Uncrossing Partitions

Jochen Könemann and David Pritchard

June 4, 2007

Abstract

We extend a well known *uncrossing* technique in linear programs (LPs) to work with partitions. Using this technique, we tie together three previously unrelated papers on Steiner trees, by showing that the following three values are equal: (1) the objective value of a subtour based LP by Polzin and Vahdati Daneshmand; (2) the objective value of a partition based LP by Könemann and Tan; (3) a "maximum gainless tree" quantity used by Karpinski and Zelikovsky. These LPs are known to be stronger than the *bidirected cut relaxation*; we conjecture that in *preprocessed* graphs, these LPs are exactly as strong as the bidirected cut relaxation, which would add a surprising fourth item to our list.

1 Introduction

The Steiner tree problem is a classical combinatorial optimization problem. The input is a graph $G = (V,E)$ with positive costs c_e for all edges, and with the vertices V divided into two sets: the terminals R and the Steiner nodes N. A feasible solution is any connected subgraph of G that connects all of R, and we want one whose sum of edge costs is minimal. It is easy to see that any optimal solution is a tree all of whose leaves are terminals; these are called *Steiner trees*. Computing a cheapest Steiner tree is NP-hard, and moreover no (polynomial time) $\frac{96}{95}$-approximation algorithm exists unless P=NP [4]. Hence, there is much interest in designing algorithms with good approximation factors.

Central to this work is the *full component decomposition* of Steiner trees, which was originally used by Zelikovsky [33] in the 1990s to get a $\frac{11}{6}$-approximation algorithm. 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 1 for an example. Zelikovsky's original algorithm (and basically all following improvements) go the other way: we compute the cheapest full components of a graph, and use that information to reconstruct a cheap Steiner tree.

More specifically, for each subset $K \subset R$ of the terminals, we would like to compute the cheapest (if any) full component with leaf set K. Then knowledge of the cheapest costs for each terminal set is sufficient to reconstruct the cheapest Steiner tree. In this way the problem is transformed into a problem about minimum-cost spanning sub-hypergraphs, which we now review. A hypergraph $(V,\mathscr{E})$ has vertices V, hyperedges $\mathscr{E} \subseteq \{E \subseteq V \mid |E| \geq 2\}$ and in this case costs C_E for each $E \in \mathscr{E}$; a spanning sub-hypergraph is $(V,\mathscr{E}')$ with $\mathscr{E}' \subset \mathscr{E}$ such that, for every partition of V into two nonempty parts, some hyperedge in $\mathscr{E}'$ intersects both parts. The correspondence between Steiner instances and hypergraphs is as follows: for each $K \subset R$, we compute $opt(K)$, the cheapest full component with leaf set K, and assign cost[1] $C_K := c(opt(K))$. Then spanning sub-hypergraphs correspond to Steiner trees of the original problem.

[1] If x is a vector and S is a set, we use the convention that $x(S) = \sum_{s \in S} x_s$; similarly if H is a graph then $c(H)$ is the total cost of the graph's edges.

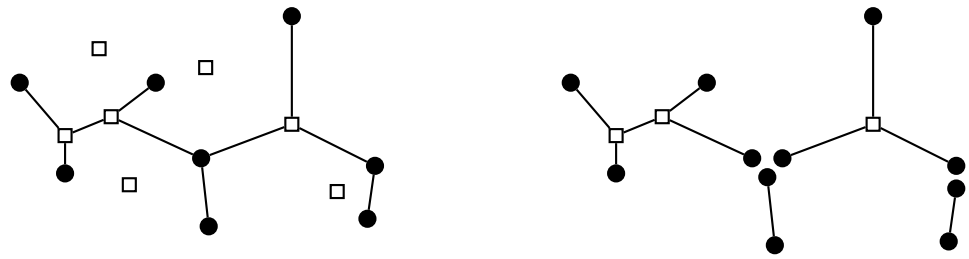


Figure 1: Black nodes are terminals and white nodes are Steiner nodes. Left: a Steiner tree for this instance. Right: the Steiner tree's edges are partitioned into full components; there are four full components.

There is a subtle point to this abstraction. The original Steiner tree instance may have about $2^{|R|}$ full components in total (one for each nonempty non-singleton subset of R) and so the usual approach is to compute only those full components of size at most k (where k is a fixed parameter). We call this *preprocessing*. Preprocessing can be accomplished in polynomial time (e.g., using dynamic programming [8]). If k is too small, the preprocessing to a hypergraph will increase the cost of a minimum-cost Steiner tree, as we now illustrate. Consider Figure 2. There is only one feasible Steiner tree and it has a full component on 5 vertices. But, say, for $k = 3$ things are still not too bad; there are full components on the terminal sets $\{r_1, r_2, r_3\}$ and $\{r_3, r_4, r_5\}$ each of cost 3, which can (as hyperedges) be joined together to produce a spanning sub-hypergraph of cost 6. Viewed in the original Steiner tree instance, the edge $r_3 n$ is now used twice; the preprocessing increased the optimal solution cost from 5 to 6. However, it is known that preprocessing increases the cost by no more than a factor of $1 + \Theta(1/\log k)$ [2]. So by taking $k \to \infty$ we get an approximation scheme where preprocessing has negligible cost.

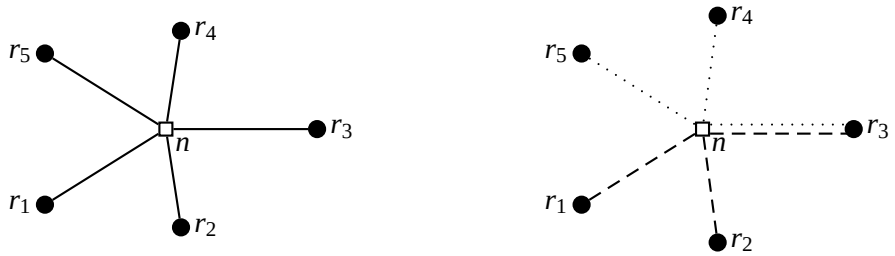


Figure 2: Left: an instance of the Steiner tree problem with five terminals. All edges have cost 1. Right: if we try to construct a feasible solution using full components on terminals of size less than 5, some edge must be used twice. Two full components on 3 terminals are shown, using dashed and dotted edges respectively.

In the rest of this section we describe some relevant related work. In Section 2 we show an equivalence between *partitions* and *subtours*. In Section 3 the partition uncrossing technique is described and we show that items (1) and (2) of the abstract are equivalent. In Section 4 we show equivalence with item (3) and give other applications of our technique. Finally, Section 5 contains ideas for future work.

1.1 Linear Relaxations

To apply linear programming to a combinatorial optimization problem, one typically assigns a continuous 0-1 variable to each possible element of a feasible solution, with 1 meaning that element should be used and

0 meaning it should not, along with constraints amongst these variables (and possibly other auxiliary variables) to enforce that all integral solutions correspond to feasible objects for the combinatorial optimization problem at hand. For NP-hard problems the linear programming approach can be used to design approximation algorithms (such as a 1.5-approximation algorithm for so-called *quasibipartite* Steiner instances [24, 25]) and also to implement exact integer programming algorithms (e.g., as applied to Steiner trees in the theses of Warme [31] and Polzin [20, 22]).

There is a great deal of literature on the study of Steiner tree linear programs themselves, and of the corresponding polyhedra (feasible regions of the LPs). Work by Edmonds [9] and later Chopra [5] on the spanning tree polytope led to a number of different linear programs for the Steiner tree problem. Chopra and Rao [6, 7] and Goemans [10] investigated these in quite a bit of detail. Goemans and Myung [12] showed that LPs from several independent lines of work were all equivalent to the *bidirected cut relaxation*. This LP was the one used to obtain the 1.5-approximation algorithm for quasibipartite graphs [24, 25] mentioned above. There are some hopes that the strength of this relaxation will lead to other algorithms, but none yet have appeared for general graphs with constant approximation factor better than 2.

In 1997, Warme [30, 31, 32] introduced a new linear program for the Steiner tree problem. In fact, this LP models the minimum-cost spanning sub-hypergraph problem, using the reduction outlined previously. Polzin et al. continued this line of work, showing [21, 23] that this program was *(sometimes strictly) stronger* than the bidirected cut relaxation. Independently, extending work of Chopra on spanning trees, Könemann and Tan [16, 28] showed that a similar LP could be used to interpret the currently best known [26] approximation algorithm for Steiner trees as a primal-dual algorithm.

2 Partitions versus Subtours

The relaxation used by Könemann and Tan has different constraints than the one considered by Warme and Polzin. In the next sections we will show that they are equivalent. We define $\mathscr{K} := \{E \subseteq V \mid |E| \geq 2\}$ to be the set of all possible hyperedges (i.e., terminal sets of full components). If there is no full component with terminal set $K \in \mathscr{K}$, or if we have only computed the full components on at most k terminals and $|K| > k$, then we set $C_K = \infty$. For a set X, we define

$$\rho(X) := \begin{cases} 0, & \text{if } X = \emptyset; \\ |X| - 1, & \text{otherwise.} \end{cases}$$

This notion of "size" or "rank" turns out to be natural in our setting.

The hypergraph formulations use a relaxed indicator variable x_K for each hyperedge $K \in \mathscr{K}$. First, we present Warme's formulation.[2]

$$\begin{aligned} \text{minimize} \quad & \sum_{K \in \mathscr{K}} C_K x_K && (\mathscr{S}') \\ & x \geq 0 && (1) \\ & \sum_{K \in \mathscr{K}} x_K \rho(K) = \rho(R) && (2) \\ \forall S \subseteq R: \quad & \sum_{K \in \mathscr{K}} x_K \rho(K \cap S) \leq \rho(S) && (3) \end{aligned}$$

[2]We adopt the following convention from [12]: a prime ($'$) symbol in an LP denotes that the feasible region is bounded.

The letter $\mathscr{S}$ is short for the *subtour elimination constraints* (3), which are a key feature of Warme's LP.

A *cycle* in a hypergraph is a sequence of distinct vertices and distinct hyperedges $v_0 \in e_0 \ni v_1 \in e_1 \ni v_2 \in \cdots \in e_t \ni v_0$. A spanning hypergraph is a *spanning (hyper)tree* if it has no cycles. If x is the characteristic vector[3] of (the edge set of) a hypergraph and x has a cycle (with notation as above), then it is easy to see that x violates the subtour constraint (3) for $S = \{v_0, v_1, \ldots, v_t\}$. Intuitively, cycles are bad since the connectivity they provide is redundant (and hence not optimal).

Warme showed that the feasible integral points of ($\mathscr{S}'$) are exactly the incidence vectors of spanning hypertrees. Informally, this correspondence holds because (3) prevents cycles and (2) ensures that all terminals are connected.

A *partition of* R is a collection of sets $\{\pi_i \mid i = 1, \ldots, t\}$ such that each π_i is nonempty, and such that each $r \in R$ occurs in exactly one π_i. The sets π_i are called the *parts* of π. For a partition π, its *rank* $r(\pi)$ is the number of parts of π, i.e., $r(\{\pi_1, \ldots, \pi_t\}) = t$. Given a partition π of R and any $K \in \mathscr{K}$, we define the *rank contribution* $\mathtt{rc}_K^\pi$ to be the number of parts of π spanned by K, minus one. Equivalently, $\mathtt{rc}_K^\pi$ is the rank drop incurred by π if we merge together all parts intersecting K. Given the incidence vector x of a spanning hypergraph and any partition π of R, Könemann and Tan showed that the following inequality is valid:

$$\sum_{K \in \mathscr{K}} x_K \mathtt{rc}_K^\pi \geq r(\pi) - 1. \tag{4}$$

Note that the special case of inequality (4) in which $r(\pi) = 2$ follows immediately from the definition of a spanning sub-hypergraph. Note also that when π is the unique rank-one partition $\{R\}$, both sides of (4) vanish. We will show that the subtour constraints (3) are equivalent to partition constraints, but first we need a lemma.

Lemma 2.1. *For a partition $\pi = \{\pi_1, \ldots, \pi_t\}$ of R, where $t = r(\pi)$, we have*

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

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

$$\sum_{i=1}^{t} \rho(K \cap \pi_i) = \sum_{i : K \cap \pi_i \neq \emptyset} (|K \cap \pi_i| - 1) = \left(\sum_{i : K \cap \pi_i \neq \emptyset} |K \cap \pi_i| \right) - (\mathtt{rc}_K^\pi + 1). \tag{5}$$

Observe that $\sum_{i : K \cap \pi_i \neq \emptyset} |K \cap \pi_i| = |K| = \rho(K) + 1$; using this fact together with Equation (5) we obtain

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

Rearranging, the proof of Lemma 2.1 is complete. □

Define Π_R to be the family of all partitions of R. We are now in a position to show that the polytope

[3] The characteristic vector of a set S is the vector $\chi(S)$ such that $(\chi(S))_s = 1$ for $s \in S$ and $(\chi(S))_s = 0$ for $s \notin S$.

defined by ($\mathscr{S}'$) is equal to the polytope defined by the following *partition based* linear program.

$$\text{minimize} \quad \sum_{K \in \mathscr{K}} C_K x_K \tag{$\mathscr{P}'$}$$

$$x \geq 0 \tag{1}$$

$$\sum_{K \in \mathscr{K}} x_K \rho(K) = \rho(R) \tag{2}$$

$$\forall \pi \in \Pi_R : \quad \sum_{K \in \mathscr{K}} x_K \mathtt{rc}_K^{\pi} \geq r(\pi) - 1 \tag{6}$$

In the rest of the paper, let us denote the feasible region of a linear program (X) by $\mathtt{fs}(X)$.

Theorem 2.2 (Subtour/partition correspondence)**.** *The polytopes* $\mathtt{fs}(\mathscr{P}')$ *and* $\mathtt{fs}(\mathscr{S}')$ *are equal.*

Proof. Notice that the constraints (1) and (2) are common to both formulations. We will show that $\mathtt{fs}(\mathscr{S}') \subseteq \mathtt{fs}(\mathscr{P}')$ and then that $\mathtt{fs}(\mathscr{P}') \subseteq \mathtt{fs}(\mathscr{S}')$.

Consider the constraint (6) for any fixed partition $\pi = \{\pi_1, \ldots, \pi_t\}$, so $t = r(\pi)$. We claim that this constraint can be written as (2) minus certain constraints (3); since π was chosen arbitrarily, this will prove that $\mathtt{fs}(\mathscr{S}') \subseteq \mathtt{fs}(\mathscr{P}')$. In particular, for each $1 \leq i \leq t$, subtract the constraint (3) with $S = \pi_i$. We obtain

$$\sum_{K \in \mathscr{K}} x_K \Big(\rho(K) - \sum_i \rho(K \cap \pi_i) \Big) \geq \rho(R) - \sum_i \rho(\pi_i). \tag{7}$$

From Lemma 2.1, $\rho(K) - \sum_i \rho(K \cap \pi_i) = \mathtt{rc}_K^{\pi}$. We also have $\rho(R) - \sum_i \rho(\pi_i) = |R| - 1 - (|R| - r(\pi)) = r(\pi) - 1$. Hence the inequality (7) that we derived as valid for $\mathtt{fs}(\mathscr{S}')$ is exactly (6).

Conversely, let us write (3) for a fixed set S as a linear combination of (2) minus an inequality of the form (6). Note when $S = \emptyset$ that (3) is vacuously true so we may assume $S \neq \emptyset$. Let $R \backslash S = \{r_1, \ldots, r_u\}$ and take $\pi = \{\{r_1\}, \ldots, \{r_u\}, S\}$. Subtract (6) for this π from (2), obtaining

$$\sum_{K \in \mathscr{K}} x_K (\rho(K) - \mathtt{rc}_K^{\pi}) \geq \rho(R) - r(\pi) + 1. \tag{8}$$

Using Lemma 2.1 and the fact that $\rho(K \cap \{r_j\}) = 0$ for any j, K, we get $\rho(K) - \mathtt{rc}_K^{\pi} = \rho(K \cap S)$. Finally, as $\rho(R) - r(\pi) + 1 = |R| - 1 - (|R \backslash S| + 1) + 1 = \rho(S)$, Equation (8) is the same as (3). So $\mathtt{fs}(\mathscr{P}') \subseteq \mathtt{fs}(\mathscr{S}')$. □

By observing that ($\mathscr{S}'$) and ($\mathscr{P}'$) have the same objective function, we obtain the following.

Corollary 2.3. *The LPs* ($\mathscr{S}'$) *and* ($\mathscr{P}'$) *have the same objective value for any cost function* C.

The technique behind the proof of Theorem 2.2 is fairly simple, and so it is not clear if it is new. However, the authors are not aware of any prior usage. We simply note that the technique can also be used to describe an existing subtour based formulation of the bidirected cut relaxation (see [12]) in terms of partitions.

3 Uncrossing Partitions

In this section we describe the new partition uncrossing technique. First, we motivate the result. The linear program ($\mathscr{P}'$) that we wrote above is actually different from the one used by Könemann and Tan. In fact, their formulation did not use the constraint (2). Note that if x is the incidence vector of the full components in a Steiner tree, then x satisfies constraint (2). However, there are some cases in which this constraint changes the strength of the LP; here is an example.

Example 3.1. *Let $R = \{1,2,3,4\}$. Define the costs $C_{\{1,2,3\}} = C_{\{1,2,4\}} = 0$, $C_{\{3,4\}} = 1$, and let all other costs be infinite. Then the optimal value of ($\mathscr{P}'$) is 1, whereas removing constraint (2) weakens the LP and reduces the optimal value to 0. For example, the characteristic vector of $\{\{1,2,3\},\{1,2,4\}\}$ meets (1) and (6) but not (2).*

It could be said that Example 3.1 is somewhat artificial. In our application (using hypergraphs to model full components of Steiner trees) the costs specified in Example 3.1 are clearly impossible. If the full component $\{1,2,3\}$ has cost zero, and all edge costs are nonnegative, then note that the path from 1 to 2 in this full component is itself a full component on terminal set $\{1,2\}$ with cost zero, which contradicts $C_{\{1,2\}} = 1$. More generally, we have the following.

Definition 3.2. *Let C be a cost function from $\mathscr{K}$ to $\mathbf{R}_+$. We call C* nondecreasing *if $C_S \geq C_T$ whenever $S \supset T$.*

Lemma 3.3. *In a Steiner tree instance with nonnegative edge costs, if C_K is the cheapest cost of any full component with terminal set K, then C is nondecreasing.*

Proof. Let $T \subset S$ be any sets of terminals, and let $opt(S)$ be a minimum-cost full component with leaf set S. For each pair (u,v) of terminals in T, there is a unique path joining those terminals in $opt(S)$; let H denote the union of these paths over all choices of u and v. Notice that H is a subset of $opt(S)$; since all edges in the original Steiner instance have nonnegative cost, $c(H) \leq c(opt(S)) = C_S$. Also, H is a full component with leaf set T, so by definition $C_T \leq c(H)$. Hence $C_T \leq c(H) \leq C_S$. □

In this section we will show that if C is nondecreasing, then removing constraint (2) from ($\mathscr{P}'$) does not change its optimal value. In other words, we can rewrite the hypergraph-in-spanning-tree LP as follows:

$$\text{minimize} \quad \sum_{K \in \mathscr{K}} C_K x_K \tag{$\mathscr{P}$}$$

$$x \geq 0 \tag{1}$$

$$\forall \pi \in \Pi_R : \quad \sum_{K \in \mathscr{K}} x_K \mathtt{rc}_K^\pi \geq r(\pi) - 1 \tag{6}$$

To be precise, Könemann and Tan interpreted the Robins-Zelikovsky algorithm [26] as an iterated primal-dual algorithm, and ($\mathscr{P}$) is the LP used in the first iteration. We will use *partition uncrossing* to show that ($\mathscr{P}$) and ($\mathscr{S}'$) are equivalent, and also equivalent to a "gainless tree" formulation. Some other nice features of this LP are:

- It is stronger than the bidirected cut relaxation.
- The optimum can be computed in polynomial time (e.g., Queyranne [31] gave a separation oracle for ($\mathscr{S}'$)).
- No extended variables are needed.
- There is essentially only one type of constraint in ($\mathscr{P}$).

Next, we introduce our new technique.

3.1 Definitions for Partitions

For any feasible point x of the program ($\mathscr{P}$), we say that partition π is *tight* for x if the constraint (6) corresponding to π is met with equality. Following notation introduced in [3], let π^0 denote the partition of R into $|R|$ singleton parts. Crucially, the constraint (2) is the same as saying that π^0 is tight for x; we want to show that π^0 is tight for some optimal x of ($\mathscr{P}$). Let $\mathscr{T}(x)$ denote the family of all tight partitions for x. We first show that the family of tight partitions has a special structure (forms a lattice). Second, we argue that some optimal solution has a "large enough" tight family.

Given two partitions π and π' of R, we say that π' *refines* π if each part of π' is contained in some part of π. See Figure 3(a). To say that π *coarsens* π' is the same as saying that π' refines π. It is easy to see that if π'' refines π' and π' refines π, then π'' refines π. The set Π_R of all partitions of R has an additional nice property which is summarized in the following definition.

Definition 3.4 (Lattice operations)**.** *Let $\pi, \pi' \in \Pi_R$. Their* join *$\pi \vee \pi'$ is the most refined partition that coarsens both π and π'. Their* meet *$\pi \wedge \pi'$ is the coarsest partition that refines both π and π'.*

See Figure 3 for an illustration. In the definition of join, when we say "the most refined partition coarsening both π and π'" we mean that, whenever π'' coarsens π and π', then $\pi \vee \pi'$ refines π''. The definition of meet is analogous. Definition 3.4 hides something, because it does not immediately obvious that such partitions exist; nonetheless the existence proof is standard, e.g., see [27]. (For reference, we remark that in general, a *lattice* is any partially ordered set where such meet and join operators are well-defined.)

Here is an intuitive construction for joins and meets.

Fact 3.5. *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',*

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

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

Fact 3.6. *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 $\pi \vee \pi'$.*

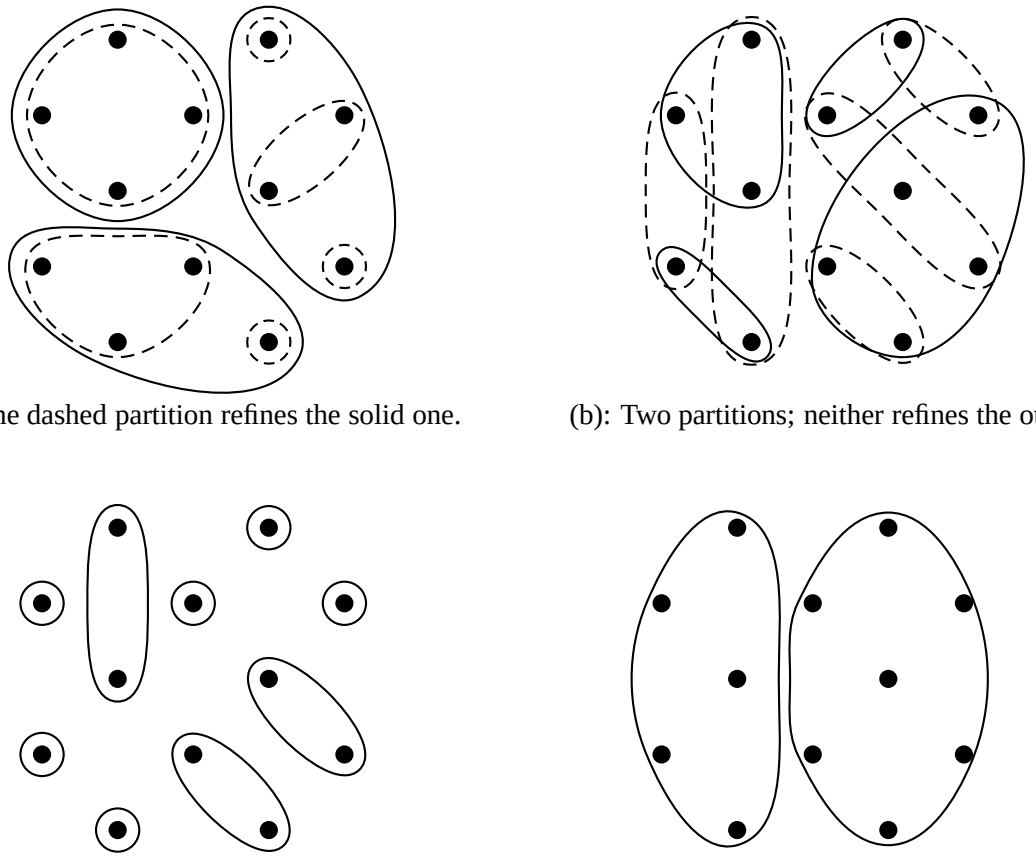
Analogously to π^0, we define π^1 to be the coarsest partition of R, i.e., the partition with just one part. The meet and join operations are each commutative due to Definition 3.4. We will later need the following fact.

Fact 3.7. *The meet and join operations are each associative.*

3.1.1 Warmup: Graphs

Before diving in to our main uncrossing argument (for partitions in hypergraphs), we will prove an analogous easier result for ordinary graphs. Let E_π denote the set of all edges of G that span two different parts of π,

$$E_\pi := \{uv \in E \mid u \text{ and } v \text{ do not lie in the same part of } \pi\}.$$



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

(b): Two partitions; neither refines the other.

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

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

Figure 3: Illustrations of some partitions. The black dots are the terminal set R.

Chopra [5] gave the following linear program for the *minimum-cost spanning tree* (MST) problem in ordinary graphs.

$$\text{minimize} \quad \sum_{e \in E} c_e x_e \tag{$\mathscr{P}_G$}$$

$$x \geq 0 \tag{9}$$

$$\forall \pi \in \Pi_R : \quad x(E_\pi) \geq r(\pi) - 1 \tag{10}$$

The linear program ($\mathscr{P}_G$) is particularly nice in that there is always an integral optimal solution (unless some $c_e < 0$, in which case the program is unbounded). Notice that we can view the hypergraph LP ($\mathscr{P}$) as a generalization of Chopra's LP ($\mathscr{P}_G$), by setting $C_K = \infty$ for all components K with $|K| > 2$ and all $K \notin E$. Similarly, ($\mathscr{S}'$) generalizes a spanning tree LP originally due to Edmonds [9]. In ($\mathscr{P}_G$), just as in ($\mathscr{P}$), we define partition π to be *tight* for x if the corresponding inequality (10) holds with equality.

Chopra showed in [5] that ($\mathscr{P}_G$) is exactly as strong as earlier MST formulations by Edmonds and Fulkerson. In proving the equivalence with Fulkerson's LP, Chopra also showed a result analogous to what we want for hypegraphs: that adding the constraint $\sum_{e \in E} x_e = |R| - 1$ to ($\mathscr{P}_G$) does not affect the optimal

value (i.e., there is an optimal solution to ($\mathscr{P}_G$) for which π^0 is tight). In contrast to our hypergraph result, where the cost function must be nondecreasing, Chopra's result holds for all nonnegative cost functions. We present a new proof of this result as a warmup to the hypergraph case. First we need the following technical lemma.

Lemma 3.8. *For any $\pi, \pi' \in \Pi_R$,*

$$r(\pi \vee \pi') + r(\pi \wedge \pi') \geq r(\pi) + r(\pi'). \tag{11}$$

Proof. Choose F_0 so that (R, F_0) is a forest that induces $\pi \wedge \pi'$. Since $\pi \wedge \pi'$ refines π we can pick a set F of edges disjoint from F_0 such that $(R, F_0 \cup F)$ is a forest that induces π. Similarly let $(R, F_0 \cup F')$ induce π' and $F_0 \cap F' = \emptyset$. Using Fact 3.6, $(R, F_0 \cup F \cup F')$ induces $\pi \vee \pi'$.

Since (R, F_0) is a forest on $|R|$ vertices with $r(\pi \wedge \pi')$ connected components, $|F_0| = |R| - r(\pi \wedge \pi')$. Similarly $|F_0 \cup F| = |R| - r(\pi)$, and $|F_0 \cup F'| = |R| - r(\pi')$, and as $(R, F_0 \cup F \cup F')$ is not necessarily a forest we have $|F_0 \cup F \cup F'| \geq |R| - r(\pi \vee \pi')$. Hence rearranging,

$$\begin{aligned} r(\pi \vee \pi') + r(\pi \wedge \pi') &\geq |R| - |F_0| + |R| - |F_0 \cup F \cup F'| = |R| - |F_0 \cup F| + |R| - |F_0 \cup F'| + |F \cap F'| \\ &\geq |R| - |F_0 \cup F| + |R| - |F_0 \cup F'| = r(\pi) + r(\pi'). \end{aligned}$$

□

We say that partitions π and π' *cross* if π neither refines nor coarsens π'. See Figure 3(b) for an example. The following lemma is trivially true if π and π' do *not* cross. In this case one refines the other, w.l.o.g. π' refines π, and so $\pi \vee \pi' = \pi$ and $\pi \wedge \pi' = \pi'$.

Lemma 3.9 (Meet/join closure of tight partitions in graphs)**.** *Let x be feasible for ($\mathscr{P}_G$). If π and π' are both tight for x, then $\pi \vee \pi'$ and $\pi \wedge \pi'$ are both tight for x.*

Proof. Let $\chi(S)$ denote the characteristic vector of $S \subseteq E$. We can write the partition inequalities of ($\mathscr{P}_G$) as $x \cdot \chi(E_\pi) \geq r(\pi) - 1$.

Since the partition inequalities (10) for $\pi \vee \pi'$ and $\pi \wedge \pi$ are valid for x,

$$x \cdot (\chi(E_{\pi \wedge \pi'}) + \chi(E_{\pi \vee \pi'})) \geq r(\pi \wedge \pi') - 1 + r(\pi \vee \pi') - 1. \tag{12}$$

Using Fact 3.5, $E_{\pi \wedge \pi'} = E_\pi \cup E'_\pi$, and using Fact 3.6, $E_{\pi \vee \pi'} \subseteq E_\pi \cap E_{\pi'}$. The component-wise vector inequality

$$\chi(E_\pi) + \chi(E_{\pi'}) = \chi(E_\pi \cup E_{\pi'}) + \chi(E_\pi \cap E_{\pi'}) \geq \chi(E_{\pi \wedge \pi'}) + \chi(E_{\pi \vee \pi'}) \tag{13}$$

then follows.

Now put everything together; in the first inequality we also need to note that $x \geq 0$.

$$x \cdot (\chi(E_\pi) + \chi(E_{\pi'})) \geq^{(13)} x \cdot (\chi(E_{\pi \wedge \pi'}) + \chi(E_{\pi \vee \pi'})) \geq^{(12)} r(\pi \wedge \pi') - 1 + r(\pi \vee \pi') - 1 \geq^{(11)} r(\pi) - 1 + r(\pi') - 1.$$

But since π and π' are tight, the leftmost and rightmost terms of the above inequality chain are equal. In particular Equation (12) holds with equality, which implies that $\pi \wedge \pi'$ and $\pi \vee \pi'$ are tight. □

Theorem 3.10 ([5])**.** *Let c be nonnegative. There is an optimal point x of ($\mathscr{P}_G$) for which the equality $\sum_{e \in E} x_e = \rho(R)$ holds (i.e., for which π^0 is tight).*

Proof. Let x be a basic optimal solution to ($\mathscr{P}_G$). Hence x is an extreme point of $\texttt{fs}(\mathscr{P}_G)$. Let $\mathscr{T}$ be shorthand for $\mathscr{T}(x)$.

Claim 3.11. *For each edge $e^* \in E$ such that $x_{e^*} > 0$, some $\pi \in \mathscr{T}$ exists with $e^* \in E_\pi$.*

Proof. Suppose otherwise, that every partition π with $e^* \in E_\pi$ is not tight. Define the point x' by

$$x'_e := \begin{cases} x_e, & e \neq e^*; \\ x_e - \varepsilon, & e = e^*. \end{cases}$$

Now there is some $\varepsilon > 0$ for which x' is feasible. Notice that $x + (x - x') \geq x$ and so, by the definition of ($\mathscr{P}_G$), $x + (x - x')$ is also feasible. But $x = x'/2 + (x + (x - x'))/2$, so x can be written as a convex combination of two elements of $\mathtt{fs}(\mathscr{P}_G)$, contradicting the fact that x is an extreme point of $\mathtt{fs}(\mathscr{P}_G)$. □

Let $\bigwedge \mathscr{T}$ denote the common meet of all tight partitions for x (we use Fact 3.7 here). Fact 3.5 implies that $\bigcup_{\pi \in \mathscr{T}} E_\pi = E_{\bigwedge \mathscr{T}}$. Claim 3.11 establishes that, for every edge e, either $x_e = 0$ or $e \in \bigcup_{\pi \in \mathscr{T}} E_\pi$. Hence $x(E) = x(E_{\bigwedge \mathscr{T}})$. By Lemma 3.9, $\bigwedge \mathscr{T}$ is tight and so

$$x(E) = x(E_{\bigwedge \mathscr{T}}) = r(\bigwedge \mathscr{T}) - 1. \tag{14}$$

On the other hand since x satisfies the inequality (10) corresponding to π^0,

$$x(E) = x(E_{\pi^0}) \geq r(\pi^0) - 1. \tag{15}$$

Putting Equations (14) and (15) together we see $r(\bigwedge \mathscr{T}) - 1 \geq r(\pi^0) - 1$, and as π^0 is the unique partition of maximum possible rank, $\bigwedge \mathscr{T} = \pi^0$. Using Lemma 3.9, we see that π^0 is tight for x. □

3.1.2 Hypergraphs

Now we return to the hypergraph LP setting. Unfortunately the "obvious" uncrossing by directly comparing the four constraints (6) for $\pi, \pi', \pi \vee \pi', \pi \wedge \pi'$ does not work. E.g., compare the following example with Equation (13).

Example 3.12. *If $K = \{1,2,3,4\}, \pi = \{\{1,2\},\{3,4\}\}, \pi' = \{\{1,3\},\{2,4\}\}$ then $\mathtt{rc}_K^\pi + \mathtt{rc}_K^{\pi'} < \mathtt{rc}_K^{\pi \vee \pi'} + \mathtt{rc}_K^{\pi \wedge \pi'}$.*

We use a slightly more complicated set of constraints to establish what we want.

Definition 3.13. *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 4 for an example. The notation introduced in Definition 3.13 allows us to restate an earlier remark:

$$\mathtt{rc}_K^\pi = r(\pi) - r(m(\pi, K)). \tag{16}$$

Here is our new uncrossing technique.

Lemma 3.14 (Partition uncrossing)**.** *Let $\pi, \pi' \in \Pi_R$ and let the parts of π be $\pi_1, \pi_2, \ldots$. The following (in)equalities hold:*

$$\forall K \in \mathscr{K}: \quad r(\pi)\left[\mathtt{rc}_K^{\pi'}\right] + \left[\mathtt{rc}_K^{\pi}\right] \quad \geq \quad \left[\mathtt{rc}_K^{\pi \wedge \pi'}\right] + \sum_{i=1}^{r(\pi)} \left[\mathtt{rc}_K^{m(\pi', \pi_i)}\right] \tag{17}$$

$$r(\pi)\left[r(\pi') - 1\right] + \left[r(\pi) - 1\right] \quad = \quad \left[r(\pi \wedge \pi') - 1\right] + \sum_{i=1}^{r(\pi)} \left[r(m(\pi', \pi_i)) - 1\right] \tag{18}$$

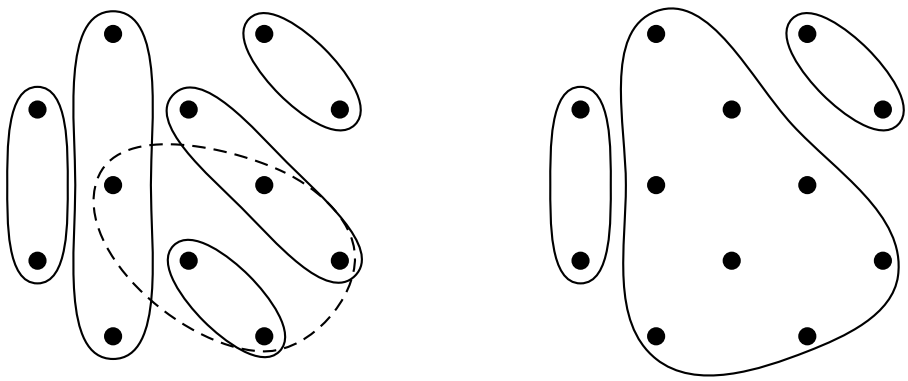


Figure 4: Illustration of merging. The left figure shows a (solid) partition π along with a (dashed) set S. The right figure shows the merged partition $m(\pi, S)$.

The brackets $[\cdot]$ have no special meaning but only emphasize the parallel structure of the equations; they will soon be interpreted as adding and subtracting multiples of the constraint (6). Before we prove Lemma 3.14, let us show how its result is used.

Lemma 3.15. *Let x be feasible for ($\mathscr{P}$), and let the parts of π be $\pi_1, \pi_2, \ldots$. If π and π' are both tight for x, then for each $K \in \mathscr{K}$ with $x_K > 0$ we have*

$$r(\pi)\left[\mathtt{rc}_K^{\pi'}\right] + \left[\mathtt{rc}_K^{\pi}\right] = \left[\mathtt{rc}_K^{\pi\wedge\pi'}\right] + \sum_{i=1}^{r(\pi)}\left[\mathtt{rc}_K^{m(\pi',\pi_i)}\right]. \tag{19}$$

Moreover, $\pi \wedge \pi'$ and each $m(\pi', \pi_i)$ are tight for x.

Proof. Let $\pi = \{\pi_1, \pi_2, \ldots\}$. Using Lemma 3.14, we have

$$\begin{aligned} r(\pi)\left[\sum_K x_K \mathtt{rc}_K^{\pi'}\right] + \left[\sum_K x_K \mathtt{rc}_K^{\pi}\right] &\geq^{(17)} \left[\sum_K x_K \mathtt{rc}_K^{\pi\wedge\pi'}\right] + \sum_{i=1}^{r(\pi)}\left[\sum_K x_K \mathtt{rc}_K^{m(\pi',\pi_i)}\right] \\ &\geq^{(*)} \left[r(\pi\wedge\pi') - 1\right] + \sum_{i=1}^{r(\pi)}\left[r(m(\pi',\pi_i)) - 1\right] \\ &=^{(18)} r(\pi)\left[r(\pi') - 1\right] + \left[r(\pi) - 1\right] \end{aligned}$$

where $(*)$ holds since (6) is valid for $x \in \mathtt{fs}(\mathscr{P})$. Since π and π' are tight, the first and last terms in the above inequality chain are equal, so all inequalities are met with equality. The fact that $(*)$ is met with equality implies that the partitions $\pi \wedge \pi'$ and $m(\pi', \pi_i)$ for all i are tight; the fact that (17) is met with equality implies that Equation (19) holds for each K with $x_K > 0$. □

Theorem 3.16 (Meet/join closure of tight partitions)**.** *Let x be feasible for ($\mathscr{P}$). If π and π' are both tight for x, then $\pi \vee \pi'$ and $\pi \wedge \pi'$ are both tight for x.*

Proof. Repeatedly applying Lemma 3.15, observe $m(\cdots m(m(\pi', \pi_1), \pi_2), \cdots) = \pi \vee \pi'$ is tight. □

So although the uncrossing operation for hypergraphs is more complicated than for graphs, Theorem 3.16 allows us to proceed more or less as we showed in Section 3.1.1. We shall now prove the partition uncrossing inequalities. For both proofs, we first modify the hypergraph by contracting each part of $\pi \wedge \pi'$. Notice that in the contracted graph, $\pi \wedge \pi' = \pi^0$, i.e., $|\pi_i \cap \pi'_j| \leq 1$ for each part π_i of π and each part π'_j of π'. To justify this approach we must remark that the contraction does not affect $r(\pi), r(\pi'), r(\pi \wedge \pi')$ or any $r(\pi', \pi_i)$.

Proof of Equation (18). 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 (16) we know that $r(m(\pi', \pi_i)) = r(\pi') - |\pi_i| + 1$. Thus adding over all i, the right-hand side of Equation (18) 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 (18). □

Let $K \in \mathscr{K}$ be a hyperedge of the original graph, which may not necessarily be a union of parts of $\pi \wedge \pi'$. Define K' to be the union of all parts of $\pi \wedge \pi'$ meeting K, i.e., K' is the part of $m(\pi \wedge \pi', K)$ containing K. Notice that K and K' have the same rank contribution with respect to $\pi, \pi', \pi \wedge \pi'$ and each $m(\pi', \pi_i)$. Hence contracting all parts of $\pi \wedge \pi'$ is justified in proving Equation (17), as we can consider K' in the contracted hypergraph instead of K.

Proof of Equation (17). Fix i. Since $|\pi_i \cap \pi'_j| \leq 1$ for all j, we have

$$\mathtt{rc}^{\pi'}_K - \mathtt{rc}^{m(\pi', \pi_i)}_K \geq \rho(\pi_i \cap K) \tag{20}$$

because, when we merge the parts of π' intersecting π_i, we make K span at least $\rho(\pi_i \cap K)$ fewer parts.

Adding the right-hand side of Equation (20) over all i gives $\sum_i \rho(\pi_i \cap K)$, which by Lemma 2.1 is equal to $\rho(K) - \mathtt{rc}^{\pi}_K$. Hence, we have

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

Finally note that $\rho(K) = \mathtt{rc}^{\pi \wedge \pi'}_K$ and the above equation, after rearranging, yields Equation (17). □

Next we explain what sort of tight partitions can be assumed to exist. The basic idea is that, for any $S \subsetneq T \subset R$, we can always decrease x_T by a little bit and increase x_S by the same amount without increasing the overall cost; so either $x_T = 0$ or this "replacement" is prevented by a tight partition.

Lemma 3.17. *Let C be nondecreasing. There is an optimal solution x^* to ($\mathscr{P}$) such that the following holds: for any K with $x^*_K > 0$ and for any $r \in K$, there exists $\pi \in \mathscr{T}(x^*)$ such that the part of π containing r contains no other vertices of K.*

Proof. Let x be any optimal solution, $x_K > 0$, and $r \in K$ as described. Suppose that no $\pi \in \mathscr{T}(x)$ exists as specified in the statement Lemma 3.17. Let e_K denote the unit basis vector for component K. For the moment assume $|K| > 2$. Let $K' = K \backslash \{r\}$ and define $x' = x - te_K + te_{K'}$ where $t \geq 0$ is a parameter. Since C is nondecreasing, x' has objective value no more than x. In order for x' to be feasible for ($\mathscr{P}$), we need $t \leq x_K$ and

$$\sum_K \mathtt{rc}^{\pi}_K x_K + (\mathtt{rc}^{\pi}_{K'} - \mathtt{rc}^{\pi}_K) t \geq r(\pi) - 1, \qquad \forall \pi \in \Pi_R$$

Notice that $\mathtt{rc}^{\pi}_{K'} - \mathtt{rc}^{\pi}_{K}$ is -1 when the part of π containing r contains no other vertices of K, and 0 otherwise. By hypothesis (i.e., our choices of π and K) all partitions π with $\mathtt{rc}^{\pi}_{K'} - \mathtt{rc}^{\pi}_{K} = -1$ are not tight, so we can take

$$t = \min\{x_K, \min_{\pi:\mathtt{rc}^{\pi}_{K'} \neq \mathtt{rc}^{\pi}_{K}} \sum_{K} \mathtt{rc}^{\pi}_{K} x_K - r(\pi) + 1\} > 0.$$

The resulting x' no longer violates the conclusion of this theorem for K and r. Finally, if $|K| = 2$ then the same approach works except that we simply define $x' = x - te_K$.

We iterate the replacement operation described in the previous paragraph until no such K and r exist; after each iteration redefine $x := x'$. We will complete the proof by showing that only finitely many iterations can occur. Notice that $\mathscr{T}(x') \supseteq \mathscr{T}(x)$. Hence, the number $|\mathscr{T}(x)|$ of tight partitions is nondecreasing. Furthermore, notice that in each iteration, if $t \neq x_K$ then $|\mathscr{T}(x)|$ increases, and otherwise the quantity

$$\sum_{S \in \mathscr{K}: x_S \neq 0} |S| \tag{21}$$

decreases. Since the quantity (21) and $|\mathscr{T}(x)|$ are integral, nonnegative, and bounded, only a finite number of iterations can occur. We define x^* to be the final x, and the proof is complete. □

With these tools, we can complete the main proof of this section.

Theorem 3.18. *Let C be nondecreasing. Then of all optimal feasible points of ($\mathscr{P}$), there is one for which (2) is valid (i.e., for which π_0 is tight).*

Proof. Let x^* be an optimal solution as specified by Lemma 3.17. Let $\mathscr{T}$ be shorthand for $\mathscr{T}(x^*)$.

For any hyperedge K such that $x^*_K > 0$, and for any $\{u, v\} \subset K$, the conclusion of Lemma 3.17 guarantees that $\{u, v\} \in E_\pi$ for some $\pi \in \mathscr{T}$. Notice furthermore that $E_{\bigwedge \mathscr{T}} = \cup_{\pi \in \mathscr{T}} E_\pi$. It follows that any hyperedge K such that $\mathtt{rc}^{\bigwedge \mathscr{T}}_{K} < |K| - 1$ must satisfy $x^*_K = 0$. Since x^* meets inequality (6) with $\pi = \pi^0$,

$$r(\pi^0) - 1 \leq \sum_K x^*_K (|K| - 1) = \sum_K x^*_K \mathtt{rc}^{\bigwedge \mathscr{T}}_{K}.$$

But $\bigwedge \mathscr{T}$ is tight and so the right-hand side of the above equation can be rewritten

$$r(\pi^0) - 1 \leq \sum_K x^*_K \mathtt{rc}^{\bigwedge \mathscr{T}}_{K} = r(\bigwedge \mathscr{T}) - 1.$$

It follows that $\bigwedge \mathscr{T} = \pi^0$. By Theorem 3.16 π^0 is tight for x^*, i.e., (2) is valid for x^*. □

Corollary 3.19. *For any Steiner tree problem instance, the optimal values of ($\mathscr{S}'$), ($\mathscr{P}'$) and ($\mathscr{P}$) are equal.*

Proof. This follows immediately from Theorem 2.2, Lemma 3.3 and Theorem 3.18. □

3.1.3 Polyhedral Results

Analogously to the bounded LP relaxations ($\mathscr{P}'$) and ($\mathscr{S}'$) of the spanning hypertree problem, the following LP is a bounded LP relaxation for the spanning tree problem in ordinary graphs.

$$\text{minimize} \quad \sum_{e \in E} c_e x_e \tag{$\mathscr{P}'_G$}$$

$$x \geq 0 \tag{22}$$

$$\sum_{e \in E} x_e = |R| - 1 \tag{23}$$

$$\forall \pi \in \Pi_R: \quad \sum_{e \in E_\pi} x_e \geq r(\pi) - 1. \tag{24}$$

As we proved, both programs ($\mathscr{P}_G$) and ($\mathscr{P}'_G$) have the same optimal value for nonnegative cost functions. However, as soon as any edge has negative cost, program ($\mathscr{P}_G$) is clearly unbounded. With a little more work we can obtain a proof of the following fact. The *dominant* of a set S is the set $\{y \mid \exists x \in S : y \geq x\}$.

Fact 3.20 ([5])**.** *The dominant of* $\mathtt{fs}(\mathscr{P}'_G)$ *is* $\mathtt{fs}(\mathscr{P}_G)$.

In contrast, $\mathtt{fs}(\mathscr{P})$ is *not* the dominant of $\mathtt{fs}(\mathscr{P}')$; we obtain an example by re-examining Example 3.1.

Example 3.21. *Let* $R = \{1,2,3,4\}$*. Then the point* $x := e_{\{1,2,3\}} + e_{\{1,2,4\}} \in \mathtt{fs}(\mathscr{P})$*. However, in order for* x *to be in the dominant of* $\mathtt{fs}(\mathscr{P}')$*, there needs to be a point of the form* $y := \alpha e_{\{1,2,3\}} + \beta e_{\{1,2,4\}}$ *in* $\mathtt{fs}(\mathscr{P}')$*. The partition inequalities* (6) *require* $\alpha, \beta \geq 1$ *but the equality* (2) *requires* $\alpha + \beta = 1$*. So no such* y *exists.*

Other more complicated generalizations of Fact 3.20 do hold, however.

4 Applications of Partition Uncrossing

Uncrossing has played a critical role in recent work in the area of network design, e.g., [11, 14, 17, 18, 19]. In that setting one uncrosses *sets* in an LP with *subtour elimination constraints*, rather than partitions as we have done here[4]. In this section we prove a property of the polyhedron $\mathtt{fs}(\mathscr{P})$ in the spirit of these results. A set of partitions is called a *chain* if no two of the partitions are crossing; equivalently, a set is a chain iff it can be written in the form $\{\pi^{[1]}, \dots, \pi^{[t]}\}$ where $\pi^{[j]}$ refines $\pi^{[i]}$ for all $t \geq j \geq i \geq 1$. The *support* of a vector is the collection of indices where it is nonzero:

$$\mathrm{supp}(x) = \{i \mid x_i \neq 0\}.$$

Using a span argument, we will establish that every extreme point of $\mathtt{fs}(\mathscr{P})$ is *defined by its support and a chain of tight partitions*, in the sense that it is the unique solution to those constraints. The common analogous result in the subtour setting uses a so-called *laminar* family of subtours in place of the chain of partitions. We have chosen to use a *span*-based argument; the original span argument, due to Jain [14], has been repeated in several places (e.g., [11, 18, 29]) and so we hope that the general idea is already familiar.

We need to rephrase the result of the previous section so as to be suitable for the span argument. Given two partitions $\pi = \{\pi_1, \pi_2, \dots\}$ and π', define the *crossing parts* $\mathtt{cp}_{\pi'}(\pi)$ to be the set

$$\mathtt{cp}_{\pi'}(\pi) := \{\pi_i \in \pi \mid m(\pi', \pi_i) \neq \pi'\}.$$

Fact 4.1. *We have that* π *refines* π' *if and only if* $\mathtt{cp}_{\pi'}(\pi) = \emptyset$*.*

Corollary 4.2. *Suppose* x *is feasible for* ($\mathscr{P}$) *and that* π *and* π' *are tight for* x*. Then for each* K *with* $x_K > 0$*,*

$$\mathtt{rc}_K^{\pi} + |\mathtt{cp}_{\pi'}(\pi)| \cdot \mathtt{rc}_K^{\pi'} = \mathtt{rc}_K^{\pi \wedge \pi'} + \sum_{\pi_i \in \mathtt{cp}_{\pi'}(\pi)} \mathtt{rc}_K^{m(\pi', \pi_i)}.$$

Proof. Apply Lemma 3.15 and subtract $(r(\pi) - |\mathtt{cp}_{\pi'}(\pi)|)\mathtt{rc}_K^{\pi'}$ from both sides of Equation (19). □

[4]The general idea is that if S, S' are tight crossing subtour sets, then so are $S \cup S'$ and $S \cap S'$. When π^0 is tight, one can show this is equivalent to the meet/join closure of tight partitions (Theorem 3.16) modulo the subtour/partition correspondence (Theorem 2.2). When π^0 is not tight however, we have not found a way to apply previous uncrossing results to our setting.

In the following, x^* is a feasible point of ($\mathscr{P}$). Without loss of generality (by deleting items from $\mathscr{K}$) we assume that $x^*_K > 0$ for all $K \in \mathscr{K}$. Let $\text{span}(\pi)$ denote the vector

$$\text{span}(\pi) := (\mathtt{rc}^{\pi}_K)_{K \in \mathscr{K}},$$

i.e., the vector of coefficients in the constraint (6). For a *set* $\mathscr{S}$ of partitions let $\text{span}(\mathscr{S})$ denote the vector space spanned by the vectors $\{\text{span}(s) \mid s \in \mathscr{S}\}$.

Lemma 4.3 (Span lemma)**.** *Let x^* be feasible for ($\mathscr{P}$). Let $\mathscr{C}$ be any inclusion-maximal chain in $\mathscr{T}(x^*)$. Then* $\text{span}(\mathscr{C}) = \text{span}(\mathscr{T}(x^*))$.

Proof. Again let $\mathscr{T} := \mathscr{T}(x^*)$. Suppose for the sake of contradiction that π is a tight partition such that $\text{span}(\pi) \notin \text{span}(\mathscr{C})$. Pick such a counterexample π having minimal rank.

Now $\mathscr{C}$ must contain some partition C that crosses π, since $\mathscr{C}$ is a maximal chain in $\mathscr{T}$, and if no such partition existed, we could add π to $\mathscr{C}$. Choose C to be the most refined partition in $\mathscr{C}$ that crosses π. The following intermediate claim will be needed later. We say π' *strictly refines* π if π' refines π and $\pi' \neq \pi$.

Claim 4.4. *If $C' \in \mathscr{C}$ and C' strictly refines C, then C' refines π.*

Proof. By our choice of C, note that C' and π don't cross. So either C' refines π or vice-versa. But if π refines C' then (since C' refines C) then π also refines C, which contradicts the fact that C and π cross. So C' refines π. □

Let the parts of C be $C_1, C_2, \ldots$, as usual. Since $x_K > 0$ for all K, Corollary 4.2 can be restated as saying that

$$\text{span}(C) + |\mathtt{cp}_\pi(C)| \cdot \text{span}(\pi) = \text{span}(\pi \wedge C) + \sum_{C_i \in \mathtt{cp}_\pi(C)} \text{span}(m(\pi, C_i))$$

and that the partitions $\pi \wedge C, m(\pi, C_i)$ are all tight. By Fact 4.1, $\mathtt{cp}_\pi(C)$ is nonempty. Since $\text{span}(C) \in \text{span}(\mathscr{C})$ and $\text{span}(\pi) \notin \text{span}(\mathscr{C})$, it follows that either $\text{span}(\pi \wedge C) \notin \text{span}(\mathscr{C})$ or $\text{span}(m(\pi, C_i)) \notin \text{span}(\mathscr{C})$ for some $C_i \in \mathtt{cp}_\pi(C)$.

Case 1: $\text{span}(\pi \wedge C) \notin \text{span}(\mathscr{C})$. We claim in fact that $\pi \wedge C$ can be added to the chain $\mathscr{C}$, contradicting the maximality of $\mathscr{C}$. We need to establish that $\pi \wedge C$ crosses no partition $C' \in \mathscr{C}$. There are two subcases:

- C refines C', in which case $\pi \wedge C$ refines C'.
- C' strictly refines C. Then using Claim 4.4, we know that C' refines π, so C' refines $\pi \wedge C$.

Indeed, in either case $\pi \wedge C$ and C' do not cross.

Case 2: $\text{span}(m(\pi, C_i)) \notin \text{span}(\mathscr{C})$. Note $m(\pi, C_i)$ is tight. Since $C_i \in \mathtt{cp}_\pi(C)$, $m(\pi, C_i)$ has smaller rank than π. This contradicts our choice of π. □

4.1 Extreme Points

The main consequence of Lemma 4.3 is that extreme points of $\mathtt{fs}(\mathscr{P})$ are zero in most of their coordinates.

Theorem 4.5. *Let x^* be an extreme point of $\mathtt{fs}(\mathscr{P})$ and let $\mathscr{C}$ be an inclusion-maximal chain in $\mathscr{T}(x^*)$. Then x^* has at most $|\mathscr{C}|$ nonzero coordinates.*

Proof. Consider the family of ($\mathscr{P}$)'s constraints that hold for x^* with equality. We have assumed that $x^*_K > 0$ for all $K \in \mathscr{K}$, so all of these constraints correspond to tight partitions. Is it a well known fact in polyhedral theory that, since x^* is extreme, the span of these constraints is full-dimensional, i.e., $\text{span}(\mathscr{T}(x^*)) = \mathbf{R}^{\mathscr{K}}$. Using Lemma 4.3, $\text{span}(\mathscr{C}) = \text{span}(\mathscr{T}(x^*)) = \mathbf{R}^{\mathscr{K}}$. However, the dimension of $\text{span}(\mathscr{C})$ is at most $|\mathscr{C}|$ and hence $|\mathscr{K}| \leq |\mathscr{C}|$. □

Corollary 4.6. *Each extreme point of* $\texttt{fs}(\mathscr{P})$ *has at most* $|R|-1$ *nonzero coordinates.*

Proof. Any chain in Π_R has at most $|R|$ members, since whenever π' strictly refines π we have $r(\pi') > r(\pi)$. Without loss of generality we can remove the vacuously true constraint corresponding to π^1 and then the longest possible chain has $|R|-1$ members. □

Note that any (graph) spanning tree meets this bound. Corollary 4.6 is reminiscent of results in a recent paper by Goemans [11] dealing with minimum-degree-plus-two spanning trees. Like Goemans, we can establish a "local sparseness" result for the extreme points of $\texttt{fs}(\mathscr{P})$. We also obtain an iterated rounding proof of the following well known result; a similar proof is given in [18] for the subtour formulation.

Corollary 4.7 ([9])**.** *Every basic solution of* ($\mathscr{P}_G$) *is the characteristic vector of a spanning tree.*

Proof. Consider a basic solution x and the support graph $H := (V, \text{supp}(x))$. Corollary 4.6 applies since ($\mathscr{P}_G$) is a special case of ($\mathscr{P}$), and so there must be some vertex $v \in V$ with degree one in H. To meet the partition inequality for $\{\{v\}, V\setminus\{v\}\}$, the single edge e of H incident on v must have $x_e \geq 1$; and if $x_e > 1$ it is easy to see the solution is not basic, so $x_e = 1$. Now delete v and e and consider the remaining graph. The projection of the old basic solution on the new edge set is once again basic. Finally, the result of Corollary 4.7 follows by induction. □

Using Corollary 4.6 we have computed all extreme points of $\texttt{fs}(\mathscr{P})$ for small values of $|R|$. We say two extreme points of R are *isomorphic* if they are the same under some relabeling of R, i.e., $e_{\{1,2,3\}} + e_{\{3,4\}}$ is isomorphic to $e_{\{1,3,4\}} + e_{\{2,3\}}$. The number of nonisomorphic extreme points of $\texttt{fs}(\mathscr{P})$ is 6 for $|R| = 4$, 27 for $|R| = 5$, and 407 for $|R| = 6$. These computational results have informed our study of these LPs. For example, we noticed and subsequently proved that as R grows, some extreme points exhibit bad fractionality; hence an iterated rounding approach as in [14] unfortunately looks impossible.

4.2 Gainless Tree Formulation

Given an ordinary spanning tree T of R 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,

$$gain_T(K) := c(T) - mst(T/K) - C_K,$$

where $mst(T/K)$ 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 a cheaper spanning hypertree is possible when K is included. We say that a tree T is *gainless* if $gain_T(K) \leq 0$ for all $K \in \mathscr{K}$.

Definition 4.8. *The quantity* $t^{\mathscr{K}}_+$ *is the maximum cost of any gainless tree* T *with nonnegative edge weights. The quantity* $t^{\mathscr{K}}$ *is the maximum cost of any gainless tree* T *with arbitrary edge weights.*

These definitions essentially come from Karpinksi and Zelikovsky [15]. They used full components and gain to devise a novel approximation algorithm for the Steiner tree problem, which had the best approximation factor known at the time. To be precise, they defined a single quantity t^k without specifying whether or not the edge weights could be negative. Using partition uncrossing, we will prove the following theorems.

Theorem 4.9. *The quantity $t^{\mathscr{K}}$ is equal to the optimum value of* ($\mathscr{P}'$).

Theorem 4.10. *The quantity $t_+^{\mathscr{K}}$ is equal to the optimum value of* ($\mathscr{P}$).

Karpinski and Zelikovsky applied preprocessing to their graphs, hence the full component cost function C is nondecreasing in their setting. Hence, using Theorem 4.9, Corollary 3.19, and Theorem 4.10 we discover an interesting fact: t^k from [15] has the same value whether or not negative tree edges are allowed.[5] Also of note is the fact that t^k was used in [15] as an upper bound in one place and as a lower bound in another; this resembles an LP optimal value already, since the optimal value in (say) a minimization program is a lower (resp. upper) bound on the objective value of any feasible primal (resp. dual) solution.

The LP dual of ($\mathscr{P}'$), which we will need, has a variable y^π for each partition $\pi \in \Pi_R$. Notice that the equality constraint (2) is just the same as saying (6) holds with equality for the partition π^0, so y^{π^0} is unconstrained while each other y^π should be nonnegative. Hence the LP dual of ($\mathscr{P}'$) is

$$\text{maximize} \quad \sum_{\pi \in \Pi_R} (r(\pi)-1)\cdot y^\pi \qquad (\mathscr{P}'^*)$$

$$\forall \pi \in \Pi_R \backslash \{\pi^0\}: \quad y^\pi \geq 0 \qquad (25)$$

$$\forall K \in \mathscr{K}: \quad \sum_{\pi \in \Pi_R} y^\pi \mathtt{rc}_K^\pi \leq C_K \qquad (26)$$

For any $y \in \mathbf{R}^{\Pi_R}$, define $c(y)$ to denote the objective value of y in ($\mathscr{P}'^*_G$). Call a (not necessarily feasible) solution y of ($\mathscr{P}'^*$) *chain-supported* if $\text{supp}(y)$ is a chain.

Our proof of Theorem 4.9 works in three steps, which we now sketch. First, by giving a dual interpretation of partition uncrossing (Lemma 3.14), we show that ($\mathscr{P}'$) has an optimum that is chain-supported. Second, we define a surjective function $\mathtt{MSTDual}$; it maps each spanning tree T of R to a chain-supported y such that satisfies (25), and such that $c(y) = c(T)$. Third, we show for each K that (26) holds for $y = \mathtt{MSTDual}(T)$ if and only if $gain_T(K) \leq 0$. Hence using these properties,

$$\begin{aligned} opt(\mathscr{P}') &= \max\{c(y) \mid y \text{ is chain-supported, (25) holds, and for all } K \in \mathscr{K} \text{ (26) holds}\} \\ &=^{(\dagger)} \max\{c(T) \mid \text{for all } K \in \mathscr{K}, gain_T(K) \leq 0\} \\ &= t^{\mathscr{K}}. \end{aligned}$$

We elaborate on (†) and discuss Theorem 4.10 after proving the supporting claims, which now follow.

Lemma 4.11 (Dual uncrossing)**.** ($\mathscr{P}'^*$) *always has an optimum y_* such that y_* is chain-supported.*

Proof. Suppose y is any feasible solution to ($\mathscr{P}'^*$) such that two crossing partitions π, π' have $y^\pi, y^{\pi'} \neq 0$. Note that π^1 does not cross any other partition, so we may assume that $y^\pi, y^{\pi'} > 0$. Let e^π denote the unit basis vector for partition π. Define

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

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

$$t = \min\left\{\frac{y^{\pi'}}{r(\pi)}, y^\pi\right\}$$

[5] Furthermore, the analysis in [15] is correct under either interpretation.

produces a feasible y' with the same objective value as y. From Equation (17) we deduce that this y' is feasible for ($\mathscr{P}'^*$); from Equation (18) we deduce that y' has the same objective value as y. By *uncrossing* π *and* π' we mean the map $y \mapsto y'$.

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

$$ranksum_i(y) = \sum_{\pi : r(\pi) = i} y^\pi.$$

Let y_* be a optimal solution ($\mathscr{P}'^*$) that is maximal with respect to lexicographic ordering on the vector $(ranksum_n(y_*), ranksum_{n-1}(y_*), \dots)$; 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 π'. By uncrossing them in y_*, we increase $y_*^{\pi \wedge \pi'}$. But when π and π' cross, it is not hard to see that

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

and it is easy to see that

$$\max\{r(\pi), r(\pi')\} \geq r(m(\pi', \pi_i))$$

for all i. Hence by uncrossing π and π' in y_*, the lexicographic value of $(ranksum_n(y_*), ranksum_{n-1}(y_*), \dots)$ strictly increases. This contradicts the maximality of y_*. Hence no such π, π' exist, and y_* is a chain-supported optimum to ($\mathscr{P}'^*$). □

4.2.1 MST Duals

The polytope $\texttt{fs}(\mathscr{P}'_G)$ is commonly called the *spanning tree polytope* because it is the convex hull of the incidence vectors of all spanning trees of G. We will need its LP dual, which follows.

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{\pi \in \Pi_R} (r(\pi) - 1) \cdot y^\pi & (\mathscr{P}'^*_G) \\ \forall \pi \in \Pi_R \backslash \{\pi^0\} : & y^\pi \geq 0 & (25) \\ \forall uv \in E(G) : & \displaystyle\sum_{\pi : \pi \text{ separates } u \text{ from } v} y^\pi \leq c_{uv} & (27) \end{array}$$

Note that the objective value in ($\mathscr{P}'^*_G$) is $c(y)$, the same objective from ($\mathscr{P}'^*$). Additionally, in ($\mathscr{P}'^*_G$) and in ($\mathscr{P}'^*$) the variable y^{π^1} is vacuous, i.e. it can have any value without affecting the feasibility or optimality of the solution. So from now on we assume $y^{\pi^1} = 0$ for convenience.

Chopra [5] used the LPs ($\mathscr{P}'_G$) and ($\mathscr{P}'^*_G$) to give a primal-dual interpretation of Krusal's MST algorithm (see also [16]). When G is a tree, although $\texttt{fs}(\mathscr{P}'_G)$ is just a single point, $\texttt{fs}(\mathscr{P}'^*_G)$ is useful for our purposes. We summarize Chopra's result (specialized to the case that G is a tree) in the procedure `MSTDual`.

Theorem 4.12 ([5]). *For any tree T, the dual solution returned by* $\texttt{MSTDual}(T)$ *is feasible and optimal for* ($\mathscr{P}'^*_T$)*, and chain-supported.*

Proof. The proof of feasibility and optimality is standard (see [5, 16]) and is therefore omitted, although here we are also allowing for negative-weight edges. The fact that y is chain-supported follows by construction, since $\pi_*^{[i]}$ refines $\pi_*^{[j]}$ for all $j \geq i$. □

Algorithm 1 The algorithm $\texttt{MSTDual}(T)$.

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 $y_* := w_1 e^{\pi_*^{[1]}} + \sum_{i=2}^{t} (w_i - w_{i-1}) e^{\pi_*^{[i]}}$ (∗Note $\pi_*^{[1]} = \pi^0$∗)

Corollary 4.13. *For any tree T on vertex set R, we have $c(T) = c(\texttt{MSTDual}(T))$.*

Proof. Apply strong LP duality to the result of Theorem 4.12, and use the fact that the characteristic vector of T is an optimal solution to $(\mathscr{P}'_T)$. □

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

Lemma 4.14. *Suppose y satisfies (25) and y is chain-supported. Then there exists a tree T on vertex set R for which $y = \texttt{MSTDual}(T)$.*

Proof. Denote the chain $\text{supp}(y) \cup \{\pi^0\}$ by $\pi^{[1]}, \pi^{[2]}, \ldots, \pi^{[t]}$ where $\pi^{[i]}$ refines $\pi^{[i+1]}$ for $1 \le i < t$. For convenience let $\pi^{[t+1]}$ denote π^1, the coarsest partition. Denote the $\pi^{[i]}$-coordinate of y by $y^{[i]}$.

We now define a set $E^{[i]}$ of edges for each $1 \le i \le t$. We claim such sets can be chosen so that $(R, \cup_{j=1}^{i} E^{[j]})$ induces $\pi^{[i]}$ for each $0 \le i \le t$. 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 $\texttt{MSTDual}(T)$ runs, $\pi_*^{[i]} = \pi^{[i]}$ for all i, $w_1 = y^{\pi^0}$, and $w_i - w_{i-1} = y^{[i]}$ for all i. Hence the output y_* is equal to y. □

The following lemma is the final technical ingredient.

Lemma 4.15 (Dual interpretation of gain)**.** *Let T be a tree on vertex set R. Let $y := \texttt{MSTDual}(T)$. Full component K has positive gain in T if and only if y violates the inequality (26) for K.*

Proof. The key fact is that

$$mst(T) - mst(T/K) = \sum_{\pi} y^{\pi} \texttt{rc}_K^{\pi}. \tag{28}$$

Once we establish Equation (28), Lemma 4.15 follows since then, by the definition of gain,

$$gain_T(K) = mst(T) - mst(T/K) - C_K = \sum_{\pi} y^{\pi} \texttt{rc}_K^{\pi} - C_K \tag{29}$$

and the right-most term of Equation (29) is positive iff y violates (26) for K.

We need to determine the cost of a minimum spanning tree in T/K. Recall that Kruskal's MST algorithm operates by examining all edges in increasing order of weight, and constructing a solution of each edge that does not create a cycle with the partial solution up to that point.

Now, the effect of contracting K is that, at the start of the algorithm, K is connected, and we need to connect the vertices of K to the rest of R. As we run Kruskal's algorithm on the graph T/K, there will be exactly $|K| - 1$ edges that form cycles with the partial solution; call these edges $e_1, \ldots, e_q$. Then the

minimum spanning tree of T/K is just $T\backslash\{e_1,\ldots,e_q\}$. Now $mst(T)-mst(T/K)=\sum_{i=1}^{q}c(e_i)$, but we also claim that

$$\sum_{i=1}^{q}c(e_i)=\sum_{\pi}y^{\pi}\mathtt{rc}_K^{\pi} \tag{30}$$

which in turn establishes Equation (28). A proof of Equation (30) (which is not difficult, but requires more notation than we wish to develop here) appears as part of [16, Lemma 5]. □

Proof of Theorem 4.9. The step needing elaboration is the equality (†) in the sketch given earlier. By Lemma 4.14,

$$\begin{aligned}&\max\{c(y)\mid y \text{ is chain-supported, (25) holds, and for all } K\in\mathscr{K} \text{ (26) holds}\}\\ =&\max\{c(\mathtt{MSTDual}(T))\mid \text{for all } K\in\mathscr{K} \text{ (26) holds at } y=\mathtt{MSTDual}(T)\}\end{aligned}$$

Then applying the dual interpretation of gain (Lemma 4.15) and the fact that $c(T)=c(\mathtt{MSTDual}(T))$,

$$\begin{aligned}&\max\{c(\mathtt{MSTDual}(T))\mid \text{for all } K\in\mathscr{K} \text{ (26) holds at } y=\mathtt{MSTDual}(T)\}\\ =&\max\{c(T)\mid \text{for all } K\in\mathscr{K}, gain_T(K)\le 0\ \}.\end{aligned}$$

□

We can obtain essentially the same result when working with the dual of the unbounded formulation ($\mathscr{P}$), which we denote by ($\mathscr{P}^*$). Notice that the only difference between ($\mathscr{P}^*$) and ($\mathscr{P}'^*$) is that in ($\mathscr{P}^*$), y^{π^0} has to be nonnegative. The analogous components that complete the proof of Theorem 4.10 are as follows.

- $t_+^{\mathscr{K}}$ and the optimum value of ($\mathscr{P}$) both are well-defined iff $C\ge 0$.
- Dual uncrossing as defined in Lemma 4.11 can only increase y^{π^0}, and so is applicable to ($\mathscr{P}^*$) .
- The constructions MSTDual and Lemma 4.14 are unchanged. We need only to note that $y^{\pi^0}\ge 0$ implies that all edge costs of T are positive, and vice versa.

5 Future Work

The *bidirected cut formulation* is one of the most well-known and deeply-studied LP relaxations of the Steiner tree problem. There are several equivalent (compact, extended) formulations, but the "natural space" of the LP consists of a relaxed indicator variable x_e for each $e\in E$. See [12] for a comprehensive survey of results in this formulation. The following result is due to Polzin and Vahdati Daneshmand [23].

Theorem 5.1. *The LP ($\mathscr{S}'$) is (sometimes strictly) stronger than the bidirected cut relaxation.*

The preprocessing we described in the first section can be viewed as a transformation that produces another graph, rather than a hypergraph. Initially let the new graph G' consist of vertex set R and no edges. Then, for each full component $K\in\mathscr{K}$, compute the cheapest full component $opt(K)$ with leaf set K, and add a new *clone* of that full component into G'. In the resulting G', every Steiner node belongs to exactly one (cloned) full component. By standard metricity assumptions (see, e.g., [13]) we can assume that every Steiner node has degree at least 3. We call this a *preprocessed* graph.

Let ($\mathscr{B}^+$) denote the bidirected cut relaxation for a preprocessed graph, strengthened with the constraint that in each (cloned) full component, all edge values are equal. The following result can be proved using techniques from [23].

Theorem 5.2. *The formulations* $(\mathscr{B}^+)$ *and* $(\mathscr{P}')$ *are equally strong.*

In fact, the result can be framed as a polyhedral equivalence, by projecting each full component's (equal) edge values onto a single variable.

We furthermore conjecture even without the strengthening, the LPs are equal in a certain natural setting. The set $\mathscr{K}$ of all full components is *down-closed* if whenever $K \in \mathscr{K}$, $J \subseteq K$ and $|J| \geq 2$, then $J \in \mathscr{K}$.

Conjecture 5.3. *Suppose G is a preprocessed graph, and $\mathscr{K}$ is down-closed. Then the bidirected cut formulation has the same optimal value as* $(\mathscr{P}')$.

We ultimately hope that the results of this study have applications beyond connecting between existing papers. One possibility is to use the LPs to get a new and improved primal-dual approximation algorithm for the Steiner problem. If Conjecture 5.3 is true, then it might be possible to use the LP $(\mathscr{P}')$ to bound the integrality gap of the bidirected cut formulation in some situations — at the moment the best lower and upper bounds [1] are $8/7$ and 2.

References

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

[2] A. Borchers and D.-Z. Du. The k-Steiner ratio in graphs. In *ACM Symp. on Theory of Computing*, pages 641–649, 1995.

[3] E. R. Canfield. Meet and join within the lattice of set partitions. *Electr. J. Comb.*, 8(1), 2001.

[4] M. Chlebík and J. Chlebíková. Approximation hardness of the Steiner tree problem on graphs. In *Proceedings, Scandinavian Workshop on Algorithm Theory*, pages 170–179, 2002.

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

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

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

[8] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[9] J. Edmonds. Matroids and the greedy algorithm. *Math. Programming*, 1:127–136, 1971.

[10] M. X. Goemans. The Steiner tree polytope and related polyhedra. *Math. Program.*, 63(2):157–182, 1994.

[11] M. X. Goemans. Minimum bounded degree spanning trees. In *FOCS*, pages 273–282. IEEE Computer Society, 2006.

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

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

[14] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. Preliminary version appeared at FOCS 1998.

[15] M. Karpinski and A. Zelikovsky. New approximation algorithms for the Steiner tree problems. *J. Combinatorial Optimization*, 1(1):47–65, 1997.

[16] J. Könemann and K. Tan. A fresh look at Steiner trees: Greedy vs primal-dual algorithms. Technical report, University of Waterloo, 2006.

[17] L. C. Lau, J. Naor, M. Salavatipour, and M. Singh. Survivable network design with degree or order constraints. In *STOC*, 2007. To appear.

[18] L. C. Lau and M. Singh. Approximating minimum bounded degree spanning trees to within one of optimal. In *STOC*, 2007. To appear.

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

[20] T. Polzin. *Algorithms for the Steiner Problem in Networks*. PhD thesis, Universität des Saarlandes, February 2003.

[21] T. Polzin and S. Vahdati Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112(1-3):241–261, 2001. Preliminary version appeared at COS 1998.

[22] T. Polzin and S. Vahdati Daneshmand. Improved algorithms for the Steiner problem in networks. *Discrete Applied Mathematics*, 112(1-3):263–300, 2001.

[23] T. Polzin and S. Vahdati Daneshmand. On Steiner trees and minimum spanning trees in hypergraphs. *Oper. Res. Lett.*, 31(1):12–20, 2003.

[24] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, 1999.

[25] 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.

[26] G. Robins and A. Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134, 2005. Preliminary version appeared as "Improved Steiner tree approximation in graphs" at SODA 2000.

[27] R. P. Stanley. *Enumerative Combinatorics*, volume 1. Wadsworth & Brooks/Cole, 1986.

[28] K. Tan. On the role of partition inequalities in classical algorithms for Steiner problems in graphs. Master's thesis, University of Waterloo, 2006.

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

[30] D. Warme. A new exact algorithm for rectilinear Steiner trees. In P. Pardalos and D.-Z. Du, editors, *Network Design: Connectivity and Facilities Location: DIMACS Workshop April 28-30, 1997*, pages 357–395. American Mathematical Society, 1997. Preliminary version appeared at ISMP 1997.

[31] D. Warme. *Spanning Trees in Hypergraphs with Applications to Steiner Trees*. PhD thesis, University of Virginia, 1998.

[32] 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.

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