup T_{i_k}$ where $\varnothing \neq T_{i_j} \subseteq \tau_{i_j}$ for each j. But these choices cannot all be made independently. For example, in Remark 4.31 $\{\{1,3\},\{2,4\}\}$ is not an expansion of x^* since it is not feasible. Next (Claim 4.32) we give a sufficient condition for the expansion x of x' to lie in $(\mathcal{P})[R]$, which is recursive. (The condition is also necessary, but we omit the proof of this unneeded fact.)

We define the linear *restriction* map $x \mapsto x|S$ from $\mathbf{R}^{\mathcal{K}(R)}$ to $\mathbf{R}^{\mathcal{K}(S)}$ by

$$(x|S)_K = \sum_{J \in \mathcal{K}(R): J \cap S = K} x_J.$$

(Note that for a J with $J \cap S = \varnothing$, this x_J never appears in the right-hand side.)

Claim 4.32. *Let* x' *be a vertex of* $(\mathcal{P}')[R/\tau]$, *and let* x *be an expansion of* x' *by* τ. *If* $x \in (\mathcal{P})[R]$ *then* $x|\tau_i \in (\mathcal{P})[\tau_i], \forall i$.

Proof. Consider any partition π of τ_i. Let $\tau \wedge \pi$ denote the partition of R obtained from τ by further refining τ_i into π. Note that $r(\tau \wedge \pi) = r(\tau) + r(\pi) - 1$. Since $x \in \mathcal{P}(R)$, x meets the partition inequality for $\tau \wedge \pi$. We also know that τ is tight for x. Thus, subtracting the equality from the inequality,

$$\sum_K x_K(\mathtt{rc}_K^{\tau\wedge\pi} - \mathtt{rc}_K^{\tau}) \geq r(\tau \wedge \pi) - 1 - (r(\tau) - 1) = r(\pi) - 1.$$

However, it is easy to see that $\mathtt{rc}_K^{\tau\wedge\pi} - \mathtt{rc}_K^{\tau} = \mathtt{rc}_{K\cap\tau_i}^{\pi}$ for all K (where we take $\mathtt{rc}_{\varnothing}^{\pi} = 0$). This implies that

$$\sum_{K \in \mathcal{K}(\tau_i)} (x|\tau_i)_K \mathtt{rc}_K^{\pi} = \sum_K x_K(\mathtt{rc}_K^{\tau\wedge\pi} - \mathtt{rc}_K^{\tau}) \geq r(\pi) - 1,$$

so $x|\tau_i$ meets the inequality of $(\mathcal{P})[\tau_i]$ corresponding to partition π. Since π and i were arbitrary, the claim is complete. □

4.7.3 Proof of Theorem 4.26

It is straightforward to see that $\texttt{pos}(\texttt{dom}(\mathcal{EP}')) \subset \mathcal{P}$ because

- if $x \in (\mathcal{EP}')$ then x meets all partition inequalities ($\mathcal{P}.2$)
- if $y \geq x, x \in (\mathcal{EP}')$ then y also meets all partition inequalities

and $(\mathcal{P})$ is a subset of the positive orthant.

We proceed by induction on $|R|$. Note that the statement remaining to be proved is equivalent to: for $x \in (\mathcal{P})[R]$, there is a $\delta \in \mathbf{R}_+^{\mathcal{K}(R)}$ such that $x - \delta \in (\mathcal{EP}')[R]$. Since $(\mathcal{P})$ has the nonnegative orthant as its recession cone, it is easy to see that we only need consider the case that x is a vertex of $(\mathcal{P})[R]$.

If $\overline{\pi}$ is tight for x then $x \in (\mathcal{EP}')[R]$ and there is nothing to prove. Let $\tau \neq \overline{\pi}$ be the most refined tight partition for x. We may assume $\tau \neq \underline{\pi}$ since the constraint ($\mathcal{P}.2$) is vacuous for $\pi = \underline{\pi}$ and the vertex x must therefore have some other nontrivial tight inequality.

We know that by Claim 4.32 x is an expansion of a vertex $x' \in (\mathcal{P})[R/\tau]$. Let the parts of τ be $\tau_1, \ldots, \tau_r$. By induction and Claim 4.32, for each part τ_i, there is a $\delta_i \in \mathbf{R}_+^{\mathcal{K}(\tau_i)}$ with $(x|\tau_i) - \delta_i \in (\mathcal{EP}')[\tau_i]$. We claim that the point y defined by

$$y := x - \sum_i \delta_i$$

lies in $(\mathcal{EP}')[R]$, which will complete the proof. To do so we will use the following alternate formulation for $(\mathcal{EP}')$,

$$(\mathcal{EP}') = \{x \in \mathbf{R}^{\mathcal{K}} : \sum_{K \in \mathcal{K}(R)} x_K \rho(K \cap S) \leq \rho(S), \quad \forall S \subset R; \quad \sum_{K \in \mathcal{K}(R)} x_K \rho(K) = \rho(R)\};$$

the equivalence of this formulation with the original can be seen using the proof technique of Theorem 4.1. It remains to show that $\sum_K y_K \rho(K \cap S) \leq \rho(S)$ for all $S \subset R$, with equality for $S = R$. This now follows by some straightforward calculations; we will explicitly show the inequality for all K, but a careful review shows that for $S = R$, the same proof goes through with all inequalities as equalities.

For a given full component K, note that $\rho(K \cap S) = \sum_i \rho(K \cap S \cap \tau_i) + \texttt{rc}^{\tau}_{K \cap S}$, and so we may decompose

$$\sum_{K \in \mathcal{K}(R)} y_K \rho(K \cap S) = \sum_{i=1}^{r} \sum_{K \in \mathcal{K}(R)} y_K \rho(K \cap S \cap \tau_i) + \sum_{K \in \mathcal{K}(R)} y_K \texttt{rc}^{\tau}_{K \cap S}. \tag{4.28}$$

For each i, the first term on the RHS of (4.28) simplifies as

$$\sum_{K\in\mathcal{K}(R)} y_K\rho(K\cap S\cap\tau_i) = \sum_{K\in\mathcal{K}(R)} (x|\tau_i-\delta_i)_K\rho(\tau_i\cap S\cap K) \le \rho(\tau_i\cap S), \tag{4.29}$$

where the last inequality holds since $(x|\tau_i)-\delta_i \in (\mathcal{E}\mathcal{P}')[\tau_i]$. Now we simplify the second term on the RHS of (4.28). Note that $\texttt{rc}^{\tau}_{K\cap S}$ is only positive when K spans more than one part of τ, but $\texttt{supp}(\delta_i)\subset\mathcal{K}(\tau_i)$, so $(\delta_i)_K\texttt{rc}^{\tau}_{K\cap S}=0$ for all i,K and S. It follows that

$$\sum_{K\in\mathcal{K}(R)} y_K\texttt{rc}^{\tau}_{K\cap S} = \sum_{K\in\mathcal{K}(R)} x_K\texttt{rc}^{\tau}_{K\cap S} = \sum_{K\in\mathcal{K}(R)} x_K\rho((K\cap S)/\tau) \tag{4.30}$$

$$\le \sum_{K'\in\mathcal{K}(R/\tau)} (x/\tau)_{K'}\rho(K'\cap(S/\tau)) \le \rho(S/\tau), \tag{4.31}$$

where the first inequality uses the fact that $\rho((K\cap S)/\tau)\le\rho((K/\tau)\cap(S/\tau))$ and the second holds since $x/\tau\in(\mathcal{P}')[R/\tau]$. Putting together equations (4.28), (4.29), (4.31), and Lemma 3.13 we get $\sum_K y_K\rho(K\cap S)\le\sum_{i=1}^{r}\rho(\tau_i\cap S)+\rho(S/\tau)=\rho(S)$, as needed.

4.8 Integrality Gaps and RatioGreedy

In this section we use $(\mathcal{P})$ to perform a novel dual fitting analysis of a greedy algorithm of Gröpl et al. [103] which we call RatioGreedy. This results in new $(\mathcal{P})$-relative approximation ratios for several classes of problems, and consequently new integrality gap bounds. Later within this section we give a new small example showing that RatioGreedy has approximation ratio at best $\frac{73}{60}$ on uniformly quasi-bipartite instances (Section 4.8.2), an alternate proof of the $\frac{73}{60}$ integrality gap upper bound for unit-cost quasibipartite instances (Section 4.8.3), and we show a lower bound of 8/7 for the integrality gap of hypergraphic LPs, which is the best known, due to Martin Skutella (Section 4.8.4).

We start by describing the algorithm of Gröpl et al. [103] in terms of full components.

Procedure RatioGreedy
1. Initialize the set of acyclic components $\mathcal{L}$ to $\varnothing$.
2. Let L^* be a minimizer of $\frac{C_L}{|L|-1}$ over all full components L such that $|L|\ge 2$ and $L\cup\mathcal{L}$ is acyclic.
3. Add L^* to $\mathcal{L}$.
4. Continue until $(R,\mathcal{L})$ is a hyper-spanning tree and return $\mathcal{L}$.

A quasi-bipartite instance of the Steiner tree problem is *uniformly quasi-bipartite* if for every Steiner node, all edges incident to that node have the same cost. This class was introduced by Gröpl et al. [103] who showed that the algorithm RatioGreedy is a

(non-LP-relative) $\frac{73}{60}$-approximation algorithm on these instances. We match this result and show RatioGreedy is a $(\mathcal{P})$-relative $\frac{73}{60}$-approximation.

For 3-restricted instances of the Steiner tree problem, we show RatioGreedy is a $(\mathcal{P})$-relative 5/4-approximation algorithm. We also outline how this approach extends to r-restricted instances for larger r.

We may also use RatioGreedy for arbitrary hypergraphs not derived from the Steiner tree problem. We restrict to down-closed instances with nondecreasing cost-functions (defined at the start of Section 3.4) since otherwise no approximation ratio is possible. In that case, where r denotes the maximum hyperedge size, we also get an integrality gap bound of $H(r-1)$ where $H(i)$ is the ith harmonic number. This complements the observation by Baudis et al. [14] that RatioGreedy has approximation ratio $H(r-1)$, which is in turn a generalization of the submodular set cover framework of Wolsey [199]. This is nearly best possible for $r \geq 4$ since "set cover with maximum set size k" to can be reduced to "spanning connected hypergraph with maximum edge size $k+1$" by creating a new root vertex and adding it to all sets. This set cover problem is APX-hard for $k \geq 3$ and Trevisan [188] showed $\ln k - O(\ln \ln k)$ inapproximability unless P=NP.

4.8.1 Integrality Gap Upper Bounds for Special Classes

Theorem 4.33. *On a uniformly quasibipartite instance,* $C(\text{RatioGreedy}) \leq \frac{73}{60}\text{OPT}(\mathcal{P})$.

Proof. Let t denote the number of iterations and $\mathcal{L} := \{L_1, \ldots, L_t\}$ be the ordered sequence of full components obtained. We now define a dual solution y to $(\mathcal{P}_D)$. Let $\pi(i)$ denote the partition induced by the connected components of $\{L_1, \ldots, L_i\}$. Let $\theta(i)$ denote $C_{L_i}/(|L_i|-1)$ and note that θ is nondecreasing. Define $\theta(0) = 0$ for convenience. We define a dual solution y with

$$y_{\pi(i)} = \theta(i+1) - \theta(i)$$

for $0 \leq i < t$, and all other coordinates of y set to zero. It is easy to see that $\sum_i y_{\pi(i)} r(\pi(i)) = C(\mathcal{L})$. We then show for any $K \in \mathcal{K}$, $\sum_i y_{\pi(i)} \mathtt{rc}_K^{\pi(i)} \leq 73/60 \cdot C_K$ — by scaling, this also proves that $\frac{60}{73}y$ is a feasible dual solution, and hence completes the proof.

Fix any $K \in \mathcal{K}$ and let $|K| = k$. Since the instance in question is uniformly quasibipartite, the full component K is a star with a Steiner centre and edges of a fixed cost c to each terminal in K. For $1 \leq i < k$, let $\tau(i)$ denote the last iteration j in which $\mathtt{rc}_K^{\pi(j)} \geq k - i$. Let K_i denote any subset of K of size $k-i+1$ such that K_i contains at most one element from each part of $\pi(\tau(i))$; i.e., $|K_i| = k-i+1$ and $\mathtt{rc}_{K_i}^{\pi(\tau(i))} = k-i$.

Our analysis hinges on the fact that K_i was a valid choice when $L_{\tau(i)+1}$ was chosen. More specifically, note that $\{L_1, \ldots, L_{\tau(i)}, K_i\}$ is acyclic, hence by the greedy nature of the

algorithm, for any $1 \le i < k$,

$$\theta(\tau(i)+1) = C_{L_{\tau(i)+1}}/(|L_{\tau(i)+1}|-1) \le C_{K_i}/(|K_i|-1) \le \frac{c\cdot(k-i+1)}{k-i}.$$

Moreover, using the definition of τ we compute

$$\begin{aligned}\sum_\pi y_\pi \mathtt{rc}_K^\pi &= \sum_{i=0}^{t-1} (\theta(i+1)-\theta(i))\mathtt{rc}_K^{\pi(i)} \\ &= \sum_{i=1}^{k-1} \theta(\tau(i)+1) \le \sum_{i=1}^{k-1} \frac{c\cdot(k-i+1)}{k-i} = c\cdot(k-1+H(k-1)).\end{aligned}$$

Finally, note that $(k-1+H(k-1)) \le \frac{73}{60}k$ for all $k \ge 2$ (achieved at $k=5$). Therefore, $\frac{60}{73}y$ is a valid solution to ($\mathcal{P}_D$). □

To get our other results, as above, we want to prove that the load of y on any full component K is at most a certain factor more than its cost, then we scale down the dual to become feasible.

Proposition 4.34. *On down-closed hypergraphs of maximum edge size r with non-decreasing costs, all K satisfy*

$$\sum_i y_{\pi(i)} \mathtt{rc}_K^{\pi(i)} \le H(r-1)\cdot C_K.$$

Proof. The argument is the same as in the proof of Theorem 4.33 except that we can only use the simple bound $C_{K_i} \le C_K$ in place of $C_{K_i} \le \frac{|K_i|-1}{|K|-1}C_K$. Hence $\theta(\tau(i)+1) \le C_K/(|K_i|-1)$ and

$$\sum_\pi y_\pi \mathtt{rc}_K^\pi = \sum_{i=1}^{k-1} \theta(\tau(i)+1) \le C_K \sum_{i=1}^{k-1} \frac{1}{k-i} = H(k-1)C_K \le H(r-1)C_K,$$

as needed. □

Proposition 4.35. *On 3-restricted instances, all K satisfy*

$$\sum_i y_{\pi(i)} \mathtt{rc}_K^{\pi(i)} \le \frac{5}{4}C_K.$$

Proof. Consider the case $k = |K| = 3$; the case $|K| = 2$ is easier. Without loss of generality the original instance is metric and so we may assume K has just three edges; let their costs be $x \le y \le z$.

Previously, K_2 denoted an arbitrary sub-full component of K such that $\texttt{rc}^{K_2}_{\pi(\tau(2))} = 1$; this time, take time K_2 to be a minimum-cost such sub-full component. The eligible choices for K_2 depend on which two of K's three terminals are connected by $\pi(\tau(2))$, but it is easy to see that at least one of the two sub-full components of cost $x + y$ or $x + z$ is eligible. Hence $C_{K_2} \leq x + z$, and $\theta(\tau(2) + 1) \leq (x + z)/1$.

Now we bound $\theta(\tau(1) + 1)$ similarly. In iteration $\tau(1) + 1$, K is an eligible choice for the algorithm, and so is the sub-full component consisting of the edges with cost x and y. Therefore $\theta(\tau(1) + 1) \leq \min\{(x + y + z)/2, x + y\}$.

Hence

$$\sum_{\pi} y_\pi \texttt{rc}^\pi_K = \sum_{i=1}^{k-1} \theta(\tau(i)+1) \leq x+z+\min\left\{\frac{x+y+z}{2}, x+y\right\} \leq x+z+\frac{3x+3y+z}{4} \leq \frac{5}{4}C_K,$$

as needed. □

Here is a sketch of how Proposition 4.35 extends to r-restricted instances for larger r — we defer the full details. We write a linear program to capture the best possible bound that can be built out of sub-full components, just as above. In addition, we observe that the restriction of $\{\pi(i)\}_i$ to K is a chain. Iterating over all possible shapes of full components, and all possible chains, we get integrality gap upper bounds $5/4 = 1.25, 11/8 = 1.375, 119/82 \approx 1.451, 3/2 = 1.5$ respectively for 3-restricted, 4-restricted, 5-restricted, and 6-restricted instances. We obtain similar, better, bounds if we know the original instance is quasi-bipartite: $5/4 = 1.25, 13/10 = 1.3, 37/28 \approx 1.321, 127/96 \approx 1.323$, and the latter bound holds also for 7- and 8-restricted quasi-bipartite instances. It appears for quasi-bipartite graphs, just as in the proof of Theorem 4.33, that the worst-case full components for this analysis have bounded size, and it would be interesting to verify this.

4.8.2 Small Example Showing 73/60 is Tight for RatioGreedy

In our studies we found a smaller (non-unit-cost) quasi-bipartite graph upon which RatioGreedy has approximation ratio $\frac{73}{60}$, which we show in Figure 4.6. There is an optimal Steiner tree of cost 10, obtained by taking all edges incident to n_1 and n_2. However, it is possible for RatioGreedy to pick the following full components in order for a total cost of $3 \cdot \frac{5}{6} + 3 \cdot \frac{8}{9} + 3 + 2 + 2 = 73/6$: first, all edges incident with n_A, since it has optimal cost-per-connection ratio of $5/4$; second, all edges incident with n_B, since then it has optimal cost-per-connection ratio of $4/3$; third, all edges incident with n_C, since then it has optimal cost-per-connection ratio of $3/2$; then two edges each incident to n_1 and n_2.

Other examples are known where RatioGreedy selects a Steiner tree with cost $\frac{73}{60}$ times the optimal cost — one with arbitrary costs in [101] and one with unit costs in [102] — but both are larger graphs than the one in Figure 4.6.

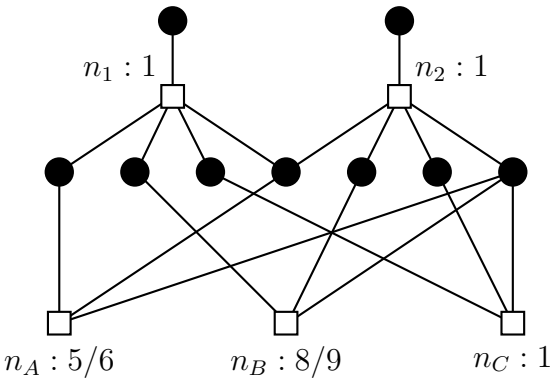


Figure 4.6: A small uniformly quasi-bipartite instance upon which the approximation ratio of RatioGreedy is 73/60. Dots are terminals and squares are Steiner vertices. The notation $n_i : c_i$ denotes that all edges incident with Steiner node n_i have cost c_i.

4.8.3 Alternate 73/60 Analysis for *Unit-cost* Quasibipartite Instances

In this section we show that in quasibipartite instances of the Steiner tree problem where all edges have unit cost, the integrality gap of the hypergraphic relaxation is at most $\frac{73}{60} \approx 1.216$. We do so by using the concept of *filtering* from [34] to analyze RatioGreedy. We already showed a more general result, Theorem 4.33; but we include the proof of the more specific result in this section because it obtains the bound $\frac{73}{60}$ through totally different means.

We actually assume that the graph is not only quasi-bipartite, but also that it has no terminal-terminal edges. (I.e., it is bipartite with bipartition $\{R, V \backslash R\}$.) We claim that this is without loss of generality on unit-cost graphs, but we omit the technical details required to prove this fact.

For unit-cost graphs, it is easy to see that RatioGreedy does the following: for each full component K in nondecreasing order of size, if adding K to $\mathcal{S}$ does not create a cycle, then we add K to $\mathcal{S}$. At the end, we have a Steiner tree T. Our filtering approach also shows there is a feasible dual solution y for ($\mathcal{P}_D$) such that $c(y) \geq \frac{60}{73} c(T)$.

By standard metricity arguments and the fact that G is unit-cost, bipartite and connected, we introduce without loss of generality a terminal spanning tree T^* consisting of cost-2 edges into G.

Let the variable w denote $|R| - 1$. Let s_6 denote the number of full components in $\mathcal{S}$ on 6 or more terminals, and e_6 denote the number of edges in these full components. For $i \in \{3, 4, 5\}$ let s_i denote the number of full components in $\mathcal{S}$ on exactly i terminals, and e_i denote the number of edges in these full components. Let T denote the final tree returned by our algorithm.

For each $k \in \{6, 5, 4, 3\}$, once no more full components of size $\geq k$ can be added by RatioGreedy, let $\mathcal{S}_k$ be the set of full components selected so far, and let T_k be the result of adding a spanning tree of $T^*/\mathcal{S}_k$ to $\mathcal{S}_k$; then T_k is a feasible hyper-spanning tree/Steiner tree. Our use of *filtering* is to consider T_k under a modified cost function $c^k \leq c$ that assigns cost $\frac{k-1}{k-2}$ to every terminal-terminal edge and cost 1 to every Steiner-terminal edge. Note T^3 is the output of the algorithm, and that $c^3 = c$.

The following lemma says that T_k is locally optimal under c^k.

Lemma 4.36. *There is no Steiner node n for which* $\mathtt{mst}(R \cup V(\mathcal{S}_k) \cup \{n\}, c^k) < \mathtt{mst}(R \cup V(\mathcal{S}_k), c^k) = c^k(T_k)$.

Proof. For a set N' of Steiner nodes, $\mathtt{mst}(R \cup N', c^k)$ can be computed in the following way. Say that two terminals are related if they are both adjacent to some $n \in N'$, and let $\pi(N')$ denote the transitive closure of this relation. Then any minimum spanning tree of $(R \cup N', c^k)$ has $r(\pi(N')) - 1$ terminal-terminal edges and $|R| + |N'| - r(\pi(N'))$ Steiner-terminal edges.

Let K be the set of all neighbours of n. Let σ be short for $\pi(V(\mathcal{S}^k))$. Note $r(\pi(V(\mathcal{S}^k) \cup \{n\})) = r(\sigma) - \mathtt{rc}^K_\sigma$. By definition of T_k and the greedy algorithm, $\mathtt{rc}^K_\sigma \leq k - 2$. The counting formula in the previous paragraph gives

$$\begin{aligned}\mathtt{mst}(R \cup V(\mathcal{S}_k) \cup \{n\}, c^k) - \mathtt{mst}(R \cup V(\mathcal{S}_k), c^k) &= -\mathtt{rc}^K_\sigma \cdot \frac{k-1}{k-2} + (1 + \mathtt{rc}^K_\sigma) \cdot 1 \\ &= 1 - \mathtt{rc}^K_\sigma/(k-2) \geq 0\end{aligned}$$

which proves the lemma. □

Now T_k is a min-cost spanning tree with respect to terminal set $R \cup V(\mathcal{S}_k)$ and the previous lemma shows it is gainless. Let $\mathtt{l}_k$ denote one edge from each full component in $\mathcal{S}_k$ — this corresponds to *loss* of these full components [178]. If we contract $\mathtt{l}_k$, T_k reduces to a tree with cost $c^k(T_k) - c^k(\mathtt{l}_k)$ but becomes a gainless MST (using Lemma 4.36), so by Corollary 3.28, the resulting contracted instance has $\mathrm{OPT}(\mathcal{P}) = c^k(T_k) - c^k(\mathtt{l}_k)$.

Let $\mathcal{L}$ denote $\mathrm{OPT}(\mathcal{P})$ on the original graph. Reducing costs and contracting can only reduce $\mathrm{OPT}(\mathcal{P})$, so the previous paragraph gives $\mathcal{L} \geq c^k(T_k) - c^k(\mathtt{l}_k)$. We simplify

$$c^k(T_k) = \frac{k-1}{k-2}(w - \sum_{j \geq k}(e_j - s_j)) + \sum_{j \geq k} e_j = \frac{k-1}{k-2}(w + \sum_{j \geq k} s_j) - \frac{1}{k-2}\sum_{j \geq k} e_j$$

and so

$$\mathcal{L} \geq c^k(T_k) - c^k(\mathtt{l}_k) = c^k(T_k) - \sum_{j \geq k} s_j = \frac{k-1}{k-2}w + \frac{1}{k-2}\sum_{j \geq k}(s_j - e_j).$$

By expanding the costs of the various trees, we obtain the following system of inequalities in the nonnegative variables $w, \{s_k, e_k\}_{3 \le k \le 6}, \mathcal{L}, w, c(T_3)$:

$$e_6 \ge 6s_6 \tag{4.32}$$
$$e_5 = 5s_5 \tag{4.33}$$
$$e_4 = 4s_4 \tag{4.34}$$
$$e_3 = 3s_3 \tag{4.35}$$
$$c(T_3) = 2(w + s_6 + s_5 + s_4 + s_3) - e_6 - e_5 - e_4 - e_3 \tag{4.36}$$
$$\mathcal{L} \ge \frac{5}{4}w + \frac{1}{4}(s_6 - e_6) \tag{4.37}$$
$$\mathcal{L} \ge \frac{4}{3}w + \frac{1}{3}(s_6 + s_5 - e_6 - e_5) \tag{4.38}$$
$$\mathcal{L} \ge \frac{3}{2}w + \frac{1}{2}(s_6 + s_5 + s_4 - e_6 - e_5 - e_4) \tag{4.39}$$
$$\mathcal{L} \ge 2w + s_6 + s_5 + s_4 + s_3 - e_6 - e_5 - e_4 - e_3 \tag{4.40}$$

Now we bound $\{c(T_3)/\mathcal{L}\}$ subject to these constraints. Since all inequalities are linear homogeneous, when we relax the integral variables to be continuous, we can compute the maximum value of this ratio by fixing $\mathcal{L} = 1$ and solving an LP that maximizes $c(T_3)$. The resulting LP has optimal value $\frac{73}{60}$ and optimal solution set, where $t > 0$ is a parameter,

$$w = 48t, \mathcal{L} = 60t, c(T_3) = 73t, f_3 = 6t, f_4 = 4t, f_5 = 3t, f_6 = 0.$$

To explicitly certify this by the dual, take the linear combination of equations

$$\frac{73}{360}(4.32) + \frac{1}{4}(4.33) + \frac{1}{3}(4.34) + \frac{1}{2}(4.35) - (4.36) + \frac{2}{15}(4.37) + \frac{1}{4}(4.38) + \frac{1}{3}(4.39) + \frac{1}{2}(4.40)$$

which is precisely $c(T_3) \le \frac{73}{60}\mathcal{L} - \frac{e_6}{72}$. In other words, this proves that RatioGreedy is an ($\mathcal{P}$)-relative $\frac{73}{60}$-approximation algorithm.

We remark that filtering over additional stages cannot reduce the approximation guarantee; Gröpl et al. [103] gave an example of a unit-cost quasi-bipartite graph where RatioGreedy has approximation ratio precisely $\frac{73}{60}$.

4.8.4 Lower Bound of 8/7 on Integrality Gap of Hypergraphic LPs

As reported by Agarwal & Charikar [1], Goemans gave a family of graphs upon which, in the limit, the integrality gap of the bidirected cut relaxation is $\frac{8}{7}$. Interestingly, it can

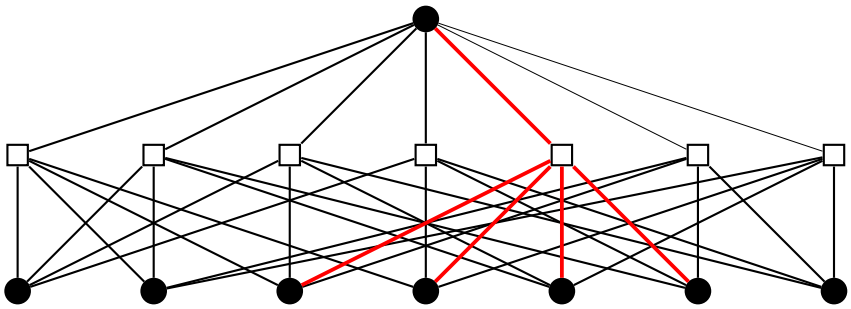


Figure 4.7: Skutella's example, which shows that the bidirected cut formulation and our new formulation both have a gap of at least $\frac{8}{7}$. The shaded edges denote one of the quasi-bipartite full components on 5 terminals.

be shown that once you preprocess these graphs, the gap completely disappears. Here we describe another example, due to Skutella [183]. It shows not only that the gap of the bidirected cut relaxation is at least $\frac{8}{7}$, but that the gap of our new formulation (including preprocessing) is at least $\frac{8}{7}$. The example is quasi-bipartite.

The Fano design is a well-known finite geometry consisting of 7 *points* and 7 *lines*, such that every point is on 3 lines, every line contains 3 points, any two lines meet in a unique point, and any two points lie on a unique common line. We construct Skutella's example by creating a bipartite graph, with one side consisting of one vertex n_p for each point p of the Fano design, and the other side consisting of one vertex n_ℓ for each line ℓ of the Fano design. Define n_p and n_ℓ to be adjacent in our graph if and only if p does *not* lie on ℓ. Then it is easy to see this graph is 4-regular, and that given any two vertices n_1, n_2 from one side, there is a vertex from the other side that is adjacent to neither n_1 nor n_2. Let one side be terminals, the other side be Steiner vertices, and then attach one additional terminal to all the Steiner vertices. Assign each edge unit cost. We illustrate the resulting graph in Figure 4.7.

Each Steiner vertex is in a unique 5-terminal quasi-bipartite full component. There are 7 such full components. Denote the family of these 7 full components by $\mathcal{C}$.

Claim 4.37. *Let $x^*_K = \frac{1}{4}$ for each $K \in \mathcal{C}$, and $x^*_K = 0$ otherwise. Then x^* is feasible for ($\mathcal{P}$).*

Proof. It is immediate that x^* satisfies constraints (5.2). It remains only to show that x^* meets constraint (5.1). Let $\pi = (V_0, V_1, \ldots, V_m)$ be an arbitrary partition such that V_0 contains the extra *top* terminal. If we can show that $\sum_K x^*_K \mathtt{rc}^\pi_K \geq m$ then we will be done, since π was arbitrary. For each $i = 1, \ldots, m$, let t_i be any terminal in V_i. Note that each t_i lies in exactly 4 full components from $\mathcal{C}$. Furthermore, every full component $K \in \mathcal{C}$

satisfies $\texttt{rc}_K^\pi \geq |K \cap \{t_1, \ldots, t_m\}|$, as K meets V_0 as well as each part V_j for which $t_j \in K$. Hence

$$\sum_{K \in \mathcal{C}} x_K^* \texttt{rc}_K^\pi = \frac{1}{4} \sum_{K \in \mathcal{C}} \texttt{rc}_K^\pi \geq \frac{1}{4} \sum_{K \in \mathcal{C}} |\{j : t_j \in K\}| = \frac{1}{4} \sum_{j=1}^{m} |\{K \in \mathcal{C} : t_j \in K\}| = \frac{1}{4} \cdot m \cdot 4 = m.$$

□

The objective value of x^* is $\frac{35}{4}$, but the optimal integral solution to the LP is 10, since at least 3 Steiner vertices need to be included. Hence, the gap of our new LP is no better than $\frac{10}{35/4} = \frac{8}{7}$.

4.9 A Dense Extreme Point for Bidirected Cut

The purpose of this section is to prove that there are extreme points for the bidirected cut LP relaxation (for the Steiner tree problem) which have $\Omega(|V|^2)$ nonzero variables. Likely the existence of this example is folklore, but as it has not appeared in print before, we explicitly describe it here. Note it is closely related to the hypergraphic extreme points mentioned in Example 3.19.

Let k be an integer parameter. We will construct a graph with $k+1$ terminals, denoted $\{r_0, r_1, \ldots, r_k\}$, and k Steiner vertices, denoted $\{n_1, \ldots, n_k\}$. The edges of this graph are all edges of the form $\{r_i, n_j\}$ where $i \neq j$. All edges have cost 1. We depict the graph in the figure below.

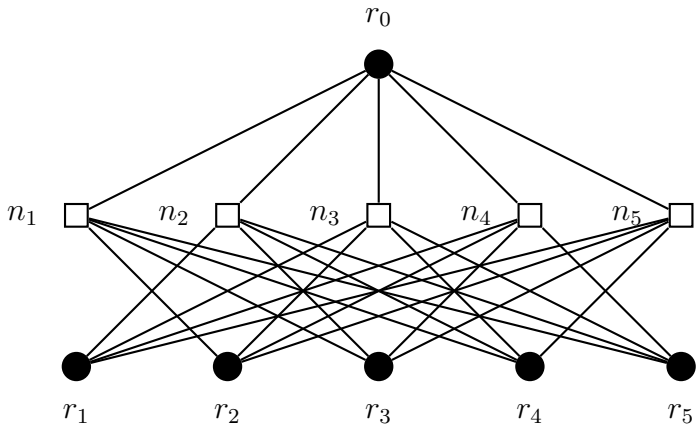


Figure 4.8: The graph used in our construction when $k = 5$. Dots are terminals and squares are Steiner vertices. Each edge has cost 1.

We choose the bidirected formulation which uses r_0 as the root and has one variable for each arc $a \in A := \{(u,v) \mid \{u,v\} \in E\}$. The LP is then

$$\min\{c \cdot x \mid x \geq 0, \delta^-(x(S)) \geq 1 \, \forall S \subseteq V \backslash \{r_0\} \text{ s.t. } V \cap \{r_1, \ldots, r_k\} \neq \varnothing\}$$

where we extend the definition of c to arcs via $c_{(u,v)} = c_{\{u,v\}}$.

Let x^* denote the solution which assigns value $1/(k-1)$ to all arcs of the form $\{(r_0, n_i) \mid 1 \leq i \leq k\}$ and $\{(n_i, r_j) \mid 1 \leq i, j \leq k, i \neq j\}$, and value 0 to all other arcs. (Informally, the "downwards" arcs in the diagram get value $1/(k-1)$, and the "upwards" ones get value 0.)

In the following claims, we will show that x^* is an extreme feasible solution for the bidirected cut relaxation. Notice that x^* is nonzero on k^2 arcs, and that there are $2k+1$ vertices; i.e. the support of x^* is of size $\Omega(|V|^2)$, which was what we wanted to show. Now we give the supporting claims.

Claim 4.38. *x^* is a feasible solution to the bidirected cut relaxation.*

Proof. We will show for each nonroot terminal r_i, there is a flow f of value 1 from r_0 to r_1 such that x^* dominates f (i.e., such that $x^* \geq f$); then standard arguments based on the max-flow min-cut theorem complete the proof. In particular, let f take value $1/(k-1)$ on all arcs of the form $(r_0, n_j), (n_j, r_i)$ for $j \neq i$, and value 0 elsewhere. It is easy to check that f is the desired flow. □

Notice that the objective value of x^* is $k^2/(k-1)$.

Claim 4.39. *x^* is an optimal solution to the bidirected cut relaxation.*

Proof. The dual LP to the bidirected cut formulation, which has a variable y_S for each set $S \subseteq V \backslash \{r_0\}$ such that $V \cap \{r_1, \ldots, r_k\} \neq \varnothing$, is

$$\min\{1 \cdot y \mid y \geq 0, \sum_{S : e \in \delta^-(S)} y_S \leq c_a \forall a \in A\}.$$

We will give a feasible dual solution y^* of objective value $k^2/(k-1)$, which by weak duality establishes that x^* is optimal.

Namely, let y^* take value $1/(k-1)$ on every set S of the form $\{r_i\}$ for $1 \leq i \leq k$, value $1/(k-1)$ on every set S of the form $\{n_j, r_i\}$ for $1 \leq i, j \leq k$ such that $i \neq j$, and value 0 elsewhere. To check feasibility of y^*, we observe that

$$\sum_{S : e \in \delta^-(S)} y^*_S = \begin{cases} 1, & \text{for } e \text{ of the form } r_0 n_i \text{ where } 1 \leq i \leq k \\ 1, & \text{for } e \text{ of the form } n_i r_j \text{ where } 1 \leq i, j \leq k \text{ and } i \neq j \\ (k-2)/(k-1), & \text{for } e \text{ of the form } r_j n_i \text{ where } 1 \leq i, j \leq k \text{ and } i \neq j \\ 0, & \text{for } e \text{ of the form } n_i r_0 \text{ where } 1 \leq i \leq k. \end{cases}$$

Since y^* assigns value $1/(k-1)$ to each of k^2 sets, its objective value is $k^2/(k-1)$ as claimed. □

Claim 4.40. *x^* is the unique optimal solution to the bidirected cut relaxation, and hence is an extreme point solution.*

Proof. Let x' be some optimal primal solution. By complementary slackness applied to the solution y^* given in the previous proof, we know that x' assigns positive value only to the "downwards" arcs.

Let N denote the set of all Steiner vertices. For any $1 \leq i \leq k$, consider the constraint for the set $S_i := N \backslash \{n_i\} \cup \{r_i\}$. Since x' is zero on all upwards arcs, this implies that $\sum_{j \neq i} x'_{(r_0,n_j)} \geq 1$. We now have a case analysis:

- if $x'_{(r_0,n_j)} > 1/(k-1)$ for some j, then $c \cdot x' \geq x'_{(r_0,n_j)} + x'(\delta^-(S_j)) + \sum_{i=1}^{k} x'(\delta^-(r_i)) > k^2/(k-1)$, contradicting the optimality of x'
- given the previous point, if $x'_{(r_0,n_j)} < 1/(k-1)$ for some j, then $x(\delta(S_i)) < 1$ for each $i \neq j$, contradicting the feasibility of x'
- so $x'_{(r_0,n_j)} = 1/(k-1)$ for all $1 \leq j \leq k$.

Now consider any arc of the form (n_i, r_j) for $i \neq j$. Let W denote the set $N \backslash \{n_i, n_j\} \cup \{r_j\}$. We have that $x'(\delta^-(W)) = x'_{(n_i,r_j)} + \sum_{\ell \neq i,j} x'_{(r_0,n_\ell)}$. Since each $x'_{(r_0,n_\ell)} = 1/(k-1)$ and x' is feasible, we see that

$$x'_{(n_i,r_j)} = x'(\delta^-(W)) - \sum_{\ell \neq i,j} x'_{(r_0,n_\ell)} \geq 1 - (k-2)/(k-1) = 1/(k-1).$$

By applying this argument to all such i and j, we see that x' dominates x^*. By the positivity of the LP cost function c, and the fact that x^* is feasible and x' is optimal, it follows that $x' = x^*$. □

Acknowledgement

I would like to thank Bill Cunningham for a discussion that was helpful in proving Theorem 4.1, and my thesis committee for noticing that we happened to be using the dual of ($\mathcal{D}$) in the proof of Section 4.5.

Chapter 5

LP Interpretations of Robins-Zelikovsky and Relative Greedy

The best approximation algorithm known for the Steiner tree problem is a combinatorial algorithm due to Robins and Zelikovsky [178]; it achieves a performance guarantee of $1 + \frac{\ln 3}{2} \approx 1.55$. The best known linear programming (LP)-based algorithms, on the other hand, [93, 2, 124] do not obtain any constant approximation ratio less than 2.

The main focus of this chapter is a link between greedy and LP-based approaches: we show that Robins and Zelikovsky's algorithm can be viewed as an iterated primal-dual algorithm with respect to generalization of the LP relaxation ($\mathcal{P}$). An instance is *b-quasi-bipartite* if each connected component of $G \backslash R$ has at most b vertices. We also show that Robins' and Zelikovsky's algorithm has an approximation ratio better than $1 + \frac{\ln 3}{2}$ for such instances, and we prove that the integrality gap of the LP ($\mathcal{P}$) is between $\frac{8}{7}$ and $\frac{2b+1}{b+1}$.

In the final section of the chapter (Section 5.6) we show that the *relative greedy* $(1+\ln 2)$-approximation algorithm can be viewed as LP-relative. The LP is derived from earlier work [199] on the submodular set cover problem; it is not clear if it can be optimized in polynomial time.

5.1 Introduction

In this chapter we provide algorithmic evidence that the primal-dual method is useful for the Steiner tree problem. We first present a novel LP relaxation for the Steiner tree

problem that generalizes the LP ($\mathcal{P}$) presented earlier. We then show that the algorithm RZ of Robins and Zelikovsky can be analyzed as an iterated primal-dual algorithm using this relaxation.

In [178], Robins and Zelikovsky showed that, for every fixed r, the performance ratio of RZ is $1.279\rho_r$ in quasi-bipartite graphs, and $1.55\rho_r$ in general graphs. We prove an interpolation of these results. For a Steiner vertex v, define its *Steiner neighbourhood* to be the collection of vertices that are in the same connected component as v in $G\backslash R$. A graph is *b-quasi-bipartite* if all of its Steiner neighbourhoods have cardinality at most b. We prove the following, where $\texttt{opt}_r$ denotes the optimal cost after r-preprocessing:

Theorem 5.1. *Given an undirected, b-quasi-bipartite graph $G = (V, E)$, terminals $R \subseteq V$, and a fixed constant $r \geq 2$, Algorithm* RZ *returns a feasible Steiner tree T s.t.*

$$c(T) \leq \begin{cases} 1.279 \cdot \texttt{opt}_r & : \quad b = 1 \\ (1 + \frac{1}{e}) \cdot \texttt{opt}_r & : \quad b \in \{2, 3, 4\} \\ \left(1 + \frac{1}{2} \ln\left(3 - \frac{2}{b}\right)\right) \texttt{opt}_r & : \quad b \geq 5. \end{cases}$$

Note that b-quasi-bipartite graphs are a natural interpolation between quasi-bipartite graphs ($b = 1$) and general graphs ($b \leq |V\backslash R|$), hence Theorem 5.1 interpolates the two main results of Robins and Zelikovsky [178].

Unfortunately, Theorem 5.1 does not imply that our new relaxation has a small integrality gap. Nonetheless, we obtain the following bounds, when G is b-quasi-bipartite:

Theorem 5.2. *Our new relaxation has an integrality gap between $\frac{8}{7}$ and $\frac{2b+1}{b+1}$.*

We remark that *filtering* approach of Chakrabarty et al. [34], can be applied to improve the gap upper bound to $\frac{2b-1}{b}$ for $b \geq 2$ [135].

5.1.1 A New LP Relaxation for Steiner Trees

In this chapter, we work on r-preprocessed graphs as introduced in Chapter 2 — so for example $\mathcal{K}$ represents element-disjoint full components, one for every possible set of up to r terminals.

We also use a type of metric assumption which holds without loss of generality. The standard one is that the costs satisfy the triangle inequality and G is a complete graph. Here we instead work under a weaker metric assumption that is also without loss of generality for the purposes of computing optimal Steiner trees. The assumption is that, for every triple u, v, w of nodes with v a Steiner node such that $uv, vw \in E$, we have $uw \in E$ and $c_{uw} \leq c_{uv} + c_{vw}$. This preserves the Steiner neighbourhoods (unlike the standard metric

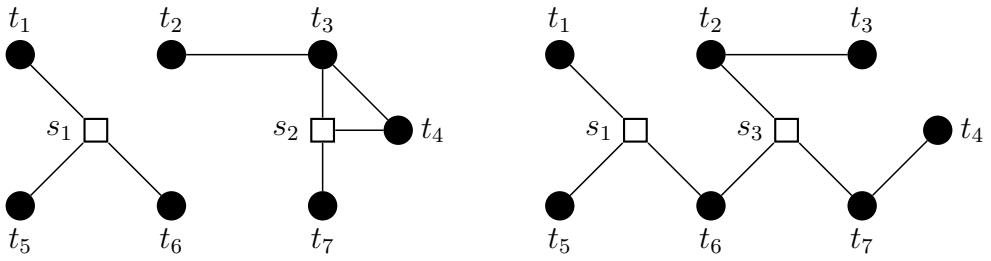


Figure 5.1: Left: a collection $\mathcal{S} = \{\{t_1, t_5, t_6\}, \{t_3, t_4, t_7\}, \{t_2, t_3\}, \{t_3, t_4\}\}$ of 4 full components. Right: a Steiner tree with $\mathcal{S}$-decomposition $(\{t_1s_1, t_5s_1, t_6s_1, t_2t_3\}, \{\{t_2, t_6, t_7\}, \{t_4, t_7\}\})$.

assumption) and has the important property that for every K, we may assume that the min-cost full component (tree) with leaf set K has degree at least 3 in all internal (Steiner) nodes. Consequently, for example, every r-restricted instance is also $(r-2)$-quasi-bipartite.

Whenever $\mathcal{S} \subset \mathcal{K}$, we will abuse notation and identify the symbol $\mathcal{S}$ with the graph obtained by deleting everything except the full components in $\mathcal{S}$. For example, $E(\mathcal{S})$ denotes the set of all edges of the full components in $\mathcal{K}$. We will always have $\binom{R}{2} \subset \mathcal{S}$, therefore the graph $\mathcal{S}$ always contains the induced subgraph $G[R]$.

Let $\mathcal{K}(T)$ denote the set of all full components of a Steiner tree T. For an arbitrary subfamily $\mathcal{S}$ of the full components $\mathcal{K}$, our new LP uses the following canonical decomposition of a Steiner tree into elements of $E(\mathcal{S})$ and $\mathcal{K}\backslash\mathcal{S}$.

Definition 5.3. *If T is an r-restricted Steiner tree, its* $\mathcal{S}$-decomposition *is the pair*

$$(E(T) \cap E(\mathcal{S}), \mathcal{K}(T)\backslash\mathcal{S}).$$

Figure 5.1 illustrates the $\mathcal{S}$-decomposition of a Steiner tree. Observe that after $\mathcal{S}$-decomposing a Steiner tree T we have

$$\sum_{e \in E(T) \cap E(\mathcal{S})} c_e + \sum_{K \in \mathcal{K}(T)\backslash\mathcal{S}} C_K = c(T).$$

We hence obtain a new higher-dimensional view of the Steiner tree polyhedron. Define

$$\mathtt{ST}^{\mathcal{S}}_{G,R} := \mathtt{conv}\{x \in \{0,1\}^{E(\mathcal{S})} \times \{0,1\}^{\mathcal{K}\backslash\mathcal{S}} \quad : \quad \exists \text{ a Steiner tree } T \text{ s.t. } x \text{ is the incidence vector of the } \mathcal{S}\text{-decomposition of } T\}.$$

The following definitions are used to generalize Steiner partition inequalities to use full components. We use $\Pi^{\mathcal{S}}$ to denote the family of all partitions of $V(\mathcal{S}) \cup R$. Let

$\pi = \{V_1, \ldots, V_p\} \in \Pi^{\mathcal{S}}$ be a partition of the set $R \cup V(\mathcal{S})$. The *Steiner rank* $\bar{r}(\pi)$ of π is defined as

$$\bar{r}(\pi) := \{\text{the number of parts of } \pi \text{ that contain terminals}\}.$$

For example, where $\mathcal{S}$ denotes the collection of full components on the left side of Figure 5.1, consider the partition $\pi = \{\{t_1, t_5, s_1\}, \{s_2\}, \{t_6, t_7\}, \{t_2, t_3\}, \{t_4\}\} \in \Pi^{\mathcal{S}}$. Its rank is $r(\pi) = 5$ but its Steiner rank is $\bar{r}(\pi) = 4$. The rank contribution of full component $K = \{t_2, t_6, t_7\}$ is $\texttt{rc}^{\pi}_K = 1$.

We describe below a new LP relaxation ($\mathcal{P}^{\mathcal{S}}$) of $\texttt{ST}^{\mathcal{S}}_{G,R}$. The relaxation has a variable x_e for each $e \in E(\mathcal{S})$ and a variable x_K for each $K \in \mathcal{K}\backslash\mathcal{S}$. For a partition $\pi \in \Pi^{\mathcal{S}}$, we define $E_\pi(\mathcal{S})$ to be the edges of $\mathcal{S}$ whose endpoints lie in different parts of π, i.e., $E_\pi(\mathcal{S}) = E(\mathcal{S}) \cap E_\pi$.

$$\begin{aligned}
\min \quad & \sum_{e \in E(\mathcal{S})} c_e \cdot x_e + \sum_{K \in \mathcal{K}\backslash\mathcal{S}} C_K \cdot x_K && && (\mathcal{P}^{\mathcal{S}})\\
\text{s.t} \quad & \sum_{e \in E_\pi(\mathcal{S})} x_e + \sum_{K \in \mathcal{K}\backslash\mathcal{S}} \texttt{rc}^{\pi}_K \cdot x_K \geq \bar{r}(\pi) - 1 && \forall \pi \in \Pi^{\mathcal{S}} && (5.1)\\
& x_e, x_K \geq 0 && \forall e \in E(\mathcal{S}), K \in \mathcal{K}\backslash\mathcal{S} && (5.2)
\end{aligned}$$

We remark that when $\mathcal{S} = \binom{R}{2}$, the LP ($\mathcal{P}^{\mathcal{S}}$) is the same as the LP ($\mathcal{P}$) studied in earlier chapters — to see this note that in this case $E(\mathcal{S})$ corresponds to the terminal-terminal edges which are precisely the full components of size 2, and $\mathcal{K}\backslash\mathcal{S}$ corresponds to all other full components.

Its LP dual has a variable y_π for each partition $\pi \in \Pi^{\mathcal{S}}$:

$$\begin{aligned}
\max \quad & \sum_{\pi \in \Pi^{\mathcal{S}}} (\bar{r}(\pi) - 1) \cdot y_\pi && && (\mathcal{P}^{\mathcal{S}}_D)\\
\text{s.t} \quad & \sum_{\pi \in \Pi^{\mathcal{S}} : e \in E_\pi(\mathcal{S})} y_\pi \leq c_e && \forall e \in E(\mathcal{S}) && (5.3)\\
& \sum_{\pi \in \Pi^{\mathcal{S}}} \texttt{rc}^{\pi}_K \cdot y_\pi \leq C_K && \forall K \in \mathcal{K}\backslash\mathcal{S} && (5.4)\\
& y_\pi \geq 0 && \forall \pi \in \Pi^{\mathcal{S}} && (5.5)
\end{aligned}$$

We conclude this section with a proof that the (primal) LP is indeed a relaxation of the convex hull of $\mathcal{S}$-decompositions for r-restricted Steiner trees. The inequalities (5.2) are obviously valid for $\texttt{ST}^{\mathcal{S}}_{G,R}$.

Lemma 5.4. *The inequalities (5.1) are valid for* $\mathtt{ST}^{\mathcal{S}}_{G,R}$.

Proof. Let T be a Steiner tree with $\mathcal{S}$-decomposition $(E(T) \cap E(\mathcal{S}), \mathcal{K}(T) \backslash \mathcal{S})$, and let $x \in \mathtt{ST}^{\mathcal{S}}_{G,R}$ be the corresponding incidence vector. Fix an arbitrary partition $\pi \in \Pi^{\mathcal{S}}$; we will now argue that the left-hand side of (5.1) for π is at least $\bar{r}(\pi) - 1$.

In order to do that we successively modify the given partition π by merging some of its parts. Initially, let $\hat{\pi} = \pi$. For each each edge uv of $E(T) \cap E(\mathcal{S})$, merge the part of $\hat{\pi}$ containing u and that containing v; if both endpoints lie in the same part of $\hat{\pi}$, the partition remains unchanged. Subsequently, consider each $K \in \mathcal{K}(T) \backslash \mathcal{S}$, and merge all parts of $\hat{\pi}$ meeting any terminal of K.

Initially, $\hat{\pi}$ has Steiner rank $\bar{r}(\pi)$, and its final Steiner rank is 1 since T connects all terminals. The Steiner rank drop of $\hat{\pi}$ due to any edge $e \in E_{\pi}(\mathcal{S})$ with $x_e = 1$ is clearly at most 1. For any other edge $e \in E(T) \cap E(\mathcal{S})$, since the endpoints of e are in the same part of π, the Steiner rank drop of $\hat{\pi}$ due to e is 0. Similarly, the Steiner rank drop of $\hat{\pi}$ due to $K \in \mathcal{K}(T) \backslash \mathcal{S}$ is at most $\mathtt{rc}^{\pi}_{K}$. This shows that x satisfies constraint (5.1). As T and π were chosen arbitrarily, the lemma follows. □

5.2 An Iterated Primal-Dual Algorithm for Steiner Trees

As described in Section 3.6.1, $\mathtt{MST}(G, c)$ denotes a call to Kruskal's minimum-spanning tree algorithm on graph G with cost-function c. It returns a minimum-cost spanning tree T and an optimal feasible dual solution y for $(\mathcal{M}_D)$. Let $\mathtt{mst}(G, c)$ denote the cost of $\mathtt{MST}(G, c)$. Since c is fixed, in the rest of the chapter we omit c where possible for brevity. Recall that we abuse notation and identify each set $\binom{R}{2} \subset \mathcal{S} \subset \mathcal{K}$ of full components with the graph $(V(\mathcal{S}), E(\mathcal{S}))$. Since $\mathcal{S} = (V(\mathcal{S}), E(\mathcal{S}))$ is connected and spans all terminals, $\mathtt{MST}(\mathcal{S})$ is a Steiner tree; namely, the one produced by running the MST heuristic on original instance after modifying the Steiner nodes of $\mathcal{S}$ to be terminals.

The reason that MST is useful in our primal-dual framework is that we can relate the dual $(\mathcal{M}_D)$ on graph $\mathcal{S}$ to the dual $(\mathcal{P}^{\mathcal{S}}_D)$. Let y be the dual returned by a call to $\mathtt{MST}(\mathcal{S})$. We treat y as a dual solution of $(\mathcal{P}^{\mathcal{S}}_D)$; note that constraints $(\mathcal{M}_D.1)$ and $(\mathcal{M}_D.2)$ of $(\mathcal{M}_D)$ imply that y also meets constraints (5.3) and (5.5) of $(\mathcal{P}^{\mathcal{S}}_D)$. If K is a full component such that (5.4) does not hold for y, we say that K is *violated* by y.

The idea behind our presentation is to find a set $\binom{R}{2} \subset \mathcal{S} \subset \mathcal{K}$ of full components such that $\mathtt{MST}(\mathcal{S})$ is a Steiner tree with small cost relative to optimal. The primal-dual algorithm finds such a set $\mathcal{S}$ in an iterative fashion. Initially, $\mathcal{S}$ is equal to $\binom{R}{2}$. In each iteration, we compute a minimum-cost spanning tree T of the graph $\mathcal{S}$. The dual solution

y corresponding to this tree is converted to a dual for $(\mathcal{P}_D^{\mathcal{S}})$, and if y is feasible for $(\mathcal{P}_D^{\mathcal{S}})$, we stop. Otherwise, we add a violated full component to $\mathcal{S}$ and continue. The algorithm clearly terminates (as $\mathcal{K}$ is finite) and at termination, it returns the final tree T as an approximately-optimum Steiner tree.

A key property of this approach is Lemma 5.5: adding a violated full component to $\mathcal{S}$ decreases $\texttt{mst}$. Hence $\texttt{MST}$, initially an infeasible dual, is decreased in value over the iterations but eventually becomes feasible.

Lemma 5.5. *If $(T, y) = \texttt{MST}(\mathcal{S})$ and K is violated by y, then $\texttt{mst}(\mathcal{S} \cup \{K\}) < \texttt{mst}(\mathcal{S})$.*

Proof. First, $\texttt{mst}(\mathcal{S}/K) \cup \{K\}$ is a spanning tree of $\mathcal{S} \cup \{K\}$, so $\texttt{mst}(\mathcal{S} \cup \{K\}) \leq \texttt{mst}(\mathcal{S}) + C_K$. Second, if K is violated by y, then Lemma 3.26 (the dual interpretation of gain) shows that $\texttt{mst}(\mathcal{S}/K) + C_K < \texttt{mst}(\mathcal{S})$. Together, this implies the lemma. □

Algorithm 5.1 summarizes the above description. Many combinatorial greedy algorithms for the Steiner tree problem follow this type of framework and differ only in how K is selected in each iteration, i.e., in the selection function $f_i : \mathcal{K} \to \mathbf{R}$. See [101, §1.4] for a well-written survey.

Algorithm 5.1 A general iterative primal-dual framework for Steiner trees.

Given: Undirected graph $G = (V, E)$, non-negative costs c_e for all edges $e \in E$, constant $r \geq 2$.
$\mathcal{S}^0 := \binom{R}{2}$, $i := 0$
repeat
  $(T^i, y^i) := \texttt{MST}(\mathcal{S}^i)$
  if y^i is not feasible for $(\mathcal{P}_D^{\mathcal{S}^i})$ **then**
    Choose a violated full component $K^i \in \mathcal{K} \backslash \mathcal{S}^i$ such that $f_i(K^i)$ is minimized
    $\mathcal{S}^{i+1} := \mathcal{S}^i \cup \{K^i\}$
  end if
  $i := i + 1$
until y^{i-1} is feasible for $(\mathcal{P}_D^{\mathcal{S}^{i-1}})$
Let $p = i - 1$ and return (T^p, y^p).

In the typical primal-dual approach [96, 190] dual feasibility is maintained and primal feasibility happens only at the end. This is true in $\texttt{MST}$ relative to $(\mathcal{M}_D)$, however if you consider the entirety of Algorithm 5.1 relative to our new LPs, we obtain a primal feasible solution in each iteration but attain dual feasibility only in the final iteration; more specifically the objective value of y^i decreases as i increases (see Lemma 5.17). We remark that the recent $\frac{4}{3}$-approximation algorithm of Chakrabarty et al. [34] for quasi-bipartite instances uses a qualitatively similar approach.

5.2.1 Cutting Losses: the RZ Selection Function

A potential weak point in Algorithm 5.1 is that once a full component is added to $\mathcal{S}$, it is never removed. On the other hand, if some cheap subgraph H connects all Steiner vertices of $\mathcal{S}$ to terminals, then adding H to any Steiner tree gives us a tree that spans $V(\mathcal{S})$, i.e., we have so far *lost* at most $c(H)$ in the final answer. This leads to the concept of the *loss* of a Steiner tree which was first introduced by Karpinski and Zelikovsky in [127].

Definition 5.6. *Let $G' = (V', E')$ be a subgraph of G. The* loss $\mathtt{L}(G')$ *is a minimum-cost set $E'' \subseteq E'$ such that every connected component of (V', E'') contains a terminal. Let $\mathtt{l}(G')$ denote the cost of $\mathtt{L}(G')$.*

See Figure 5.2 for an example of the loss of a graph. The above discussion amounts to saying that $\min\{\mathtt{mst}(\mathcal{S}') \mid \mathcal{S}' \supseteq \mathcal{S}\} \leq \mathtt{opt}_r + \mathtt{l}(\mathcal{S})$. Consequently, our selection function f_i in step 6 of the algorithm should try to keep the loss small. The following fact holds because full components in $\mathcal{K}$ meet only at terminals.

Fact 5.7. *If $\mathcal{S} \subseteq \mathcal{K}$, then $\mathtt{L}(\mathcal{S}) = \cup_{K \in \mathcal{S}} \mathtt{L}(K)$ and so $\mathtt{l}(\mathcal{S}) = \sum_{K \in \mathcal{S}} \mathtt{l}(K)$.*

For a set $\mathcal{S}$ of full components, where y is the dual solution returned by $\mathtt{MST}(\mathcal{S})$, define

$$\overline{\mathtt{mst}}(\mathcal{S}) := \sum_{\pi \in \Pi^{\mathcal{S}}} (\bar{r}(\pi) - 1) y_\pi. \tag{5.6}$$

If y is feasible for $(\mathcal{P}_D^{\mathcal{S}})$ then by weak LP duality, $\overline{\mathtt{mst}}(\mathcal{S})$ provides a lower bound on $\mathtt{opt}_r$. If y is infeasible for $(\mathcal{P}_D^{\mathcal{S}})$, then which full component should we add? Robins and Zelikovsky propose minimizing the ratio of the added loss to the change in potential lower bound (5.6). Their selection function f_i is defined by

$$f_i(K) := \frac{\mathtt{l}(K)}{\overline{\mathtt{mst}}(\mathcal{S}^i) - \overline{\mathtt{mst}}(\mathcal{S}^i \cup \{K\})} = \frac{\mathtt{l}(\mathcal{S}^i \cup \{K\}) - \mathtt{l}(\mathcal{S}^i)}{\overline{\mathtt{mst}}(\mathcal{S}^i) - \overline{\mathtt{mst}}(\mathcal{S}^i \cup \{K\})}, \tag{5.7}$$

where the equality uses Fact 5.7.

5.3 Analysis

Fix an optimum r-Steiner tree T^*. There are several steps in proving the performance guarantee of Robins and Zelikovsky's algorithm, and they are encapsulated in the following result, whose complete proof appears in Section 5.5.

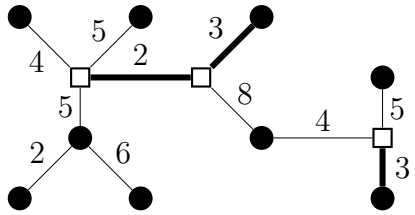


Figure 5.2: The figure shows a Steiner tree instance with costs on the edges. The loss of the Steiner tree in this figure is shown in thick edges; the loss has cost is 8.

Lemma 5.8. *The cost of the tree T^p returned by Algorithm 5.1 is at most*

$$\texttt{opt}_r + \texttt{l}(T^*) \cdot \ln\left(1 + \frac{\overline{\texttt{mst}}(G[R], c) - \texttt{opt}_r}{\texttt{l}(T^*)}\right).$$

The main observation in the proof of the above lemma can be summarized as follows: from the discussion in Section 2.1, we know that the tree T^p returned by Algorithm 5.1 has cost

$$\texttt{mst}(\mathcal{S}^p) = \sum_{\pi \in \Pi^{\mathcal{S}^p}} (r(\pi) - 1) y^p_\pi$$

and the corresponding lower-bound on $\texttt{opt}_r$ returned by the algorithm is

$$\overline{\texttt{mst}}(\mathcal{S}^p) = \sum_{\pi \in \Pi^{\mathcal{S}^p}} (\bar{r}(\pi) - 1) y^p_\pi.$$

We know that $\overline{\texttt{mst}}(\mathcal{S}^p) \leq \texttt{opt}_r$ but how large is the difference between $\texttt{mst}(\mathcal{S}^p)$ and $\overline{\texttt{mst}}(\mathcal{S}^p)$? We show that the difference

$$\sum_{\pi \in \Pi^{\mathcal{S}^p}} (r(\pi) - \bar{r}(\pi)) y^p_\pi$$

is exactly equal to the loss $\texttt{l}(T^p)$ of tree T^p — this is proved in Lemma 5.14. We then bound the loss of each selected full component K^i, and putting everything together finally yields Lemma 5.8.

The following lemma states the performance guarantee of Moore's minimum-spanning tree heuristic as a function of the optimum loss and the maximum cardinality b of any Steiner neighbourhood in G.

Lemma 5.9. *Fix an arbitrary optimum r-restricted Steiner tree T^*. Given an undirected, b-quasi-bipartite graph $G = (V, E)$, a set of terminals $R \subseteq V$, and non-negative costs c_e for all $e \in E$, we have*

$$\texttt{mst}(G[R], c) \leq 2\texttt{opt}_r - \frac{2}{b}\texttt{l}(T^*)$$

for any $b \geq 1$.

Proof. Recall that $\mathcal{K}(T^*)$ is the set of full components of tree T^*. Now consider a full component $K \in \mathcal{K}(T^*)$. We will now show that there is a minimum-cost spanning tree of $G[K]$ whose cost is at most $2C_K - \frac{2}{b}\mathtt{l}(K)$. By repeating this argument for all full components $K \in \mathcal{K}(T^*)$, adding the resulting bounds, and applying Fact 5.7, we obtain the lemma.

For terminals $r, s \in K$, let P_{rs} denote the unique r, s-path in K. Pick $u, v \in K$ such that $c(P_{uv})$ is maximal. Define the *diameter* $\Delta(K) := c(P_{uv})$. Do a depth-first search traversal of K starting in u and ending in v. The resulting walk in K traverses each edge not on P_{uv} twice while each edge on P_{uv} is traversed once. Hence the walk has cost $2C_K - \Delta(K)$. Using standard short-cutting arguments it follows that the minimum-cost spanning tree of $G[K]$ has cost at most

$$2C_K - \Delta(K) \tag{5.8}$$

as well.

Each Steiner vertex $s \in V(K)\backslash R$ can connect to some terminal $v \in K$ at cost at most $\frac{\Delta(K)}{2}$. Hence, the cost $\mathtt{l}(K)$ of the loss of K is at most $b\frac{\Delta(K)}{2}$. In other words we have $\Delta(K) \geq \frac{2}{b}\mathtt{l}(K)$. Plugging this into (5.8) yields the lemma. □

For small values of b we can obtain additional improvements via case analysis.

Lemma 5.10. *Suppose $b \in \{3, 4\}$. Fix an arbitrary optimum r-restricted Steiner tree T^*. Given an undirected, b-quasi-bipartite graph $G = (V, E)$, a set of terminals $R \subseteq V$, and non-negative costs c_e for all $e \in E$, we have*

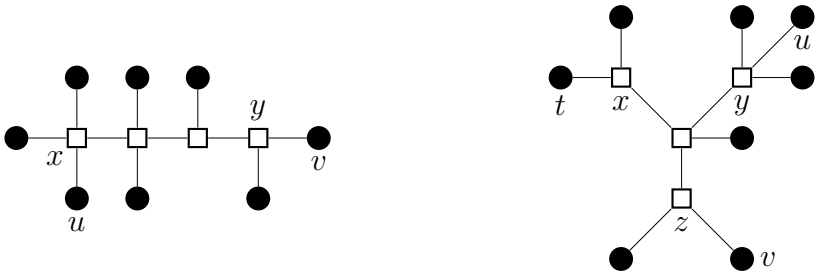
$$\mathtt{mst}(G[R], c) \leq 2\mathtt{opt}_r - \mathtt{l}(T^*).$$

Proof. As in the proof of Lemma 5.9 it suffices to prove that, for each full component $K \in \mathcal{K}(T^*)$, there is a minimum-cost spanning tree of $G[K]$ whose cost is at most $2C_K - \mathtt{l}(K)$, for then we can add the bound over all such K to get the desired result. For terminals $r, s \in K$, let P_{rs} again denote the unique r, s-path in K.

Notice that the Steiner vertices (there are at most b of them) in the full component K either form a path, or else there are 4 of them and they form a star.

Case 1: the Steiner vertices in K form a path. Let x and y be the Steiner vertices on the ends of this path. Let u (resp. v) be any terminal neighbour of x (resp. y); see Figure 5.3(i) for an example. Perform a depth-first search in K starting from u and ending at v; the cost of this search is $2C_K - c(P_{uv})$. By standard short-cutting arguments it follows that $2C_K - c(P_{uv})$ is an upper bound on $\mathtt{mst}(G[K])$. On the other hand, since $P_{uv}\backslash\{ux\}$ is a candidate for the loss of K, we know that $\mathtt{l}(K) \leq c(P_{uv}\backslash\{ux\}) \leq c(P_{uv})$. Therefore we obtain

$$\mathtt{mst}(G[K]) \leq 2C_K - c(P_{uv}) \leq 2C_K - \mathtt{l}(K). \tag{5.9}$$



(i) (ii)

Figure 5.3: The figure shows the two types of full components when $b \leq 4$. On the left is a full component where the Steiner vertices form a path, and on the right is a full component where the Steiner vertices form a star with 3 tips.

Case 2: the Steiner vertices in K form a star. Let the tips of the star be x, y, z and let t, u, v be any terminal neighbours of x, y, z respectively; see Figure 5.3(ii) for an example. Without loss of generality, we may assume that $c_{xt} \leq c_{yu} \leq c_{zv}$. As before, a depth-first search in K starting from u and ending at v has cost $2C_K - c(P_{uv})$ and this is an upper bound on $\texttt{mst}(G[K])$. On the other hand, $P_{uv} \backslash \{yu\} \cup \{xt\}$ is a candidate for the loss of K and so $\texttt{l}(K) \leq c(P_{uv}) - c_{yu} + c_{xt} \leq c(P_{uv})$. We hence obtain Equation (5.9) as in the previous case. □

We are ready to prove our main theorem. We restate it using the notation introduced in the last two sections.

Theorem 5.1. *Given an undirected, b-quasi-bipartite graph $G = (V, E)$, terminals $R \subseteq V$, and a fixed constant $r \geq 2$, Algorithm 5.1 returns a feasible Steiner tree T^p with*

$$c(T^p) \leq \begin{cases} 1.279 \cdot \texttt{opt}_r & : \quad b = 1 \\ (1 + 1/e) \cdot \texttt{opt}_r & : \quad b \in \{2, 3, 4\} \\ \left(1 + \frac{1}{2} \ln\left(3 - \frac{2}{b}\right)\right) \texttt{opt}_r & : \quad b \geq 5. \end{cases}$$

Proof. Using Lemma 5.8 we see that

$$\begin{aligned} c(T^p) &\leq \texttt{opt}_r + \texttt{l}(T^*) \cdot \ln\left(1 + \frac{\overline{\texttt{mst}}(G[R], c) - \texttt{opt}_r}{\texttt{l}(T^*)}\right) \\ &= \texttt{opt}_r + \texttt{l}(T^*) \cdot \ln\left(1 + \frac{\texttt{mst}(G[R], c) - \texttt{opt}_r}{\texttt{l}(T^*)}\right). \end{aligned} \tag{5.10}$$

The second equality above holds because $G[R]$ has no Steiner vertices. Applying the bound on $\texttt{mst}(G[R], c)$ from Lemma 5.9 yields

$$c(T^p) \leq \texttt{opt}_r \cdot \left[1 + \frac{\texttt{l}(T^*)}{\texttt{opt}_r} \cdot \ln\left(1 - \frac{2}{b} + \frac{\texttt{opt}_r}{\texttt{l}(T^*)}\right)\right]. \tag{5.11}$$

Karpinski and Zelikovsky [127] show that $\texttt{l}(T^*) \leq \frac{1}{2}\texttt{opt}_r$. We can therefore obtain an upper-bound on the right-hand side of (5.11) by bounding the maximum value of function $x \ln(1 - 2/b + 1/x)$ for $x \in [0, 1/2]$. We branch into cases:

$b = 1$: The maximum of $x \ln(1/x - 1)$ for $x \in [0, 1/2]$ is attained for $x \approx 0.2178$. Hence, $x \ln(1/x - 1) \leq 0.279$ for $x \in [0, 1/2]$.

$b = 2$: The maximum of $x \ln(1/x)$ is attained for $x = 1/e$ and hence $x \ln(1/x) \leq 1/e$ for $x \in [0, 1/2]$.

$b \in \{3, 4\}$: We use Equation (5.10) together with Lemma 5.10 in place of Lemma 5.9; the subsequent analysis and result are the same as in the previous case.

$b \geq 5$: The function $x \ln(1 - 2/b + 1/x)$ is increasing in x and its maximum is attained for $x = 1/2$. Thus, $x \ln(1 - 2/b + 1/x) \leq \frac{1}{2} \ln(3 - 2/b)$ for $x \in [0, 1/2]$.

The four cases above conclude the proof of the theorem. □

We remark that under the original analysis of Robins and Zelikovsky, for RZ to achieve an approximation ratio better than the MST heuristic requires $(1 + \frac{1}{2}\ln(3))\rho_r < 2$ which occurs for $r \geq 12$. Note the graph resulting from preprocessing under a given choice of r is $(r-2)$-quasi-bipartite; hence, Theorem 5.1 shows that for $r = 5$, RZ achieves ratio $\rho_5 \cdot (1 + \frac{1}{e}) = \frac{13}{9} \cdot (1 + \frac{1}{e}) < 2$ and does better than the MST heuristic.

5.4 Properties of $(\mathcal{P}^{\mathcal{S}})$

We prove in this section that the linear program $(\mathcal{P}^{\mathcal{S}})$ is gradually weakened as the algorithm progresses (i.e., as more full components are added to $\mathcal{S}$).

Lemma 5.11. *If $\mathcal{S} \subset \mathcal{S}'$, then the integrality gap of $(\mathcal{P}^{\mathcal{S}})$ is at most the integrality gap of $(\mathcal{P}^{\mathcal{S}'})$.*

Proof. We consider only the case where $\mathcal{S}' = \mathcal{S} \cup \{J\}$ for some full component J; the general case then follows by induction on $|\mathcal{S}' \backslash \mathcal{S}|$.

Let x be any feasible primal point for $(\mathcal{P}^{\mathcal{S}})$ and define the *extension* x' of x to be a primal point of $(\mathcal{P}^{\mathcal{S}'})$, with $x'_e = x_J$ for all $e \in E(J)$ and $x'_Z = x_Z$ for all $Z \in (\mathcal{K} \backslash \mathcal{S}') \cup E(\mathcal{S})$. We claim that x' is feasible for $(\mathcal{P}^{\mathcal{S}'})$. Since x and x' have the same objective value, this will prove Lemma 5.11.

It is clear that x' satisfies constraints (5.2), so now let us show that x' satisfies the partition inequality (5.1) in $(\mathcal{P}^{\mathcal{S}'})$. Fix an arbitrary partition π' of $V(\mathcal{S}')$, and let π be the restriction of π' to $V(\mathcal{S})$. We get

$$\sum_{e \in E_{\pi'}(\mathcal{S}')} x'_e + \sum_{K \in \mathcal{K} \backslash \mathcal{S}'} \mathtt{rc}_K^{\pi'} x'_K = \left(\sum_{e \in E_\pi(\mathcal{S})} x_e + \sum_{K \in \mathcal{K} \backslash \mathcal{S}} \mathtt{rc}_K^{\pi} x_K \right) + |E_{\pi'} \cap E(J)| x_J - \mathtt{rc}_J^{\pi} x_J. \tag{5.12}$$

Now J spans at least $\mathtt{rc}_J^{\pi} + 1$ parts of π', and it follows that $|E_{\pi'} \cap E(J)| \geq \mathtt{rc}_J^{\pi}$. Hence, using Equation (5.12), the fact that x satisfies constraint (5.1) for π, and the fact that $\bar{r}(\pi) = \bar{r}(\pi')$, we have

$$\sum_{e \in E_{\pi'}(\mathcal{S}')} x'_e + \sum_{K \in \mathcal{K} \backslash \mathcal{S}'} \mathtt{rc}_K^{\pi'} x'_K \geq \sum_{e \in E_\pi(\mathcal{S})} x_e + \sum_{K \in \mathcal{K} \backslash \mathcal{S}} \mathtt{rc}_K^{\pi} x_K \geq \bar{r}(\pi) - 1 = \bar{r}(\pi') - 1.$$

So x' satisfies (5.1) for π'. □

5.4.1 Integrality Gap Upper Bound for b-Quasi-Bipartite Instances

In [176] Rajagopalan and Vazirani show that the bidirected cut relaxation has a gap of at most $\frac{3}{2}$ if the graph is quasi-bipartite. Since $(\mathcal{P}^{\binom{R}{2}})$ is the same as $(\mathcal{P})$ which is stronger than the bidirected cut relaxation, its gap is also at most $\frac{3}{2}$ for such graphs. We are able to generalize this result as follows.

Theorem 5.2. *On b-quasi-bipartite graphs, $(\mathcal{P}^{\binom{R}{2}})$ has an integrality gap between $\frac{8}{7}$ and $\frac{2b+1}{b+1}$ in the worst case.*

Proof. The lower bound comes from Section 4.8.4. We assume G is b-quasi-bipartite, we let T^* be an optimal Steiner tree, which by our metric assumption, has degree at least 3 at all internal nodes.

Let $\mathcal{S}^*$ be its set of full components, together with $\binom{R}{2}$. Since T^* is a minimum spanning tree for $\mathcal{S}^*$, there is a corresponding feasible dual y for $(\mathcal{M}_D)$. When we convert y to a dual

for $(\mathcal{P}_D^{\mathcal{S}^*})$, we claim that y is feasible: indeed, by Corollary 3.27 a violated full component could be used to improve the solution, but T^* is already optimal. The next lemma is the cornerstone of our proof.

Lemma 5.12. *Let π be a partition of $V(\mathcal{S}^*)$ with $y_\pi > 0$. Then $(\bar{r}(\pi)-1) \geq \frac{b+1}{2b+1}(r(\pi)-1)$.*

Proof. For each part V_i of π, let us identify all of the vertices of V_i into a single pseudo-vertex v_i. We may assume by Theorem 3.23 that each induced subgraph $T^*[V_i]$ is connected, hence this identification process yields a tree T'. Let us say that v_i is *Steiner* if and only if all vertices of V_i are Steiner. Note that T' has $r(\pi)$ pseudo-vertices and $r(\pi) - \bar{r}(\pi)$ of these pseudo-vertices are Steiner. The full components of T' are defined analogously to the full components of a Steiner tree.

Consider any full component K' of T' and let K' contain exactly s Steiner pseudo-vertices. It is straightforward to see that $s \leq b$. Each Steiner pseudo-vertex in K' has degree at least 3 by our metric assumption, and at most $s-1$ edges of K' join Steiner vertices to other Steiner vertices. Hence K' has at least $3s-(s-1) = 2s+1$ edges, and so

$$|E(K')| \geq \frac{2s+1}{s} \cdot s \geq \frac{2b+1}{b} \cdot s.$$

Now summing over all full components K', we obtain

$$|E(T')| \geq \frac{2b+1}{b} \cdot \#\{\text{Steiner pseudo-vertices of } T'\}.$$

But $|E(T')| = r(\pi) - 1$ and T' has $r(\pi) - \bar{r}(\pi)$ Steiner pseudo-vertices, therefore

$$r(\pi) - 1 \geq \frac{2b+1}{b}((r(\pi)-1) - (\bar{r}(\pi)-1)) \quad \Rightarrow \quad \frac{2b+1}{b}(\bar{r}(\pi)-1) \geq \frac{b+1}{b}(r(\pi)-1).$$

This proves what we wanted to show. □

It follows that the objective value of y in $(\mathcal{P}_D^{\mathcal{S}^*})$ is

$$\sum_{\pi \in \Pi^{\mathcal{S}}} (\bar{r}(\pi)-1)y_\pi \geq \sum_{\pi \in \Pi^{\mathcal{S}}} \frac{b+1}{2b+1}(r(\pi)-1)y_\pi = \frac{b+1}{2b+1}c(T^*)$$

and since T^* is an optimum integer solution of $(\mathcal{P}^{\mathcal{S}^*})$, it follows that the integrality gap of $(\mathcal{P}^{\mathcal{S}^*})$ is at most $\frac{2b+1}{b+1}$. Then, finally, by applying Lemma 5.11 to $(\mathcal{P}^{\binom{R}{2}})$ and $(\mathcal{P}^{\mathcal{S}^*})$ we obtain Theorem 5.2. □

5.5 Proof of Lemma 5.8

In this section we present a proof of Lemma 5.8. The methodology follows that proposed by Gröpl et al. [101]; see also the presentation of Korte & Vygen [141, Ch. 20] which corrects a small bug. The essential novelty of our approach is an integral-based interpretation of $\mathtt{mst}$, $\overline{\mathtt{mst}}$ and loss, which leads to the cornerstone $\mathtt{mst} = \overline{\mathtt{mst}} + \mathtt{l}$ (Lemma 5.14). This also results in a new, short proof of the ubiquitous *contraction lemma* (Lemma 5.18).

When G is a graph and τ is a real number, let $G_{\leq\tau}$ denote the subgraph of G obtained by deleting all edges with weight greater than τ. For a graph G, let $\kappa(G)$ denote the number of connected components of G.

Lemma 5.13. $\mathtt{mst}(G) = \int_{\tau=0}^{\infty}(\kappa(G_{\leq\tau}) - 1)d\tau$.

Proof. We re-use the notation from Section 3.6.1. At time τ, Kruskal's primal-dual algorithm raises the objective function of $(\mathcal{M}_D)$ at a rate of $r(\pi(\tau)) - 1$ per unit time. By Theorem 3.23,

$$\mathtt{mst}(G) = c(T) = \sum_{\pi} y^*_{\pi}(r(\pi) - 1) = \int_{\tau=0}^{\tau^*}(r(\pi(\tau)) - 1)d\tau.$$

Now, since $\pi(\tau)$ is the same as the partition induced by the connected components of $G_{\leq\tau}$, and since $\kappa(G_{\leq\tau}) = 1$ for $\tau \geq \tau^*$, we are done. □

We first relate the cost of a minimum-cost spanning tree of $\mathcal{S}$ for some set $\mathcal{S}$ of full components to the (potential) lower-bound $\overline{\mathtt{mst}}(\mathcal{S})$ on $\mathtt{opt}_r$ that it provides.

Lemma 5.14. *For any graph G and terminal set $R \subset V(G)$,*

$$\mathtt{mst}(G) = \overline{\mathtt{mst}}(G) + \mathtt{l}(G).$$

Proof. Run $\mathtt{MST}$ on input G, obtaining output (T, y) Let us adopt the notation from the proof of Theorem 3.23. The difference $\mathtt{mst}(G) - \overline{\mathtt{mst}}(G)$ satisfies

$$\mathtt{mst}(G) - \overline{\mathtt{mst}}(G) = \sum_{\pi} y_{\pi} r(\pi) - \sum_{\pi} y_{\pi}\bar{r}(\pi) = \int_{0}^{\tau^*}(r(\pi(\tau)) - \bar{r}(\pi(\tau)))d\tau. \tag{5.13}$$

Let a *Steiner part* of a partition be a part which contains only Steiner vertices. The quantity $r(\pi(\tau)) - \bar{r}(\pi(\tau))$ counts the number of Steiner parts of $\pi(\tau)$. Recall from Section 3.6.1 that G^τ denotes the forest maintained by Kruskal's algorithm at time $\tau \geq 0$. We then obtain G^τ/R from G^τ by identifying the set of all terminals; G^τ/R has one connected component

for each Steiner part of $\pi(\tau)$, and one additional connected component containing all other vertices. Therefore, the right-hand side of (5.13) is equal to

$$\int_0^{\tau^*} (\kappa(G^\tau/R) - 1)d\tau = \int_0^{\infty} (\kappa((G/R)_{\leq\tau}) - 1)d\tau = \mathtt{mst}(G/R),$$

where the last equality uses Lemma 5.13.

Finally, note that $\mathtt{l}(G) = \mathtt{mst}(G/R)$, since the loss is the minimum-cost set of edges to connect every Steiner vertex to some terminal, which is the same as the minimum-cost set of edges to connect every Steiner vertex to the pseudo-vertex corresponding to R in G/R, which is in turn the minimum spanning tree of G/R. □

We obtain the following immediate corollary:

Corollary 5.15. *In iteration i of Algorithm 5.1, adding full component $K \in \mathcal{K}$ to $\mathcal{S}$ reduces the cost of $\mathtt{mst}(\mathcal{S})$ if and only if $f_i(K) < 1$.*

Proof. By applying Lemma 5.14 we see that

$$\mathtt{mst}(\mathcal{S}^i) - \mathtt{mst}(\mathcal{S}^i \cup \{K\}) = \overline{\mathtt{mst}}(\mathcal{S}^i) + \mathtt{l}(\mathcal{S}^i) - \overline{\mathtt{mst}}(\mathcal{S}^i \cup \{K\}) - \mathtt{l}(\mathcal{S}^i \cup \{K\}).$$

Whereas the left-hand side is positive iff adding K to $\mathcal{S}^i$ causes a reduction in $\mathtt{mst}$, the right-hand side is positive iff $f_i(K) < 1$, due to the definition of f_i (equation (5.7)). □

Using Corollaries 3.27 and 5.15, we obtain the following.

Corollary 5.16. *For all $1 \leq i \leq p$, $f_i(K^i) < 1$.*

Fix an optimum r-Steiner tree T^*. The next two lemmas give bounds that are needed to analyze RZ's greedy strategy. Informally, the first says that $\overline{\mathtt{mst}}$ is non-increasing, while the second says that $\overline{\mathtt{mst}}$ is supermodular.

Lemma 5.17. *If $\binom{R}{2} \subseteq \mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{K}$, then $\overline{\mathtt{mst}}(\mathcal{S}') \leq \overline{\mathtt{mst}}(\mathcal{S})$.*

Proof. Using Lemma 5.14 and Fact 5.7 we see

$$\overline{\mathtt{mst}}(\mathcal{S}) - \overline{\mathtt{mst}}(\mathcal{S}') = \mathtt{mst}(\mathcal{S}) + \mathtt{l}(\mathcal{S}' \backslash \mathcal{S}) - \mathtt{mst}(\mathcal{S}').$$

However, the right hand side of the above equation is non-negative, as $\mathtt{MST}(\mathcal{S}) \cup \mathtt{L}(\mathcal{S}' \backslash \mathcal{S})$ is a spanning tree of $\mathcal{S}'$. Lemma 5.17 then follows. □

Lemma 5.18 (Contraction Lemma)**.** *Let $\mathcal{S}, \mathcal{S}', \mathcal{S}'' \subset \mathcal{K}$ be disjoint collections of full components with $\binom{R}{2} \subseteq \mathcal{S}$. Then*

$$\overline{\texttt{mst}}(\mathcal{S}) - \overline{\texttt{mst}}(\mathcal{S} \cup \mathcal{S}'') \geq \overline{\texttt{mst}}(\mathcal{S} \cup \mathcal{S}') - \overline{\texttt{mst}}(\mathcal{S} \cup \mathcal{S}' \cup \mathcal{S}'').$$

Proof. The statement to be proved is equivalent to

$$\texttt{mst}(\mathcal{S}) - \texttt{mst}(\mathcal{S} \cup \mathcal{S}'') \geq \texttt{mst}(\mathcal{S} \cup \mathcal{S}') - \texttt{mst}(\mathcal{S} \cup \mathcal{S}' \cup \mathcal{S}''), \tag{5.14}$$

due to Lemma 5.14 and Fact 5.7. Our proof is centred around proving that for all $\tau \geq 0$,

$$\kappa(\mathcal{S}_{\leq\tau}) - \kappa(\mathcal{S}_{\leq\tau} \cup \mathcal{S}''_{\leq\tau}) \geq \kappa(\mathcal{S}_{\leq\tau} \cup \mathcal{S}'_{\leq\tau}) - \kappa(\mathcal{S}_{\leq\tau} \cup \mathcal{S}'_{\leq\tau} \cup \mathcal{S}''_{\leq\tau}). \tag{5.15}$$

If we prove Equation (5.15), then by adding $-1+1$ to each side, integrating along τ, and using Lemma 5.13, we obtain Equation (5.14) as needed.

Define a function μ on graphs by $\mu(G) = |V(G)| - \kappa(G)$. Note μ doesn't depend on isolated vertices and thus can be viewed as a function of edge sets. The crux is that μ is the rank function for graphic matroids, and is hence submodular over full components. The function $|V(\mathcal{S})|$ for $\mathcal{S} \supset \binom{R}{2}$ is modular with respect to full components, and so $\kappa(\mathcal{S}) = |V(\mathcal{S})| - \mu(\mathcal{S})$ is supermodular with respect to full components, which proves Equation (5.15). □

Note that the proof of Lemma 5.18 easily generalizes to other matroids. This seems not to have been noticed before, and is not evident from early proofs of the Contraction Lemma (e.g. [16, Lemma 3.9], [101], [177, Lemma 2]) — although the presentation of Korte & Vygen [141] leads this way.

Remark 5.19. *Let $\beta(M)$ denote the minimum cost of any basis of the matroid M on ground set E, which equals $\int_{\mathbf{R}}(\sigma(t)r(M) - r(\{e \in E \mid c_e \leq t\}))dt$ for the unit step function σ, and let A, B be disjoint subsets of E. If either $c \geq 0$ or $r(A) + r(B) = r(A \cup B)$ we have that $\beta(M/A) + \beta(M/B) \leq \beta(M) + \beta(M/A/B)$. If either $c \leq 0$ or $r(M\backslash A) + r(M\backslash B) = r(M) + r(M\backslash A\backslash B)$, we have that $\beta(M\backslash A) + \beta(M\backslash B) \leq \beta(M) + \beta(M\backslash A\backslash B)$. Unconditionally, we have $\beta(M/A) + \beta(M\backslash B) \geq \beta(M/A\backslash B) + \beta(M)$.*

We are finally near the end of the analysis, where the Contraction Lemma comes into play. We can now bound the value $f_i(K^i)$ for all $0 \leq i \leq p-1$ in terms of the cost of T^*'s loss. In the remainder of the section, let $\texttt{l}^*$ denote $\texttt{l}(T^*)$, let $\overline{\texttt{mst}}^i$ denote $\overline{\texttt{mst}}(\mathcal{S}^i)$ and let $\overline{\texttt{mst}}^*$ denote $\overline{\texttt{mst}}(T^*)$.

Lemma 5.20. *For all $0 \leq i \leq p-1$, if $\overline{\texttt{mst}}^i - \overline{\texttt{mst}}^* > 0$, then $f_i(K^i) \leq \texttt{l}^*/(\overline{\texttt{mst}}^i - \overline{\texttt{mst}}^*)$.*

Proof. Let the full components of T^* be $K^{*,1}, \ldots, K^{*,q}$. By the choice of K^i in Algorithm 5.1, we have $f_i(K^i) \leq \min_j f_i(K^{*,j})$. A standard fraction averaging argument implies that

$$\begin{aligned} f_i(K^i) &\leq \frac{\sum_{j=1}^{q} \mathtt{l}(K^{*,j})}{\sum_{j=1}^{q} \left(\overline{\mathtt{mst}}(\mathcal{S}^i) - \overline{\mathtt{mst}}(\mathcal{S}^i \cup \{K^{*,j}\})\right)} \\ &\leq \frac{\mathtt{l}^*}{\sum_{j=1}^{q} \left(\overline{\mathtt{mst}}(\mathcal{S}^i \cup \{K^{*,1}, \ldots, K^{*,j-1}\}) - \overline{\mathtt{mst}}(\mathcal{S}^i \cup \{K^{*,1}, \ldots, K^{*,j}\})\right)} \end{aligned} \tag{5.16}$$

where the last inequality uses Fact 5.7 and Lemma 5.18. The denominator of the right-hand side of Equation (5.16) is a telescoping sum. Canceling like terms, and using Lemma 5.17 to replace $\overline{\mathtt{mst}}(\mathcal{S}^i \cup \{K^{*,1}, \ldots, K^{*,q}\})$ with $\overline{\mathtt{mst}}^*$, we are done. □

We can now bound the cost of T^p.

Proof of Lemma 5.8. We first bound the loss $\mathtt{l}(T^p)$ of tree T^p. Using Fact 5.7,

$$\mathtt{l}(T^p) = \sum_{i=0}^{p-1} \mathtt{l}(K^i) = \sum_{i=0}^{p-1} f_i(K^i) \cdot (\overline{\mathtt{mst}}^i - \overline{\mathtt{mst}}^{i+1}) \tag{5.17}$$

where the last equality uses the definition of f_i from (5.7). Using Corollary 5.16 and Lemma 5.20, the right hand side of Equation (5.17) is bounded as follows:

$$\sum_{i=0}^{p-1} f_i(K^i) \cdot (\overline{\mathtt{mst}}^i - \overline{\mathtt{mst}}^{i+1}) \leq \sum_{i=0}^{p-1} \frac{\mathtt{l}^*}{\max\{\mathtt{l}^*, \overline{\mathtt{mst}}^i - \overline{\mathtt{mst}}^*\}} \cdot (\overline{\mathtt{mst}}^i - \overline{\mathtt{mst}}^{i+1}). \tag{5.18}$$

The right hand side of Equation (5.18) can in turn be bounded from above by the following integral:

$$\sum_{i=0}^{p-1} \frac{\mathtt{l}^* \cdot (\overline{\mathtt{mst}}^i - \overline{\mathtt{mst}}^{i+1})}{\max\{\mathtt{l}^*, \overline{\mathtt{mst}}^i - \overline{\mathtt{mst}}^*\}} \leq \int_{\overline{\mathtt{mst}}^p}^{\overline{\mathtt{mst}}^0} \frac{\mathtt{l}^*}{\max\{\mathtt{l}^*, x - \overline{\mathtt{mst}}^*\}} dx = \int_{\overline{\mathtt{mst}}^p - \overline{\mathtt{mst}}^*}^{\overline{\mathtt{mst}}^0 - \overline{\mathtt{mst}}^*} \frac{\mathtt{l}^*}{\max\{\mathtt{l}^*, x\}} dx. \tag{5.19}$$

Notice that $\overline{\mathtt{mst}}^0 = \mathtt{mst}(G[R], c) \geq \mathtt{opt}_r = \mathtt{l}^* + \overline{\mathtt{mst}}^*$. The termination condition in Algorithm 5.1 and Lemma 5.4 imply that $\overline{\mathtt{mst}}^p \leq \mathtt{opt}_r$. Hence the result of evaluating the integral in the right-hand side of Equation (5.19) is

$$\mathtt{l}^* - (\overline{\mathtt{mst}}^p - \overline{\mathtt{mst}}^*) + \mathtt{l}^* \cdot \int_{\mathtt{l}^*}^{\overline{\mathtt{mst}}^0 - \overline{\mathtt{mst}}^*} \frac{1}{x} dx = \mathtt{opt}_r - \overline{\mathtt{mst}}^p + \mathtt{l}^* \cdot \ln\left(\frac{\overline{\mathtt{mst}}^0 - \overline{\mathtt{mst}}^*}{\mathtt{l}^*}\right) \tag{5.20}$$

where the equality uses Lemma 5.14. Applying Lemma 5.14 two more times, and combining Equations (5.17)–(5.20), we obtain

$$\begin{aligned} c(T^p) = \overline{\mathtt{mst}}^p + \mathtt{l}(T^p) &\leq \mathtt{opt}_r + \mathtt{l}^* \cdot \ln\left(\frac{\overline{\mathtt{mst}}^0 - \overline{\mathtt{mst}}^*}{\mathtt{l}^*}\right) \\ &= \mathtt{opt}_r + \mathtt{l}^* \cdot \ln\left(1 + \frac{\overline{\mathtt{mst}}^0 - (\overline{\mathtt{mst}}^* + \mathtt{l}^*)}{\mathtt{l}^*}\right) \\ &= \mathtt{opt}_r + \mathtt{l}^* \cdot \ln\left(1 + \frac{\overline{\mathtt{mst}}^0 - \mathtt{opt}_r}{\mathtt{l}^*}\right) \end{aligned}$$

as wanted. □

5.6 A Steiner Tree LP with Integrality Gap $1 + \ln 2$

In this section we show a new LP formulation for the Steiner tree problem. It is based on using the *contraction lemma* (Lemma 5.18) to view the Steiner tree as an instance of the *submodular set cover* [199] problem. Then we show that the *relative greedy algorithm* of Zelikovsky [203], which he showed was a $(1 + \ln 2)$-approximation algorithm, is an *LP-relative* $(1 + \ln 2)$-approximation algorithm. We are motivated by the LP-based work of Wolsey [199]; in fact the relative greedy algorithm and the greedy algorithm of Wolsey are the same in our setting.

It has been an open problem for quite some time to find an LP relaxation for Steiner tree with integrality gap less than 2. Nonetheless, we feel that this problem is essentially still open because the LP we describe lacks certain essential properties. We discuss these problems in Section 5.6.2.

Let $\mathcal{K}$ denote the set of all full components as usual. We view the Steiner tree problem as one of picking the minimum-cost connected spanning subhypergraph from $\mathcal{K}$. Also let G denote the original underlying graph, and $\mathtt{mst}(G)$ denote the minimum-cost spanning tree of $G[R]$, assuming as usual that G is a complete graph with c metric. For a collection $\mathcal{S}$ of full components, let $\mathtt{mst}(G/\mathcal{S})$ denote the cost of a minimum spanning tree in the graph $G[R]/\mathcal{S}$.

First, we make explicit the relation to submodular set cover. Let $z(\mathcal{S}) = \mathtt{mst}(G) - \mathtt{mst}(G/\mathcal{S})$ for any $\mathcal{S} \subset \mathcal{K}$. Then z is nondecreasing and the Contraction Lemma implies that z is submodular. The problem of finding a minimum-cost connected spanning subhypegraph is the same as finding finding a min-cost $\mathcal{S}$ such that $z(\mathcal{S}) = z(\mathcal{K})$.

We give two proofs of the $(1 + \ln 2)$ integrality gap but both rely on similar ingredients. The first proof follows the line of Zelikovsky [203] and is non-constructive. The second

follows the line of Wolsey [199, Thm. 1.ii], using dual fitting, and gives in polynomial-time an explicit dual (lower bound).

We first state the algorithm, which we denote by RG. It is straightforward to observe that it outputs a feasible solution.

Procedure RG
1: Let $\mathcal{S}^0 := \varnothing$ and $i = 1$.
2: Let $f^i := \min_K C_K/(\mathtt{mst}(G/\mathcal{S}^{i-1}) - \mathtt{mst}(G/\mathcal{S}^{i-1}/K))$ over all $K \in \mathcal{K}$ with nonzero denominator, and let K^i be a full component which achieves this minimum.
3: Let $\mathcal{S}^i := \mathcal{S}^{i-1} \cup \{K^i\}$
4: If $\mathcal{S}$ is not a connected spanning hypergraph, increment i and goto line 2.

The LP is based on the following observation.

Claim 5.21. *If $\mathcal{S}$ is a spanning subhypergraph and $\mathcal{J}$ is an arbitrary subset of $\mathcal{K}$, then*

$$\sum_{K \in \mathcal{S}} \mathtt{mst}(G/\mathcal{J}) - \mathtt{mst}(G/\mathcal{J}/K) \geq \mathtt{mst}(G/\mathcal{J}).$$

Proof. We know that $\mathtt{mst}(G/\mathcal{S}) = 0$. Enumerate $\mathcal{S} = \{S_1, \ldots, S_t\}$. By the Contraction Lemma we get

$$\begin{aligned}\sum_{i=1}^{t} \mathtt{mst}(G/\mathcal{J}) - \mathtt{mst}(G/\mathcal{J}/S_i) &\geq \sum_{i=1}^{t} \mathtt{mst}(G/\mathcal{J}/\{S_j \mid j < i\}) - \mathtt{mst}(G/\mathcal{J}/\{S_j \mid j \leq i\}) \\ &= \mathtt{mst}(G/\mathcal{J}) - \mathtt{mst}(G/\mathcal{J}/\mathcal{S}) = \mathtt{mst}(G/\mathcal{J}),\end{aligned}$$

as needed. □

For a subset $\mathcal{S} \subset \mathcal{K}$ and $K \in \mathcal{K}$, define

$$\rho(\mathcal{S}, K) = \mathtt{mst}(G/\mathcal{S}) - \mathtt{mst}(G/\mathcal{S}/K).$$

Then Claim 5.21 shows the LP ($\mathcal{W}$) (with dual ($\mathcal{W}_D$)) given in Figure 5.4 is a relaxation for the Steiner tree problem.

Let t denote the number of iterations performed by RG, i.e. $\mathcal{S}^t$ is the output. We defer the proofs of the following supporting claims, which are used in both approaches.

Claim 5.22. *$f^i \leq 1$ for all i.*

Claim 5.23. *$f^1 \geq 1/2$ and $\mathtt{mst}(G) \leq 2\mathrm{OPT}(\mathcal{W})$.*

$$\min \Big\{ \sum_{K \in \mathcal{K}} C_K x_K : \quad x \in \mathbf{R}^{\mathcal{K}} \tag{$\mathcal{W}$}$$

$$\sum_K x_K \rho(\mathcal{J}, K) \geq \mathtt{mst}(G/\mathcal{J}), \quad \forall \mathcal{J} \subset \mathcal{K} \tag{5.21}$$

$$x_K \geq 0, \quad \forall K \in \mathcal{K} \Big\} \tag{5.22}$$

$$\max \Big\{ \sum_U \mathtt{mst}(G/\mathcal{J}) y_{\mathcal{J}} : \quad y \in \mathbf{R}^{\mathcal{P}(\mathcal{K})} \tag{$\mathcal{W}_D$}$$

$$\sum_{\mathcal{J}} y_{\mathcal{J}} \rho(\mathcal{J}, K) \leq C_K, \quad \forall K \in \mathcal{K} \tag{5.23}$$

$$y_{\mathcal{J}} \geq 0, \quad \forall \mathcal{J} \subset \mathcal{K} \Big\} \tag{5.24}$$

Figure 5.4: The submodular set cover relaxation and its dual.

We need the following lemma to give our variant of Zelikovsky's proof.

Lemma 5.24. $f^i \leq \mathrm{OPT}(\mathcal{W})/\mathtt{mst}(G/\mathcal{S}^{i-1})$.

Proof. By the algorithm's greedy choice, $f^i \leq C_K/(\mathtt{mst}(G/\mathcal{S}^{i-1}) - \mathtt{mst}(G/\mathcal{S}^{i-1}/K))$ for each K. Let x^* be an optimal solution to ($\mathcal{W}$), then we have

$$f^i \leq \sum_K \frac{x^*_K C_K}{x^*_K(\mathtt{mst}(G/\mathcal{S}^{i-1}) - \mathtt{mst}(G/\mathcal{S}^{i-1}/K))} \leq \frac{\mathrm{OPT}(\mathcal{W})}{\mathtt{mst}(G/\mathcal{S}^{i-1})}$$

where the first inequality holds by a standard averaging argument, and the second holds since x^* satisfies (5.21). □

Here is our nonconstructive variant of Zelikovsky's proof.

Theorem 5.25. *The integrality gap of* ($\mathcal{W}$) *is at most* $1 + \ln 2$.

Proof. For conciseness, let OPT stand for $\mathrm{OPT}(\mathcal{W})$. We know by Claim 5.22 and Lemma 5.24 that $C(\mathcal{S}^t)$ equals

$$\sum_{i=1}^{t} f^i(\mathtt{mst}(G/\mathcal{S}^{i-1}) - \mathtt{mst}(G/\mathcal{S}^i)) \leq \sum_{i=1}^{t} (\mathtt{mst}(G/\mathcal{S}^{i-1}) - \mathtt{mst}(G/\mathcal{S}^i)) \min\left\{1, \frac{\mathrm{OPT}}{\mathtt{mst}(G/\mathcal{S}^{i-1})}\right\}.$$

A standard integral bound, plus $\texttt{mst} \geq \text{OPT} \geq \texttt{mst}/2$ (Claim 5.23) gives

$$C(\mathcal{S}^t) \leq \int_{u=0}^{\texttt{mst}(G)} du \min\{1, \text{OPT}/u\} = \text{OPT} + \text{OPT} \ln \frac{\texttt{mst}(G)}{\text{OPT}} \leq (1 + \ln 2)\text{OPT},$$

as needed. □

Next we give a constructive proof following Wolsey's approach. It relies on the following lemma.

Lemma 5.26. *The sequence $\{f^i\}$ is nondecreasing in i.*

Proof. Use the greedy definition of f^i and note by the Contraction Lemma that $(\texttt{mst}(G/\mathcal{S}^{i-1}) - \texttt{mst}(G/\mathcal{S}^{i-1}/K))$ is decreasing in i. □

Define $f^0 = 0$ for convenience. Define the dual solution y to ($\mathcal{W}_D$) by

$$y_{\mathcal{S}^{i-1}} = f^i - f^{i-1},$$

for $1 \leq 1 \leq t$, and y zero on all other sets.

Theorem 5.27. *The objective value of y equals $C(\mathcal{S}^t)$, and $y/(1 + \ln 2)$ is feasible for ($\mathcal{W}_D$).*

Proof. First, using the fact that $f^0 = 0$ and $\texttt{mst}(G/\mathcal{S}^t) = 0$, we get that the objective value of y is

$$\sum_{i=1}^{t} (f^i - f^{i-1})\texttt{mst}(G/\mathcal{S}^{i-1}) = \sum_{i=1}^{t} f^i(\texttt{mst}(G/\mathcal{S}^{i-1}) - \texttt{mst}(G/\mathcal{S}^i)) = \sum_{i=1}^{t} C_{K^i},$$

as needed.

Second, to show that $y/(1 + \ln 2)$ is feasible, we need to show that for all K,

$$\sum_{i=1}^{t} (f^i - f^{i-1})\rho(\mathcal{S}^{i-1}, K) \leq (1 + \ln 2)C_K. \tag{5.25}$$

Due to the greedy choices of the algorithm we know that $\rho(\mathcal{S}^{i-1}, K) \leq C_K/f^i$. Therefore

$$\sum_{i=1}^{t} (f^i - f^{i-1})\rho(\mathcal{S}^{i-1}, K) \leq C_K \sum_{i=1}^{t} (f^i - f^{i-1})/f^i = C_K(1 + \sum_{i=2}^{t} (f^i - f^{i-1})/f^i) \tag{5.26}$$

Since the f^i are increasing, a standard integral estimate shows that

$$\sum_{i=2}^{t} (f^i - f^{i-1})/f^i \leq \int_{u=f^1}^{f^t} \frac{du}{u} = \ln\left(\frac{f^t}{f^1}\right). \tag{5.27}$$

Combining Equations (5.26) and (5.27) with Claims 5.22 and 5.23 gives Equation (5.25), as needed. □

5.6.1 Proofs of Supporting Claims

Proof of Claim 5.22, $f^i \leq 1$ for all i. This is equivalent to saying that some K satisfies

$$C_K \leq \mathtt{mst}(G/\mathcal{S}^{i-1}) - \mathtt{mst}(G/\mathcal{S}^{i-1}/K),$$

note that any full component K of size 2 (i.e., terminal-terminal edge) in the minimum spanning tree of $G[R]/\mathcal{S}^{i-1}$ meets this inequality with equality. □

Proof of Claim 5.23, $f^1 \geq 1/2$ and $\mathtt{mst}(G) \leq 2\mathrm{OPT}(\mathcal{W})$. Considering the definition of f^1, the statement $f^1 \geq 1/2$ is equivalent to

$$\mathtt{mst}(G) \leq \mathtt{mst}(G/K) + 2C_K \tag{5.28}$$

for all $K \in \mathcal{K}$. Consider an Eulerian tour of the tree representing the full component K: since the edge costs are metric, we can shortcut which establishes that in $G[R]$, there is a tree spanning K with cost at most $2C_K$. Taking the union of this tree with a minimum spanning tree of G/K gives a spanning tree of G, hence Equation (5.28) holds.

Now to see that $\mathtt{mst}(G) \leq 2\mathrm{OPT}(\mathcal{W})$, take any optimal solution x^* to $(\mathcal{W})$. Constraint (5.21) for $\mathcal{J} = \varnothing$, together with $f^1 \geq 1/2$, implies

$$\mathrm{OPT}(\mathcal{W}) = \sum_K x_K C_K \geq \sum_K x_K \rho(\varnothing, K)/2 \geq \mathtt{mst}(G)/2.$$ □

Note that in r-restricted graphs, the above analysis can be slightly improved, giving an integrality gap bound of $1 + \ln(2 - 2/r)$.

5.6.2 Difficulties with the LP

We are, unfortunately, unaware of a polynomial-time algorithm to optimize over $(\mathcal{W})$ in our case. (A closely related separation problem [192, §3.7.2] is NP-hard, even to approximate.) We present a more precise negative structural result momentarily. We also remark that all relaxations studied before for the Steiner tree problem (undirected cut, bidirected cut, hypergraphic) are *combinatorial relaxations* in the sense that the LP feasible region is independent of the edge costs; the submodular cover relaxation $(\mathcal{W})$ is not combinatorial in this sense.

We also remark that, while a critic of $(\mathcal{W})$ may point out that there is a trivial LP for the Steiner tree problem with integrality gap 1 which also is not polynomial-time solvable, the property Theorem 5.27 of having an explicit poly-time primal-dual certificate of approximate optimality is non-trivial.

Because our negative results are somewhat vague and this LP is esoteric to begin with, it is helpful to keep in mind a sample potential application of the LP, the *bounded-degree Steiner tree* problem [148]. For this problem, there is no bicriteria approximation algorithm with cost factor smaller than 2, so one would hope to use ($\mathcal{W}$) to get a bicriteria algorithm with cost factor $1 + \ln 2$.

We are not aware of any structure for basic solutions of ($\mathcal{W}$). It is easy to verify that the usual *set uncrossing* (e.g., see Section 3.2) of $\mathcal{J}, \mathcal{J}'$ into $\mathcal{J} \cap \mathcal{J}', \mathcal{J} \cup \mathcal{J}'$ fails. A nice observation is that we can reformulate ($\mathcal{W}$) in terms of partitions. Namely, we can replace the only nontrivial constraint (5.21) with

$$\sum_K x_K \rho(\pi, K) \geq \mathtt{mst}(G/\pi), \quad \forall \pi \in \Pi_R$$

where $\rho(\pi, K)$ denotes $\mathtt{mst}(G/\pi) - \mathtt{mst}(G/\pi/K)$.

Remark 5.28. *The integral-based formulation of* mst *(see Section 5.5) permits us to reformulate the new LP, since*

$$\mathtt{mst}(G/\pi) = \int_0^\infty (r(\pi \vee \tau(t)) - 1)dt \quad \textit{and similarly} \quad \rho(\pi, K) = \int_0^\infty \mathtt{rc}_K^{\pi \vee \tau(t)} dt$$

where $\tau(t)$ is the partition induced by the connected components of $(R, \{e \in E \mid c(e) \leq t\})$. Also, in the special case that $G[R]$ has a min-cost spanning tree consisting of unit-cost edges, the new LP is the same as ($\mathcal{P}$).

In general, the partition-based formulation cannot be uncrossed in the sense of (3.2).

Example 5.29. *Consider the Steiner tree instance with $R = \{1, 2, 3\}$ and costs $C_{12} = 1, C_{13} = C_{23} = 2, C_{123} = 3$. The constraint for $\underline{\pi}$ is vacuous and the other constraints (labeled by their corresponding partitions) are*

$$\begin{array}{rrrrll} x_{12} & +2x_{23} & +2x_{13} & +3x_{123} & \geq 3 & (1,2,3) \\ & 2x_{23} & +2x_{13} & +2x_{123} & \geq 2 & (12,3) \\ x_{12} & & +x_{13} & +x_{123} & \geq 1 & (1,23) \\ x_{12} & +x_{23} & & +x_{123} & \geq 1 & (13,2) \end{array}$$

and the extreme point solution $[0, 1, 1, 0]$ is tight for $(13, 2)$ and $(1, 23)$ but not their meet $(1, 2, 3)$.

The main negative facts about this LP — that it is non-combinatorial and not known to be separable in polynomial time — do not alone preclude that ($\mathcal{W}$) could be useful. For example, the *knapsack cover inequalities* [32], have also these two properties, and still

were useful in developing a variety of good polynomial-time approximation algorithms, e.g. [32, 134] and Section 7.2.1. But note this is due to the following special property of the knapsack-cover formulation: given an x, we can either round it up to a feasible integral solution with cost at most $\alpha \cdot c(x)$, or find an equality which x violates. The new LP ($\mathcal{W}$) does not seem to have this special property; one might blame this on the fact that ($\mathcal{W}$)'s integrality gap bound comes from dual fitting, in contrast to how knapsack-cover was engineered for rounding.

Chapter 6

Integral Multicommodity Flow in Trees: Using Covers to Pack

We consider the max-weight integral multicommodity flow problem in trees. In this problem we are given an edge-, arc-, or vertex-capacitated tree and weighted pairs of terminals, and the objective is to find a max-weight integral flow between terminal pairs subject to the capacities. This problem is APX-hard [87], and a 4-approximation for the edge- and arc-capacitated versions is known [39]. Some special cases are known to be exactly solvable in polynomial time, including when the graph is a path or a star.

We show that all three versions of this problems fit in a common framework. The two ingredients are that *iterated LP relaxation* applies to each and that an *LP-relative approximation algorithm* exists for their covering analogues. The result of the framework is a $1 + O(1/\mu)$-approximation algorithm where μ denotes the minimum capacity, for all three versions. We give a complementary hardness result which shows that this is asymptotically best possible. We also get a similar algorithmic result for the covering analogue of multicommodity flow.

We extend the range of edge-capacitated instances for which exact solutions can be found efficiently. When the tree is a *spider* (i.e. if only one vertex has degree greater than 2) we give a polynomial-time algorithm to find an optimal solution, as well as a polyhedral description of the convex hull of all integral feasible solutions.

6.1 Introduction

In the max-weight integral multicommodity flow problem (WMF), we are given an undirected *supply* graph $G = (V, E)$, terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ where $s_i, t_i \in V$, non-negative weights $w_1, \ldots, w_k$ and non-negative integral capacities. We distinguish between three versions of the problem: in edge-WMF each edge $e \in E$ has a capacity c_e; in arc-WMF each of the $2|E|$ directed arcs (u, v) with $\{u, v\} \in E$ has a capacity c_{uv}; in vertex-WMF each vertex $v \in V$ has a capacity c_v. The goal is to simultaneously route integral s_i-t_i flows of value y_i, subject to the capacities, so as to maximize the weight $\sum w_i y_i$. Note that edge-WMF can be reduced to vertex-WMF by subdividing each edge, moving that edge's capacity onto the new vertex, and setting all other vertex capacities to be infinite. When we make statements that apply to all three versions, we simply say WMF.

The single-commodity version ($k = 1$) of WMF is well-known to be solvable in polynomial time. If we drop the integrality restriction the problem can be solved in polynomial time via linear programming for any k. However, when integrality is required, even the 2-commodity unit-capacity, unit-weight arc- and edge-versions are NP-complete — see Even, Itai, and Shamir [64]. Let $n := |V|$. Recent results [4, 107] on the *edge-disjoint paths* problem show more strongly that arc-WMF is NP-hard to $|E|^{\frac{1}{2}-\epsilon}$-approximate, and edge-WMF cannot be approximated better than $\log^{\frac{1}{2}-\epsilon}(n)$ unless $\mathsf{NP} \subset \mathsf{ZPTIME}(n^{\mathrm{polylog}(n)})$. Randomized rounding gives a $(1+\epsilon)$-approximation algorithm for WMF when all capacities are at least $\log n/\epsilon^2$, for suitably small ϵ [62, 175].

An easier and significant special case of WMF is where the supply graph G is a tree, which we denote by WMFT. Garg, Vazirani and Yannakakis [87] considered the unit-weight case of edge-WMFT and showed APX-hardness even if G's height is at most 3 and all capacities are 1 or 2; but on the positive side, they gave a 2-approximate polynomial-time primal-dual algorithm. Techniques of Garg et al. show that WMFT can be solved in polynomial time when G has unit capacity (using dynamic programming and matching) or is a star (this problem is essentially equivalent to b-matching). The case where G is a path (so-called *interval packing*) is also polynomial-time solvable [29, 39, 109], e.g. by linear programming since the natural LP has a totally unimodular constraint matrix. Arc-WMF on unit-capacity bidirected trees admits a $(\frac{5}{3} + \epsilon)$-approximation algorithm [63]. In general, the best result for edge- and arc-WMFT is a 4-approximation of Chekuri, Mydlarz and Shepherd [39]. Vertex-WMFT has not been explicitly studied as far as we are aware, although we observe later (Proposition 6.11) that techniques of [39] yield a 5-approximation.

Results. Throughout the chapter, we use μ to denote the minimum capacity in the WMFT instance. Our first important technical contribution is to show that *iterated relaxation* yields an integral solution with optimal value or better but exceeding edge capacities additively — by 2 for edge-WMFT and by 6 for arc-WMFT and vertex-WMFT. This re-

solves a conjecture stated in Chekuri et al. [39] — that the "c-relaxed integrality gap" is 1 for some constant c — which we show for $c = 2$.[1] We feel we give a fairly clean illustration of the iterated relaxation paradigm since we do not require *uncrossing* as in the original applications [182, 148].

When the minimum capacity μ is $\Omega(\log |V|)$, randomized rounding gives a $1+O(\log |V|/\mu)$-approximation for WMFT. This suggests that it is easier to approximate as the minimum capacity increases. This was anticipated by Chekuri et al. [39], and indeed, by plugging our iterated rounding results into [39, Cor. 3.5] we get a $1+O(1/\sqrt{\mu})$-approximation when μ is suitably large. In this chapter, our first main result is an improvement on the best ratio known for edge-WMFT when $\mu \geq 2$.

Theorem 6.1. *For edge-WMFT, there are polynomial-time algorithms achieving (a) approximation ratio* 3 *for* $\mu \geq 2$*, and (b) approximation ratio* $(1 + 4/\mu + 6/(\mu^2 - \mu))$ *for general* μ*.*

A slight modification of Garg et al.'s hardness proof shows that for some $\epsilon > 0$, for all $\mu \geq 2$, there are instances with minimum capacity μ which are NP-hard to approximate within a ratio of $1 + \epsilon/\mu$; we detail this modification in Section 6.2.4. Thus (if P $\neq$ NP) the ratio in Theorem 6.1(b) is tight up to the constant in the $\frac{1}{\mu}$ term.

Our methodology for Theorem 6.1 is to decrease the additively-violating solution towards feasibility, without losing too much weight. Part (a) uses an argument of Cheriyan, Jordán and Ravi [42]. Part (b) relies on an auxiliary *covering* problem; every feasible cover, when subtracted from the +2-violating edge-WMFT solution, results in a feasible edge-WMFT solution. An approach due to Jain [124] shows that iterated LP rounding, applied to the auxiliary problem, leads to a provably low-weight integral solution for the covering problem. One particular property of the iterated rounding framework is crucial in obtaining Theorem 6.1(b). For a combinatorial optimization problem P with linear relaxation $\mathcal{L}$, we say that an α-approximation algorithm for P is *$\mathcal{L}$-relative* if it has the stronger property that its output is within a factor α of $\mathrm{OPT}(\mathcal{L})$. The crucial property is that the iterated rounding algorithm's guarantee is LP-relative, for the natural LP relaxation.

Second, using the same technology and variants of earlier techniques we obtain the following result.

Theorem 6.2. *For arc-WMFT and vertex-WMFT, there are polynomial-time* $1+O(1/\mu)$*-approximation algorithms.*

The most novel technical contribution to obtain these results is a counting lemma for these variants of WMFT.

[1] To say that the c-relaxed integrality gap of the maximization LP $\mathcal{L}$ is 1 means there is an integral solution of value at least $\mathrm{OPT}(\mathcal{L})$ but violating each constraint by up to $+c$.

Using the same framework we also derive results for multicommodity covering, which is the problem of finding integral s_i-t_i flows of value y_i so as to minimize $\sum_i w_i y_i$ and so that the amount of flow through each item (edge, arc, or vertex) is at *least* its capacity. In Section 6.3.1 we show the following.

Theorem 6.3. *For multicommodity integral covering in a tree, with either edge, arc, or vertex capacities, there is a polynomial-time* $1 + O(1/\mu)$*-approximation algorithms.*

Our final result maps out more of the landscape of tractable edge-WMFT instances. An *all-ror* instance is one in which, for some choice of root vertex, each commodity path either goes through the root or is *radial*, defined to be a path with one of its endpoints an ancestor of the other with respect to the root. For example, every instance of WMFT in which G is a spider is an all-ror instance.

Theorem 6.4. *All-ror edge-WMFT instances can be solved in strongly polynomial time.*

One way to view this result is as an efficient solution for a common generalization of b-matching and interval packing. Our proof of Theorem 6.4 is via a combinatorial reduction to *bidirected flow* [61]. This reduction also yields a polyhedral characterization of the feasible solutions for all-ror instances.

Related Work. Edge-WMFT appears in the literature under a variety of names including *cross-free-cut matching* [87] in the unit-capacity case and *packing of a laminar family* [42]. One generalization is the *demand* version [39] in which each commodity i is given a requirement r_i and we require $y_i \in \{0, r_i\}$ for each feasible solution.

Arc- and edge-WMFT with unit capacities are equivalent to the weighted *edge-disjoint paths* (EDP) problem on trees. The hardness results [4, 107] apply to EDP; however for fixed k, edge-EDP with at most k commodities is polynomial-time solvable by results of the graph minors project. See e.g. [179, §70.5] for further discussion.

The extreme points of the natural LP for edge-WMFT arise frequently in the literature of LP-based network design [39, 42, 63, 83, 84, 99, 124, 148, 182]. From this perspective, edge-WMFT is a natural starting point for an investigation of how large capacities/requirements affect the difficulty of weighted network design problems. We remark that the *min-cardinality k-edge-connected spanning subgraph* problem has a similar history to WMFT: a $1 + 2/(k+1)$-approximation was given by Cheriyan & Thurimella [43], $(1+\epsilon/k)$-hardness of approximation was shown by Gabow et al. [84], and the best current algorithm for the problem, due to Gabow & Gallagher [83], uses iterated rounding.

Organization of the Chapter. Section 6.1.1 contains some basic definitions and notation. In Section 6.2 we prove Theorem 6.1 and the complementary hardness result. In Section 6.3 we prove Theorem 6.2. In Section 6.4 we prove Theorem 6.4. In Section 6.4.1, we state our polyhedral results. Finally, we suggest some directions for future work in Section 6.5.

6.1.1 WMFT Formulation

In this section we formulate edge-WMFT; we will also use the analogous formulations for arc-WMFT and vertex-WMFT. We define the commodities by a set of *demand edges* $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ on vertex set V with a weight w_d assigned to each demand edge $d \in D$; this is without loss of generality since the supply graph and demand edges are undirected. (In the arc-capacitated case, D is a set of arcs.) Since we discuss WMF only on trees, each commodity has a unique path along which flow is sent. For each demand edge d, let its *demand path* p_d be the unique path in G joining the endpoints of d. We thus may represent a multicommodity flow by a vector $\{y_d\}_{d \in D}$ where y_d is the amount of commodity d that is routed (along p_d). Then a flow y is *feasible* for edge-WMFT if it meets the two conditions

$$\forall d \in D : \qquad y_d \geq 0 \tag{6.1}$$

$$\forall e \in E : \qquad \sum_{d : e \in p_d} y_d \leq c_e \tag{6.2}$$

The objective of edge-WMFT is to find a feasible *integral* y that maximizes $w \cdot y$.

6.2 Improved Approximation for Edge-WMFT

In this section we obtain a $\min\{3, 1+4/\mu+6/(\mu^2-\mu)\}$-approximation algorithm for edge-WMFT, assuming $c_e \geq \mu \geq 2$ for each edge e. The algorithm uses the iterated rounding paradigm, which was used first by Jain [124] and more recently by Lau et al. [148] and others [10, 83, 182] for network design. The idea is that in each iteration, we use an extreme fractional optimum y^* of the natural LP to develop an integral solution. If some demand edge d has value 1 or greater in y^*, we route the integer part and decrease the capacity of edges on p_d accordingly. If $\mathbf{0} < y < \mathbf{1}$ we perform a *relaxation* step; a counting argument (Section 6.2.3) guarantees that a particular relaxation can always be performed. At the end, we obtain an integral solution which exceeds the capacity of each edge by at most 2, and which has weight at least as large as that of an optimal feasible solution. In Sections 6.2.1 and 6.2.2 we show how to compute high-weight feasible solutions from this +2-violating solution. In Section 6.2.4 we show it is NP-hard to $(1+\epsilon/\mu)$-approximate edge-WMFT, fox any fixed $\mu \geq 2$, and for some ϵ independent of μ.

The natural LP relaxation of edge-WMFT, which we denote by ($\mathcal{F}$), is as follows:

$$\text{maximize } w \cdot y \text{ subject to (6.1) and (6.2).} \tag{$\mathcal{F}$}$$

This program has a linear number of variables and constraints, and thus can be solved in polynomial time. Any integral vector y is feasible for ($\mathcal{F}$) iff it is feasible for the edge-WMFT instance. However, the linear program has fractional extreme points in general,

and thus solving the LP does not give us the type of solution we seek. Nonetheless, optimal solutions to the LP have certain properties that permit an iterated rounding approach, such as the following.

Lemma 6.5. *Let y^* be an optimal solution to $(\mathcal{F})$, define* $\text{OPT} = w \cdot y^*$*, and suppose $y^*_d \geq t$ for some $d \in D$ and some integer $t \geq 1$. Reduce the capacity of each edge $e \in p_d$ by t and let* OPT' *denote the new optimal value of $(\mathcal{F})$. Then* $\text{OPT}' = \text{OPT} - tw_d$.

Proof. Let z denote the vector such that $z_d = t$ and $z_{d'} = 0$ for each $d' \neq d$. Then it is easy to see that $y^* - z$ is feasible for the new LP, and hence $\text{OPT}' \geq w \cdot (y^* - z) = \text{OPT} - tw_d$. On the other hand, where y' denotes the optimal solution to the new LP, it is easy to see that $y' + z$ is feasible for the original LP; so $\text{OPT} \geq \text{OPT}' + tw_d$. Combining these inequalities, we are done. □

From now on, let OPT denote the optimal value to $(\mathcal{F})$.

In general terms, our iterated rounding approach works on the following principles. Define the following restricted version of $(\mathcal{F})$:

$$\text{maximize } w \cdot y \text{ subject to (6.1) and (6.2) and } \forall d \in D : y_d \leq 1. \qquad (\mathcal{F}_1)$$

Assume for the moment that $(\mathcal{F}_1)$ also has optimal value OPT. We iteratively build an integral solution to $(\mathcal{F}_1)$ with value at least equal to OPT. The first step in each iteration is to solve $(\mathcal{F}_1)$, obtaining solution y^*. If $y^*_d = 0$ for some demand edge d, then we can discard d without affecting the optimal value of $(\mathcal{F}_1)$. If $y^*_d = 1$ for some d, then we can route one unit of flow along p_d and update capacities accordingly. Similar to Lemma 6.5, the optimal LP value will drop by an amount equal to the weight of the flow that was routed. If neither of these cases applies, we use the following lemma, whose proof appears in Section 6.2.3.

Lemma 6.6. *Suppose that y^* is an extreme point solution to $(\mathcal{F})$, and that $0 < y^*_d < 1$ for each demand edge $d \in D$. Then there is an edge $e \in E$ so that $|\{d \in D : e \in p_d\}| \leq 3$.*

Our algorithm discards the capacity constraint (6.2) for e from our LP. We call this *contracting* e because the effect is the same as if we had merged the two endpoints of e in the tree G. Pseudocode for our algorithm is given in Figure 6.1.

To justify our assumption that $(\mathcal{F})$ and $(\mathcal{F}_1)$ have the same optimal value, we preprocess the problem as follows. First, we compute any optimal solution y^* to $(\mathcal{F})$. Then we route the integer part $\lfloor y^* \rfloor$ of the solution (i.e., we initialize $\widehat{y} = \lfloor y^* \rfloor$) and reduce each capacity c_e by $\sum_{d:e \in p_d} \lfloor y^*_d \rfloor$. The residual problem has $\langle y^* \rangle := y^* - \lfloor y^* \rfloor$ as an optimal solution, and since $0 \leq \langle y^* \rangle \leq 1$, our assumption is justified.

IteratedFlowSolver

1. Set $\widehat{y} = \mathbf{0}$
2. If $D = \varnothing$ terminate and return $\widehat{y}$
3. Let y^* be an optimal extreme point solution to ($\mathcal{F}_1$)
4. For each d such that $y^*_d = 0$, discard d
5. For each d such that $y^*_d = 1$, increase $\widehat{y}_d$ by 1, decrease c_e by 1 for each $e \in p_d$, and discard d
6. If neither step 4 nor 5 applied, find e as specified by Lemma 6.6 and contract e
7. Go to step 2

Figure 6.1: The iterated rounding algorithm.

Assuming Lemma 6.6, we now prove the main properties of our iterated rounding algorithm: it runs in polynomial time, it exceeds each capacity by at most 2, and it produces a solution of value at least OPT.

Property 1. IteratedFlowSolver *runs in polynomial time.*

Proof. Recall that ($\mathcal{F}$) and ($\mathcal{F}_1$) can be solved in polynomial time. In each iteration we decrease $|D| + |E|$, so polynomially many iterations occur, and the result follows. □

Property 2. *The integral solution computed by* IteratedFlowSolver *violates each capacity constraint* (6.2) *by at most* 2.

Proof. Consider what happens to any given edge e during the execution of the algorithm. In the preprocessing and in each iteration, the flow routed through e equals the decrease in its residual capacity. If in some iteration, e's residual capacity is decreased to 0, all demand paths through e will be discarded in the following iteration. Thus if e is not contracted, its capacity constraint (6.2) will be satisfied by the final solution.

The other case is that we contract e in step 6 of some iteration because e lies on at most 3 demand paths. The residual capacity of e is at least 1, and at most one unit of flow will be routed along each of these 3 demand paths in future iterations. Hence the final solution violates (6.2) for e by at most +2. □

Property 3. *The integral solution computed by* IteratedFlowSolver *has objective value at least equal to* OPT.

Proof. When we contract an edge e we just remove a constraint from $(\mathcal{F}_1)$, which cannot decrease the optimal value of $(\mathcal{F}_1)$ since it is a maximization LP. In every other iteration and in preprocessing, Lemma 6.5 implies that the LP optimal value drops by an amount equal to the increase in $w \cdot \widehat{y}$. When termination occurs, the optimal value of $(\mathcal{F}_1)$ is 0. Thus the overall weight of flow routed must be at least as large as the initial value of OPT. □

6.2.1 Minimum Capacity $\mu = 2$

As per Property 2, our iterated solver may exceed some of the edge capacities. When $c_e \geq \mu \geq 2$ for each edge e we can invoke the following theorem, which appears as Thm. 6 in [42], to produce a high-weight feasible solution.

Theorem 6.7 (Cheriyan, Jordán, Ravi)**.** *Suppose that $\widehat{y}$ is a nonnegative integral vector so that for each edge e, the constraint (6.2) is violated by at most a* multiplicative *factor of 2 by $\widehat{y}$. Then in polynomial time, we can find an integral vector y' with $w \cdot y' \geq (w \cdot \widehat{y})/3$, and $0 \leq y' \leq \widehat{y}$, and such that y' satisfies all constraints (6.2).*

The algorithm as literally described in [42] is actually pseudo-polynomial, but it is straightforward to modify it to have polynomial running time; we defer the proof for a moment. Assuming this fact, we now prove part (a) of Theorem 6.1.

Proof of Theorem 6.1(a). Let $\widehat{y}$ be the output of IteratedFlowSolver. Since $c_e \geq 2$ for each edge e, and since by Property 2 each edge's capacity is additively violated by at most +2, Theorem 6.7 applies. Thus y' is a feasible solution to the edge-WMFT instance with objective value $w \cdot y' \geq w \cdot \widehat{y}/3 \geq \text{OPT}/3$, using Property 3. Finally, since $(\mathcal{F})$ is an LP-relaxation of the edge-WMFT problem, OPT is at least equal to the optimal edge-WMFT value, and so y' is a 3-approximate feasible integral solution. □

Now we detail the polynomial-time implementation of Cheriyan et al.'s algorithm. It applies to a multiset $\mathcal{P}$ of demand edges, and has running time polynomial in $|\mathcal{P}|$. Directly applying their algorithm to the multiset containing y_d copies of each demand edge d would give us a correct answer y', but the running time would be pseudo-polynomial in the length of the edge-WMFT instance's description, i.e., it would run in $\text{poly}(\sum_d y_d) = \text{poly}(|E| \max_e c_e)$ time, whereas we can only allow polylogarithmic dependence on $\max_e c_e$.

To modify this for our purposes, define $\widetilde{y}_d = \lfloor y_d/2 \rfloor$ for each demand edge d and let $\overline{y} = y - 2\widetilde{y}$, so $\overline{y}$ is a 0-1 vector. Define, for each edge e, $c'_e := c_e - \sum_{d:e \in p_d} \widetilde{y}_d$ and notice that

$\overline{y}$ violates the capacities c' by at most a factor of 2. Apply the algorithm of Theorem 6.7 to $\overline{y}$, producing a solution $\overline{y}'$ which is feasible for c' with $w \cdot \overline{y}' \geq w \cdot \overline{y}/3$. This application is polynomial-time since $\sum_d \overline{y}_d \leq |D|$. Define $y' = \overline{y}' + \widetilde{y}$; then it is immediate that y' is feasible for capacities c. Furthermore,

$$w \cdot y' = w \cdot (\overline{y}' + \widetilde{y}) \geq \frac{1}{3} w \cdot \overline{y} + w \cdot \widetilde{y} \geq \frac{1}{3} w \cdot (\overline{y} + 2\widetilde{y}) = (w \cdot y)/3,$$

as needed.

6.2.2 Arbitrary Minimum Capacity

Given the infeasible solution $\widehat{y}$ produced by IteratedFlowSolver, we want to reduce $\widehat{y}$ in a minimum-weight way so as to attain feasibility. For each edge e let $f_e = \max\{0, \sum_{d:e\in p_d} \widehat{y}_d - c_e\}$, i.e. f_e is the amount by which $\widehat{y}$ violates the capacity of e. Note now that a reduction z with $0 \leq z \leq \widehat{y}$ makes $\widehat{y} - z$ a feasible (integral) edge-WMFT solution if and only if z is a feasible (integral) solution to the following linear program.

$$\text{minimize } w \cdot z \text{ subject to } 0 \leq z \leq \widehat{y} \text{ and } \forall e \in E : \sum_{d:e\in p_d} z_d \geq f_e. \qquad (\mathcal{F}_c)$$

Notice that $(\mathcal{F}_c)$ is a covering analogue of $(\mathcal{F})$ (with added upper bounds). Furthermore, Jain's iterated rounding framework [124] gives an $(\mathcal{F}_c)$-relative 2-approximation algorithm to find an optimal integral solution.

Theorem 6.8. *There is a polynomial-time algorithm which returns an integral feasible solution $\widehat{z}$ for $(\mathcal{F}_c)$ such that $w \cdot \widehat{z} \leq 2\mathrm{OPT}(\mathcal{F}_c)$.*

Proof. The idea is very similar to the main result of [124] but simpler in that no *uncrossing* is needed (in [124] this simplifies an arbitrary LP basis into one having a tree-like laminar structure), because we already have a tree structure. Hence we only sketch the details. In each iteration, we obtain an extreme point optimal solution z^* to the linear program $(\mathcal{F}_c)$. We increase $\widehat{z}$ by the integer part of z^* and accordingly decrease the requirements f. If $z^*_d = 0$, d is discarded. Finally if $0 < z^* < 1$ a token redistribution argument of Jain shows that some $d^* \in D$ has $z^*_{d^*} \geq 1/2$. In this case we increase $\widehat{z}_{d^*}$ by 1 and update the requirements accordingly. Standard arguments then give the claimed bound on the cost of $\widehat{z}$ and polynomial running time. □

Here is how we use Theorem 6.8 to approximate edge-WMFT instances on trees.

Proof of Theorem 6.1(b). Notice that $z = \frac{2}{\mu+2}\widehat{y}$ is a feasible fractional solution to ($\mathcal{F}_c$) by the definition of f. Hence, the optimal value of ($\mathcal{F}_c$) is at most $\frac{2}{\mu+2}\widehat{y}\cdot w$. Thus the solution $\widehat{z}$ produced by Theorem 6.8 satisfies $\widehat{z}\cdot w \le \frac{4}{\mu+2}\widehat{y}\cdot w$, so $\widehat{y}-\widehat{z}$ is a feasible solution to the edge-WMFT problem, with $w\cdot(\widehat{y}-\widehat{z}) \ge (1-\frac{4}{\mu+2})\widehat{y}\cdot w \ge (1-\frac{4}{\mu+2})\mathrm{OPT}$. This gives us a $1/(1-\frac{4}{\mu+2}) = 1+4/\mu+O(1/\mu^2)$ approximation algorithm for edge-WMFT.

To obtain the exact bound claimed in Theorem 6.1(b), we refine this slightly by taking a two-round approach. In the first round we set f_e to be the characteristic vector of those edges which $\widehat{y}$ violates by $+2$, obtaining $\widehat{y}' := \widehat{y}-\widehat{z}$. Then $\widehat{y}'$ has only $+1$ additive violation, and the same reasoning as before shows $\widehat{y}'\cdot w \ge (1-\frac{2}{\mu+2})\mathrm{OPT}$. The second round analogously extracts from $\widehat{y}'$ a solution with $+0$ violation, i.e. a feasible solution, with weight at least $(1-\frac{2}{\mu+1})\widehat{y}'\cdot w$. This gives approximation ratio $1/(1-\frac{2}{\mu+2})(1-\frac{2}{\mu+1}) = 1+4/\mu+6/(\mu^2-\mu)$, as desired. □

6.2.3 Proof of Lemma 6.6

First, we need the following simple counting argument.

Lemma 6.9. *Let T be a tree with n vertices and let n_i denote the number of its vertices that have degree i. Then $n_1 > (n-n_2)/2$.*

Proof. Using the handshake lemma and the fact that T has $n-1$ edges, we have $2(n-1) = \sum_i i\cdot n_i$. But $\sum_i i\cdot n_i \ge n_1+2n_2+3(n-n_1-n_2) = 3n-2n_1-n_2$ and hence $2n-2 \ge 3n-2n_1-n_2$. Solving for n_1 gives $n_1 \ge (n-n_2+2)/2$ as needed. □

Proof of Lemma 6.6. Using basic facts from polyhedral combinatorics, it follows that there exists a set $E^* \subset E$ of edges with $|E^*| = |D|$ such that y^* is the unique solution to

$$\sum_{d\in D: e\in p_d} y_d = c_e \qquad \forall e \in E^*. \tag{6.3}$$

In particular, the characteristic vectors of the sets $\{d : e \in p_d\}$ for $e \in E^*$ are linearly independent.

Contract each edge of $E\backslash E^*$ in (V,E), resulting in the tree $T' = (V', E^*)$; call elements of V' *nodes*. We now use a counting argument to establish the existence of the desired edge e within E^*. We call the two ends of each $d \in D$ *endpoints* and say that node $v' \in V'$ *owns* k endpoints when the degree of v' in (V', D) is k.

First, consider any node $v' \in V'$ that has degree 2 in T'; let e_1, e_2 be its incident edges in T'. If v' owns no endpoints then $\{d : e_1 \in p_d\} = \{d : e_2 \in p_d\}$, contradicting linear

independence. If v' owns exactly one endpoint, the symmetric difference $\{d : e_1 \in p_d\} \triangle \{d : e_2 \in p_d\}$ consists of a single demand edge; but since y^* satisfies (6.3), $\mathbf{0} < y^* < \mathbf{1}$, and c is integral, this is a contradiction. Hence v' owns two or more endpoints.

If there exists a leaf node v' of T' that owns at most 3 endpoints then we are done, since this implies that the edge of E^* incident to v', viewed in the original graph, lies on a most 3 demand paths. Otherwise, we apply a counting argument to T', seeking a contradiction. Let n_i denote the number of nodes of T' of degree i. Then our previous arguments establish that the total number of endpoints is at least $4n_1 + 2n_2$. Lemma 6.9 then shows that the total number of endpoints is more than $2(|V'| - n_2) + 2n_2 = 2|V'| > 2|E^*| = 2|D|$. This is the desired contradiction, since there are only $2|D|$ endpoints in total. □

We remark that Lemma 6.6 is tight in the following sense: if we replace the bound $|\{d \in D : e \in p_d\}| \leq 3$ with $|\{d \in D : e \in p_d\}| \leq 2$, the resulting statement is false. An example of an extreme point solution for which the modified version fails, due to Cheriyan et al. [42], is given in Figure 6.2.

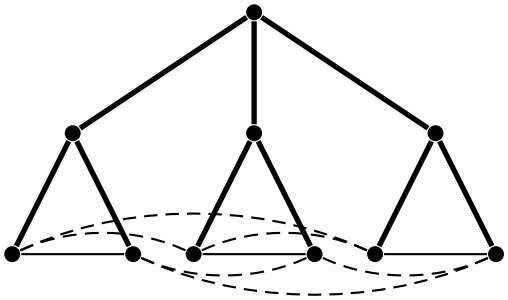


Figure 6.2: An extreme point solution to ($\mathcal{F}$). There are 9 edges in the supply graph, shown as thick lines; each has capacity 1. There are 9 demand edges, shown as thin lines; the solid ones have value 1/2, and the dashed ones have value 1/4. This is a tight example for Lemma 6.6 because each edge lies on at least three demand paths.

6.2.4 Hardness of MFT with Constant Lower Bounds

We use MFT to stand for the special case of WMFT where all costs are 1.

Theorem 6.10. *For some $\epsilon > 0$, for all integers $\mu \geq 2$, it is* NP*-hard to approximate edge-MFT to a factor of less than $1 + \epsilon/\mu$, even when restricted to instances where all capacities are at least μ.*

We remark that this directly implies similar hardness for vertex-MFT, and that the proof can be modified to give similar hardness for arc-MFT.

Proof. We extend Garg et al.'s original proof [87, Thm. 4.2] that edge-MFT is APX-complete. Their reduction is from *3-bounded maximum three-dimensional matching* (MAX 3DM–3); an instance consists of three disjoint sets X, Y, Z of size n and a family of triples $S \subset X \times Y \times Z$ such that each element is in at most 3 triples. The objective is to find a maximum-size disjoint set of triples from S. If $|S| < n$, elements appearing in no triples can be discarded, so WOLOG $|S| \geq n$.

Kann [125] showed that MAX 3DM–3 is MAX SNP-complete, hence by the PCP theorem, for some $\delta > 0$ it is NP-hard to approximate it within a ratio of $1 + \delta$. A greedy argument easily shows the optimal value of MAX 3DM–3 is always at least $|S|/7$. Hence it is NP-hard to additively approximate MAX 3DM–3 within $|S|\delta/7$.

The construction given in [87] is an instance of MFT which has optimal value $t + |S|$ where t is the maximum number of disjoint triples in S. Let (V, E) denote the tree obtained from this construction; it satisfies $|E| = 3(|S| + n)$. To obtain a lower bound μ on capacity, we perform the following for each edge $uv \in E$: (1) add a new leaf u' and a new edge uu' to the tree; (2) add a new unit-weight demand edge $u'v$ to D; (3) increase the capacity of uv by μ and set the capacity of uu' to be μ. We illustrate the overall modification in Figure 6.3.

An easy alteration argument shows that there exists an optimal solution y to the modified MFT problem such that $y_d = \mu$ for each new demand edge d. It then follows that its optimal value is $t + |S| + \mu|E|$. Furthermore, since $t + |S| + \mu|E| \leq (6\mu + 2)|S|$, approximating the MFT instance to a ratio less than $1 + \frac{|S|\delta/7}{(6\mu+2)|S|} = 1 + \Theta(1/\mu)$ is NP-hard. □

6.3 Vertex-WMFT and Arc-WMFT

The natural LP relaxation for edge-WMFT, given in Section 6.1.1, admits straightforward analogues for vertex- and arc-WMFT; we replace constraint (6.2) with a vertex- or arc-capacity constraint. Let us denote these LPs by *vertex*-($\mathcal{F}$) and *arc*-($\mathcal{F}$). Analogously to the methods of Section 6.2, the crux of our work can be performed under the assumption that $y_d \leq 1$ for each d, so we similarly define *vertex*-($\mathcal{F}_1$) and *arc*-($\mathcal{F}_1$). As a preliminary step, we show that the framework of Chekuri et al. [39] for edge- and arc-WMFT extends to vertex-WMFT.

Proposition 6.11. *There is a 5-approximation algorithm for vertex-WMFT.*

Proof. Since the whole proof for edge-WMFT is lengthy and we modify it only slightly, our presentation assumes the reader is already familiar with the approach in [39]. We define Binned Tree Colouring the same way except that $n_v \leq c_v$ for each leaf vertex v, and we respect vertex capacities.

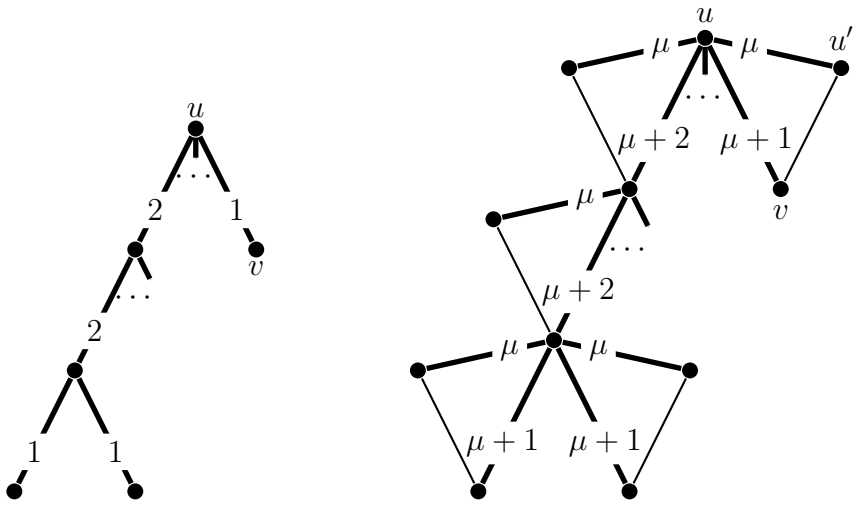


Figure 6.3: Left: a portion of the capacitated tree used in the hardness construction of [87]. Demand edges are not shown. Right: the modified capacitated tree (thick) used in the proof of Theorem 6.10, and the new demand edges (thin). The old demand edges are still not shown, but present.

Now we claim that an analogue of [39, Thm. 2.2] holds with $5k$ colours instead of $4k$. The main thing to check is that we can complete the partial colouring returned by the recursive call — checking these details pertains to the last paragraph of their proof. An edge e joining v_i and v_j cannot use

- any colour already assigned within e's bin at v_i, of which there are less than $2k$
- any colour already assigned within e's bin at v_j, of which there are less than $2k$
- any colour which is already assigned to c_v edges passing through v, of which there are less than k, since at most $k \cdot c_v$ edges pass through v

Hence one of the $5k$ colours is still available. □

The key in our approaches to arc-WMFT and vertex-WMFT are analogues of Lemma 6.6; here is the latter.

Lemma 6.12. *(Counting Lemma, Vertex Version) Suppose that y^* is an extreme point solution to vertex-($\mathcal{F}_1$), and that $0 < y^*_d < 1$ for each demand edge $d \in D$. Then there is a vertex $v \in V$ so that $|\{d \in D : v \in p_d\}| \leq 7$.*

Proof. Using basic facts from polyhedral combinatorics, it follows that there exists a set $V^* \subset V$ of *tight vertices* with $|V^*| = |D|$ such that y^* is the unique solution to

$$\sum_{d \in D : v \in p_d} y_d = c_v \qquad \forall v \in V^*. \tag{6.4}$$

In particular, the characteristic vectors of the sets $\{d : v \in p_d\}$ for $v \in V^*$ are linearly independent.

We now introduce several properties that hold without loss of generality (that is to say, without affecting the fact that y^* is the unique solution to Equation (6.4)). First, for each demand edge $d = yz$, we may assume both y and z are tight, since otherwise we can replace d by $y'z'$ where y' is the closest tight vertex to y on p_d and z' is defined similarly (note, possibly $y' = z'$). Second, every degree-1 vertex is tight, since we can iteratively delete degree-1 vertices that are non-tight. If v is a degree-2 vertex with neighbours u and w, define *contracting the vertex* v to mean removing v from the graph and making u, w adjacent. Third, every degree-2 vertex is tight, since we can iteratively contract degree-2 vertices that are non-tight.

Now we will apply a counting argument. Let t_i denote the number of tight vertices of degree i, and u_i denote the number of non-tight vertices of degree i. So $u_1 = u_2 = 0$ and all other values are non-negative. We give the high-level argument and then fill in the details. Say an edge uv of the supply graph is *special* if both u and v are degree-2 (tight) vertices, and let s be the total number of special edges. We re-use the notion from the proof of Lemma 6.6 that each demand edge $d = yz$ has two endpoints, one *owned* by y and the other *owned* by z. (I.e. the number of endpoints owned by v is equal to its degree in (V, D) where a loop counts twice to the degree.) Let $t_{\geq 3} = \sum_{i \geq 3} t_i$ and define $u_{\geq 3}$ similarly.

Claim 6.13. *The degree-2 tight vertices own at least* $2s$ *endpoints.*

Claim 6.14. $s \geq t_2 - (u_{\geq 3} + t_1 + t_{\geq 3} - 1)$.

Claim 6.15. $4t_1 \geq 4t_{\geq 3} + 4u_{\geq 3}$.

Given these claims, we combine them as follows. Count the number w of endpoints owned by degree-1 (tight) vertices; this value satisfies

$$\begin{aligned}
w &\leq 2|V^*| - 2s && \text{by Claim 6.13 and since } |D| = |V^*| \\
&= 2(t_1 + t_2 + t_{\geq 3}) - 2s \\
&\leq 2(t_1 + t_2 + t_{\geq 3}) - 2(t_2 - (u_{\geq 3} + t_1 + t_{\geq 3} - 1)) && \text{by Claim 6.14} \\
&= -2 + 4t_1 + 4t_{\geq 3} + 2u_{\geq 3} \\
&\leq 8t_1 - 2. && \text{by Claim 6.15}
\end{aligned}$$

So $w < 8t_1$, in particular there is some degree-1 tight vertex v which owns at most 7 endpoints. Clearly $|\{d \in D : v \in p_d\}| \leq 7$, which completes the proof of Lemma 6.12. We now move on to the supporting claims.

Proof of Claim 6.13. By definition of "special edge," note that the special edges can be partitioned into (maximal) paths. Let $v_0, v_1, \ldots, v_k$ be one such path, i.e. suppose the edges $v_{i-1}v_i$ are special for $1 \leq i \leq k$, so each v_i is a degree-2 tight vertex. We will show that the vertices of the path own at least $2k$ endpoints; then by adding over all paths it follows that the set of all degree-2 vertices own at least $2s$ endpoints.

Define v_{-1} to be the neighbour of v_0 which is not equal to v_1 and define v_{k+1} similarly. For $1 \leq i \leq k$, say that a demand edge $d = yz$ has a "left endpoint at v_i" if one of y, z is equal to v_i and $v_{i-1} \notin p_d$; define right endpoints similarly. Notice that the number of endpoints owned by $\{v_i \mid 0 \leq i \leq k\}$ equals the total number of left and right endpoints therein.

For each $1 \leq i \leq k$, since the constraints (6.4) for tight vertices v_{i-1} and v_i are linearly independent, there is at least one demand path p_d containing exactly one of v_{i-1} and v_i. Since the vertex capacities are integral and both are tight, in fact there are at least 2 demand paths containing exactly one of v_{i-1} or v_i. Thus the number of right endpoints at v_{i-1} plus the number of left endpoints at v_i is at least 2. Adding over all k, we are done. □

Proof of Claim 6.14. Define T' to be the tree obtained from T by contracting all degree-2 tight vertices; viewing this process in reverse, T can be obtained from T' by subdividing its edges. For each edge of T', if it is subdivided $x \geq 1$ times, that corresponds to a path of $x - 1$ special edges in T. Note T' has $u_{\geq 3} + t_1 + t_{\geq 3}$ vertices and thus $u_{\geq 3} + t_1 + t_{\geq 3} - 1$ edges. Hence the number of special edges is at least

$$s \geq t_2 - (u_{\geq 3} + t_1 + t_{\geq 3} - 1). \quad \square$$

Proof of Claim 6.15. Using the handshake lemma (as in the proof of Lemma 6.9) we know that $t_1 = 2 + \sum_{i \geq 3}(i-2)(t_i + u_i)$, and the desired result follows. □

(End of proof of Lemma 6.12.) □

Now we give the analogous result for arc-WMFT.

Lemma 6.16. *(Counting Lemma, Arc Version) Suppose that y^* is an extreme point solution to arc-($\mathcal{F}_1$), and that $0 < y^*_d < 1$ for each demand edge $d \in D$. Then there is an arc $a \in A$ so that $|\{d \in D : a \in p_d\}| \leq 7$.*

Proof. Let $|A^*|$ denote a maximum size linearly independent set of tight arcs, so $|D^*| = |A^*|$. Whenever both uv and vu are non-tight, we contract $\{u, v\}$. Furthermore, for each edge $\{u, v\}$ such that both uv and vu are tight, subdivide uv with a new vertex w so that $c_{uw} = c_{uv}$, $c_{vw} = c_{vu}$, and consider uw and vw as tight instead of uv and vu. What results is a directed tree where every edge is tight in exactly one direction. Its vertex set V^* satisfies $|V^*| = |A^*| + 1$.

In (V^*, A^*) let n_1 be the number of degree-1 vertices, n_2 the number of degree-2 vertices, and $n_{\geq 3}$ the number of vertices of degree at least 3. Define an edge to be *special* if both endpoints have degree-2, as in the proof of Lemma 6.12. Analogously to Claim 6.14, we have $s \geq n_2 - n_1 - n_{\geq 3} + 1$. Analogously to Claim 6.13, we have $n_1 \geq n_{\geq 3}$. We will show moreover that the degree-2 vertices own at most $2s$ endpoints; then we will be done since the number of endpoints owned by degree-1 vertices is at most

$$2|A^*| - 2s = 2(n_1 + n_2 + n_{\geq 3} - 1) - 2s \leq 4n_1 + 4n_{\geq 3} - 4 < 8n_1.$$

To show that the degree-2 vertices own at least $2s$ endpoints, we proceed along the lines of the proof of Claim 6.13. Let $P := v_0, v_1, \ldots, v_{k+1}$ be a sequence of arcs in (V^*, A^*) such that $\delta(v_i) = 2$ iff $1 \leq i \leq k$. (P is a path when ignoring directions, but P is not in general a dipath since the orientations of arcs on P is arbitrary.) This path P contains $k - 1$ special edges, and $k + 1$ edges in total. We will show that the vertices $\{v_1, \ldots, v_k\}$ own at least $2k - 2$ endpoints; then by adding over all such paths P we will have shown that the degree-2 vertices own at least $2s$ endpoints.

For convenience we call arcs in P of the form $v_i v_{i-1}$ *leftwards*, and other arcs *rightwards*. Similarly, for any demand arc $d \in D^*$ such that $p_d \cap P \neq \varnothing$, either p_d intersects only leftwards arcs or rightwards arcs; we think of p_d as *leftwards* or *rightwards* correspondingly. Let there be r rightwards arcs and $k + 1 - r$ leftwards arcs on this path. We will show that $\{v_1, \ldots, v_k\}$ own at least $2r - 2$ endpoints of rightward demand arcs and at least $2k - 2r$ endpoints of leftward demand arcs, which gives the desired result.

Let $v_i v_{i+1}$ and $v_j v_{j+1}$ with $k \geq j > i \geq 0$ be rightwards arcs of P such that all intermediate arcs of P are leftwards (i.e., "consecutive" rightwards arcs). By linear independence there is at least one rightwards demand path using exactly one of $v_i v_{i+1}$ or $v_j v_{j+1}$; in fact since both arcs are tight with integral capacities and y^* is fractional, there are at least two such rightwards demand paths. This implies that the vertices $\{v_u\}_{u=i+1}^{j}$ own at least 2 endpoints of rightwards demand paths. By considering all possible choices of i, j it follows that $\{v_1, \ldots, v_k\}$ own at least $2r - 2$ endpoints of rightwards demand paths; an analogous argument works for leftwards demand paths. □

We get two algorithmic corollaries from Lemma 6.12. The first one says that we can efficiently compute super-optimal +6-violating solutions.

Corollary 6.17. *There is a polynomial-time algorithm which, when given a vertex-WMFT or arc-WMFT instance, produces y such that $w \cdot y$ is at least as large as the optimum for the instance, and such that the y is feasible when each capacity is increased by* 6.

Proof. We use IteratedFlowSolver but instead of discarding the capacity constraint for an edge, we discard the capacity constraint for the vertex/arc specified by Lemma 6.12. (Alternatively, we set the capacity to be $+\infty$.) Analogously to the proof of Property 2, we get a $+6$-violating solution. □

The second corollary to Lemma 6.12 deals with vertex-($\mathcal{F}_c$) and arc-($\mathcal{F}_c$), i.e. multicommodity covering problems. We prove only the vertex version; the arc version is analogous.

Corollary 6.18. *There is an algorithm which computes an integral solution z to vertex-($\mathcal{F}_c$) such that $w \cdot z \leq 7\text{OPT}(\textit{vertex-}(\mathcal{F}_c))$.*

Proof. This follows almost immediately from the iterated rounding framework [124]. It suffices to show any nonzero extreme point solution z^* to vertex-($\mathcal{F}_c$) has $z^*_d \geq 1/7$ for some d.

We need only consider the case that $0 < z^*_d < 1$ for all d; then z^* is the unique solution to Equation (6.4) (with c replaced by f), for some set V^* with $|V^*| = |\,\mathtt{supp}(z^*)|$. So using iterated rounding as in Lemma 6.12, there is a tight vertex v on at most 7 demand paths. Since that vertex is tight, $\sum_{d:v\in p_d} z^*_d = f_v \geq 1$. Thus some d with $v \in p_d$ has $z^*_d \geq 1/7$, as needed. □

This gives us the main result.

Theorem 6.2. *There are $1+O(1/\mu)$-approximation algorithms for arc-WMFT and vertex-WMFT.*

Proof. We prove the vertex-WMFT version; the arc version is analogous. We do not attempt to optimize the constants. As in the proof of Theorem 6.1, it is no loss of generality to assume the additional constraints $y_d \leq 1$, hence we work with vertex-($\mathcal{F}_1$) instead of vertex-($\mathcal{F}$).

Corollary 6.17 gives us a solution y to vertex-($\mathcal{F}_1$) with $w \cdot y \geq \text{OPT}$ and such that y violates each vertex capacity by at most $+6$. Let f_v equal the amount by which the capacity for v is violated, or 0 if the capacity is not violated. Apply Corollary 6.18 to vertex-($\mathcal{F}_c$) for this choice of f and $\widehat{y} = y$; we get a z such that $w \cdot z \leq 7\text{OPT}(\text{vertex-}(\mathcal{F}_c))$. Moreover, $y - z$ is feasible for vertex-($\mathcal{F}_1$).

Just as before, it is easy to verify that $\frac{6}{\mu+6}y$ is a feasible solution to vertex-($\mathcal{F}_c$). Hence we have

$$w\cdot(y-z) \geq w\cdot y - 7\mathrm{OPT}(\text{vertex-}(\mathcal{F}_c)) \geq w\cdot y - 7\frac{6}{\mu+6}w\cdot y \geq \left(1-\frac{42}{\mu+6}\right)\mathrm{OPT}(\text{vertex-}(\mathcal{F}_1)).$$

For large μ, this implies that y is an $1+O(1/\mu)$-approximately optimal solution to the vertex-WMFT instance. For small μ, we know there is a 5-approximation algorithm due to Proposition 6.11. Combining these facts, we are done. □

Remark 6.19. *A sensible generalization of the problems we have considered is {arc,vertex}-WMFT, where there are capacities on both arcs and vertices. The method of Chekuri et al. [39] gives a constant-factor approximation for this problem, however it is not clear if the analogue of the counting lemma (Lemma 6.6) is true in this case.*

6.3.1 Multicommodity Covering

Our framework also shows that covering analogues of WMFT can be approximated within ratio $1+O(1/\mu)$, provided that there are no upper bounds on variables. We use the specific example of edge-WMFT but the same result holds for the arc- and vertex-versions. We use the following natural LP relaxation for multicommodity covering in a tree:

$$\text{minimize } w\cdot z \text{ subject to } 0 \leq z \text{ and } \forall e\in E : \sum_{d:e\in p_d} z_d \geq f_e. \qquad (\mathcal{F}'_c)$$

We use μ to denote $\min_e f_e$. Then the main result is as follows.

Theorem 6.3. *There is a $(\mathcal{P}'_c)$-relative $(1+2/\mu)$-approximation algorithm for multicommodity integer covering in a tree.*

Proof. First, we need the following algorithmic result on additive violation for the covering problem under consideration.

Corollary 6.20. *There is a polynomial-time algorithm which produces $z \geq 0$ such that $w\cdot z \leq \mathrm{OPT}(\mathcal{F}'_c)$ and $\forall e\in E : \sum_{d:e\in p_d} z_d \geq f_e - 2$.*

Proof. We use an analogue of IteratedFlowSolver. In each iteration, if some $y_d = 0$ or some $y_d \geq 1$, we reduce the problem. If $0 < y_d < 1$ for every d, then the counting argument (Lemma 6.6) shows that some tight edge lies on at most 3 demand paths. Since $y_d < 1$ for each d, $f_e \leq 2$. We reset $f_e = 0$ (i.e., discard the constraint) and continue iterating. The usual analysis completes the proof. □

To obtain Theorem 6.3, define $f'_v := f_v + 2$ for each vertex v and run the modified algorithm on f'. The output z is a feasible solution to the original instance. Moreover, if z^* is an optimal solution to the original ($\mathcal{F}'_c$), then $(1 + 2/\mu)z^*$ is a feasible solution to the new LP. Hence $w \cdot z$, which is at most the optimum of the new ($\mathcal{F}'_c$), is at most $(1 + 2/\mu)$ times the optimum of the original ($\mathcal{F}'_c$). □

6.4 Exact Solution for Spiders

In this section we show that edge-WMFT can be exactly solved in polynomial time when the supply graph is a spider. (A *spider* is a tree with exactly one vertex of degree greater than 2.) Call the vertex of degree ≥ 3 the *root* of the spider. Call each maximal path having the root as an endpoint a *leg* of the spider. Observe that in edge-WMFT when (V, E) is a spider, each demand path p_d either goes through the root, or else lies within a single leg. We generalize this observation into the following definition.

Definition 6.21. *Consider an instance of edge-WMFT on graph (V, E). With respect to a chosen root vertex $r \in V$, a demand edge d is said to be*

root-using*, if r is an internal vertex of p_d;*

radial*, if one endpoint of d is a descendant of the other, with respect to the orientation of (V, E) induced by the root r.*

The instance is all-ror *(short for "all root-using or radial") if there exists a choice of $r \in V$ for which every demand edge is either root-using or radial.*

Instances with only radial demand paths can be exactly solved via the LP ($\mathcal{F}$) since the constraint matrix is unimodular. Instances with only root-using demand paths can be solved using a matching approach, see e.g. the work of Nguyen [163]. To solve all-ror instances in general we use *bidirected flows*, which were introduced by Edmonds and Johnson [61]. Bidirected flow problems can be solved via a combinatorial reduction to b-matching (e.g., see [179]) which increases the instance size by a constant factor. We present in this section a reduction from all-ror edge-WMFT to bidirected flow.

A bidirected graph is an undirected graph together with, for each edge e and each of its endpoints $v \in e$, a sign $\sigma_{v,e} \in \{-1, +1\}$. Thus an edge can have two negative ends, two positive ends, or one end of each type; these are respectively called *negative* edges, *positive* edges, and *directed* edges. We will speak of directed edges as having the $+1$ end as their

head and -1 end as their tail. An instance of capacitated max-weight bidirected flow is an integer program of the following form.

$$\text{maximize} \sum_{e \in E} \pi_e x_e \tag{6.5}$$

$$\forall v \in V : \quad a_v \leq \sum_{e \ni v} \sigma_{v,e} x_e \leq b_v \tag{6.6}$$

$$\forall e \in E : \quad \ell_e \leq x_e \leq u_e \tag{6.7}$$

$$x \text{ integral} \tag{6.8}$$

When $a = b = \mathbf{0}$ and all edges are directed, (6.5)–(6.8) becomes a max-weight circulation problem; when all edges are positive and $a = \mathbf{0}$, (6.5)–(6.8) becomes a b-matching problem. We now describe the reduction.

Proof of Theorem 6.4. Let r denote the root vertex, i.e., assume every demand edge is either radial or root-using with respect to r. We construct a bidirected graph whose underlying undirected graph is $(V, E \cup D)$. Make each edge $e \in E$ directed, with head pointing towards r in the tree (V, E). We make each root-using $d \in D$ a positive edge; we make each radial $d \in D$ a directed edge, with head pointing away from r. See Figure 6.4 for an illustration.

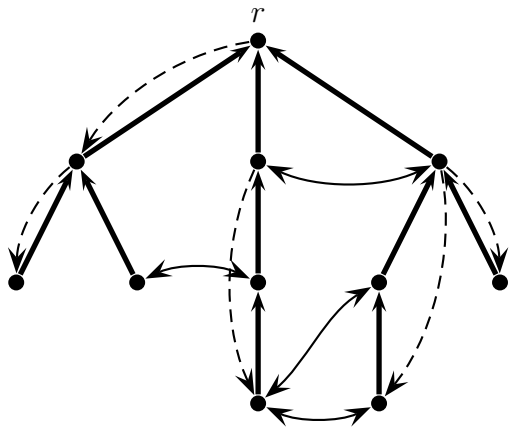


Figure 6.4: An all-ror multicommodity flow instance. The tree graph (V, E) is depicted using thick lines, and the demand edges D are thin. Radial demand edges are dashed and root-using demand edges are solid. The root is r. An arrowhead denotes a positive endpoint, while the remaining endpoints are negative; these signs correspond to the reduction in the proof of Theorem 6.4.

Set $a_r = -\infty$, $b_r = +\infty$ and $a_v = b_v = 0$ for each $v \in V \backslash \{r\}$ in the bidirected flow problem (6.5)–(6.8). For each demand edge $d \in D$, call $C(d) := \{d\} \cup p_d$ the *demand cycle*

of d. For a set F let χ^F denote the characteristic vector of F. Our choices of signs for the endpoints ensure that for each demand cycle $C(d)$, its characteristic vector $x = \chi^{C(d)}$ satisfies constraint (6.6). Moreover, any linear combination of these vectors is easily seen to satisfy (6.6), and the following converse holds.

Claim 6.22. *Any x satisfying (6.6) is a linear combination of characteristic vectors of demand cycles.*

Proof. Let $x' = x - \sum_d x_d \chi^{C(d)}$, and observe that x' also satisfies (6.6). Moreover, as each particular demand edge d^* occurs only in one demand cycle, namely $C(d^*)$, we have $x'_{d^*} = x_{d^*} - \sum_d x_d \chi^{C(d)}_{d^*} = x_{d^*} - x_{d^*} = 0$ for each $d^* \in D$. In other words, x' vanishes on D.

Now consider any leaf $v \neq r$ of G and its incident edge $uv \in E$. Since x' satisfies (6.6) at v and x' is zero on every edge incident to v except possibly uv, we deduce that $x_{uv} = 0$. By induction we can repeat this argument to show that x' also vanishes on all of E, so $x' = 0$. Then $x = x' + \sum_d x_d \chi^{C(d)} = \sum_d x_d \chi^{C(d)}$, which proves Claim 6.22. □

By Claim 6.22, we may change the variables in the optimization problem from x to instead have one variable y_d for each $d \in D$; the variables are thus related by $x = \sum_d y_d \chi^{C(d)}$. In the bidirected optimization problem, set $\ell_e = 0, u_e = c_e$ for each $e \in E$, and $\ell_d = 0, u_d = +\infty$ for each $d \in D$. Rewriting (6.7) in terms of the new variables gives precisely the constraints (6.1) and (6.2). In other words, feasible integral flows x correspond bijectively to feasible integral solutions y for the edge-WMFT instance. Setting $\pi_d = w_d$ for $d \in D$ and $\pi_e = 0$ for $e \in E$, the objective function of (6.5) represents the weight for y, completing the reduction.

As mentioned earlier, this bidirected flow problem can in turn be reduced to a b-matching problem with a constant factor increase in the size of the problem. Using the strongly polynomial b-matching algorithm of Anstee [6], the proof of Theorem 6.4 is complete. □

6.4.1 Polyhedral Results

The reduction used in the proof of Theorem 6.4 can also be used to derive the following polyhedral characterization; note that it is independent of which vertex is the root.

Theorem 6.23. *The convex hull of all integral feasible solutions in an all-ror edge-WMFT*

problem has the following description:

$$y_d \geq 0, \qquad \forall d \in D \tag{6.9}$$

$$\sum_{e \in p_d} y_d \leq c_e, \qquad \forall e \in E \tag{6.10}$$

$$\sum_{d \in D} y_d \lfloor |p_d \cap F|/2 \rfloor \leq \lfloor c(F)/2 \rfloor, \qquad \forall F \subset E \text{ with } c(F) \text{ odd} \tag{6.11}$$

Proof. It is obvious that constraints (6.9) and (6.10) are valid. To see that the constraint (6.11) is valid, notice that it can be obtained as a Chvátal-Gomory cut: give coefficient $1/2$ to each constraint (6.10) for $e \in F$. This establishes necessity, and the rest of the proof will establish sufficiency of the constraints (6.9)–(6.11).

Our starting point is the following polyhedral characterization, which appears as Cor. 36.3a in Schrijver [179], and deals with the special case of bidirected flow when $a = b$ and $\ell = \vec{0}$.

Proposition 6.24. *Let σ denote the signs of a bidirected graph. The convex hull of the integer solutions to*

$$\forall e \in E : 0 \leq x_e \leq u_e \qquad \forall v \in V : \sum_{e \ni v} \sigma_{v,e} x_e = b_v \tag{6.12}$$

(i.e. the convex hull of all feasible integral bidirected flows) is determined by Equation (6.12) together with the constraints

$$x(\delta(U) \backslash F) - x(F) \geq 1 - u(F) \tag{6.13}$$

where $U \subseteq V$ and $F \subseteq \delta(U)$ with $b(U) + u(F)$ odd.

In order to apply Proposition 6.24 to the construction in the proof of Theorem 6.4, we set $a_r = b_r = 0$ and add a loop at r with both of its endpoints negative. Further, we change the definition of $C(d)$ to include this loop whenever d is a radial demand edge; then it is not hard to show that, just as before, feasible bidirected flows x correspond bijectively to feasible multicommodity flows y.

Now apply Proposition 6.24 to the construction. Recall that the edge set of the bidirected graph is $D \cup E$. Since $u_d = +\infty$ for $d \in D$, the constraint (6.13) is vacuously true unless $F \subset E$. Furthermore, recall that b is the all-zero vector and $u_e = c_e$ for $e \in E$. Rearranging, we obtain the following description of the convex hull of all integral feasible bidirected flows:

$$\forall e \in E : 0 \leq x_e \leq c_e \qquad \forall d \in D : 0 \leq x_d \qquad \forall v \in V : \sum_{e \ni v} \sigma_{v,e} x_e = 0 \qquad \text{and}$$

$$x(F) - x(\delta(U))/2 \leq (c(F)-1)/2 \quad \text{for } U \subseteq V, F \subseteq E \cap \delta(U), c(F) \text{ odd}$$

Rewriting in terms of the y variables, and collecting like terms, yields

$$\forall d \in D : 0 \leq y_d \qquad \forall e \in E : \sum_{e \in p_d} y_d \leq c_e \qquad \text{and} \tag{6.14}$$

$$\sum_d y_d(|p_d \cap F| - |C(d) \cap \delta(U)|/2) \leq (c(F)-1)/2 \quad \text{for } U \subseteq V, F \subseteq E \cap \delta(U), c(F) \text{ odd} \tag{6.15}$$

For any fixed choice of $F \subseteq E$, let $U_F^* \subseteq V$ be the unique set such that $\delta(U_F^*) \cap E = F$ and $r \notin U_F^*$. We claim that for this F, constraint (6.15) is tightest for $U = U_F^*$. To see this, note first that $|C(d) \cap \delta(U)|$ is always even (since in traversing the cycle $C(d)$, we enter U as many times as we leave); second, that $|C(d) \cap \delta(U_F^*)|/2 = \lceil |p_d \cap F|/2 \rceil$; third, that for any other U' such that $F \subseteq \delta(U')$, $|C(d) \cap \delta(U')|/2$ is an integer greater than or equal to $|p_d \cap F|/2$.

Hence, there is no loss of generality in assuming $U = U_F^*$ in constraint (6.15). Rewriting, it becomes

$$\sum_d y_d(|p_d \cap F| - \lceil |p_d \cap F|/2 \rceil) \leq (c(F)-1)/2 \quad \text{for } c(F) \text{ odd;}$$

finally, since $t = \lfloor t/2 \rfloor + \lceil t/2 \rceil$ for all integers t, the theorem follows. □

Interestingly, results of Garg et al. [87, preliminary version] show that (6.9)–(6.11) is also integral in *unit-capacity* edge-WMFT. (In fact they show that an intermediate LP between the naive one and (6.9)–(6.11) is integral — specifically they only require (6.11) for subsets of edges forming a star of odd size. They don't give a full proof; the detailed proof is obtained by observing that the LP is locally (around each vertex) just the matching LP for edges around that vertex, and one needs to observe that you can paste together convex combinations around various vertices.) Tangentially, we observe their results imply a good approximation for another special case of edge-WMFT.

Proposition 6.25. *There is a 3/2-approximation algorithm for edge-WMFT when all edges have the same capacity.*

Proof. (Sketch.) Let μ be the common capacity. Let x be an optimal integral solution; $\frac{x}{\mu}$ is a feasible fractional flow for unit capacities; then $\frac{2}{3}\frac{x}{\mu}$ is a feasible fractional solution to (6.9)–(6.11) (for unit capacities). By the algorithmic result of [87] one may find the optimal integral solution y for unit capacities; by the polyhedral result of [87] its value is at least $\frac{2}{3}\frac{cx}{\mu}$. Then μy is a feasible integral solution (for the original capacities) with value at least $\frac{2}{3}cx$, i.e. at least 2/3 optimal. □

A similar argument establishes that the integrality gap of the LP (6.9)–(6.11) is at most 3/2, when all capacities are unit.

It is possible to synthesize our Theorems 6.4 and 6.23 with corresponding results of [87] for the unit-capacity case, by "gluing" all-ror instances at capacity-1 edges. For this more general class of problems we find that (6.9)–(6.11) is integral (and hence the optimal integral flow can be found in polynomial time) although we omit full details of this rather esoteric result.

6.5 Closing Remarks

Caprara and Fischetti [30] gave a strongly polynomial-time algorithm to separate over the family (6.11) of inequalities. Is the polyhedral formulation (6.9)–(6.11) useful in designing a better approximation algorithm for edge-WMFT? One roadblock is that normal uncrossing techniques seem to fail on that LP.

There is a close relation between WMFT and its "demand" version where every flow variable is restricted according to $y_d \in \{0, r_d\}$ for some constants $\{r_d\}_{d \in D}$. E.g., combining IteratedFlowSolver and Cor. 3.5 of [39], we obtain a $1 + O(\sqrt{r_{\max}/\mu})$ approximation for demand WMFT where $r_{\max}$ is the maximum demand. Shepherd and Vetta [180] showed that when the tree is a star, the $O(r_{\max})$-relaxed integrality gap of the demand analogue of ($\mathcal{F}$) is 1, and we can develop this result to get a $1+O(r_{\max}/\mu)$-approximation algorithm for demand WMFT on stars (this follows from Theorem 7.14 in the next chapter). It would be interesting to demonstrate the same results on arbitrary trees.

When every edge has capacity 1, edge-WMFT is exactly solvable [87], and Theorem 6.1(a) gives a 3-approximation when there are no capacity-1 edges. Can we combine these results to improve upon the 4-approximation by Chekuri et al. [39] for general instances?

Acknowledgement

We would like to thank Joseph Cheriyan, Jim Geelen, András Sebő, and Chaitanya Swamy for helpful discussions.

Chapter 7

Approximability of Sparse Integer Programs

The main focus of this chapter is a pair of new approximation algorithms for sparse integer programs. First, for covering integer programs $\{\min cx : Ax \geq b, \mathbf{0} \leq x \leq d\}$ where A has at most k nonzeroes per row, we give a k-approximation algorithm. (We assume A, b, c, d are nonnegative.) For any $k \geq 2$ and $\epsilon > 0$, if $\mathsf{P} \neq \mathsf{NP}$ this ratio cannot be improved to $k - 1 - \epsilon$, and under the unique games conjecture this ratio cannot be improved to $k - \epsilon$. One key idea is to replace individual constraints by others that have better rounding properties but the same nonnegative integral solutions; another critical ingredient is knapsack-cover inequalities. Second, for packing integer programs $\{\max cx : Ax \leq b, \mathbf{0} \leq x \leq d\}$ where A has at most k nonzeroes per column, we give a $(2k^2 + 2)$-approximation algorithm. Our approach builds on the iterated LP relaxation framework.

7.1 Introduction and Prior Work

In this chapter we investigate the following problem: what is the best possible approximation ratio for integer programs where the constraint matrix is sparse? To put this in context we recall a famous result of Lenstra [150]: integer programs with a constant number of variables or a constant number of constraints can be solved in polynomial time. Our investigations analogously ask what is possible if the constraints each involve at most k variables, or if the variables each appear in at most k constraints.

Rather than consider the full class of all integer programs, we consider only packing and covering problems. One sensible reason for this is that *every* integer program can be

rewritten (possibly with additional variables) in such a way that each constraint contains at most 3 variables and each variable appears in at most 3 constraints, if mixed positive and negative coefficients are allowed. Aside from this, packing programs and covering programs represent a substantial portion of the literature on integer programs; and sparse programs of this type are interesting in their own right as "multiple-knapsack" problems where each item affects a bounded number of knapsacks, or each knapsack is affected by a bounded number of items.

We use CIP (resp. PIP) as short for *covering* (resp. *packing*) *integer program*, which is any integer program of the form $\{\min cx : Ax \geq b, \mathbf{0} \leq x \leq d\}$ (resp. $\{\max cx : Ax \leq b, \mathbf{0} \leq x \leq d\}$) with A, b, c, d nonnegative and rational. Note that CIPs are sometimes called *multiset multicover* when A and b are integral. We call constraints $x \leq d$ *multiplicity constraints* (also known as *capacity constraints*). We allow for entries of d to be infinite, and without loss of generality, all finite entries of d are integral. An integer program with constraint matrix A is *k-row-sparse*, or *k-RS*, if each row of A has at most k entries; we define *k-column-sparse (k-CS)* similarly. As a rule of thumb we ignore the case $k = 1$, since such problems trivially admit fully polynomial-time approximation schemes (FPTAS's) or poly-time algorithms. The symbol $\mathbf{0}$ denotes the all-zero vector, and similarly for $\mathbf{1}$. For covering problems an *α-approximation algorithm* is one that always returns a solution with objective value at most α times optimal; for packing, the objective value is at least $1/\alpha$ times optimal. We use n to denote the number of variables and m the number of constraints (i.e. the number of rows of A).

7.1.1 k-Row-Sparse Covering IPs: Previous and New Results

The special case of 2-RS CIP where A, b, c, d are 0-1 is the same as Min Vertex Cover, which is APX-hard. More generally, 0-1 k-RS CIP is the same as k-Bounded Hypergraph Min Vertex Cover (a.k.a. Set Cover with maximum frequency k) which is not approximable to $k - 1 - \epsilon$ for any fixed $\epsilon > 0$ unless P=NP [55] ($k - \epsilon$ under the unique games conjecture [130]). This special case is known to admit a matching positive result: set cover with maximum frequency k can be k-approximated by direct rounding of the naive LP [113] or local ratio/primal-dual methods [12].

The following results are known for other special cases of k-RS CIP with multiplicity constraints: Hochbaum [108] gave a k-approximation in the special case that A is 0-1; Hochbaum et al. [115] and Bar-Yehuda & Rawitz [13] gave pseudopolynomial 2-approximation algorithms for the case that $k = 2$ and d is finite. For the special case $d = \mathbf{1}$, Carr et al. [32, §2.6] gave a k-approximation, and Fujito & Yabuta [80] gave a primal-dual k-approximation. Moreover [32, 80] claim a k-approximation for general d, but there seems to have been some oversights as the papers do not provide full proofs and their methods alone seem to be insufficient for general d. Briefly, [32] claims $d = \mathbf{1}$ holds

without loss of generality, which is true for some of the applications in [32], but seems false for k-RS CIPs; [80] sketches a reduction that seems faulty, and omits many crucial details for lack of space. Our first main result, given in Section 7.2, is a simple and correct proof of the same claim.

Theorem 7.1. *There is a polynomial time k-approximation algorithm for k-RS CIPs with multiplicity constraints.*

Our approach is to first consider the special case that there are no multiplicity constraints (i.e. $d_j = +\infty$ for all j); we then extend to the case of finite d via *knapsack-cover inequalities*, using linear programming (LP) techniques from Carr et al. [32]. A $(k+1)$-approximation algorithm is relatively easy to obtain using LP rounding; in order to get the tighter ratio k, we replace constraints by other "$\mathbf{Z}_+$-equivalent" constraints (see Definition 7.7) with better rounding properties. The algorithm requires a polynomial-time linear programming subroutine.

Independently of our work, recent work of Koufogiannakis & Young [143, 146, 144, 145] also gives a full and correct proof of Theorem 7.1. Their primal-iterative approach works for a broad generalization of k-RS CIPs and runs in low-degree strongly polynomial time. Our approach has the generic advantage of giving new ideas that can be used in conjunction with other LP-based methods, and the specific advantage of giving integrality gap bounds. See Section 7.2.2 for more details.

7.1.2 k-Column-Sparse Packing IPs: Previous and New Results

Before 2009, no constant-factor approximation was known for k-CS PIPs, except in special cases. If every entry of b is $\Omega(\log m)$ then randomized rounding provides a constant-factor approximation. *Demand matching* is the special case of 2-CS PIP where (i) in each column of A all nonzero values in that column are equal to one another and (ii) no two columns have their nonzeroes in the same two rows. Shepherd & Vetta [180] showed demand matching is APX-hard but admits a $(\frac{11}{2} - \sqrt{5})$-approximation algorithm when $d = \mathbf{1}$; their approach also gives a $\frac{7}{2}$-approximation for 2-CS PIP instances satisfying (i). Results of Chekuri et al. [39] yield a $11.542k$-approximation algorithm for k-CS PIP instances satisfying (i) and such that the maximum entry of A is less than the minimum entry of b.

The special case of k-CS PIP where A, b are 0-1 is the same as *min-weight k-set packing*, *hypergraph matching with edges of size $\leq k$*, and *strong independent sets in hypergraphs with degree at most k*. The best approximation ratio known for this problem is $(k+1)/2+\epsilon$ [17] for general weights, and $k/2+\epsilon$ when $c = \mathbf{1}$ [119]. For $k = 3$ a 2-approximation in the weighted case is known due to Lau and Chan [37]. The best lower bound is due to Hazan et al. [111], who showed $\Omega(k/\ln k)$-inapproximability unless P=NP, even for $c = \mathbf{1}$.

Our second main result, given in Section 7.3, is the following result.

Theorem 7.2. *There is a polynomial time $(2k^2 + 2)$-approximation algorithm for k-CS PIPs with multiplicity constraints.*

Our methodology begins by using *iterated LP relaxation* [181] to find an integral solution with super-optimal value, but violating some constraints in an additively-bounded way. Then we use a colouring argument to decompose the violating solution into $O(k^2)$ feasible solutions.

The original arXiv eprint and conference version [172] of this work gave a $O(k^2 2^k)$-approximation for k-CS PIP using iterated relaxation plus a randomized decomposition approach; that was the first approximation algorithm for this problem with ratio that depends only on k. The $O(k^2 2^k)$-approximation does not appear in this version. Subsequent to the initial arXiv version [172] in April 2009, several better approximations to k-CS PIP were discovered. C. Chekuri, A. Ene and N. Korula (personal communication) obtained an $O(k2^k)$ algorithm using randomized rounding, and an $O(k^2)$-approximation in May. The latter was independently matched by the author and D. Chakrabarty, as it turns out using the same simple approach, which appears in this version. Finally, Bansal, Korula, and Nagarajan [11] gave in August 2009 a simple and elegant $O(k)$-approximation algorithm based on randomized rounding with a careful alteration argument.

7.1.3 Other Related Work

Srinivasan [184, 185] showed that k-CS CIPs admit a $O(\log k)$-approximation. Kolliopoulos and Young [134] extended this result to handle multiplicity constraints. There is a matching hardness result: it is NP-hard to approximate k-Set Cover, which is the special case where A, b, c are 0-1, better than $\ln k - O(\ln \ln k)$ for any $k \geq 3$ [188]. Hence for k-CS CIP the best possible approximation ratio is $\Theta(\log k)$. A $(k + \epsilon)$-approximation algorithm can be obtained by separately applying an approximation scheme to the knapsack problem corresponding to each constraint. Hochbaum [114] showed 2-CS CIPs are NP-hard to optimize and gave a bicriteria approximation algorithm. Although 0-1 2-CS CIP is Edge Cover which lies in P, 2-CS CIP in general is NP-hard to $(17/16 - \epsilon)$-approximate, due to methods from [35], even if A has 2 *equal* nonzeroes per column and d is 0-1 or d is all-$+\infty$. See Appendix 7.5 for details.

The special case of 2-RS PIP where A, b, c are 0-1 is the same as Max Independent Set, which is not approximable within $n/2^{\log^{3/4+\epsilon} n}$ unless $\mathsf{NP} \subset \mathsf{BPTIME}(2^{\log^{O(1)} n})$ [129]. On the other hand, n-approximation of any packing problem is easy to accomplish by looking at the best singleton-support solution. A slightly better n/t-approximation, for any fixed t, can be accomplished by exhaustively guessing the t most profitable variables in the optimal solution, and then solving the resulting t-dimensional integer program to optimality via Lenstra's result [150].

A closely related problem is k-Dimensional Knapsack, which are PIPs or CIPs with at most k constraints (in addition to nonnegativity constraints). See Chapter 9 for more background on this problem.

Semimodular Optimization

The recent results mentioned here all assume $d = \mathbf{1}$.

For minimizing a submodular objective subject to k-row sparse covering constraints, three k-approximations were recently published, one in the framework of Koufogiannakis & Young [143, 146, 144, 145], one by Iwata and Nagano [123], and one by Goel et al. [92] for the case $k = 2$.

For maximizing a monotone submodular function subject to k-column sparse packing constraints, Bansal et al. [11] gave a $O(k)$-approximation algorithm.

It is not clear whether anyone has considered the problem of submodular k-CS CIPs. Some work has been done already on submodular set cover, for example see Wolsey [199]. One might hope to use results of Kolliopoulos & Young [134] (see also the references therein including work by Srinivasan) to get an $O(\ln k)$ approximation for this problem.

7.1.4 Summary

We summarize the existing and new results in Table 7.1. Note that in all four cases, the strongest known lower bounds are obtained even in the special case that A, b, c, d are 0-1.

	k-Column-Sparse		k-Row-Sparse	
	lower bound	upper bound	lower bound	upper bound
Packing	$\Omega(k/\ln k)$	$\mathbf{2k^2 + 2}$	$n^{1-o(1)}$	ϵn
Covering	$\ln k - O(\ln \ln k)$	$O(\ln k)$	$k - \epsilon$	$\mathbf{k}$

Table 7.1: The landscape of approximability of sparse integer programs. Our main results are in boldface.

7.2 k-Approximation for k-Row-Sparse CIPs

By scaling rows and clipping coefficients that are too high (i.e. setting $A_{ij} = \min\{1, A_{ij}\}$), there is no loss of generality in the following definition.

Definition 7.3. *A* k-RS CIP *is an integer program* $\{\min cx : Ax \geq \mathbf{1}, \mathbf{0} \leq x \leq d\}$ *where* A *is* k*-RS and all entries of* A *are at most 1.*

To begin with, we focus on the case $d_j = +\infty$ for all j, which we will call *unbounded* k*-RS CIP*, since it already illustrates the essence of our new technique. Motivated by LP rounding methods, we make the following definition, in which x is a vector-valued variable and α is a vector of real coefficients. Throughout, we assume coefficients are nonnegative. When we apply $\lfloor \cdot \rfloor$ to vectors we mean the component-wise floor.

Definition 7.4. *A constraint* $\alpha x \geq 1$ *is* ρ-roundable *for some* $\rho > 1$ *if for all nonnegative real* x, $(\alpha x \geq 1)$ *implies* $(\alpha \lfloor \rho x \rfloor \geq 1)$.

Note that ρ-roundability implies ρ'-roundability for $\rho' > \rho$. The relevance of this property is explained by the following proposition.

Proposition 7.5. *If every constraint in an unbounded covering integer program is* ρ*-roundable, then there is a* ρ*-approximation algorithm for the program.*

Proof. Let x^* be an optimal solution to the program's linear relaxation. Then cx^* is a lower bound on the cost of any optimal solution. Thus, $\lfloor \rho x^* \rfloor$ is a feasible solution with cost at most ρ times optimal. □

Another simple observation helps us get started.

Proposition 7.6. *The constraint* $\alpha x \geq 1$ *is* $(1 + \sum_i \alpha_i)$*-roundable.*

Proof. Let $\rho = (1 + \sum_i \alpha_i)$. Since $\lfloor t \rfloor > t - 1$ for any t, if $\alpha x \geq 1$ for a nonnegative x, then

$$\alpha \lfloor \rho x \rfloor \geq \sum_i \alpha_i (\rho x_i - 1) = \rho \sum_i \alpha_i x_i - \sum_i \alpha_i \geq \rho \cdot 1 - (\rho - 1) = 1,$$

as needed. □

Now consider an unbounded k-RS CIP. Since each constraint has at most k coefficients, each less than 1, it follows from Proposition 7.6 that every constraint in these programs is $(k+1)$-roundable, and so such programs admit a $(k+1)$-approximation algorithm by Proposition 7.5. It is also clear that we can tighten the approximation ratio to k for programs where the sum of the coefficients in every constraint (row) is at most $k - 1$. What we will now do is show that rows with sum in $(k-1, k]$ can be replaced by other rows which are k-roundable.

Definition 7.7. *Two constraints* $\alpha x \geq 1$ *and* $\alpha' x \geq 1$ *are* $\mathbf{Z}_+$-equivalent *if for all nonnegative integral* x, $(\alpha x \geq 1) \Leftrightarrow (\alpha' x \geq 1)$.

In other words, $\alpha x \geq 1$ and $\alpha' x \geq 1$ are $\mathbf{Z}_+$-equivalent if $\alpha x \geq 1$ is valid for $\{x : x \geq 0, \alpha' x \geq 1\}$ *and* $\alpha' x \geq 1$ is valid for $\{x : x \geq 0, \alpha x \geq 1\}$.

Proposition 7.8. *Every constraint $\alpha x \geq 1$ with at most k nonzero coefficients is $\mathbf{Z}_+$-equivalent to a k-roundable constraint.*

Before proving Proposition 7.8, let us illustrate its use.

Theorem 7.9. *There is a polynomial time k-approximation algorithm for unbounded k-RS CIPs.*

Proof. Using Proposition 7.8 we replace each constraint with a $\mathbf{Z}_+$-equivalent k-roundable one. The resulting IP has the same set of feasible solutions and the same objective function. Therefore, Proposition 7.5 yields a k-approximately optimal solution. □

With the framework set up, we begin the technical part: a lemma, then the proof of Proposition 7.8.

Lemma 7.10. *For any positive integers k and v, the constraint $\sum_{i=1}^{k-1} x_i + \frac{1}{v} x_k \geq 1$ is k-roundable.*

Proof. Let $\alpha x \geq 1$ denote the constraint. If x satisfies the constraint, then the maximum of $x_1, x_2, \ldots, x_{k-1}$ and $\frac{1}{v} x_k$ must be at least $1/k$. If $x_i \geq 1/k$ for some $i \neq k$ then $\lfloor k x_i \rfloor \geq 1$ and so $\alpha \lfloor kx \rfloor \geq 1$ as needed. Otherwise x_k must be at least v/k and so $\lfloor k x_k \rfloor \geq v$ which implies $\alpha \lfloor kx \rfloor \geq 1$ as needed. □

Proof of Proposition 7.8. If the sum of coefficients in the constraint is $k-1$ or less, we are done by Proposition 7.6, hence we assume the sum is at greater than $k-1$. Without loss of generality (by renaming) such a constraint is of the form

$$\sum_{i=1}^{k} x_i \alpha_i \geq 1 \tag{7.1}$$

where $\mathbf{0} < \alpha \leq \mathbf{1}$, $k - 1 < \sum_i \alpha_i \leq k$, and the α_i's are nonincreasing in i.

Define the *support* of x to be $\mathtt{supp}(x) := \{i \mid x_i > 0\}$. Now $\alpha_{k-1} + \alpha_k > 1$ since $k - 1 < \sum_{i<k-1} \alpha_i + \alpha_{k-1} + \alpha_k \leq \alpha_{k-1} + \alpha_k + (k-2)$. Since the α_i are nonincreasing, $\alpha_i + \alpha_j > 1$ for any $i < k, j \leq k$; more generally, any integral $x \geq 0$ with $|\mathtt{supp}(x)| \geq 2$ must satisfy $\alpha x \geq 1$. To express the set of *all* feasible integral solutions, let t be the maximum i for which $\alpha_i = 1$ (or $t = 0$ if no such i exists), let e_i denote the ith unit basis vector, and

let $v = \lceil 1/\alpha_k \rceil$. Then it is not hard to see that the nonnegative integral solution set to constraint (7.1) is the disjoint union

$$\begin{aligned} &\{x \mid x \geq 0, |\,\mathtt{supp}(x)| \geq 2\} \uplus \{ze_i \mid 1 \leq i \leq t, z \geq 1\} \\ \uplus &\{ze_i \mid t < i < k, z \geq 2\} \uplus \{ze_k \mid z \geq v\}. \end{aligned} \tag{7.2}$$

The special case $t = k$ (i.e. $\alpha_1 = \alpha_2 = \cdots = \alpha_k = 1$) is already k-roundable by Lemma 7.10, so assume $t < k$. Consider the constraint

$$\sum_{i=1}^{t} x_i + \sum_{i=t+1}^{k-1} \frac{v-1}{v} x_i + \frac{1}{v} x_k \geq 1. \tag{7.3}$$

Every integral $x \geq 0$ with $|\,\mathtt{supp}(x)| \geq 2$ satisfies constraint (7.3). By also considering the cases $|\,\mathtt{supp}(x)| \in \{0, 1\}$, it is easy to check that constraint (7.3) has precisely Equation (7.2) as its set of feasible solutions, i.e. constraint (7.3) is $\mathbf{Z}_+$-equivalent to $\alpha x \geq 1$. If $t < k-1$, the sum of the coefficients of constraint (7.3) is $k-1$ or less, so it is k-roundable by Proposition 7.6. If $t = k-1$, constraint (7.3) is k-roundable by Lemma 7.10. Thus in either case we have what we wanted. □

7.2.1 Multiplicity Constraints

We next obtain approximation guarantee k even with multiplicity constraints $x \leq d$. For this we use *knapsack-cover inequalities*. These inequalities represent residual covering problems when a set of variables is taken at maximum multiplicity. Wolsey [199] studied inequalities like this for 0-1 problems to get a primal-dual approximation algorithm for submodular set cover. The LP we use is most like what appears in Carr et al. [32] and Kolliopoulos & Young [134], but we first replace each row with a k-roundable one.

Specifically, given a CIP $\{\min cx \mid Ax \geq \mathbf{1}, \mathbf{0} \leq x \leq d\}$ with A, d nonnegative, we now define the knapsack cover LP. Note that we allow d to contain some entries equal to $+\infty$. For a subset F of $\mathtt{supp}(A_i)$ such that $\sum_{j \in F} A_{ij} d_j < 1$, define $A_{ij}^{(F)} = \min\{A_{ij}, 1 - \sum_{j \in F} A_{ij} d_j\}$. Following [32, 134] we define the *knapsack cover LP* for our problem to be

$$\text{KC-LP} = \Big\{ \min cx : \mathbf{0} \leq x \leq d; \\ \forall i, \forall F \subset \mathtt{supp}(A_i) \text{ s.t. } \sum_{j \in F} A_{ij} d_j < 1 : \sum_{j \notin F} A_{ij}^{(F)} x_j \geq 1 - \sum_{j \in F} A_{ij} d_j \Big\}.$$

Theorem 7.1. *There is a polynomial time k-approximation algorithm for k-RS CIPs.*

Proof. Using Proposition 7.8, we assume all rows of A are k-roundable. Let x^* be the optimal solution to KC-LP. Define $\widehat{x} = \min\{d, \lfloor kx^* \rfloor\}$, where min denotes the component-wise minimum. We claim that $\widehat{x}$ is a feasible solution to the CIP, which will complete the proof. In other words, we want to show for each row i that $A_i\widehat{x} \geq 1$.

Fix any row i and define $F = \{j \in \mathtt{supp}(A_i) \mid x^*_j \geq d_j/k\}$, i.e. F is those variables in the constraint that were rounded to their maximum multiplicity. If $F = \varnothing$ then, by the k-roundability of $A_ix \geq 1$, we have that $A_i\widehat{x} = A_i\lfloor kx^* \rfloor \geq 1$ as needed. So assume $F \neq \varnothing$.

If $\sum_{j\in F} A_{ij}d_j \geq 1$ then the constraint $A_i\widehat{x} \geq 1$ is satisfied; consider otherwise. Since $\lfloor kx^*_j \rfloor > kx^*_j - 1$ for $j \notin F$, since x^* satisfies the knapsack cover constraint for i and F, and since $A^{(F)}_{ij} \leq 1 - \sum_{j\in F} A_{ij}d_j$ for each j, we have

$$\begin{aligned}\sum_{j\notin F} A^{(F)}_{ij}\widehat{x}_j &\geq k\sum_{j\notin F} A^{(F)}_{ij}x^*_j - \sum_{j\notin F} A^{(F)}_{ij} \\ &\geq k\Big(1 - \sum_{j\in F} A_{ij}d_j\Big) - \Big|\{j : j \in \mathtt{supp}(A_i)\backslash F\}\Big|\Big(1 - \sum_{j\in F} A_{ij}d_j\Big).\end{aligned}$$

Since $F \neq \varnothing$ and $|\,\mathtt{supp}(A_i)| \leq k$, this gives $\sum_{j\notin F} A^{(F)}_{ij}\widehat{x}_j \geq 1 - \sum_{j\in F} A_{ij}d_j$. Rearranging, and using the facts $(\forall j : A_{ij} \geq A^{(F)}_{ij})$ and $(\forall j \in F : d_j = \widehat{x}_j)$, we deduce $A_i\widehat{x} \geq 1$, as needed.

For fixed k, we may solve KC-LP explicitly, since it has polynomially many constraints. For general k, we follow the ellipsoid algorithm-based approach of [32, 134]: rather than solve KC-LP in polynomial time, we obtain a solution x^* which is optimal for a modified KC-LP having not all knapsack-cover constraints, but at least all those for the the m specific (i, F) pairs (depending on x^*) used in our proof; thus we still get a k-approximation in polynomial time. □

7.2.2 Integrality Gap Bounds

Recall the discussion of integrality gaps and LP-relativity from Section 1.2 to motivate this section. In discussing integrality gaps for k-RS CIP problems, we say that the *naive LP relaxation* of $\{\min cx \mid x \text{ integral}, Ax \geq b, \mathbf{0} \leq x \leq d\}$ is the LP obtained by removing the restriction of integrality. Earlier, we made the assumption that $A_{ij} \leq b_i$ for all i, j; let us call this the *clipping assumption*. The clipping assumption is without loss of generality for the purposes of approximation guarantees, however, it affects the integrality gap of the naive LP for unbounded k-RS CIP, as we now illustrate. Without the clipping assumption, the integrality gap of k-RS CIP problems can be unbounded as a function of k; indeed for any integer $M \geq 1$ the simple covering problem $\{\min x_1 \mid [M]x_1 \geq 1, 0 \leq x\}$ has integrality gap M. With the clipping assumption, our discussion of roundability shows

in unbounded instances the integrality gap of the naive LP is at most $k+1$, since the roundability framework gives an LP-relative approximation algorithm.

Even under the clipping assumption, k-RS CIPs with *multiplicity constraints* can have large integrality gaps — e.g. $\{\min x_2 \mid \left[\begin{smallmatrix} M \\ M \end{smallmatrix}\right] x \geq M+1, 0 \leq x, x_1 \leq 1\}$ has integrality gap M. For bounded instances, the knapsack-cover inequalities represent a natural generalization of the clipping assumption, namely, we perform a sort of clipping even considering that any subset of the variables are chosen to their maximum extent. (Note also that if $d_i = +\infty$ for all i, then KC-LP is just the naive LP with the clipping assumption.)

The methods in Section 7.2 show that KC-LP has integrality gap at most $k+1$ on k-RS CIP instances (since our algorithms are LP-relative). Our methods also show that if we replace each row with a k-roundable one (Proposition 7.8), then the resulting KC-LP has integrality gap at most k. We are actually unaware of any k-RS CIP instance with $k > 1$ where the integrality gap of KC-LP (without applying Proposition 7.8) is greater than k; resolving whether such an instance exists would be interesting. Some special cases are understood, e.g. Koufogiannakis and Young [145] give a primal-dual k-approximation for k-CS PIP in the case A is 0-1, also known as hypergraph b-matching.

7.3 Column-Sparse Packing Integer Programs

In this section we give an approximation algorithm for k-column-sparse packing integer programs with approximation ratio $2k^2+2$, and better results for $k = 2$. The results hold even in the presence of multiplicity constraints $x \leq d$. Broadly speaking, our approach is rooted in the demand matching algorithm of Shepherd & Vetta [180]; their path-augmenting algorithm can be viewed as a restricted form of *iterated relaxation*, which is the main tool in our new approach. Iterated relaxation yields a superoptimal solution that violates some constraints, and with a colouring/decomposition approach we are able to obtain a feasible solution while retaining at least a constant fraction of the weight.

For a k-CS PIP $\mathcal{P}$ let $\mathcal{L}(\mathcal{P})$ denote its linear relaxation $\{\max cx \mid Ax \leq b, \mathbf{0} \leq x \leq d\}$. We use the set I to index the constraints and J to index the variables in our program. We note a simple assumption that is without loss of generality for the purposes of obtaining an approximation algorithm: $A_{ij} \leq b_i$ for all i, j. To see this, note that if $A_{ij} > b_i$, then every feasible solution has $x_j = 0$ and we can simply delete x_j from the instance.

Now we give the iterated rounding method which is key for our approach. Let the term *entry* mean a pair $(i, j) \in I \times J$ such that $A_{ij} > 0$. Our iterated rounding algorithm computes a set S of *special* entries; for such a set we let $A_{S \to 0}$ denote the matrix obtained from A by zeroing out the special entries.

Lemma 7.11. *Given a k-CS PIP $\mathcal{P}$, we can in polynomial time find S and nonnegative integral vectors x^0, x^1 with $x^0 + x^1 \leq d$ and $x^1 \leq \mathbf{1}$ such that*

(a) $c(x^0 + x^1) \geq \mathrm{OPT}(\mathcal{L}(\mathcal{P}))$

(b) $\forall i \in I$, *we have* $|\{j : (i,j) \in S\}| \leq k$

(c) $Ax^0 + A_{S \to 0} x^1 \leq b$.

In particular, since x^1 is 0-1, $x^0 + x^1$ is a solution with super-optimal objective value, and it violates each constraint in an additively bounded way (i.e., $A_i(x^0 + x^1) \leq b_i + k \max_j A_{ij}$ for all i). We now give the proof.

Proof of Lemma 7.11. First, we give a sketch. Let $\mathtt{supp}(A_j) := \{i \in I \mid A_{ij} > 0\}$, which has size at most k, and similarly $\mathtt{supp}(A_i) := \{j \in J \mid A_{ij} > 0\}$. Let x^* be an extreme optimal solution to $\mathcal{L}(\mathcal{P})$. The crux of our approach deals with the case that x^* has no integral values: then x^* is a *basic feasible solution*, and there is a set of $\mathtt{supp}(x^*) = |J|$ linearly independent tight constraints for x^*, so $|I| \geq |J|$. By double-counting there is some i with $|\mathtt{supp}(A_i)| \leq k$, which is what permits iterated relaxation: we discard the constraint for i and go back to the start.

Figure 7.1 contains pseudocode for our iterated rounding algorithm, IteratedSolver.

IteratedSolver(A, b, c, d)
1: Let x^* be an extreme optimum of $\{\max cx \mid x \in \mathbf{R}^J; \mathbf{0} \leq x \leq d; Ax \leq b\}$
2: Let $x^0 = \lfloor x^* \rfloor, x^1 = \mathbf{0}, J' = \{j \in J \mid x^*_j \notin \mathbf{Z}\}, I' = I$
3: **loop**
4: Let x^* be an extreme optimum of $\{\max cx \mid x \in [0,1]^{J'}; Ax^0 + A_{S \to 0}(x + x^1) \leq b\}$
5: For each $j \in J'$ with $x^*_j = 0$, delete j from J'
6: For each $j \in J'$ with $x^*_j = 1$, set $x^1_j = 1$ and delete j from J'
7: If $J' = \varnothing$, terminate and return S, x^0, x^1
8: **for** each $i \in I'$ with $|\mathtt{supp}(A_i) \cap J'| \leq k$ **do**
9: Mark each entry $\{(i,j) \mid j \in \mathtt{supp}(A_i) \cap J'\}$ special and delete i from I'
10: **end for**
11: **end loop**

Figure 7.1: Algorithm for k-CS PIP.

Now we explain the pseudocode. The x^0 term can be thought of as a preprocessing step which effectively reduces the general case to the special case that $d = \mathbf{1}$. The term

$x^1 \in \{0,1\}^J$ grows over time. The set J' represents all j that could be added to x^1 in the future, but have not been added yet. The set $I \backslash I'$ keeps track of constraints that have been dropped from the linear program so far.

Since x^* is a basic feasible solution we have $|I'| \geq |J'|$ in Step 8. Each set $|\,\mathtt{supp}(A_j) \cap I'|$ for $j \in J'$ has size at most k. By double-counting $\sum_{i \in I'} |\,\mathtt{supp}(A_i) \cap J'| \leq k|J'| \leq k|I'|$ and so some $i \in I'$ has $|\,\mathtt{supp}(A_i) \cap J'| \leq k$. Thus $|I'|$ decreases in each iteration, and the algorithm has polynomial running time. (In fact, it is not hard to show that there are at most $O(k \log |I|)$ iterations.)

The algorithm has the property that $c(x^0 + x^1 + x^*)$ does not decrease from one iteration to the next, which implies property (a). Properties (b) and (c) can be seen immediately from the definition of the algorithm. □

Now we give the proof of the main result in this section. Here and later we abuse notation and identify vectors in $\{0,1\}^J$ with subsets of J, for notational convenience.

Theorem 7.2. *There is a polynomial time $(2k^2+2)$-approximation algorithm for k-CS PIPs with multiplicity constraints.*

Proof. The main idea in the proof is to partition the set x^1 into $2k^2+1$ sets which are all feasible (i.e., we get $x^1 = \sum_{j=1}^{2k^2+1} y^j$ for 0-1 vectors y^j each with $Ay^j \leq b$). If we can establish the existence of such a partition, then we are done as follows: the total profit of the $2k^2+2$ feasible solutions $x^0, y^1, \ldots, y^{2k^2+1}$ is $c(x^0 + x^1) \geq \text{OPT}$, so the most profitable is a $(2k^2+2)$-approximately optimal solution.

Call $j, j' \in x^1$ *in conflict at* i if $A_{ij} > 0, A_{ij'} > 0$ and at least one of (i,j) or (i,j') is special. We claim that if $y \subset x^1$ and no two elements of y are in conflict, then y is feasible; this follows from Lemma 7.11(c) together with the fact that $A_{ij} \leq b_i$ for all i, j. (Explicitly, for each constraint we either just load it with a single special entry, or all non-special entries, both of which are feasible.) In the remainder of the proof, we find a $(2k^2+1)$-colouring of the set x^1 such that similarly-coloured items are never in conflict; then the colour classes give the needed sets y^j and we are done.

To find our desired colouring, we create a *conflict digraph* which has node set x^1 and an arc (directed edge) from j to j' whenever j, j' are in conflict at i and (i,j) is special. Rewording, there is an arc (j,j') iff some $(i,j) \in S$ and $A_{ij'} > 0$. (If (i,j') is also special, this also implies an arc (j',j).) The key observation is that each node $j \in x^1$ has indegree bounded by k^2, i.e. there are at most k^2 choices of j such that (j,j') is an arc: to see this note $\#\{i \mid A_{ij'} > 0\} \leq k$, and each i in this set has $\#\{j \mid (i,j) \in S\} \leq k$. Now we use the following lemma, which completes the proof.

Lemma 7.12. *A digraph with maximum indegree d has a $2d+1$-colouring.*

Proof. We use induction on the number of nodes in the graph, with the base case being the empty graph Now suppose the graph is nonempty. The average indegree is at most d, and the average indegree equals the average outdegree. Hence some node n has outdegree at most the average, which is d. In total, this node has at most $2d$ neighbours. By induction there is a $(2d+1)$-colouring when we delete n, then we can extend it to the whole digraph by assigning n any colour not used by its neighbours. □

(We remark that Lemma 7.12 is tight, e.g. arrange $2d+1$ vertices on a circle and include an arc from each vertex to its d clockwise-next neighbours; this directed K_{2d+1} cannot be $2d$-coloured.) This ends the proof of Theorem 7.2. □

Finally, we give some small improvements for the case $k = 2$, using some insights due to Shepherd & Vetta [180]. A 2-CS PIP is *non-simple* if there exist distinct j, j' with $\texttt{supp}(A_j) = \texttt{supp}(A_{j'})$ and $|A_j| = 2$. Otherwise, it is simple.

Theorem 7.13. *There is a deterministic* 4*-approximation algorithm for 2-CS PIPs. There is also a randomized* $6 - \sqrt{5} \approx 3.764$*-approximation algorithm for simple 2-CS PIPs with* $d = \mathbf{1}$.

(Sketch). Since we are dealing with a 2-CS PIP, each $\texttt{supp}(A_j)$ is an edge or a loop on vertex set I; we abuse notation and directly associate j with an edge/loop. Consider the initial value of J', i.e. after executing Step 2. Then we claim that the graph (I, J') has at most one cycle per connected component; to see this, note that any connected component with two cycles would have more edges than vertices, which contradicts the linear independence of the tight constraints for the initial basic solution x^*.

We modify IteratedSolver slightly. Immediately after Step 2, let $M \subset J'$ consist of one edge from each cycle in (I, J'), and set $J' := J' \backslash M$. Then M is a matching (hence a feasible 0-1 solution) and the new J' is acyclic. Modify the cardinality condition in Step 8 to $|\texttt{supp}(A_i) \cap J'| \leq 1$ (instead of ≤ 2); since J' is acyclic, it is not hard to show the algorithm will still terminate, and $\forall i \in I$, we have $|\{j : (i, j) \in S\}| \leq 1$.

To get the first result, we use a colouring argument from [180, Thm. 4.1] which shows that x^1 can be decomposed into two feasible solutions $x^1 = y^1 + y^2$. We find that the most profitable of x^0, M, y^1, y^2 is a 4-approximately optimal solution.

For the second result, we instead apply a probabilistic technique from [180, §4.3]. They define a distribution over subsets of the forest x^1; let z be the random variable indicating the subset. Let $p = \frac{1}{20}(5 + \sqrt{5})$. Say that an edge ii' is *compatible* with z if z neither contains an edge with a special endpoint at i, nor at i'. The distribution has the properties that z is always feasible for the PIP, $\Pr[j \in z] = p$ for all $j \in x^1$, and $\Pr[\texttt{supp}(A_j)$ compatible with $z] \geq p$ for all $j \in x^0$. (Simplicity implies that x^0 and

x^1 have no edge in common, except possibly loops, which is needed here.) Finally, let w denote the subset of x^0 compatible with z. Then $z + w$ is a feasible solution, and $\mathrm{E}[c(z+w)] \geq pc(x^1+x^0)$. Hence the better solution of $z+w$ and M is a $1+1/p = (6-\sqrt{5})$-approximately optimal solution. □

7.3.1 Improvements For High Width

The *width* W of an integer program is $\min_{ij} b_i/A_{ij}$, taking the inner term to be $+\infty$ when $A_{ij} = 0$. Note that without loss of generality, $W \geq 1$. From now on let us normalize each constraint so that $b_i = 1$; then a program has width $\geq W$ iff every entry of A is at most $1/W$.

In many settings better approximation can be obtained as W increases. For example in k-RS CIPs with $b = \mathbf{1}$, the sum of each row of A is at most k/W, so Propositions 7.5 and 7.6 give a $(1 + k/W)$-approximation algorithm. Srinivasan [184, 185] gave a $(1 + \ln(1 + k)/W)$-approximation algorithm for unbounded k-CS CIPs. Using *grouping and scaling* techniques introduced by Kolliopoulos and Stein [133], Chekuri et al. [39] showed that no-bottleneck demand multicommodity flow in a tree admits a $(1+O(1/\sqrt{W}))$-approximation algorithm, and gave general sufficient conditions for a problem to admit a $(1 + O(1/\sqrt{W}))$-approximation algorithm. Along the same vein, using iterated rounding, in the previous chapter (Chapter 6) we obtained a $(1+O(1/W))$-approximation algorithm for multicommodity flow and covering in a tree.

Whereas Theorem 7.2 gives an approximation which is quadratic in k, the following result gives a significant improvement for high width. It treats a somewhat specialized case, but its main technique — using an LP to guide the reduction of an additively-violating solution to a feasible solution — is interesting and may have other applications.

Theorem 7.14. *There is a polynomial time* $(1+k/W)/(1-k/W)$*-approximation algorithm to solve* k*-column-sparse PIPs with* $k/W < 1$.

To make Theorem 7.14 more concrete, notice this implies a $1+O(k/W)$-approximation for $W > 1.01k$. This is tight in the sense that for any fixed $k \geq 4$, $1+o(1/W)$-approximation is NP-hard, by results from [87, 137] on approximating multicommodity flows in trees (see Section 6.2.4).

We remark that after the initial publication of this work [172], Bansal et al. [11] gave for each fixed integer $t \geq 2$ a $O(k^{1/t})$ approximation for k-CS PIPs with $W \geq t$, which is superior in many cases.

Proof of Theorem 7.14. Run IteratedSolver. From Lemma 7.11 we see that $c(x^0 + x^1) \geq \mathrm{OPT}$ and, using the width bound,

$$A(x^0 + x^1) \leq (1 + k/W)\mathbf{1}. \tag{7.4}$$

Define $\mathcal{V}(x)$ by $\mathcal{V}(x) := \{i \in I \mid A_i x > 1\}$, e.g. the set of violated constraints in $Ax \leq \mathbf{1}$.

We want to reduce $(x^0 + x^1)$ so that no constraints are violated. In order to do this we employ a linear program. Let $\chi(\cdot)$ denote the characteristic vector. Our LP, which takes a parameter $\widehat{x}$, is

$$\mathcal{R}(\widehat{x}) : \max\{cx \mid \mathbf{0} \leq x \leq \widehat{x}, Ax \leq \mathbf{1} - \frac{k}{W}\chi(\mathcal{V}(\widehat{x}))\}.$$

We can utilize this LP in an iterated rounding approach, described by the following pseudocode.

IteratedReducer
1: Let $\widehat{x} := x^0 + x^1$
2: **while** $\mathcal{V}(\widehat{x}) \neq \varnothing$ **do**
3: Let x^* be an extreme optimum of $\mathcal{R}(\widehat{x})$
4: Let $\widehat{x} = \lceil x^* \rceil$
5: **end while**

We claim that this algorithm terminates, and that the value of $c\widehat{x}$ upon termination is at least

$$\frac{1 - k/W}{1 + k/W} c(x^0 + x^1) \geq \frac{1 - k/W}{1 + k/W} \mathrm{OPT}.$$

Once we show these facts, we are done, since the for the final $\widehat{x}$, $\mathcal{V}(\widehat{x}) = \varnothing$ implies $\widehat{x}$ is feasible. As an initial remark, note that each coordinate of $\widehat{x}$ is monotonically nonincreasing, and so $\mathcal{V}(\widehat{x})$ is also monotonically nonincreasing.

Observe that $\mathcal{R}$ in the first iteration has $\frac{1-k/W}{1+k/W}(x^0 + x^1)$ as a feasible solution, by Equation (7.4). Next, note that x which is feasible for $\mathcal{R}$ in one iteration is also feasible for $\mathcal{R}$ in the next iteration since $\mathcal{V}(\widehat{x})$ is monotonically nonincreasing; hence the value of cx^* does not decrease between iterations.

To show the algorithm terminates, we will show that $\mathcal{V}(\widehat{x})$ becomes strictly smaller in each iteration. Note first that if $i \notin \mathcal{V}(\widehat{x})$, the constraint $A_i x \leq 1$ is already implied by the constraint $x \leq \widehat{x}$. Hence $\mathcal{R}(\widehat{x})$ may be viewed as having only $|\mathcal{V}(\widehat{x})|$ many constraints other than the box constraints $0 \leq x \leq \widehat{x}$. Then x, a basic feasible solution to $\mathcal{R}(\widehat{x})$, must have at most $|\mathcal{V}(\widehat{x})|$ non-integral variables. In particular, using the fact that the program is k-CS, by double counting, there exists some $i \in \mathcal{V}(\widehat{x})$ such that $\#\{j \mid x^*_j \notin \mathbf{Z}, A_{ij} > 0\} \leq k$. Thus (using the fact that all entries of A are at most $1/W$) we have $A_i\lceil x^* \rceil < A_i x^* + k(1/W) \leq 1$ — so $i \notin \mathcal{V}(\lceil x^* \rceil)$, and $\mathcal{V}(\widehat{x})$ is strictly smaller in the next iteration, as needed. □

7.4 Open Problems

There remain a few gaps in the types of problems we studied:

- It would be natural to determine whether submodular k-CS CIPs have an $O(\ln k)$ approximation (see Section 7.1.3).
- The approximability of k-CS CIPs is known so far only to be between $\Omega(k/\ln k)$ and $O(k)$; improving this would be significant, since even for the 0-1 case (hypergraph matching) no $o(k)$ approximation is known.
- Is there any general reason why the special 0-1 cases are essentially as hard as the general cases?

Although 2-RS IPs are very hard to optimize (at least as hard as Max Independent Set), the problem of finding a *feasible* solution to a 2-RS IP is still interesting. Hochbaum et al. [115] gave a pseudopolynomial-time 2-SAT-based feasibility algorithm for 2-RS IPs with finite upper and lower bounds on variables. They asked if there is a pseudopolynomial-time feasibility algorithm when the bounds are replaced by just the requirement of nonnegativity, which is still open as far as we know. It is strongly NP-hard to determine if IPs of the form $\{x \geq 0 \mid Ax = b\}$ are feasible when A is 2-CS [114], e.g. by a reduction from 3-Partition; but for IPs where each variable appears at most twice *including* in upper/lower bounds, it appears all that is known is NP-completeness (for example, via the *unbounded knapsack problem* [159]).

Acknowledgement.

We would like to thank Glencora Borradaile, Christina Boucher, Deeparnab Chakrabarty, Stephane Durocher, Jochen Könemann and Christos Koufogiannakis for helpful discussions, and the ESA referees for useful feedback.

7.5 Hardness of Column-Restricted 2-CS CIP

Theorem 7.15. *It is* NP*-hard to approximate 2-CS CIPs of the form* $\{\min cx \mid Ax \geq b, x$ *is* 0-1$\}$ *and* $\{\min cx \mid Ax \geq b, x \geq 0, x$ *integral*$\}$ *within ratio* $17/16 - \epsilon$ *even if the nonzeroes of every column of* A *are equal and* A *is of the block form* $\left[\begin{smallmatrix} A_1 \\ A_2 \end{smallmatrix}\right]$ *where each* A_i *is 1-CS.*

Proof. Our proof is a modification of a hardness proof from [35] for a budgeted allocation problem — we thank Deeparnab Chakrabarty for mentioning this to us. We focus on the version where x is 0-1; for the other version, no major modifications to the proof are needed. The specific problem described in the statement of the theorem is easily seen equivalent to the following problem, which we call *demand edge cover in bipartite multigraphs*: given a bipartite multigraph (V, E) where each vertex v has a demand b_v and each edge e has a cost c_e and value d_e, find a minimum-cost set E' of edges so that for each vertex v its demand is satisfied, meaning that $\sum_{e \in E' \cap \delta(v)} d_e \geq b_v$. Our construction also has the property that $c_e = d_e$ for each edge — so from now on we denote both by the variable d_e.

The proof uses a reduction from Max-3-Lin(2), which is the following optimization problem: given a collection $\{x_i\}_i$ of 0-1 variables and a family of three-variable modulo-2 equalities called *clauses* (for example, $x_1 + x_2 + x_3 \equiv 1 \pmod 2$), find an assignment of values to the variables which satisfies the maximum number of clauses. Håstad [110] showed that for any $\epsilon > 0$, it is NP-hard to distinguish between the two cases that (1) a $(1 - \epsilon)$ fraction of clauses can be satisfied and (2) at most a $(1/2 + \epsilon)$ fraction of clauses can be satisfied.

Given an instance of Max-3-Lin(2) we construct an instance of demand edge cover as follows. For each variable x_i there are three vertices "x_i", "$x_i = 0$" and "$x_i = 1$"; these vertices have b-value $4 \deg(x_i)$ where $\deg(x_i)$ denotes the number of clauses containing x_i. For each clause there are four vertices labelled by the four assignments to its variables that do *not* satisfy it; for example for the clause $x_1 + x_2 + x_3 \equiv 1 \pmod 2$ we would introduce four vertices, one of which would be named "$x_1 = 0, x_2 = 0, x_3 = 0$." These vertices have b-value equal to 3. Each vertex "$x_i = C$" is connected to "x_i" by an edge with d-value $4 \deg(x_i)$; each vertex v of the form "$x_{i_1} = C_1, x_{i_2} = C_2, x_{i_3} = C_3$" is incident to a total of nine edges each with d-value 1: three of these edges go to "$x_{i_j} = C_j$" for each $j = 1, 2, 3$. The construction is illustrated in Figure 7.2.

Let m denote the total number of clauses; so $\sum_i \deg(x_i) = 3m$. We claim that the optimal solution to this demand edge cover instance has cost $24m + 3t$ where t is the least possible number of unsatisfied clauses for the underlying Max-3-Lin(2) instance. If we can show this then we are done since Håstad's result shows we cannot distinguish whether the optimal cost is $\geq 24m + 3m(1/2 - \epsilon)$ or $\leq 24m + 3(\epsilon m)$; this gives an inapproximability ratio of $\frac{24+3/2-3\epsilon}{24+3\epsilon} \to 17/16$ as $\epsilon \to 0$, since no approximation ratio could produce a result within this range of indistinguishability on instances of value $\leq (24 + 3\epsilon)m$.

Let x^* denote a solution to the Max-3-Lin(2) instance with t unsatisfied clauses; we show how to obtain a demand edge cover E' of cost $24m + 3t$. We include in E' the edge between "x_i" and "$x_i = x_i^*$" for each i; this has total cost $\sum_i 4 \deg(x_i) = 12m$. For each satisfied clause $x_i + x_j + x_k \equiv C \pmod 2$, we include in E' all three edges between "$x_i = 1 - x_i^*$" and "$x_i = 1 - x_i^*, x_j = x_j^*, x_k = x_k^*$" and similarly for j, k, and one of each of the parallel triples between "$x_i = 1 - x_i^*, x_j = 1 - x_j^*, x_k = 1 - x_k^*$" and its

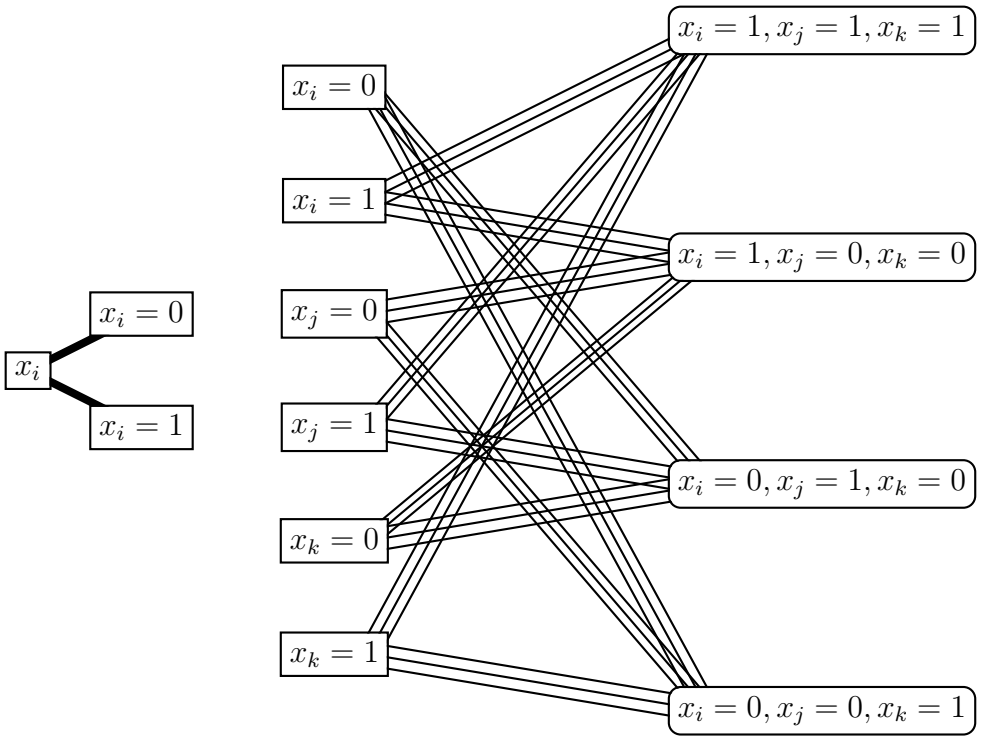


Figure 7.2: Left: the gadget constructed for each variable x_i. The vertices shown as rectangles have b-value $4\deg(x_i)$; the thick edges have d-value and cost $4\deg(x_i)$. Right: the gadget constructed for the clause $x_i + x_j + x_k \equiv 0 \pmod 2$. The vertices shown as rounded boxes have b-value 3; the thin edges each have unit d-value and cost.

neighbours; this has cost 12 for that clause. For each unsatisfied clause $x_i + x_j + x_k \equiv C \pmod 2$, we include in E' any three edges incident to "$x_i = x_i^*, x_j = x_j^*, x_k = x_k^*$," as well as the following twelve edges: the nodes "$x_i = 1 - x_i^*$" (symmetrically for j, k) and "$x_i = 1 - x_i^*, x_j = 1 - x_j^*, x_k = x_k^*$" (symmetrically for j, k) induce a 6-cycle of parallel triples, and we take two edges out of each triple; this has cost 15 for that clause. It is not hard to see that this solution is feasible — perhaps the trickiest vertices to check are those of the form "$x_i = 1 - x_i^*$," which are covered by 4 edges for each clause containing them. The total cost is $c(E') = 12m + 12(m - t) + 15t = 24m + 3t$.

To finish the proof we show the following.

Claim 7.16. *Given a feasible edge cover E', we can find a solution x^* such that t, the number of unsatisfied clauses for x^*, satisfies $24m + 3t \leq c(E')$.*

Proof. First we claim it is without loss of generality that for each i, E' contains exactly one of the edges incident to "x_i". Clearly at least one of these two edges lies in E'; if both do, then remove one (say, the edge between "x_i" and "$x_i = 0$") and add to E' any subset of the other $6\deg(x_i)$ edges incident to "$x_i = 0$" so that the total number of edges incident on "$x_i = 0$" in E' becomes at least $4\deg(x_i)$. The removed edge has d-value $4\deg(x_i)$ and all other incident edges have d-value 1, so clearly the solution is still feasible and the cost has not increased.

Define x^* so that for each i, E' contains the edge between "x_i" and "$x_i = x_i^*$." Let E'' denote the edges of E' incident on clause vertices (i.e. the edges of E' with unit d-value). For $F \subset E''$ their *left-contribution*, denoted $\ell(F)$, is the number of them incident on vertices of the form "$x_i = 1 - x_i^*$." Note that $\ell(F) \leq |F|$ for any F. Furthermore for each unsatisfied clause, all edges incident on its vertex "$x_i = x_i^*, x_j = x_j^*, x_k = x_k^*$" have zero left-contribution, but E' contains at least three of these edges. Thus the edges of E'' incident on that clause's vertices have $\ell(F) \leq |F| - 3$. Finally, consider $\ell(E'')$. Each edge of E'' is in the gadget for a particular clause, and it follows that $\ell(E'') \leq |E''| - 3t$ where t is the number of unsatisfied clauses for x^*. However, E'' needs to have $4\deg(x_i)$ edges incident on each "$x_i = 1 - x_i^*$" so $\ell(E'') \geq \sum_i 4\deg(x_i) = 12m$. Thus $|E''| \geq 12m + 3t$ and considering the edges incident on the vertices "x_i" we see that $c(E') \geq 24m + 3t$. □

This completes the proof of the reduction. □

Chapter 8

Extreme Points for Traveling Salesperson LP, and k-ECSS

In this chapter we describe a new family of extreme point solutions for a ubiquitous LP in network design. Up to scaling, the LP is the simplest relaxation for the traveling salesperson problem, the Steiner tree problem, and the k-edge connected spanning multi-subgraph problem. We find an extreme point on n vertices with minimum nonzero value exponentially small in n and whose support graph has maximum degree $n/2$. We also discuss approximability of the k-edge connected spanning subgraph problem; we show it is NP-hard to $(1+\epsilon)$-approximate for some fixed $\epsilon > 0$ independent of k, but conjecture some constant c exists such that there is a $(1 + c/k)$-approximation algorithm for the multi-subgraph version.

8.1 Introduction

In this section we study two closely related problems. In the k-edge-connected spanning subgraph problem (k-ECSS), we are given an input graph G with edge costs, and wish to select a minimum-cost subset of edges so that the resulting graph has edge-connectivity k between all vertices. The k-edge-connected spanning multi-subgraph problem (k-ECSM) differs only in that we seek a multi-subset of E, i.e. we are allowed to purchase each edge as many times as we want instead of just once.

The natural LP relaxation for k-ECSM, denoted ($\mathcal{N}_k$) and shown in Figure 8.1, is closely related to other network design problems, as we now explain. The objective cost function on edges is *metric* if $c_{uw} \leq c_{uv} + c_{vw}$ holds for all triples u, v, w of distinct vertices. In k-ECSM, we may assume without loss of generality that the cost function is metric,

$$\min \Big\{ \sum_{e\in E} c_e x_e : \quad x \in \mathbf{R}^E \qquad (\mathcal{N}_k)$$

$$\sum_{e\in\delta(S)} x_e \geq k, \quad \forall \varnothing \neq S \subsetneq V \qquad (8.1)$$

$$x_e \geq 0, \quad \forall e \in E \Big\} \qquad (8.2)$$

$$\min \Big\{ \sum_{e\in E} c_e x_e : \quad x \in \mathbf{R}^E \qquad (\mathcal{N}'_k)$$

$$\sum_{e\in\delta(v)} x_e = k, \quad \forall v \in V \qquad (8.3)$$

$$\sum_{e\in\delta(S)} x_e \geq k, \quad \forall \varnothing \neq S \subsetneq V \qquad (8.4)$$

$$x_e \geq 0, \quad \forall e \in E \Big\} \qquad (8.5)$$

Figure 8.1: The undirected relaxation for k-edge connected spanning multi-subgraph. The unbounded version ($\mathcal{N}_k$) is on the left, the bounded version ($\mathcal{N}'_k$) is on the right.

since at least one of the two edges uv, vw crosses every cut that the edge uw crosses. Further, Goemans & Bertsimas [93] proved that for metric cost functions, the constraint $\forall v \in V : x(\delta(v)) = 2$ can be added to ($\mathcal{N}_k$) without affecting the value of the LP; this is a consequence of what they called the *parsimonious property*, proved using the *splitting-off* operation of Mader and Lovász. We denote the resulting strengthened LP by ($\mathcal{N}'_k$). We also remark that by the parsimonious property, any ($\mathcal{N}'_k$)-relative approximation ratio for k-ECSM implies the same for *subset* k-ECSM, in which the vertices are partitioned into terminal and Steiner nodes, and we require edge-connectivity k only between each pair of terminals.

It is not too hard to show that in ($\mathcal{N}'_k$), the family (8.4) of constraints can be replaced by the family

$$\sum_{\{u,v\}\subset S} x_{uv} \leq \frac{k}{2}(|S| - 1), \quad \forall \varnothing \neq S \subsetneq V \qquad (8.6)$$

without affecting the feasible region. This shows that ($\mathcal{N}'_2$) is the same polyhedron as the *subtour relaxation* for the traveling salesperson problem [53] (TSP), which also equals the so-called 1-tree bound [112].

There are many network design problems which rely on structural properties of LPs derived from ($\mathcal{N}'_k$) and ($\mathcal{N}_k$): Jain [124] for skew-submodular network design, Gabow, Goemans, Tardos & Williamson [84] for unweighted k-edge connected spanning subgraph, Goemans [99] and Lau & Singh [182] for bounded-degree min-cost spanning tree, and extensions [148, 10]. For example, Jain's algorithm [124] relies on the fact that for every extreme point x of a certain LP, some edge has $x_e \geq 1/2$. Motivated by these results, we ask what other structural properties these LPs have, with an eye to designing approximation algorithms. We give a new lower bound for k-ECSS and conjecture for k-ECSM in Section 8.3. Then we give the construction of complex extreme points for ($\mathcal{N}'_2$) in Section 8.4,

followed by some discussion.

8.2 Literature Review

8.2.1 k-ECSS and k-ECSM

We assume for the rest of the chapter, in k-ECSM, that costs are metric since this is without loss of generality. When discussing k-ECSS we do not assume costs are metric. We conflate k-ECSS with its more general capacitated version where there is an upper limit d_e on the number of times each edge could be purchased, i.e. $x_e \leq d_e$ for all e, since this is similar to k-ECSS on a multigraph.

The case $k = 1$. A 1-ECSS or 1-ECSM is just an spanning connected subgraph, hence the 1-ECSS problem is essentially just the spanning tree problem. The quality of the relaxations is relatively poor; the worst-case integrality gap of ($\mathcal{N}_k$) is $2(1 - 1/n)$ [93].

The case $k = 2$. The 2-ECSS problem is closely related to its vertex-connectivity variant which we denote 2-VCSS, and the TSP problem. Frederickson & Jájá [76] gave 2-approximation algorithms for 2-VCSS and 2-ECSS — by Jain [124] we can also obtain an ($\mathcal{N}_2$)-relative 2-approximation. (Fleischer [67] gives a similar result for 2-VCSS.)

Using a simple version of splitting-off, Frederickson & Jájá [77] observed that the optimal values of 2-ECSS and 2-VCSS are the same for metric cost functions. Frederickson & Jájá [77] gave 3/2-approximation algorithms for metric min-cost 2-VCSS and 2-ECSS. Wolsey [198] showed that the integrality gap of ($\mathcal{N}_2'$) is at most 3/2 for metric costs by providing an LP-relative analysis of the Christofides heuristic [49]; there is a long-standing open conjecture that the integrality gap is at most 4/3, and this conjecture is supported by recent exhaustive computational experiments on graphs with at most 12 vertices [15, 23].

Fernandes [66] showed that 2-ECSS is APX-hard for unit costs. Czumaj & Lingas [52] extended this to 2-VCSS.

The problem of 2-ECSM is APX-hard and admits a 3/2-approximation algorithm. The approximation result was first given in [22]. To see hardness, first we note that splitting off parallel edges implies that 2-ECSM and 2-ECSS have the same optimal value on metric costs.[1] Hence, we need only to show that metric 2-ECSS is APX-hard. Now by [165], TSP is APX-hard even on complete graphs with costs 1 and 2, and an observation of [20] shows that a 2-ECSS can be transformed to a Hamiltonian cycle (TSP tour) by repeatedly replacing two edges with one edge, which does not incerase the overall cost if edge costs

[1] This gives a simpler proof of a 3/2-approximation for 2-ECSM, which is metric WOLOG, by using the 3/2-approximation algorithm of [77] for 2-ECSS.

are 1 and 2. Therefore, on these same (metric) instances, finding the min-cost 2-ECSM is APX-hard.

General k. Khuller & Vishkin [131] gave a 2-approximation for k-ECSS, and later results of Jain [124] imply more strongly that ($\mathcal{N}_k$) has an integrality gap of at most 2. Goemans & Bertsimas [97, 93] give a ($\mathcal{N}_k$)-relative approximation algorithm for (subset) k-ECSM with ratio $\frac{3}{2}$ when k is even, and $(\frac{3}{2} + \frac{1}{2k})$ when k is odd. For unit costs on simple graphs with unit upper bounds, Cheriyan & Thurimella [43] gave a $(1+2/(k+1))$-approximation algorithm for k-ECSS; for unit costs in general there is a simple ($\mathcal{N}_k$)-relative $(1+2/k)$-approximation algorithm for k-ECSS by Goemans et al. [84]. The latter two bounds were improved by Gabow & Gallagher [83] to $1 + \frac{1}{2k} + O(\frac{1}{k^2})$ and $1 + \frac{21}{11k}$, respectively.

Hardness Results, $k > 2$. Kortsarz et al. [142] showed that there is a fixed $\epsilon > 0$ so that for all $k > 1$, it is NP-hard to approximate k-VCSS better than $1+\epsilon$. This is obtained by taking hard instances of 2-VCSS [52] and adding $(k-2)$ new vertices, connected to all other vertices by 0-cost edges. As an aside, we mention that the best approximation ratio known for k-VCSS is currently $O(\log k \log \frac{n}{n-k})$ [164].

Gabow et al. [84] show that for some fixed $\epsilon > 0$, k-ECSS with unit costs is $(1+\epsilon/k)$-hard to approximate.

We remark that neither the construction of [142] nor that of [84] is useful to get a hardness result for k-ECSM, $k > 2$; indeed, no such hardness seems to be known.

8.2.2 Extreme Points

Existing work on extreme points of ($\mathcal{N}_k$) and ($\mathcal{N}'_k$) deals mostly with TSP and the case $k = 2$. The *min-norm* of an extreme point solution x is the minimum of its nonzero values. The *denominator* of x is the least integer d for which dx is integral. The *support graph* of a solution x is the graph $(V, \mathtt{supp}(x))$ obtained by retaining only edges e with $x_e > 0$; we abuse notation and identify the *maximum degree* and *number of edges* of the support graph as belonging to x.

Boyd and Pulleyblank [26, 24] showed that for any $t \geq 3$, there is an extreme point of ($\mathcal{N}'_2$) on $2t+4$ vertices with denominator t and values in $\{1/t, 2/t, 1-2/t, 1-1/t, 1\}$, hence min-norm $1/t$. In unpublished work, Cunningham & Zhang [51] observed that in ($\mathcal{N}_2$), any two extreme points with supports G_1, G_2 can be glued together by identifying one vertex of G_1 with one vertex of G_2; thereby it is easy to get an extreme point of ($\mathcal{N}_2$) on $2t+1$ vertices with maximum degree t, and using the Boyd-Pulleyblank construction, an extreme point on n vertices with denominator $\Omega(\sqrt{n}!)$. However, this gluing construction fails for ($\mathcal{N}'_2$). Also, from our approximation-algorithmic perspective, the non-2-vertex-connected extreme point solution obtained from gluing is not a serious obstacle to a good ratio for

k-ECSS since the algorithm could reduce to 2-vertex-connected components. We further suggest that any algorithm for arbitrary metric costs which uses the relaxation $(\mathcal{N}_2)$ may as well use the relaxation $(\mathcal{N}_2')$; to support this claim, we note that taking the metric closure and adding ϵ to every edge cost makes it *strictly metric* at which point splitting-off & parsimony imply that *every* optimal extreme point of $(\mathcal{N}_2)$ is an extreme point of $(\mathcal{N}_2')$. (Observe also every extreme point of $(\mathcal{N}_2')$ is an extreme point of $(\mathcal{N}_2)$.) Hence, we suggest that the central LP for studying bad extreme points should be $(\mathcal{N}_2')$.

Cheung [44] showed that extreme points of $(\mathcal{N}_2')$ can have arbitrarily high maximum degree. He gave a family of extreme points on $\Theta(t^2)$ vertices with maximum degree $4t+2$ and entries in $\{1/(2t+1), 1-1/(2t+1), 1\}$, for every integer $t \geq 1$.

There is substantial literature on the structure of extreme points for various supermodular network design problems, especially the max-norm since this relates to iterated LP rounding (e.g., see [82]). We mention one classical result on the max-norm of extreme points for $(\mathcal{N}_2')$ due to Boyd and Pulleyblank [24]: any extreme point has at least three edges e with $x_e = 1$.

8.3 Approximability of k-ECSM and k-ECSS

We now focus on qualitative properties of the approximability of k-ECSS. One of the striking features of the unit-cost version of k-ECSS is that it gets *easier* to approximate as k increases [84]. It is natural, therefore, to ask which other versions share this property. We provide a new lower bound and conjecture a new upper bound that will help to resolve this question. First, we introduce the lower bound, whose proof we defer to Section 8.3.1.

Theorem 8.1. *There is a fixed $\epsilon > 0$ so that for all $k \geq 2$, it is* NP*-hard to approximate k-ECSS within ratio $1+\epsilon$.*

The actual value of ϵ is quite minuscule, as is typical in APX-hardness proofs. Our conjectured positive result is the following:

Conjecture 8.2. *There is an $(\mathcal{N}_k)$-relative $(1+O(1/k))$-approximation algorithm for k-ECSM, i.e. one which produces a solution of value at most $(1+C_1/k)\mathrm{OPT}(\mathcal{N}_k)$ for some constant C_1.*

Note that for $k \in \{1,2\}$ the conjecture holds with $C_1 = 1$. As remarked earlier, a positive answer to this conjecture would also imply a $(1+C_1/k)$-approximation algorithm for subset k-ECSM.

With these results in mind, we now summarize known approximability results.

	Unit Costs		Arbitrary Costs	
	lower bound	upper bound	lower bound	upper bound
k-ECSM	$1+\epsilon$ for $k=2$	$1+O(1/k)$	$1+\epsilon$ for $k=2$	$\frac{3}{2}\{+\frac{1}{2k}\}_{\text{odd } k}$ & Conj. 8.2
k-ECSS	$1+\epsilon/k$	$1+O(1/k)$	$1+\epsilon$ (Thm. 8.1)	2

Table 8.1: Approximability of k-ECSS and k-ECSM. Note that all the upper bounds can be obtained by $(\mathcal{N}_k)$-relative approximation algorithms. Each ϵ represents a small fixed constant independent of k.

8.3.1 Proof of Hardness of k-ECSS (Theorem 8.1)

Our hardness reduction uses the following problem. (Here $\uplus$ denotes disjoint union.)

PATH-COVER-OF-TREE
Input: A tree $T=(V,E)$ and a set $\mathcal{X} \subset \binom{V}{2}$ of pairs.
Output: A subset of $\mathcal{Y}$ of $\mathcal{X}$ so that $(V, E \uplus \mathcal{Y})$ is 2-edge-connected.
Objective: Minimize $|\mathcal{Y}|$.

We call the problem PATH-COVER-OF-TREE for the following reason; for $X=\{x_1,x_2\} \in \mathcal{X}$ let P_X denote the unique path in T from x_1 to x_2.

Proposition 8.3. *$\mathcal{Y}$ is feasible for* PATH-COVER-OF-TREE *if and only if* $\bigcup_{X\in\mathcal{Y}} P_X = E$.

Proof. For every edge e or T, a *fundamental cut* of e and T means the vertex set of either connected component of $T \backslash e$.

First, $\mathcal{Y}$ is feasible if $|\delta_{(V,E\uplus\mathcal{Y})}(U)| \geq 2$ for every set U with $\varnothing \neq U \subsetneq V$. But $|\delta_{(V,E)}(U)|$ is 1 when U is a fundamental cut and at least 2 otherwise; hence $\mathcal{Y}$ is feasible iff $|\delta_{(V,\mathcal{Y})}(U)| \geq 1$ for every fundamental cut U.

Second, when U is a fundamental cut for e, $|\delta_{(V,\mathcal{Y})}(U)| \geq 1$ iff $\bigcup_{X\in\mathcal{Y}} P_X$ contains e. Taking this together with the previous paragraph, we are done. □

Consider the problem of augmenting a connected graph to become 2-edge-connected, using a minimum number of additional edges from a given set $\mathcal{X}$. It can be reduced to PATH-COVER-OF-TREE with T the tree of cut-edges/bridges (obtained by contracting every 2-edge-connected component). This problem is NP-hard by [76]; APX-hardness follows from [66]. (More strongly, one can adapt the hardness construction in [87] to show PATH-COVER-OF-TREE is APX-hard even for trees of depth 2; note depth 1 is poly-time since it is equivalent to the edge cover problem.)

Now we give our reduction. Let $T = (V, E), \mathcal{X}$ denote an instance of Path-Cover-of-Tree. We construct a k-ECSS instance on vertex set V. For each $e \in E$, our k-ECSS instance includes the edge e with multiplicity limit $d_e = k - 1$ and cost $c_e = 0$. For each $x = \{x_1, x_2\} \in \mathcal{X}$, our k-ECSS instance includes the edge $\{x_1, x_2\}$ with multiplicity limit $d_x = 1$ and cost $c_x = 1$.

Clearly, there is an optimal solution for the k-ECSS instance which includes $k - 1$ copies of each $e \in E$, which we denote by $(k-1)E$. Logic as above shows that $\mathcal{Y}$ is a feasible solution for the Path-Cover-of-Tree instance if and only if $(k-1)E \uplus \mathcal{Y}$ is a feasible solution for the k-ECSS instance. It immediately follows that an α-approximation algorithm for k-ECSS would also give an α-approximation algorithm for Path-Cover-of-Tree, and we are done.

Finally, we remark that it is possible to obtain a reduction in which the k-ECSS instance is simple with unit upper bounds (and still 0-1 costs). To do this, we replace every vertex V of the tree by a clique on $k + 1$ vertices connected by 0-cost edges; we replace every edge $uv \in E$ of the tree by any $k - 1$ 0-cost edges between members of the two cliques corresponding to u and v; each pair $\{u, v\} \in \mathcal{X}$ maps to a unit-cost edge between the cliques for u and v.

8.4 Complex Extreme Points for $(\mathcal{N}_2')$

Now we give our construction of a new family of extreme points for the TSP subtour relaxation $(\mathcal{N}_2')$. Let F_i denote the ith Fibonacci number, where $F_1 = F_2 = 1$. For a parameter $t \geq 3$, we denote the extreme point by x^*; the support graph of x^* has $2t$ vertices and $4t - 3$ edges with denominator F_t and maximum degree t.

The edges of the graph and x^* are given in the list below; it is pictured in Figure 8.2. (Note x^* has min-norm $1/F_t$.)

- For i from 1 to t, an edge $(2i - 1, 2i)$ of x^*-value 1
- For i from 2 to $t - 1$, an edge $(1, 2i)$ of x^*-value F_{t-i}/F_t
- An edge $(1, 2t)$ of x^*-value $1/F_t$
- For i from 3 to t, an edge $(2i - 3, 2i - 1)$ of x^*-value F_{t-i+1}/F_t
- For i from 3 to t, an edge $(2i - 4, 2i - 1)$ of x^*-value $1 - F_{t-i+2}/F_t$
- An edge $(2, 3)$ of x^*-value F_{t-1}/F_t
- An edge $(2t - 2, 2t)$ of x^*-value $1 - 1/F_t$

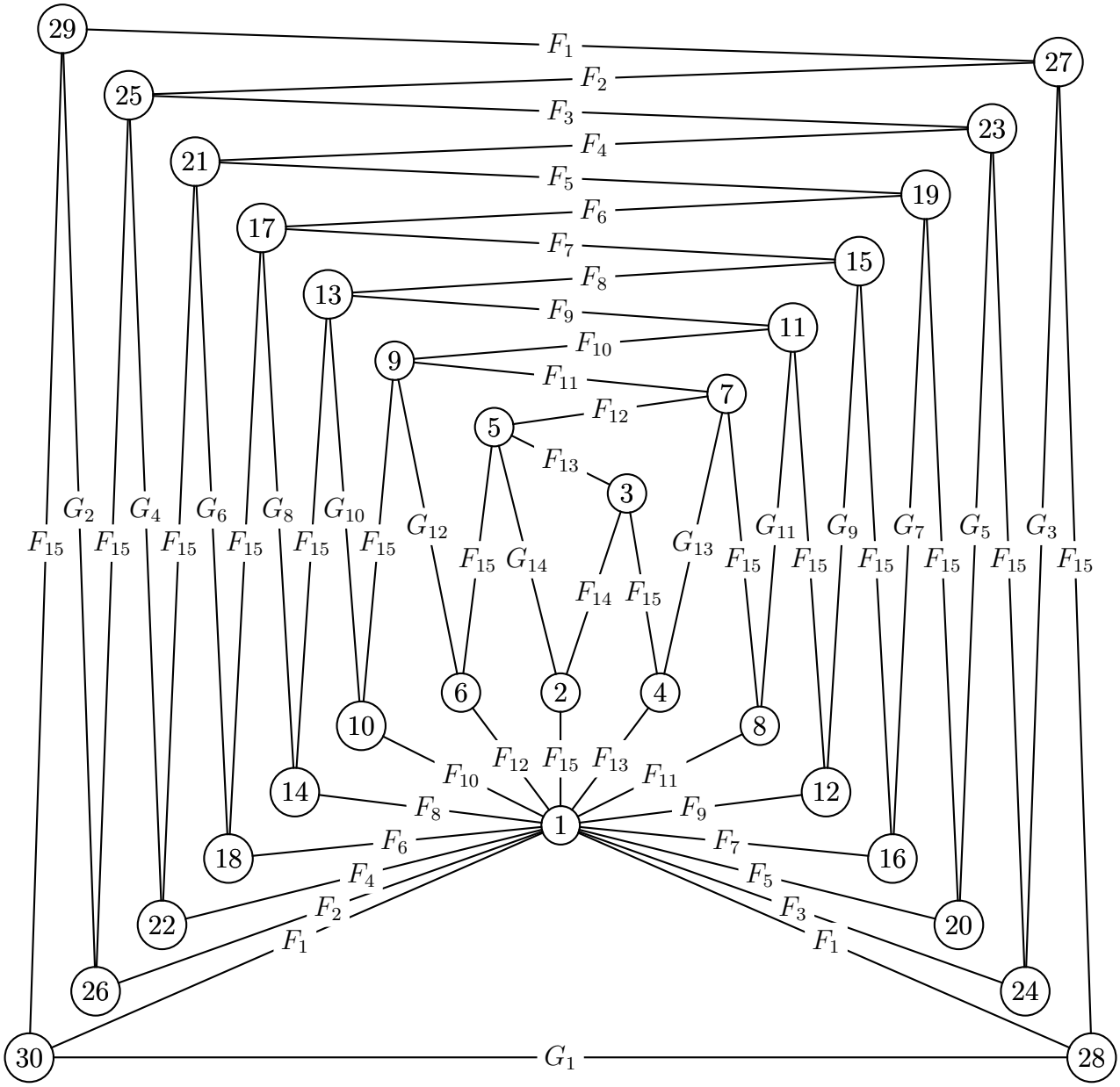


Figure 8.2: Our new construction of a complex extreme point x^* for the subtour TSP polytope $(\mathcal{N}_2')$, illustrated for $t = 15$. Scaled edge values are shown: the label F_i on an edge e indicates that $x_e^* = F_i/F_t$. The symbol G_i denotes $F_t - F_i$, i.e. an edge e with $x_e^* = 1 - (F_i/F_t)$.

Theorem 8.4. *The solution x^* described above is an extreme point solution for $(\mathcal{N}_2')$.*

Proof. With foresight, we write down the following family of $4t-3$ sets:

$$\mathcal{L} := \{\{i\}_{i=1}^{2t}, \{2i-1, 2i\}_{i=1}^{t}, \{1, \ldots, 2i\}_{i=2}^{t-2}\}.$$

The plan of our proof is to first show that x^* is the unique solution to $\{x(\delta(T)) = 2 \mid T \in \mathcal{L}\}$. It is easy to verify that x^* indeed satisfies all these conditions, so let us focus on the harder task of showing that x^* is the *only* solution. (Note, we are not assuming that x^* is feasible, so possibly $x^*(\delta(S)) < 2$ for some other sets, but we will deal with this later.)

Consider any solution which is tight for all sets in the given family. We first need a simple lemma, where for disjoint sets S, T, the symbol $\delta(S : T)$ denotes the set of edges with one end in S and the other in T.

Lemma 8.5. *If for some solution x, S, T are disjoint tight sets and $S \cup T$ is also tight, then $x(\delta(S : T)) = 1$.*

Proof. We have $\delta(S) = \delta(S : T) \uplus \delta(S : V \backslash S \backslash T)$ and $\delta(T) = \delta(S : T) \uplus \delta(T : V \backslash S \backslash T)$. Also, $\delta(S \cup T) = \delta(S : V \backslash S \backslash T) \uplus \delta(T : V \backslash S \backslash T)$. Thus $2 = x(\delta(S)) + x(\delta(T)) - x(\delta(S \cup T)) = 2x(\delta(S : T))$. □

Consider a hypothetical solution x with $x(\delta(S)) = 2, \forall x \in \mathcal{L}$. The lemma shows all edges $\{2i-1, 2i\}_{i=1}^{t}$ have x-value 1 (take $S = \{2i-1\}, T = \{2i\}$). Define y_i to denote be $x_{(2i+1,2i+3)}$ for i from 1 to $t-2$. The degree constraint at 3 (i.e., $x(\delta(3)) = 1$) forces $x_{(2,3)} = 1 - y_1$. The degree constraint at 2 forces $x_{(5,2)} = y_1$. Note $\{1, \ldots, 2t-2\}$ is tight since this set has the same constraint as $\{2t-1, 2t\}$. For i from 1 to $t-2$, note that the sets $\delta(\{1, \ldots, 2i\} : \{2i+1, 2i+2\})$ and $\delta(2i+1)$ differ only in that the former contains the edge $(2i+2, 1)$ and the latter contains the edges $\{(2i+1, 2i+2), (2i+1, 2i+3)\}$. Thus, using the lemma and degree constraint at $2i+1$, we see $x_{(2i+2,1)} + x_{(2i+1,2i+3)} = y_i$. The degree constraint at $2i+2$ then forces $x_{(2i+2,2i+5)} = 1 - y_i$ for $1 \le i \le t-3$. The degree constraint at $2t-2$ forces $x_{(1,2t-2)} = 1 - y_{t-2}$; the degree constraint at $2t$ forces $x_{(1,2t)} = y_{t-2}$. The degree constraint at $2t-1$ forces $y_{t-2} = y_{t-3}$, and the degree constraint at $2i+5$ forces $y_i = y_{i+1} + y_{i+2}$ for i from 1 to $t-4$; together this shows $y_i = F_{t-1-i} \cdot y_{t-2}$ for i from $t-4$ to 1 by induction. The degree constraint at 5 forces $2y_1 + y_2 = 1$, so $(2F_{t-2} + F_{t-3})y_{t-2} = 1$ and consequently $y_{t-2} = 1/F_t$. Thus it is true that $x = x^*$.

Now, we show x^* is feasible using standard uncrossing arguments, plus the fact that $|\mathcal{L}| = 4t-3$. For $(\mathcal{N}_2')$, it is not hard to show that the constraints (8.2) for S and $V \backslash S$ are equivalent, in the sense that they can be obtained from one another by adding or subtracting multiples of equality constraints. We fix any root vertex $r \in V$ and keep only the constraints for sets S not containing r, which does not change the LP. Correspondingly,

we change $\mathcal{L}$ by complementing the sets that contain r, and it is easy to see $\mathcal{L}$ is a laminar family on $V\backslash\{r\}$. (This is along the lines of the standard argument by Cornuéjols et al. [50].) In fact $\mathcal{L}$ is a maximal laminar family, since any laminar family of nonempty subsets of X contains at most $2|X|-1$ elements, for any set X.

Finally, suppose for the sake of contradiction that x^* is not feasible, so there is a set S, with $r \notin S$, having $x^*(\delta(S)) < 2$. Clearly $S \notin \mathcal{L}$. Two sets S, T, neither containing r, *cross* if all three of $S\backslash T$, $T\backslash S$, and $T \cap S$ are non-empty. Take S that crosses a minimal number of sets in $\mathcal{L}$. If S crosses zero sets in $\mathcal{L}$, then $\mathcal{L} \cup \{S\}$ is laminar, but this is a contradiction since $S \notin \mathcal{L}$ and, crucially, $\mathcal{L}$ was maximal. Otherwise, set S crosses some tight set $T \in \mathcal{L}$, then since

$$2 + 2 > x^*(\delta(S)) + x^*(\delta(T)) \geq x^*(\delta(S \cup T)) + x^*(\delta(S \cap T)),$$

either $x^*(\delta(S \cup T)) < 2$ or $x^*(\delta(S \cap T)) < 2$. It is easy to verify that both $S \cup T$ and $S \cap T$ cross fewer sets of $\mathcal{L}$ than S, contradicting our choice of S. □

8.4.1 Methodology

To investigate extreme points of $(\mathcal{N}_2')$, we first used computational methods to try to find the most "interesting" small examples. There are a number of properties that the support graph must have, e.g. no more than $2n-3$ edges, 3-vertex-connected (or else it is essentially a 2-sum of smaller solutions), and our method was to compute all extreme points on all such graphs. See Boyd [15, 23] for more discussion of how these steps can be implemented. We used `nauty` [161] to generate the graphs, and the Maple package `convex` [75] to enumerate extreme points. The Maple package available at the time did not have a good interface for laying out graphs, so we created a procedure [171] to export the graphs to GeoGebra [117], which is well-suited for layout (and exporting for diagrams in this document). We found the following interesting examples, which are pictured in Figure 8.3. Note "unique" means unique up to graph isomorphism.

(a) for $n \leq 6$, there is a unique extreme point with denominator ≥ 2

(b) for $n \leq 7$, there is a unique extreme point with maximum degree ≥ 4

(c) for $n \leq 8$, there is a unique extreme point with denominator ≥ 3

(d) for $n \leq 9$, there is a unique extreme point with maximum degree ≥ 5

(e) for $n \leq 9$, there is a unique extreme point with denominator ≥ 4

(f) for $n \leq 10$, the maximum degree that occurs is 5 and the maximum denominator is 5; there is a unique solution on 10 vertices that attains both simultaneously

We found that there was some primal structure and dual structure to the 10-vertex example which was shared with the smaller examples (a) and (c); these observations led to the family described in Section 8.4. We remark that the extreme points pictured, and more generally our new construction, do not coincide with the families of Boyd and Pulleyblank [24] or Cheung [44] for any choice of parameters.

8.4.2 Discussion

The construction given shows that extreme points on n vertices of the Held-Karp relaxation may have maximum support degree as big as $n/2$ and min-norm as small as $1/F_{n/2}$, for even n. A natural question is whether these bounds are maximal. Boyd, with Benoit [15] and Elliott-Magwood [23], has computed and posted online [25] a list of all vertices of the subtour elimination polytope for up to 12 vertices. Filtering through that data, we find the following facts.

Remark 8.6. *For 11-vertex solutions, the largest maximum degree is 6, the largest denominator is 8, and of 11-vertex solutions with maximum degree 6, the maximum denominator is 5 which is uniquely attained. For 12-vertex solutions, the largest maximum degree is 6, the largest denominator is 9, and of 12-vertex solutions with maximum degree 6, the maximum denominator is 8 which is uniquely attained.*

Hence for even n, $F_{n/2}$ is not the maximum possible denominator. Based on the available data, we conjecture the following.

Conjecture 8.7. *The maximum degree of extreme points on n vertices is exactly $\lceil n/2 \rceil$.*

The best upper bound we are aware of is $n - 3$, which follows from the fact that each basic solution has at most $2n - 3$ edges, plus an easy argument to eliminate degree-2 vertices.

Asymmetric TSP. Asymmetric TSP is the analogue of TSP for directed graphs: we are given a metric directed cost function on the complete digraph (V, A), and seek a min-cost directed Hamiltonian cycle. Recently Asadpour et al. [7] obtained a breakthrough $O(\log n / \log \log n)$ approximation for this problem; its analysis uses crucially the fact that extreme points of the natural LP relaxation

$$\{y \in \mathbf{R}_+^A : \forall \varnothing \neq U \subsetneq V, y(\delta^{\text{out}}(U)) \geq 1\} \tag{8.7}$$

have denominator bounded by $2^{O(n \ln n)}$. Our construction for (symmetric) TSP implies that for *asymmetric* TSP, the extreme points attain denominator exponential in n, as we now show.

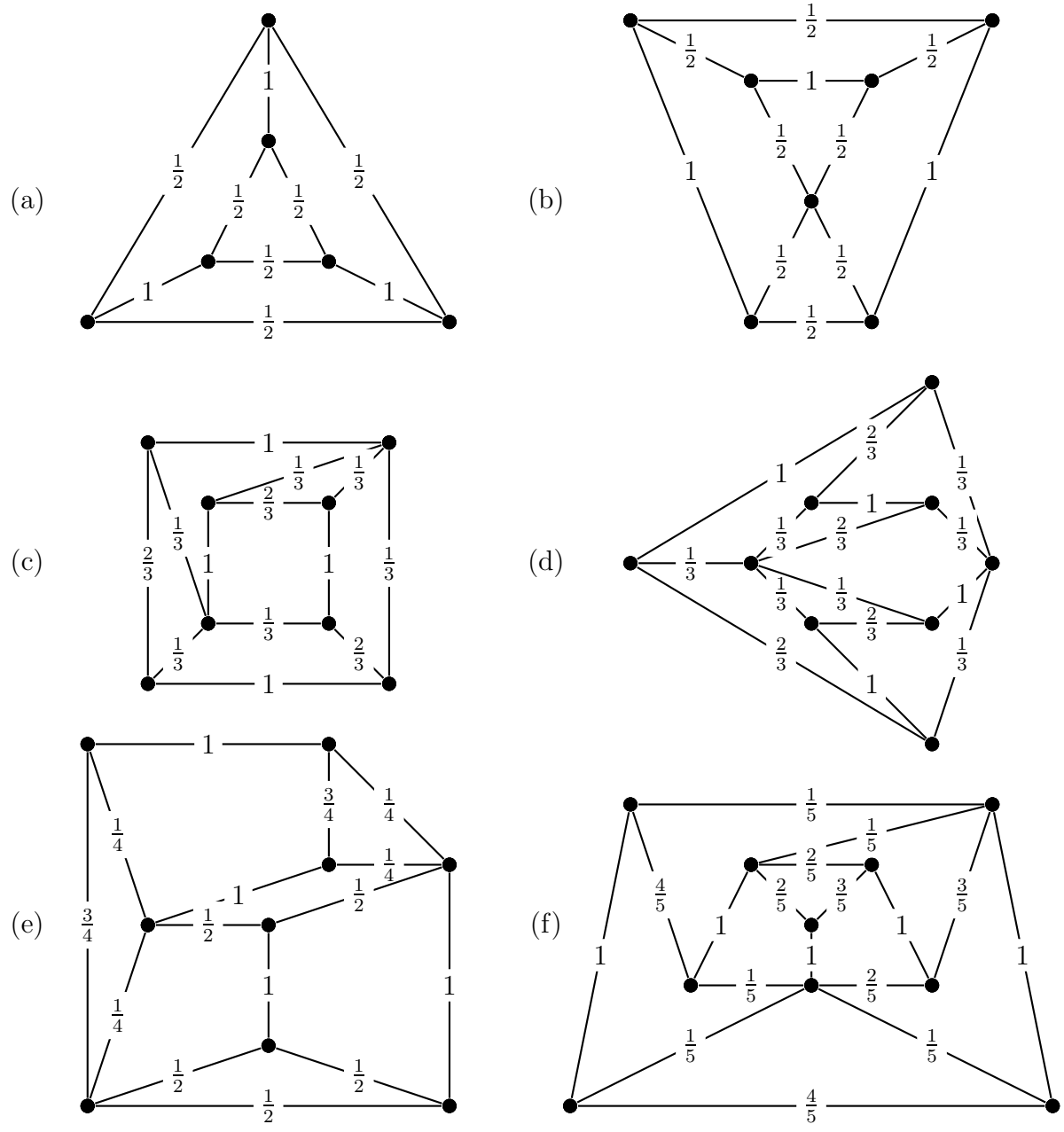


Figure 8.3: Six extreme points for the subtour TSP polytope ($\mathcal{N}_2'$) with extremal properties.

Proposition 8.8. *For every even $n \geq 6$ there are extreme points for (8.7) on n vertices with denominator at least $F_{n/2}$.*

Proof. The key is to note that $(\mathcal{N}_2)$ equals the projection of (8.7) to $\mathbf{R}_+^E$ obtained by setting $x_{\{u,v\}} = y_{(u,v)} + y_{(v,u)}$ for all $\{u,v\} \in \binom{V}{2}$ (call this map *dropping directions*). One direction is evident: given y, it has value at least 1 both coming into and coming out of every nontrivial cut set U, hence its undirected image x has value at least 2 spanning the cut it defines, i.e. $x(\delta(U)) \geq 2$. Conversely, to show that for every $x \in (\mathcal{N}_2)$, there is a $y \in$ (8.7) of this type, just assign $y_{(u,v)} = y_{(v,u)} = x_{\{u,v\}}/2$ for all $\{u,v\} \in \binom{V}{2}$.

Now we prove Proposition 8.8. Consider x^* given by the construction in this chapter, and consider the set of all y in (8.7) such that y becomes x^* when dropping directions. The argument in the previous paragraph establishes that this set is nonempty, and it is not hard to see this set is a face of (8.7). Finally, let y^* be any extreme point of this face. Our construction includes an edge e with $x^*_e = 1/F_{n/2}$, hence at least one of the two arcs corresponding to e has y^*-value in $(0, 1/F_{n/2}]$, giving the claimed result. □

An interesting future project is to check whether the extreme points for (8.7) obtained by the above construction have *strictly* larger denominator than $F_{n/2}$.

Integrality gap. Several papers of Boyd and coauthors investigate TSP LP extreme points with the goal of lower-bounding the integrality gap, therefore it is natural to ask what integrality gap is implied by the construction given in this paper. It does not appear that our construction gives a good integrality gap lower bound; for $6, 8, 10, 12$ vertices we have computed that the integrality gap obtained is only $\frac{9}{8}, \frac{23}{21}, \frac{22}{20}, \frac{35}{32}$. (Specifically, this value is the least $t \geq 0$ such the extreme point is dominated by t times a convex combination of indicator vectors of Hamiltonian cycles.)

Acknowledgement

I thank Ashkan Aazami, Deeparnab Chakrabarty, Michel Goemans, Nick Harvey, and Jochen Könmenann for useful discussions on the topics in this chapter.

Chapter 9

LP-Relative Approximation Scheme for k-Dimensional Knapsack

In this chapter we give a new LP construction for packing (or covering) nonnegative integer programs with at most k constraints. Specifically, for any fixed integer k and any $\epsilon > 0$, we get a polynomial-size linear program admitting an LP-relative $(1+\epsilon)$-approximation algorithm, and hence integrality gap at most $1+\epsilon$. The framework handles 0-1 variables, unbounded variables, or arbitrary upper bounds. This generalizes a recent result of Bienstock [18] for packing with $k = 1$, i.e., the knapsack problem.

9.1 Introduction

The classical *knapsack problem* is the following: given a collection of items each with a value and a weight, and given a weight limit, find a subset of items whose total weight is at most the weight limit, and whose value is maximized. If n denotes the number of items, this can be formulated as the integer program $\{\max \sum_{i=1}^{n} x_i v_i \mid x \in \{0,1\}^n, \sum_{i=1}^{n} x_i w_i \leq \ell\}$ where n denotes the number of items, v_i denotes the value of item i, w_i denotes the weight of item i, and ℓ denotes the weight limit.

In the more general *k-dimensional knapsack* (or k-constrained knapsack) problem, there are k different kinds of "weight" and a limit for each kind. An example for $k = 3$ would be a robber who is separately constrained by the weight, volume, and mercury content of the items he is stealing. An orthogonal generalization is that the robber could take multiple items of each item i, up to d_i copies. We therefore model the k-dimensional knapsack problem as

$$\{\max cx \mid x \in \mathbf{Z}^n, 0 \leq x \leq d, Ax \leq b\} \tag{9.1}$$

where A is a k-by-n matrix, b is a vector of length k, and d is a vector of length n, all non-negative. If $d = \mathbf{1}$ we call it the 0-1 *knapsack problem*. If $d = +\infty$, we call it the *unbounded knapsack problem*; otherwise we call it the *bounded knapsack problem*.

Another natural generalization is the *k-dimensional knapsack-cover problem*,

$$\{\min cx \mid x \in \mathbf{Z}^n, 0 \leq x \leq d, Ax \geq b\}$$

which has analogous bounded, unbounded, and 0-1 variants.

On the positive side, for any fixed k, all above variants admit a simple pseudo-polynomial-time dynamic programming solution. Chandra et al. [38] gave the first PTAS for k-dimensional knapsack in 1976, and later an LP-based scheme was given by Frieze and Clarke [78]. See the book by Kellerer et al. [128, §9.4.2] for a more comprehensive literature review. Also, for any fixed k, all above variants admit a polynomial-time approximation scheme (PTAS). But whereas the case $k = 1$ also admits a fully polynomial-time approximation scheme (FPTAS), for $k \geq 2$ there is no FPTAS unless P=NP. This was originally shown for 0-1 k-dimensional knapsack by Gens & Levner [88] and Korte & Schrader [140] (see also [128]) and subsequently for arbitrary d by Magazine & Chern [160]. For any optimization problem with a polynomially bounded objective the following is known: if it is strongly NP-hard, it has no FPTAS unless P=NP [86]. We remark that the converse is false, as shown by k-dimensional knapsack.

We recall for comparison a result of Lenstra [150]: for any fixed integer k, integer programs with k constraints can be solved in polynomial time. Compare this with the unbounded k-dimensional knapsack problem $\{\max cx \mid x \in \mathbf{Z}^n, x \geq 0, Ax \leq b\}$ which has k constraints *in addition to* nonnegativity constraints. Lueker [159] showed this is NP-complete even for $k = 1$; so the nonnegativity constraints make an important difference.

This chapter draws on a recent paper of Bienstock [18] which gives an LP-relative approximation scheme for the standard (1-dimensional) knapsack problem, using disjunctive programming. This chapter also draws on the earlier LP-based approximation schemes for k-dimensional knapsack and knapsack-cover. A key technique in both is to exhaustively guess the g max-cost items in the knapsack for some constant g, which we extend. Bienstock & McClosky [19] extend this line of work to covering problems and other settings, and also give an LP of size $O((1/\epsilon)^{1/\epsilon}n^2)$ with integrality gap $1 + \epsilon$ for 1-dimensional, 0-1 knapsack. In this chapter we give a much more general result, but with size $O(n^{O(k/\epsilon)})$. There is some current work [31] on obtaining primal-dual algorithms (that is, not needing the ellipsoid method or interior-point subroutines) for knapsack-type covering problems with good approximation ratio and [19] reports that the methods of [31] extend to a combinatorial LP-relative approximation scheme for 1-dimensional covering knapsack. It would be interesting to see if these methods also apply to the work stated in this chapter.

We give two other recent developments in this field. There is a line of work in *counting* the number of feasible solutions to a given k-dimensional knapsack problem (in which case

there is no objective function c) and Dyer [57] recently gave a simple dynamic programming-based FPRAS (fully-polynomial time randomized approximation scheme) to count the number of feasible solutions for k-dimensional bounded packing knapsack. Separate from this, there is a line of work on maximizing constrained submodular functions. For non-monotone submodular maximization subject to k linear packing constraints, the state of the art is by Lee et al. [149] who give a $(5+\epsilon)$-approximation algorithm. For monotone submodular maximization the state of the art is by Chekuri & Vondrák [40] who give a $(e/(e-1)+\epsilon)$-approximation subject to k knapsack constraints and a matroid constraint. We note it is NP-hard to obtain any factor better than $e/(e-1)$ for monotone submodular maximization over a matroid [65], so as in the original knapsack problems, the knapsack constraints only affect the best possible approximation ratio by ϵ.

9.2 Rounding

We follow the exposition of [128] and generalize the construction of [18]. Each knapsack instance (9.1) is determined by the parameters (A, b, c, d). The natural LP relaxation of the knapsack problem is

$$\{\max cx \mid 0 \le x \le d, Ax \le b\}. \qquad \mathcal{K}(A, b, c, d)$$

In the following, *fractional* means non-integral. We omit the easy proof of the following lemma.

Lemma 9.1. *Let x^* be an extreme point solution to the linear program ($\mathcal{K}(A, b, c, d)$). Then x^* is fractional in at most k coordinates.*

Therefore, we obtain the following primitive guarantee on a rounding strategy. Let $c_{\max} := \max_i c_i$.

Corollary 9.2. *Let x^* be an extreme point solution to the linear program ($\mathcal{K}(A, b, c, d)$). Then $c\lfloor x^* \rfloor \ge cx^* - kc_{\max}$.*

Now the idea is to take x^* to be an optimal solution, and use a standard exhaustive guessing step to turn the additive guarantee into a multiplicative factor of $1+\epsilon$. Let γ denote a parameter, which represents the size of a multi-set we will guess. For a non-negative vector z let the notation $\|z\|_1$ mean $\sum_i z_i$. A *guess* is an integral vector g with $0 \le g \le d, Ag \le b$ and $\|g\|_1 \le \gamma$. It is easy to see there are no more than $(n+1)^\gamma$ possible guesses, so for constant γ we can iterate through all guesses in polynomial time. First, we can therefore determine the guess with best objective value in polynomial time. Second, this simplifies the task of determining an approximately-optimal solution, as we now explain.

From now on we assume without loss of generality (by reordering items if necessary) that $c_1 \leq c_2 \leq \cdots \leq c_n$. For a guess g with $\|g\|_1 = \gamma$ we now define the *residual knapsack problem* for g; under the restriction that the γ most profitable[1] items chosen (counting multiplicity) are g, the residual problem models how to optimally select the remaining objects. Let $\mu(g)$ denote $\min\{i \mid g_i > 0\}$. Define d^g to be the first $\mu(g)$ coordinates of $d - g$ followed by $n - \mu(g)$ zeroes, and $b^g = b - Ag$. The *residual knapsack problem* for g is (A, b^g, c, d^g). The residual problem for g only permits taking items with index at most $\mu(g)$ and so essentially has its $c_{\max}$ value bounded by $c_{\mu(g)}$, which is at most $c \cdot g / \|g\|_1 = c \cdot g / \gamma$.

If a guess g has $\|g\|_1 < \gamma$, define b^g and d^g to be all-zero. Then Corollary 9.2 gives the following.

Corollary 9.3. *Let* OPT *be an optimal integral knapsack solution for (A, b, c, d). Let g be the γ most profitable items in* OPT *(or all, if there are less than γ). Let x^* be an optimal extreme point solution to $\mathcal{K}(A, b^g, c, d^g)$. Then $g + \lfloor x^* \rfloor$ is a feasible knapsack solution for (A, b, c, d) with value at least $1 - k/\gamma$ times optimal.*

By taking $\gamma = k/\epsilon$ and solving $\mathcal{K}(A, b^g, c, d^g)$ for all possible g we get the previously known PTAS for k-dimensional knapsack; we now refine the approach to get a single LP with small integrality gap.

9.3 Disjunctive Programming

In this section we give a brief introduction to *disjunctive programming*, e.g. see early work of Balas [8]. For our purposes we will only derive one central result, that it is possible to write a compact LP for the convex hull of the union of several polytopes, provided that we we have compact LPs for each one.

Suppose we have polyhedra $P^1 = \{x \in \mathbf{R}^n \mid A^1 x \leq b^1\}$ and $P^2 = \{x \in \mathbf{R}^n \mid A^2 x \leq b^2\}$. Both of these sets are convex and it is therefore easy to see that the convex hull of their union is the set

$$\text{conv.hull}(P^1 \cup P^2) = \{x \in \mathbf{R}^n \mid x = \lambda x^1 + (1-\lambda)x^2, 0 \leq \lambda \leq 1, A^1 x^1 \leq b^1, A^1 x^2 \leq b^2\}.$$

However, this is not a *linear* program, e.g. since we multiply the variable λ by the variables x^1. Nonetheless, it is not hard to see that the following is a linear formulation of the same set:

$$\text{conv.hull}(P^1 \cup P^2) = \{x \in \mathbf{R}^n \mid x = x^1 + x^2, 0 \leq \lambda \leq 1, A^1 x^1 \leq \lambda b^1, A^1 x^2 \leq (1-\lambda) b^2\}.$$

[1] To simplify the description, even if $c_{i+1} = c_i$ we think of item $i+1$ as more profitable than item i.

A similar construction gives the convex hull of the union of any number of polyhedra; we now apply this to the knapsack setting.

It is convenient to "shift" the LP $\mathcal{K}(A, b^g, c, d^g)$ of each residual problem by taking the Minkowski sum with g; feasible integral solutions of the shifted version correspond to knapsack solutions in which the γ most profitable items are g. The shifted LP is $\{y = x + g \mid x \in \mathbf{R}^n, 0 \le x \le d^g, Ax \le b^g\}$ (in which g is a constant).

Let $\mathcal{G}$ denote the set of all possible guesses g. Then the convex hull of the union of the shifted polyhedra is

$$\left\{ y \mid y = \sum_{g \in \mathcal{G}} y^g; \sum_{g \in \mathcal{G}} \lambda^g = 1; \lambda \ge \mathbf{0}; \forall g : y^g = x^g + \lambda^g g, \mathbf{0} \le x^g \le \lambda^g d^g, Ay^g \le \lambda^g b^g \right\}. \quad (\widetilde{\mathcal{K}})$$

We also think of $(\widetilde{\mathcal{K}})$ as a linear program by giving it the objective function $\max c \cdot y$. Finally, we complete our analysis.

Theorem 9.4. *If $\gamma = k/\epsilon$, there is a polynomial-time $(\widetilde{\mathcal{K}})$-relative $1/(1-\epsilon)$-approximation algorithm for the (0-1, unbounded, or bounded) k-dimensional knapsack problem.*

Proof. Let y be an optimal extreme point solution for $(\widetilde{\mathcal{K}})$. It is easy to argue that any extreme point solution has $\lambda^{g^*} = 1$ for some particular g^*, and $\lambda^g = 0$ for all other g. Hence $y = x^{g^*} + g^*$ where x^{g^*} is an optimal extreme point solution to $\mathcal{K}(A, b^{g^*}, c, d^{g^*})$. We now show that $\lfloor y \rfloor$ is a $(1-\epsilon)$-approximately optimal solution, re-using the previous arguments.

If $\|g^*\|_1 < \gamma$, then $x^{g^*} = 0$ so y is integral, hence y is an optimal knapsack solution. Otherwise, if $\|g^*\|_1 = \gamma$, then Corollary 9.2 shows that

$$c \cdot \lfloor y \rfloor = c \cdot \lfloor x^{g^*} \rfloor + c \cdot g^* \ge c \cdot x^{g^*} - k\frac{c \cdot g^*}{\gamma} + c \cdot g^* = c \cdot y - k\frac{c \cdot g^*}{\gamma} \ge (1-\epsilon) c \cdot y,$$

which completes the proof. □

A proof along the similar lines gives an LP-relative approximation scheme for the knapsack-cover problem. Let g be an integral vector with $0 \le g \le d, \|g\|_1 \le \gamma$; we define $\mu(g), d^g$ as before and call g a *guess* if $A(g + d^g) \ge b$, in which case we set b^g to be the component-wise maximum of $\mathbf{0}$ and $b - Ag$. Informally, this excludes g's for which the residual problem would be infeasible. Then the approach goes through as before, except we round up instead of down.

Somewhat imaginatively, we see it possible that the LPs constructed in this section could have applications; for example, we have seen before (the use of Jain's algorithm in Theorem 6.1(b)) that LP-relativity can be a crucial property when using an LP as a building block in a larger scheme. However, we do not see a concrete application, and the large size $n^{O(k/\epsilon)}$ of the LPs makes them seem mostly of theoretical interest.

Open Problems/Future Work

In this final section we collect the most pressing open questions mentioned in the thesis.

Steiner Tree LPs (Chapters 2–5)

The first and foremost question would be to improve the best known approximation ratio for the Steiner tree problem, as well as integrality gaps for the hypergraphic and bidirected LPs. After this thesis was submitted to its examiners, a preprint of Byrka et al. [27] was circulated to the author with new progress on these issues: a novel analysis of RZ giving a ($\mathcal{D}$)-relative 1.55-approximation, and a non-LP-relative ($\frac{3}{2} + \epsilon$)-approximation using ($\mathcal{D}$). (Other than this paragraph, the rest of the thesis has not been updated to reflect these developments.) Improving the hardness result of [45] would also be important progress towards the ideal of finding matching approximation and hardness bounds.

We noted that the hypergraphic relaxations can be solved in polynomial time on r-restricted instances, and approximately solved in general, but it is open whether we can solve them exactly in general.

One of our results is that the bidirected and hypergraphic LPs have the same value on quasi-bipartite instances. We give a 4-quasi-bipartite example in Section 4.6.4 where they have different values. Are the LPs equal on 2- and 3-quasi-bipartite instances?

We give in Section 5.6 an LP ($\mathcal{W}$) for the Steiner tree problem whose integrality gap is at most $1 + \ln 2$. Can we solve ($\mathcal{W}$) in polynomial time? Is there any relation between ($\mathcal{W}$) and the hypergraphic ones like ($\mathcal{P}$) — for example, does one strengthen the other?

Are there any interesting applications of the contraction lemma (Lemma 5.18) in its generalization to matroids?

Multicommodity Flow in Trees (Chapter 6)

Can we get $(1 + \epsilon/\mu)$-hardness of approximation for multicommodity *covering* in trees (where μ denotes the minimum capacity and $\epsilon > 0$ is a constant independent of μ)? Unlike multicommodity flow in trees, the problem is APX-hard even when all capacities are unit — see Section 8.3.1.

Can we make algorithmic use of the LP (6.9)–(6.11) for multicommodity flow in trees? Recall that it has several nice properties: integrality in unit-cost instances, integrality in spiders, and polynomial-time solvability.

For demand multicommodity flow in a tree (possibly with the *no-bottleneck assumption* [39]) can we get an approximation algorithm of the form $1+O(1/W)$ where W denotes the width?

Sparse Integer Programs (Chapter 7)

The follow-up work of Bansal et al. [11] — an $O(k)$-approximation algorithm for k-column sparse packing integer problems — answers the most important open question from Chapter 7. Further directions in sparse integer programming research are to map out the approximability of versions with semimodular objectives (Section 7.1.3) and/or high width (Section 7.3.1).

k-Edge Connectivity Problems (Chapter 8)

It would be interesting to obtain any hardness result for the k-edge connected spanning multi-subgraph problem when $k > 3$.

Is Conjecture 8.2 true? More weakly, can we find a non-LP-based $1+O(1/k)$-approximation algorithm for the k-edge connected spanning multi-subgraph problem?

Multidimensional Knapsack LPs (Chapter 9)

For k-dimensional knapsack problems, we gave a new LP admitting an LP-relative $(1 + \epsilon)$ approximation algorithm. Can we get a parallel result in which the algorithm is combinatorial (e.g., primal-dual, along the lines of [31])?

Appendix A

Integrality of Cost-Polytope from Section 4.6.3

```
restart: with(convex): with(combinat): with(networks):

# compute all maximal laminar families on a ground set S
LF := proc(S) local mg, rep, result:
if nops(S)=1 then return {{S}}: end:
rep := rand();
result := NULL;
for mg in choose(S, 2) do
  result := result, op(map(e->e union {{mg[1]}, {mg[2]}},
            subs(rep=(op(mg)), LF(S minus mg union {rep}))));
end:
return {result}:
end:

# convert list of vertices in a path to list of edges in a path
PathHelper := L -> seq({L[i], L[i+1]}, i=1..nops(L)-1):

# find all edges on the unique s-t path in tree G
Path := (G, s, t) -> PathHelper(path([s, t], shortpathtree(G, s))):

# find all edges on subfullcomponent for terminal set S in full component G
FullComp := (G, S) -> {seq(Path(G, st[1], st[2]), st in choose(S, 2))}:
```

```
# check whether the polyhedron for terminals {1,...,k}, laminar family Q,
# Steiner node set N, and edge set e is integral
# It prints out any non-integral vertices found, and is silent otherwise.
is01 := proc(k, Q, N, e)
local result, E, R, A, P, G, S, fc, P1, i, ineq, V;
R := {seq(i,i=1..k)};
A := [op(e), op(map(x->[x[2], x[1]], e))];
E := map(x->{op(x)}, e);
P := [op(Q)];
G := graph(N union R, E):
ineq := NULL:
for i from 1 to k do ineq := ineq, [seq(0, i=1..nops(E)),
 seq(`if`(i in P[j], 1, 0), j=1..nops(P))] = 1; end:
for S in combinat[powerset](R) do
    fc := FullComp(G, S):
    ineq := ineq, ([seq(`if`(E[i] in fc, 1, 0), i=1..nops(E)),
     seq(`if`((nops(P[j] intersect S)>0), -1, 0), j=1..nops(P))] >= -1);
end:
P1 := intersection(ineq, convert(posorthant(nops(E)+nops(P)), affine));
result := true:
for V in convex[vertices](P1) do
 if (nops({op(V)} minus {0, 1})>0) then
  print(Q, N, e, V):
  result := false:
 end:
end:
return result:
end:

# check all possible laminar families for terminals {1,...,k},
# Steiner node set N, and each edge set e in E
Test := proc(k, N, E) local W, R, i, e, result:
R := {seq(i,i=1..k)};
W := LF(R):
result := true;
for i from 1 to nops(W) do for e in E do
  result := result and is01(k, W[i], N, e):
end: end:
if (result) then return "Testing Complete, all 0-1"
```

```
 else return "Testing Complete, not all 0-1": end:
end:

### end of preliminary procedures, start of main session
# this fails to be integral and demonstrates laminarity is needed
is01(3, [{1, 2}, {1, 3}, {2, 3}], {a}, [[1, a], [2, a], [3, a]]);
```

$$[\{1, 2\}, \{1, 3\}, \{2, 3\}], \{a\}, [[1, a], [2, a], [3, a]], [0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$$

$$[\{1, 2\}, \{1, 3\}, \{2, 3\}], \{a\}, [[1, a], [2, a], [3, a]], [\frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$$

$$[\{1, 2\}, \{1, 3\}, \{2, 3\}], \{a\}, [[1, a], [2, a], [3, a]], [\frac{1}{2}, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$$

$$[\{1, 2\}, \{1, 3\}, \{2, 3\}], \{a\}, [[1, a], [2, a], [3, a]], [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$$

false

```
# now try with a laminar family
is01(3, [{1}, {2}, {3}, {1, 2}, {1, 2, 3}], {a}, [[1, a], [2, a], [3, a]]);
```

true

```
Test(3, {a}, [[[1, a], [2, a], [3, a]]]);
```

"Testing Complete, all 0-1"

```
Test(4, {a, b}, [[[1, a], [2, a], [a, b], [b, 3], [b, 4]]]);
```

"Testing Complete, all 0-1"

```
Test(5, {a, b, c}, [[[1, a], [2, a], [a, b], [b, 3], [b, c], [c, 4], [c, 5]]]);
```

"Testing Complete, all 0-1"

```
# this laminar family on 6 terminals fails to be integral
# it is taken from Figure 4.2 in the thesis
is01(6, [{1, 6}, {2, 3}, {4, 5}], {a, b, c, d},
[[1, a], [2, a], [a, b], [b, c], [c, 3], [c, 4], [b, d], [d, 5], [d, 6]]);
```

$$[\{1, 6\}, \{2, 3\}, \{4, 5\}], \{a, b, c, d\},$$
$$[[1, a], [2, a], [a, b], [b, c], [c, 3], [c, 4], [b, d], [d, 5], [d, 6]],$$
$$[\frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, 1, 1, 1]$$

false

Appendix B

Enumerating Vertices of $(\mathcal{P})$

```
# helper to compute partitions
allAugs := proc(list_of_sets, elt) local res, i, j, tmp;
 res := NULL;
 for j from 1 to nops(list_of_sets) do
  tmp := NULL;
  for i from 1 to nops(list_of_sets) do
   if (i=j) then tmp := tmp, (list_of_sets[i] union {elt});
   else tmp := tmp, list_of_sets[i]; end;
  end;
  res := res, {tmp};
 end;
 return res;
end:

# compute all partitions of set S
partitionsOf := proc(S) local elt, rec, Sprime, i, res;
  if (nops(S)=1) then return [{S}]; end;
  elt := S[1];
  Sprime := S minus {elt};
  rec := partitionsOf(Sprime);
  res := NULL;
  for i from 1 to nops(rec) do res := res, ({{elt}} union rec[i]),
   allAugs(rec[i], elt); end;
  return {res};
```

```
end:

# lexicographic & length-graded lexicographic sorting
grlex := (X,Y)->(`if`(nops(X)<nops(Y), true, `if`(nops(Y)<nops(X),
false, lex(X, Y)))):
lexi := proc(X, Y, C);
if (C in X and not (C in Y)) then return true;
elif (C in Y and not (C in X)) then return false;
else return lexi(X, Y, C+1); end;
end: lex := (X, Y) -> lexi(X, Y, 1):

# convert partition to a list
P2L := proc(p) local n, i, j, r, q;
q := [op(p)];
n := `+`(op(map(nops, q)));
r := NULL;
for i from 1 to n do
  for j from 1 to nops(q) do
    if evalb(i in q[j]) then r := r, j;
  end;
end;
end;
return [r];
end:

# convert list to a partition
L2P := proc(l) local p, r, u, n, i, j;
n := nops(l);
u := {};
r := NULL;
for i from 1 to n do
  if not evalb(l[i] in u) then
    p := {i};
    for j from 1 to n do if l[j]=l[i] then
     p := p union {j};
     u := u union {j} end; end;
    r := r, p;
  end;
end;
return {r};
```

```
end:

# join of two partitions
wedge := (p, q) -> L2P(ListTools[Transpose]([P2L(p), P2L(q)])):

# meet of two partitions
vee := proc(p1, p2) local m, n, s, i, j, k, r, p;
n := `+`(op(map(nops, [op(p1)])));
m := Matrix(n, n);
for r in {p1, p2} do
for s in r do
  p := [op(s)];
  for i from 1 to nops(p)-1 do
    m[p[i], p[i+1]] := 1;
    m[p[i+1], p[i]] := 1;
  end;
end;
end;
for k from 1 to n do for i from 1 to n do for j from 1 to n do
  if m[i, k]=1 and m[k, j]=1 then m[i, j]:=1; end;
end; end; end;
r := NULL;
for i from 1 to n do
  p := i;
  for j from 1 to n do if m[i, j]=1 then p := p, j; end; end;
  r := r, {p};
end;
return {r};
end:

with(LinearAlgebra):

# rank of a set, denoted rho in the thesis
rank := x -> max(0, nops(x)-1):

# rank-contribution of pi and k
rc := proc(pi, k) local i, res;
  res := 0;
  for i from 1 to nops(pi) do
    if (nops(pi[i] intersect k) > 0) then res := res + 1; end;
```

```
  end;
  return res-1;
end:
sc := (S, T) -> rank(S intersect T):

# all maximal chains of partitions on items {1, 2, ..., i}
PC := proc(i) option memo; local res, j, k, l, t, u, T, U, sq, krange;
if (i=1) then return [[{{1}}]]; end;
res := NULL;
for j from 1 to i/2 do
 for T in PC(j) do
  for U in shift(PC(i-j), j) do
   krange := map(x->{op(x)}, combinat[choose](i-2, j-1));
   if (j*2 = i and i>2) then krange := select(elt->(1 in elt), krange); end;
   for k in krange do
    t := 1;
    u := 1;
    sq := {{seq(x, x=1..i)}}, (T[t] union U[u]);
    for l from 1 to i-2 do
     if l in {op(k)} then t:=t+1; else u:=u+1; end;
     sq := sq, T[t] union U[u];
    end;
    res := res, [sq];
   end;
  end;
 end;
end;
return [res];
end:
shift := (exp, sh) -> `if`(type(exp, numeric), exp+sh, map(shift, exp, sh)):

# used to eliminate multiple isomorphic items
containsSimilar := proc(set, item) local p;
for p in combinat[permute](nops(R)) do
  if subs({seq(r[i]=r[p[i]], i=1..nops(R))}, item) in set then return true; end;
end;
return false;
end:

# terminal set
```

```
R := {1, 2, 3, 4}:
# nonempty subsets of R
S := sort([op(combinat[powerset](R) minus ({{}, op(map(x->{x}, R))}))], grlex):
K := S:
# partitions of R
PI := [op(partitionsOf(R) minus {{R}})]:
# pre-compute rank-contribution and partition rank
RC := Matrix(nops(PI), nops(K), (i, j)->rc(PI[i], K[j])):
PR := Vector(nops(PI), i->(rank(PI[i]))):

# iterator for non-singular square submatrices
nextBasis := proc(L, supp) local m, n, res, pos, i, j;
m := op(1, L)[2];
n := op(1, L)[1];
if (supp = 0) then
 res := [seq(0, i=1..n)];
 pos := 1;
else
 res := supp;
 pos := n;
end;
while (true) do
 if (pos = 0) then return 0; end;
 res[pos] := res[pos]+1;
 if (res[pos] > m+pos-n) then pos := pos-1;
 else
  if Rank(SubMatrix(L, [1..n], res[1..pos])) >= pos then
   if (pos = n) then return res; end;
   res[pos+1] := res[pos];
   pos := pos+1;
  end;
 end;
end;
end:

# pretty-printing routines
pr := x->printf("%a\n", x):
foo := e -> `if`(type(e, numeric), e, `if`(type(e, set),
   {cat(op(e))}, map(foo, e))):
bar := pt -> foo([op(subs({seq(r[i]=i, i=1..nops(R))}, [op(pt)]))]):
```

```
### end of preliminary procedures, start of main session
foundpoints := {}:
usedchains := {}:
count := 0:
printf("%d chains to check\n", nops(PC(nops(R)))*(2^(nops(R)-1)-1)):
for ch0 in PC(nops(R)) do
 for chain in {op(combinat[powerset](ch0[2..nops(ch0)]))} minus {[]} do
  count := count+1;
  printf("checking chain number %d\n", count);
  if (not containsSimilar(usedchains, chain)) then
   usedchains := usedchains union {chain};
   n := nops(chain);
   Kco := Matrix(n, nops(K), (j, i)->rc(chain[j], K[i]));
   supp := nextBasis(Kco, 0);
   while supp <> 0 do
    x := LinearSolve(SubMatrix(Kco, [1..n], supp),
     Vector([seq(nops(chain[i])-1, i=1..n)]));
    valid := true;
    for i from 1 to n do
     valid := valid and x[i]>0 and x[i]<=1;
    end;
    pt := {seq(x[i]*map(e->r[e], K[supp[i]]), i=1..n)};
    valid := valid and not containsSimilar(foundpoints, pt);
    for i from 1 to nops(PI) do
     tmp := `+`(seq(x[j]*RC[i, supp[j]], j=1..n)) - PR[i];
     valid := valid and tmp >= 0;
    end;
    if valid then
     printf("foundpoints = %d\n", nops(foundpoints)+1);
     pr(bar(pt));
     pr(subsop(seq(supp[j]=x[j], j=1..n), [seq(0, j=1..nops(K))]));
     foundpoints := foundpoints union {pt};
    end;
    supp := nextBasis(Kco, supp);
   end;
  end;
 end;
end;
pr(map(bar, foundpoints));
```

Bibliography

[1] A. Agarwal and M. Charikar. On the advantage of network coding for improving network throughput. In *Proc. IEEE Information Theory Workshop*, pages 105–109, 2004. 12, 14, 63, 74

[2] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem in networks. *SIAM J. Comput.*, 24:440–456, 1995. Preliminary version appeared in *Proc. 23rd STOC*, pages 134-144, 1991. 3, 12, 79

[3] Ernst Althaus, Tobias Polzin, and Siavash Vahdati Daneshmand. Improving linear programming approaches for the Steiner tree problem. In *Proc. 2nd WEA*, pages 620–621, 2003. 3

[4] Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Electronic Colloquium on Computational Complexity*, 14(113), 2007. Preliminary versions appeared in *Proc. 46th FOCS*, pages 226–244, 2005 and *ECCC* 13(141), 2006. 104, 106

[5] Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980. 5, 12

[6] Richard P. Anstee. A polynomial algorithm for b-matchings: An alternative approach. *Inf. Process. Lett.*, 24(3):153–157, 1987. 123

[7] A. Asadpour, M.X. Goemans, A. Madry, S. Oveis Gharan, and A. Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proc. 21st SODA*, 2010. To appear. 157

[8] E. Balas. Disjunctive programming. In *Discrete Optimization II (Proc. Advanced Research Institute on Discrete Optimization and Systems Applications, Banff, Alta., 1977)*, volume 5 of *Annals of Discrete Mathematics*, pages 3–51. North-Holland, 1979. 164

[9] Nikhil Bansal, Zachary Friggstad, Rohit Khandekar, and Mohammad R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *Proc. 20th SODA*, pages 702–709, 2009. 4

[10] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree bounded directed network design. In *Proc. 40th STOC*, pages 769–778, 2008. 5, 107, 148

[11] Nikhil Bansal, Nitish Korula, and Viswanath Nagarajan. On k-column sparse packing programs. arXiv:0908.2256, 2009. 130, 131, 140, 168

[12] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2:198–203, 1981. 128

[13] Reuven Bar-Yehuda and Dror Rawitz. Efficient algorithms for integer programs with two variables per constraint. *Algorithmica*, 29(4):595–609, 2001. 128

[14] G. Baudis, S. Hougardy C. Gröpl, T. Nierhoff, and H.J. Prömel. Approximating minimum spanning sets in hypergraphs and polymatroids. Technical report, Humboldt-Universität zu Berlin, Institut für Informatik, 2000. 69

[15] Geneviève Benoit and Sylvia Boyd. Finding the exact integrality gap for small traveling salesman problems. *Math. Oper. Res.*, 33(4):921–931, 2008. 149, 156, 157

[16] P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. *J. Algorithms*, 17(3):381–408, 1994. Preliminary version appeared in *Proc. 3rd SODA*, pages 325-334, 1992. 2, 3, 5, 12, 94

[17] Piotr Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. *Nordic J. of Computing*, 7(3):178–184, 2000. Preliminary version appeared in *Proc. 7th SWAT*, pages 214–219, 2000. 129

[18] Daniel Bienstock. Approximate formulations for 0-1 knapsack sets. *Oper. Res. Lett.*, 36(3):317–320, 2008. 7, 161, 162, 163

[19] Daniel Bienstock and Benjamin McClosky. Tightening simple mixed-integer sets with guaranteed bounds. Manuscript, July 2008. 162

[20] Hans-Joachim Böckenhauer, Dirk Bongartz, Juraj Hromkovič, Ralf Klasing, Guido Proietti, Sebastian Seibert, and Walter Unger. On the hardness of constructing minimal 2-connected spanning subgraphs in complete graphs with sharpened triangle inequality. *Theor. Comput. Sci.*, 326(1-3):137–153, 2004. 149

[21] A. Borchers and D. Du. The k-Steiner ratio in graphs. *SIAM J. Comput.*, 26(3):857–869, 1997. Preliminary version appeared in *Proc. 27th STOC*, pages 641-649, 1995. 11

[22] S. Boyd and A. Cameron. A 3/2-approximation algorithm for the multi-two-edge connected subgraph problem. Technical Report TR-2008-01, SITE, University of Ottawa, 2008. 149

[23] S. Boyd and P. Elliott-Magwood. The structure of the extreme points of the subtour elimination polytope of the STSP. Technical Report TR-2007-09, SITE, University of Ottawa, 2007. 149, 156, 157

[24] S. C. Boyd and William R. Pulleyblank. Optimizing over the subtour polytope of the travelling salesman problem. *Math. Program.*, 49:163–187, 1991. 8, 150, 151, 157

[25] Sylvia Boyd. Vertices of the subtour elimination polytope. http://www.site.uottawa.ca/~sylvia/subtourvertices/index.htm. Accessed September 25, 2009. 157

[26] Sylvia Boyd. *The Subtour Polytope of the Travelling Salesman Problem*. PhD thesis, University of Waterloo, 1986. 8, 150

[27] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An LP-based $(3/2 + \epsilon)$-approximation for Steiner tree. Manuscript, 2009. 167

[28] Mao-Cheng Cai. The number of vertices of degree k in a minimally k-edge-connected graph. *J. Comb. Theory Ser. B*, 58(2):225–239, 1993. 8

[29] Gruia Calinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. Improved approximation algorithms for resource allocation. In *Proc. 9th IPCO*, pages 401–414, 2002. 104

[30] Alberto Caprara and Matteo Fischetti. $\{0, \frac{1}{2}\}$-Chvátal–Gomory cuts. *Math. Program.*, 74(3):221–235, 1996. 126

[31] Tim Carnes and David B. Shmoys. Primal-dual schema for capacitated covering problems. In *Proc. 13th IPCO*, pages 288–302, 2008. 162, 168

[32] Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proc. 11th SODA*, pages 106–115, 2000. 6, 101, 102, 128, 129, 134, 135

[33] Robert D. Carr, Toshihiro Fujito, Goran Konjevod, and Ojas Parekh. A $2\frac{1}{10}$-approximation algorithm for a generalization of the weighted edge-dominating set problem. *J. Comb. Optim.*, 5(3):317–326, 2001. Preliminary version appeared in *Proc. 8th ESA*, pages 132–142, 2000. 4

[34] Deeparnab Chakrabarty, Nikhil R. Devanur, and Vijay V. Vazirani. New geometry-inspired relaxations and algorithms for the metric Steiner tree problem. In *Proc. 13th IPCO*, pages 344–358, 2008. 3, 12, 14, 41, 72, 80, 84

[35] Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. In *Proc. 49th FOCS*, pages 687–696, 2008. 130, 143

[36] Deeparnab Chakrabarty, Jochen Könemann, and David Pritchard. Hypergraphic LP relaxations for Steiner trees. arXiv:0910.0281, October 2009. vii

[37] Yuk Hei Chan and Lap Chi Lau. On linear and semidefinite programming relaxations for hypergraph matching. In Proc. 21st SODA, 2010. To appear. 129

[38] A. Chandra, D. Hirschberg, and C. Wong. Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science*, 3(3):293–304, 1976. 162

[39] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3(3):27, 2007. Preliminary version appeared in *Proc. 30th ICALP*, pages 410–425, 2003. 103, 104, 105, 106, 114, 115, 120, 126, 129, 140, 168

[40] Chandra Chekuri and Jan Vondrák. Randomized pipage rounding for matroid polytopes. arXiv:0909.4348, September 2009. 163

[41] Xiuzhen Cheng and Ding-Zhu Du, editors. *Steiner Trees in Industry*, volume 11 of *Combinatorial Optimization*. Springer, 2002. 2

[42] Joseph Cheriyan, Tibor Jordán, and R. Ravi. On 2-coverings and 2-packings of laminar families. In *Proc. 7th ESA*, pages 510–520, 1999. 105, 106, 110, 113

[43] Joseph Cheriyan and Ramakrishna Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM J. Comput.*, 30(2):528–560, 2000. Preliminary version appeared in *Proc. 37th FOCS*, pages 292–301, 1996. 106, 150

[44] Kevin Cheung. *Subtour Elimination Polytopes and Graphs of Inscribable Type*. PhD thesis, University of Waterloo, 2003. 151, 157

[45] Miroslav Chlebík and Janka Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.*, 406(3):207–214, 2008. Preliminary version appeared in *Proc. 8th SWAT*, pages 170–179, 2002. 2, 9, 167

[46] S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8:25–29, 1989. 12, 13, 17, 22, 39, 63

[47] S. Chopra and M. R. Rao. The Steiner tree problem 1: Formulations, compositions, and extension of facets. *Mathematical Programming*, 64:209–229, 1994. 5, 12, 14, 15

[48] S. Chopra and M. R. Rao. The Steiner tree problem 2: Properties and classes of facets. *Mathematical Programming*, 64:231–246, 1994. 5, 12

[49] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976. 149

[50] G. Cornuéjols, D. Naddef, and J. Fonlupt. The traveling salesman problem on a graph and some related integer polyhedra. *Math. Programming*, 33:1–27, 1985. 8, 28, 156

[51] W. Cunningham and F. Zhang. On the dominant of the cut polyhedron. Manuscript, July 1992. 150

[52] Artur Czumaj and Andrzej Lingas. On approximability of the minimum-cost -connected spanning subgraph problem. In *Proc. 10th SODA*, pages 281–290, 1999. 149, 150

[53] G. B. Dantzig, D. R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *J. Oper. Res. Soc. America*, 2:393–410, 1954. 148

[54] Mohamed Didi Biha, Hervé Kerivin, and Ali Ridha Mahjoub. Steiner trees and polyhedra. *Discrete Applied Mathematics*, 112(1-3):101–120, 2001. 5, 12

[55] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM J. Comput.*, 34(5):1129–1146, 2005. Preliminary version appeared in *Proc. 35th STOC*, pages 595–601, 2003. 128

[56] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972. 10

[57] Martin Dyer. Approximate counting by dynamic programming. In *Proc. 35th STOC*, pages 693–699, 2003. 163

[58] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71B:233–240, 1967. 5, 12, 14

[59] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanam, N. Sauer, and J. Schonheim, editors, *Combinatorial structures and their applications (Proc. 1969 Calgary Conference)*, pages 69–87, New York, 1970. Gordon and Breach. 7

[60] J. Edmonds and R. Giles. A min-max relation for submodular functions on graphs. In *Studies in Integer Programming (1975 Bonn, Germany)*, volume 1 of *Annals of Discrete Mathematics*, pages 185–204. North-Holland, 1977. 7, 8, 25, 28

[61] J. Edmonds and E. Johnson. Matching: A well-solved class of integer linear programs. In R. Guy, H. Hanani, N. Sauer, and J. Schonheim, editors, *Combinatorial structures and their applications (Proc. 1969 Calgary Conference on Combinatorial Structures and Their Applications)*, pages 89–92. Gordon and Breach, 1970. 106, 121

[62] Thomas Erlebach. Approximation algorithms for edge-disjoint paths and unsplittable flow. In Evripidis Bampis, Klaus Jansen, and Claire Kenyon, editors, *Efficient Approximation and Online Algorithms*, pages 97–134. Springer, 2006. 104

[63] Thomas Erlebach and Klaus Jansen. Conversion of coloring algorithms into maximum weight independent set algorithms. In J. D. P. Rolim et al., editor, *Proc. ICALP Satellite Workshops*, pages 135–146, 2000. 104, 106

[64] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976. Preliminary version appeared in *Proc. 16th FOCS*, pages 184–193, 1975. 104

[65] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45, 1998. Preliminary version appeared in *Proc. 28th STOC*, pages 314–318, 1996. 163

[66] Cristina G. Fernandes. A better approximation ratio for the minimum size k-edge-connected spanning subgraph problem. *J. Algorithms*, 28(1):105–124, 1998. Preliminary version appeared in *Proc. 8th SODA*, pages 629–638, 1997. 149, 152

[67] Lisa Fleischer. A 2-approximation for minimum cost $\{0, 1, 2\}$ vertex connectivity. In *Proc. 8th IPCO*, pages 115–129, 2001. 149

[68] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962. 7

[69] A. Frank. Edge-connection of graphs, digraphs, and hypergraphs. In E. Győri, G. Katona, and L. Lovász, editors, *More sets, graphs and numbers*, volume 15 of *Bolyai Mathematical Society Math. Studies*, pages 93–142. Springer Verlag, 2006. 8

[70] András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Discret. Math.*, 5(1):25–53, 1992. Preliminary version appeared in *Proc. 31st FOCS*, pages 708–718, 1990. 8

[71] András Frank. Submodular functions in graph theory. *Discrete Math.*, 111(1-3):231–243, 1993. 8

[72] András Frank and Tamás Király. A survey on covering supermodular functions. In William J. Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization (Bonn 2008)*, chapter 6, pages 87–126. Springer, 2009. 19, 46

[73] András Frank, Tamás Király, and Zoltán Király. On the orientation of graphs and hypergraphs. *Discrete Appl. Math.*, 131(2):385–400, 2003. 19, 29, 46

[74] András Frank, Tamás Király, and Matthias Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Appl. Math.*, 131(2):373–383, 2003. 29

[75] Matthias Franz. Convex - a maple package for convex geometry. http://www.math.uwo.ca/~mfranz/convex/. Accessed September 27, 2009. vii, 35, 62, 156

[76] G. N. Frederickson and J. JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981. 149, 152

[77] G. N. Frederickson and J. JáJá. On the relationship between biconnectivity augmentation and the traveling salesman problem. *Theoretical Computer Science*, 19:189–201, 1982. 149

[78] A. M. Frieze and M. R. B. Clarke. Approximation algorithms for the m-dimensional 0-1 knapsack problem: Worst-case and probabilistic analyses. *European Journal of Operational Research*, 15(1):100–109, January 1984. 162

[79] Toshihiro Fujito and Hiroshi Nagamochi. A 2-approximation algorithm for the minimum weight edge dominating set problem. *Discrete Appl. Math.*, 118(3):199–207, 2002. 4

[80] Toshihiro Fujito and Takatoshi Yabuta. Submodular integer cover and its application to production planning. In *Proc. 2nd WAOA*, pages 154–166, 2004. 6, 128, 129

[81] D. R. Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Math. Programming*, 1:168–194, 1971. 12, 47

[82] Harold N. Gabow. On the L_∞-norm of extreme points for crossing supermodular directed network LPs. *Mathematical Programming*, 110:111–144, 2007. Preliminary version appeared in *Proc. 11th IPCO*, pages 392–406, 2005. 151

[83] Harold N. Gabow and Suzanne Gallagher. Iterated rounding algorithms for the smallest k-edge-connected spanning subgraph. In *Proc. 19th SODA*, pages 550–559, 2008. 106, 107, 150

[84] Harold N. Gabow, Michel X. Goemans, Éva Tardos, and David P. Williamson. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. *Networks*, 53(4):345–357, 2009. Preliminary version appeared in *Proc. 16th SODA*, pages 562–571, 2005. 106, 148, 150, 151

[85] Harold N. Gabow, Michel X. Goemans, and David P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Math. Program.*, 82:13–40, 1998. Preliminary version appeared in *Proc. 3rd IPCO*, pages 57–74, 1993. 8

[86] Garey, M. R. and Johnson, D. S. Strong NP-completeness results : Motivation, examples and implications. *J. ACM*, 25:499–508, 1978. 162

[87] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, May 1997. Preliminary version appeared in *Proc. 20th ICALP*, pages 64–75, 1993. 103, 104, 106, 114, 115, 125, 126, 140, 152

[88] G. V. Gens and E. V. Levner. Complexity and approximation algorithms for combinatorial problems: A survey. Technical report, Central Economic and Mathematical Institute, Academy of Sciences of the USSR, Moscow, 1979. 162

[89] Dion Gijswijt and Gyula Pap. An algorithm for weighted fractional matroid matching. arXiv:0806.1818, 2008. 17

[90] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16(1):1–29, 1968. 2, 3, 9

[91] Frederick Richard Giles. *Submodular Functions, Graphs and Integer Polyhedra*. PhD thesis, University of Waterloo, 1975. 7

[92] Gagan Goel, Chinmay Karande, Pushkar Tripathi, and Lei Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *Proc. 50th FOCS*, 2009. To appear. 131

[93] M. X. Goemans and D. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Math. Programming*, 60:145–166, 1993. Preliminary version appeared in *Proc. 1st SODA*, pages 388-396, 1990. 3, 12, 44, 79, 148, 149, 150

[94] M. X. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proc. 5th SODA*, pages 223–232, 1994. 8

[95] M. X. Goemans and Y. Myung. A catalog of Steiner tree formulations. *Networks*, 23:19–28, 1993. 5, 12, 13, 14, 21, 44, 47

[96] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 4. PWS, Boston, 1997. 8, 38, 84

[97] Michel X. Goemans. *Analysis of Linear Programming Relaxations for a Class of Connectivity Problems*. PhD thesis, MIT, 1990. 150

[98] Michel X. Goemans. The Steiner tree polytope and related polyhedra. *Math. Program.*, 63(2):157–182, 1994. 14, 18, 19

[99] Michel X. Goemans. Minimum bounded degree spanning trees. In *Proc. 47th FOCS*, pages 273–282, 2006. 5, 106, 148

[100] Michel X. Goemans and David P. Williamson. Approximating minimum-cost graph problems with spanning tree edges. *Operations Research Letters*, 16(4):183 – 189, 1994. 8

[101] 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 industries*, pages 235–279. Kluwer Academic Publishers, Norvell, Massachusetts, 2001. 12, 71, 84, 92, 94

[102] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel. Lower bounds for approximation algorithms for the Steiner tree problem. In *Proc. 26th Workshop on Graph-Theoretic Concepts in Computer Science*, 2001. 71

[103] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel. Steiner trees in uniformly quasi-bipartite graphs. *Inform. Process. Lett.*, 83(4):195–200, 2002. Preliminary version appeared as a Technical Report at TU Berlin, 2001. 22, 23, 68, 74

[104] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988. 7, 27

[105] M. Grötschel, C. Monma, and M. Stoer. Design of survivable networks. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*. North-Holland, 1995. 15

[106] Martin Grötschel and Clyde L. Monma. Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM J. Discrete Math.*, 3(4):502–523, 1990. 15

[107] Venkatesan Guruswami, Sanjeev Khanna, Rajmohan Rajaraman, F. Bruce Shepherd, and Mihalis Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *J. Comput. Syst. Sci.*, 67(3):473–496, 2003. Preliminary version appeared in *Proc. 31st STOC*, pages 19–28, 1999. 104, 106

[108] N. G. Hall and D. S. Hochbaum. A fast approximation algorithm for the multicovering problem. *Discrete Appl. Math.*, 15(1):35–40, 1986. 128

[109] Irith Ben-Arroyo Hartman. Optimal k-colouring and k-nesting of intervals. In *Proc. 4th Israel Symposium on Theory of Computing and Systems*, pages 207–220, 1992. 104

[110] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. 143

[111] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Comput. Complex.*, 15(1):20–39, 2006. Preliminary versions appeared in *Proc. 6th APPROX*, pages 83–97, 2003 and ECCC-TR03-020, 2003. 129

[112] M. Held and R. M. Karp. The traveling-salesman and minimum cost spanning trees. *Operations Research*, 18:1138–1162, 1970. 18, 148

[113] Dorit S. Hochbaum. Approximation algorithms for set covering and vertex cover problems. *SIAM J. Comput.*, 11:555–556, 1982. 128

[114] Dorit S. Hochbaum. Monotonizing linear programs with up to two nonzeroes per column. *Oper. Res. Lett.*, 32(1):49–58, 2004. 130, 142

[115] Dorit S. Hochbaum, Nimrod Megiddo, Joseph (Seffi) Naor, and Arie Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Program.*, 62(1):69–83, 1993. 128, 142

[116] A.J. Hoffman and D.E. Schwartz. On lattice polyhedra. In A. Hajnal and V.T. Sós, editors, *Combinatorics*, volume 18 of *Colloquia Mathematica Societatis János Bolyai*, pages 593–598. North-Holland, Amsterdam, 1976. 7, 8, 29

[117] Markus Hohenwarter. GeoGebra. http://www.geogebra.org. Accessed September 27, 2009. vii, 156

[118] S. Hougardy and H. J. Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In *Proc. 10th SODA*, pages 448–453, 1999. 2, 3, 5, 12

[119] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Discret. Math.*, 2(1):68–72, 1989. 129

[120] C.A.J. Hurkens, L. Lovász, A. Schrijver, and É. Tardos. *How to tidy up your set-system?*, volume 52 of *Colloquia Mathematica Societies Janos Bolyai*, pages 309–314. North-Holland, Amsterdam, 1987. 38

[121] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. Number 53 in Annals of Discrete Mathematics. North-Holland, 1992. 9

[122] Clay Mathematics Institute. P vs NP problem. http://www.claymath.org/millennium/P_vs_NP/. Accessed October 10, 2009. 2

[123] Satoru Iwata and Kiyohito Nagano. Submodular function minimization under covering constraints. In *Proc. 50th FOCS*, 2009. To appear. 131

[124] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. Preliminary version appeared in *Proc. 39th FOCS*, pages 448–457, 1998. 3, 4, 8, 12, 25, 28, 79, 105, 106, 107, 111, 119, 148, 149, 150

[125] Viggo Kann. *On the Approximability of NP-complete Optimization Problems*. PhD thesis, Royal Institute of Technology Stockholm, 1992. 114

[126] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, NY, 1972. 2, 9

[127] M. Karpinski and A. Zelikovsky. New approximation algorithms for the Steiner tree problems. *J. Combinatorial Optimization*, 1(1):47–65, 1997. 2, 3, 5, 12, 18, 37, 85, 89

[128] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004. 162, 163

[129] Subhash Khot and Ashok Kumar Ponnuswami. Better inapproximability results for MaxClique, Chromatic Number and Min-3Lin-Deletion. In *Proc. 33rd ICALP*, pages 226–237, 2006. 130

[130] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. Preliminary version appeared in *Proc. 18th CCC*, pages 379–386, 2003. 128

[131] Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994. Preliminary version appeared in *Proc. 24th STOC*, pages 759–770, 1992. 150

[132] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998. 14

[133] S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Proc. 38th FOCS*, pages 426–436, 1997. 140

[134] Stavros G. Kolliopoulos and Neal E. Young. Approximation algorithms for covering/packing integer programs. *J. Comput. Syst. Sci.*, 71(4):495–505, 2005. 102, 130, 131, 134, 135

[135] J. Könemann, D. Pritchard, and Y. Wei. Filtering for the Steiner tree problem. Manuscript, 2008. 80

[136] J. Könemann and K. Tan. A fresh look at Steiner trees: Greedy vs primal-dual algorithms. Technical Report CORR 2006-08, Department of Combinatorics & Optimization, University of Waterloo, 2006. vii, 17

[137] Jochen Könemann, Ojas Parekh, and David Pritchard. Max-weight integral multicommodity flow in spiders and high-capacity trees. In *Proc. 6th WAOA*, pages 1–14, 2008. vii, 140

[138] Jochen Könemann and David Pritchard. Uncrossing partitions. Technical Report CORR 2007-11, Department of Combinatorics & Optimization, University of Waterloo, 2007. vii, 17, 34, 63

[139] Jochen Könemann, David Pritchard, and Kunlun Tan. A partition-based relaxation for Steiner trees. *Mathematical Programming*, 2009. In press. vii, 17

[140] B. Korte and R. Schrader. On the existence of fast approximation schemes. In *Proc. 4th Symp., Madison, Wisc.*, volume 4 of *Nonlinear Programming*, pages 415–437. Academic Press, 1980. 162

[141] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer, New York, 2008. 92, 94

[142] Guy Kortsarz, Robert Krauthgamer, and James R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720, 2004. Preliminary version appeared in *Proc. 5th APPROX*, pages 185-199, 2002. 150

[143] Christos Koufogiannakis and Neal E. Young. Flooding overcomes small covering constraints. arXiv:0807.0644, 2008. 129, 131

[144] Christos Koufogiannakis and Neal E. Young. Distributed and parallel algorithms for weighted vertex cover and other covering problems. In *Proc. 28th PODC*, pages 171–179, 2009. 129, 131

[145] Christos Koufogiannakis and Neal E. Young. Distributed fractional packing and maximum weighted b-matching via tail-recursive duality. In *Proc. 23rd DISC*, 2009. To appear. 129, 131, 136

[146] Christos Koufogiannakis and Neal E. Young. Greedy δ-approximation algorithm for covering with arbitrary constraints and submodular cost. In *Proc. 36th ICALP*, pages 634–652, 2009. 129, 131

[147] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings, American Mathematical Society*, 7:48–50, 1956. 13

[148] Lap Chi Lau, Joseph (Seffi) Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. In *Proc. 39th STOC*, pages 651–660, 2007. 5, 101, 105, 106, 107, 148

[149] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proc. 41st STOC*, pages 323–332, 2009. 163

[150] H.W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983. 127, 130, 162

[151] L. Lovász. Solution to problem 11 in: Report on the Memorial Mathematical Contest Miklós Schweitzer of the year 1968 (in Hungarian). *Matematikai Lapok*, 20:145–171, 1969. 8

[152] L. Lovász. A generalization of Kónig's theorem. *Acta Mathematica Hungarica*, 21:443–446, 1970. 7

[153] L. Lovász. A remark on Menger's theorem. *Acta Mathematica Hungarica*, 21:365–368, 1970. 7

[154] L. Lovász. Connectivity in digraphs. *J. Combin. Theory Ser. B*, 15:174–177, 1973. 7

[155] L. Lovász. 2-matchings and 2-covers of hypergraphs. *Acta Math. Ac. Sc. Hung.*, 26:433–444, 1975. 7

[156] L. Lovász. On two minimax theorems in graph theory. *J. Combin. Theory Ser. B*, 21:96–103, 1976. 7, 8

[157] C. Lucchesi and D. Younger. A minimax theorem for directed graphs. *J. London Math. Soc.*, s2-17(3):369–374, 1978. 7, 8, 25

[158] Cláudio Leonardo Lucchesi. *A Minimax Equality for Directed Graphs*. PhD thesis, University of Waterloo, 1976. 7

[159] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report 178, Computer Science Laboratory, Princeton University, 1975. 142, 162

[160] Michael J. Magazine and Maw-Sheng Chern. A note on approximation schemes for multidimensional knapsack problems. *Math. of Oper. Research*, 9(2):244–247, 1984. 162

[161] Brendan McKay. nauty. http://cs.anu.edu.au/~bdm/nauty/. Accessed September 27, 2009. 156

[162] C.St.J.A. Nash-Williams. On orientations, connectivity and odd-vertex-pairings in finite graphs. *Canadian Journal of Mathematics*, 12:555–567, 1960. 7

[163] Thành Nguyen. On the disjoint paths problem. *Oper. Res. Lett.*, 35(1):10–16, 2007. 121

[164] Zeev Nutov. An almost $(\log k)$-approximation for k-connected subgraphs. In *Proc. 20th SODA*, pages 912–921, 2009. 150

[165] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18:1–11, 1993. 149

[166] Ojas Parekh. *Polyhedral Techniques for Graphic Covering Problems*. PhD thesis, Carnegie Mellon University, 2002. 4

[167] Tobias Polzin. *Algorithms for the Steiner Problem in Networks*. PhD thesis, Universität des Saarlandes, February 2003. 9, 12

[168] Tobias Polzin and Siavash Vahdati Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112(1-3):241–261, 2001. 5, 12, 13, 14, 44

[169] Tobias Polzin and Siavash Vahdati Daneshmand. Improved algorithms for the Steiner problem in networks. *Discrete Applied Mathematics*, 112(1-3):263–300, 2001. 12

[170] Tobias Polzin and Siavash Vahdati Daneshmand. On Steiner trees and minimum spanning trees in hypergraphs. *Oper. Res. Lett.*, 31(1):12–20, 2003. 5, 13, 16, 18, 19, 20, 21, 44, 50

[171] David Pritchard. Maple and GeoGebra. http://daveagp.wordpress.com/maple-geogebra/. Accessed September 27, 2009. 156

[172] David Pritchard. Approximability of sparse integer programs. In *Proc. 17th ESA*, pages 83–94, 2009. Preliminary version appears at arXiv:0904.0859. vii, 130, 140

[173] H. J. Prömel and A. Steger. A new approximation algorithm for the Steiner tree problem with performance ratio 5/3. *J. Algorithms*, 36(1):89–101, 2000. Preliminary version appeared in *Proc. 14th STACS*, pages 559–570, 1997. 2, 3, 5, 9, 11, 12

[174] H. J. Prömel and A. Steger. *The Steiner Tree Problem — A Tour through Graphs, Algorithms, and Complexity*. Vieweg Verlag, Braunschweig-Wiesbaden, 2002. 2, 12

[175] P. Raghavan and C. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987. 104

[176] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem. In *Proc. 10th SODA*, pages 742–751, 1999. 3, 12, 14, 90

[177] R. Rizzi. On Rajagopalan and Vazirani's 3/2-approximation bound for the Iterated 1-Steiner heuristic. *Information Processing Letters*, 86(6):335–338, 2003. 3, 12, 94

[178] G. Robins and A. Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134, 2005. Preliminary version appeared in *Proc. 11th SODA*, pages 770–779, 2000. v, 2, 3, 5, 12, 73, 79, 80

[179] A. Schrijver. *Combinatorial optimization*. Springer, New York, 2003. 7, 8, 27, 29, 106, 121, 124

[180] F. B. Shepherd and A. Vetta. The demand-matching problem. *Mathematics of Operations Research*, 32(3):563–578, 2007. Preliminary version appeared in *Proc. 9th IPCO*, pages 457-474, 2002. 126, 129, 136, 139

[181] Mohit Singh. *Iterative Methods in Combinatorial Optimization*. PhD thesis, Carnegie Mellon University, 2008. 6, 130

[182] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proc. 39th STOC*, pages 661–670, 2007. 5, 25, 28, 105, 106, 107, 148

[183] M. Skutella. Personal communication, 2006. 75

[184] Aravind Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM J. Comput.*, 29(2):648–670, 1999. Preliminary version appeared in *Proc. 27th STOC*, pages 268–276, 1995. 130, 140

[185] Aravind Srinivasan. An extension of the Lovász Local Lemma, and its applications to integer programming. *SIAM J. Comput.*, 36(3):609–634, 2006. Preliminary version appeared in *Proc. 7th SODA*, pages 6–15, 1996. 130, 140

[186] Richard P. Stanley. *Enumerative Combinatorics*, volume 1. Wadsworth & Brooks/Cole, 1986. 22, 27, 29

[187] Kunlun Tan. On the role of partition inequalities in classical algorithms for Steiner problems in graphs. Master's thesis, University of Waterloo, 2006. vii

[188] Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proc. 33rd STOC*, pages 453–461, 2001. 69, 130

[189] John H. Vande Vate. Fractional matroid matchings. *J. Comb. Theory, Ser. B*, 55(1):133–145, 1992. 17, 29

[190] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001. 84

[191] V.V. Vazirani. Recent results on approximating the Steiner tree problem and its generalizations. *Theoret. Comput. Sci.*, 235(1):205–216, 2000. 12, 14

[192] Jan Vondrák. *Submodularity in Combinatorial Optimization*. PhD thesis, Charles University, 2007. 100

[193] D. Warme, P. Winter, and M. Zachariasen. Exact Algorithms for Plane Steiner Tree Problems: A Computational Study. In D.-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, pages 81–116. Kluwer Academic Publishers, 2000. Abstract appeared in *Proc. 10th SODA*, pages, 979-980, 1999. 3

[194] David Warme. *Spanning Trees in Hypergraphs with Applications to Steiner Trees*. PhD thesis, University of Virginia, 1998. 5, 12, 15, 17

[195] D.M. Warme. A new exact algorithm for rectilinear Steiner trees. In P.M. Pardalos and D.-Z. Du, editors, *Network Design: Connectivity and Facilities Location*, pages 357–395. American Mathematical Society, 1997. 9

[196] K. White, M. Farber, and W. Pulleyblank. Steiner tress, connected domination and strongly chordal graphs. *Networks*, 15:109–124, 1985. 15

[197] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. An approximation algorithm for general graph connectivity problems. *Combinatorica*, 15:435–454, 1995. Preliminary version appeared in *Proc. 25th STOC*, pages 708–717, 1993. 8

[198] L. Wolsey. Heuristic analysis, linear programming and branch and bound. *Math. Programming Study*, 13:121–134, 1980. 149

[199] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982. 69, 79, 96, 97, 131, 134

[200] R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Math. Programming*, 28:271–287, 1984. 5, 12, 14

[201] D. H. Younger. Maximum families of disjoint directed cut sets. In W. T. Tutte, editor, *Recent Progress in Combinatorics (Proc. 3rd Waterloo Conference on Combinatorics)*, pages 329–333. Academic Press, 1969. 7

[202] A. Z. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993. 2, 3, 5, 9, 12

[203] Alexander Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. Technical Report CS-96-06, University of Virginia, Charlottesville, VA, USA, 1996. 2, 3, 5, 12, 96