Polyhedral and Algorithmic Methods in Network Connectivity

Michael (Mik) Zlatin

April 5, 2024

Abstract

This dissertation contains five chapters, each addressing an algorithmic or polyhedral problem in network connectivity and combinatorial optimization.

The first three chapters consider problems, both classical and novel, in the field of network connectivity augmentation. In these problems, we seek to augment a given graph by making it more resilient to edge failures. The latter two chapters address problems related to Set Cover polyhedra.

In the first and second chapters, we consider connectivity augmentation in the "Steiner" setting. In this setting, we are only interested in the connectivity between a specified set of terminal nodes, rather than the global connectivity of the network. In chapter one, we define the Steiner Tree Augmentation Problem, the natural generalization of the Tree Augmentation Problem to this setting. We employ a relative greedy algorithm to give a $1 + \ln(2) + \varepsilon$ approximation, the first algorithm for this problem with approximation ratio below 2. We then show how to use the local search methodology of Traub and Zenklusen to bring down the approximation ratio to $1.5 + \varepsilon$.

In chapter two, we define the more general Steiner Connectivity Augmentation Problem (k-SCAP): given a Steiner k-edge-connected graph, augment it so that the Steiner edge-connectivity increases to $k + 1$. We also define the k-Steiner Augmentation of a Graph problem (k-SAG), which is a special case of k-SCAP but still generalizes the global connectivity augmentation problem. We employ a relative greedy technique to give a $1 + \ln(2) + \varepsilon$ approximation for k-SCAP when $k = 2$, and a $1.5 + \varepsilon$ approximation for k-SAG for any k.

In the third chapter, we consider the well-known Tree Augmentation Problem (TAP) where the goal is to increase the edge-connectivity of a tree by 1. We take a polyhedral perspective, and show that the integrality gap of a linear relaxation for TAP (known as the ODD-LP) has integrality gap upper bounded by $(2 - \frac{1}{2^{k-1}})$ for depth-k trees. For constant-depth trees, this is the first known linear relaxation for TAP with gap less than 2 which can still be optimized over in polynomial time. The proofs also yield approximation algorithms with matching guarantees.

In chapter four, we turn to a matroidal generalization of the Tree Augmentation Problem, which we call the Base Augmentation Problem. In the Base Augmentation Problem, the goal is to add elements to a given base of a matroid so that we obtain a set which remains full rank after any of its elements are deleted. The Tree Augmentation Problem arises when the matroid is graphic. We characterize the approximability of the Base Augmentation Problem for several common classes of matroids, showing that it is Set-Cover hard for Transversal and Binary matroids, but polynomial time solvable for Laminar Matroids. We finish this chapter with a Conjecture on the complexity of the Base Augmentation Problem for regular matroids.

In the final chapter, we consider a problem related to packing and covering in directed graphs. Woodall conjectured in 1976 that in any digraph, the minimum cardinality of a directed cut is equal to the maximum number of pairwise arc-disjoint directed joins. In this chapter, we describe progress towards resolving this question. In particular, we reduce the problem to a class of bipartite, "semi-regular" instances and use this to show the existence of an admissible dijoin, and various special cases of Woodall's conjecture depending on the "balancedness" of the digraph.

Contents

Chapter 1

Introduction

In this introduction, I will introduce the necessary background to put the results of this thesis into context, and explain why these results are important. I will also explain some basic results which may be of independent interest to those who are new to the field.

My aim is to write the introduction in such a way that a reader can come away with an understanding of all the results proved in this thesis just by reading the introduction. Those interested in proofs can find them in the referenced chapters. Let's jump right in.

1.1 Approximation Algorithms for Network Design

Many fundamental problems in combinatorial optimization stem from applications to the efficient design of networks. These problems have a wide range of applications from transportation networks to communication networks to supply chain networks, and many more.

However, the combinatorial optimization problems which arise in these applications are usually NP-hard, meaning that designing a fast, exact algorithm is highly implausible. There are many ways around this barrier, both theoretically and from a practical perspective. A common theme is the design of *approximation algorithms* - fast algorithms which return a (potentially) suboptimal feasible solution whose cost is nevertheless bounded with respect to the optimum.

Many problems considered in this thesis pertain to a specific measure of the resilience of a network known as "edge-connectivity". Given a graph $G = (V, E)$, the edge-connectivity $\lambda_G(u, v)$ between a pair of vertices u and v in V, is the maximum number of pairwise edge-disjoint $u - v$ paths in G. Equivalently, due to Menger's Theorem, $\lambda(u, v)$ is the cardinality of the minimum edge cut which separates u from v.

With this notion of network resilience, we can define a general class of network design problems for which many classical problems in network design are a special case. The following is known as the general Survivable Network Design Problem (SNDP).

Problem 1.1.1 (Survivable Network Design Problem)**.** Given a graph $G = (V, E)$ with non-negative costs on edges $c : E \to \mathbb{R}_{\geq 0}$ and integers $r_{uv} \in \mathbb{Z}$ for each $u, v \in V \times V$, the goal is to select a set of edges $F \subseteq E$ of minimum cost so that the graph $H := (V, F)$ satisfies $\lambda_H(u, v) \geq r_{uv}$ for all pairs of vertices u, v.

This problem captures many classical problems in network design. For example:

- Suppose $r_{uv} = 1$ for all $u, v \in V \times V$. This is the Minimum Spanning Tree (MST) problem, which can be solved optimally in polynomial time.
- Suppose $r_{uv} = k$ for all pairs $u, v \in V \times V$. This is the (Weighted) Minimum k-edge-connected Spanning Subgraph problem (min k-ECSS). When $k \geq 2$, the best known approximation ratio for this problem is 2.
- Suppose $R \subseteq V$, and $r_{uv} = 1$ for all $u, v \in R \times R$. Then this problem is the Minimum Steiner Tree problem. The current best approximation ratio for Steiner tree is $\ln(4) < 1.39$.
- Similar to the above problem, let $R \subseteq V$, and $r_{uv} = k$ for all $u, v \in R \times R$. This is the Minimum k-edge-connected Steiner subgraph problem. If $k \geq 2$, the current best approximation ratio for this problem is 2.
- Suppose that $r_{uv} \in \{0, 1\}$ for all $u, v \in V \times V$. This is the Minimum Steiner Forest problem, for which the current best approximation algorithm has ratio 2.

You may notice that some of the problems in the above list have an approximation ratio of 2, while for others, we can do better. However, there are no cases of SNDP with approximation ratio greater than 2. This is because in 2001, Kamal Jain used iterative rounding to give a 2 approximation for general SNDP [Jai01]. Consider the following mathematical program, which is the natural linear relaxation of an integer program capturing SNDP.

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & & \\ \text{s.t} \quad & \sum_{e \in \delta(S)} x_e \geq \max_{\{u \in S, v \notin S\}} r_{uv}, & & \text{for all } S \subseteq V \\ & x_e \geq 0, & & \text{for all } e \in E. \end{aligned}$$

Let the polyhedron defined above be denoted by P. Jain showed the following.

Lemma 1.1.2 ([Jai01])**.** *Let x^* be an extreme point of P. Then there is some $e \in E$ for which $x^*_e \geq \frac{1}{2}$.*

The above Lemma is enough to employ an "iterative rounding" algorithm. Namely, we can find an extreme point of P using linear programming, then round the largest edge up to a value 1. We then re-solve the LP to find an extreme point of the residual SNDP problem and repeat. Since P can be optimized in polynomial time, this yields the following.

Corollary 1.1.3. *There is a polynomial time 2-approximation algorithm for SNDP.*

It is perhaps surprising that even highly specialized cases of SNDP (such as when $r_{uv} = 2$ for all pairs), has the same complexity as general SNDP, as far as we know. This leads to a major question in the field of network design.

Question: For which cases of SNDP can we achieve an approximation ratio below 2?

As we shall see, there have been major advances in our understanding of this question in recent years.

1.1.1 Connectivity Augmentation Problems

An important subclass of network design problems are *network augmentation* problems. In these problems, a given network structure already exists and we seek to augment it by adding additional edges or vertices (of minimum cost) to improve the network in some way - typically by increasing its robustness to failures, reducing its diameter, or increasing the amount of flow it can support between supply and demand nodes.

As we discussed, one important measure of network robustness is the edge-connectivity between pairs of nodes in the network. If $\lambda(u, v) \geq k$ for all pairs of vertices u and v in a graph G, we say that G is (globally) k-edge-connected. Notice that if G is k-edge-connected, then it can sustain failures of up to $k-1$ edges without being disconnected.

The problem of augmenting the edge-connectivity of a given network by 1 is called the (Weighted) Connectivity Augmentation Problem (CAP).

Problem 1.1.4 (k-Connectivity Augmentation Problem)**.** Given a k-edge-connected graph $G = (V, E)$, and collection of links $L \subseteq V \times V$ with non-negative costs $c : L \to \mathbb{R}_{\geq 0}$. Return a subset $F \subseteq L$ of minimum cost so that $G' := (V, E \cup F)$ is $(k+1)$-edge-connected.

Notice that CAP is a special case of SNDP. Indeed, we can set $r_{uv} = k+1$ for all pairs and ensure that there is a k-edge-connected graph H of cost 0 in the input to SNDP.

One can also think of the CAP problem as the problem of "covering" the minimum-cuts in a network with links of varying costs. Interestingly, because of the properties of minimum cuts, the k-CAP problem reduces to two different problems depending on the parity of k.

When G is a tree (implying $k = 1$), the k-CAP problem is called the Tree Augmentation Problem (TAP). A cactus is a graph in which every edge is contained in exactly one cycle. When G is a cactus (implying $k = 2$), the k-CAP problem is called the Cactus Augmentation Problem (CacAP).

The following proposition follows from the results of Fleiner and Frank on structure of minimum cuts in a network [FF09].

Proposition 1.1.5. *If k is odd, then k-CAP is equivalent to TAP. If k is even, then k-CAP is equivalent to CacAP.*

Both TAP and CAP have simple 2-approximations, with the first 2-approximation for TAP dating back to 1981, due to Frederickson and Jaja [FJ81]. The TAP problem was originally introduced some years earlier by Eswaran and Tarjan, who showed that it is NP-complete [ET76]. Since then, TAP has received much study, and improved algorithms were obtained in various special cases, such as when the tree has constant diameter [Nut10], when the costs are bounded [Adj18] [FGKS18], or when there is some lower bound on smallest non-zero value in an LP solution [IR22].

After around four decades of active study, the first better-than-2 approximation for general weighted TAP was given by Traub and Zenklusen in [TZ22b]. They use a relative greedy algorithm to give a $1 + \ln(2) + \varepsilon$ approximation. As discussed earlier, this yields the same guarantee for the k-CAP problem as long as k is odd. They subsequently improve their approximation ratio down to $1.5 + \varepsilon$ in [TZ22c]. Finally, in [TZ22a], they give a matching guarantee of $1.5 + \varepsilon$ for k-CAP for any k. Thus, k-CAP was at last added to the short list of SNDP problems for which a better-than-2 approximation is known.

1.1.2 Beyond Global Connectivity

It is important to note that for the above network augmentation problems, the parameter of interest is the *global* edge-connectivity of the graph. However, in many applications, we are not interested in connectivity between all nodes in the graph, but rather the connectivity only between a specified subset of "important" nodes called terminals. In this case, intermediary "Steiner" vertices are used in the network simply to help establish connectivity between these terminal nodes. Consider Figure 1.1. There are four terminal vertices arranged in a unit square. To connect these terminals using only direct links, the cheapest solution is to purchase the three links of cost 1 in any minimum spanning tree. However, with the introduction of Steiner vertices, we can create a cheaper network which still maintains connectivity amongst the terminals.

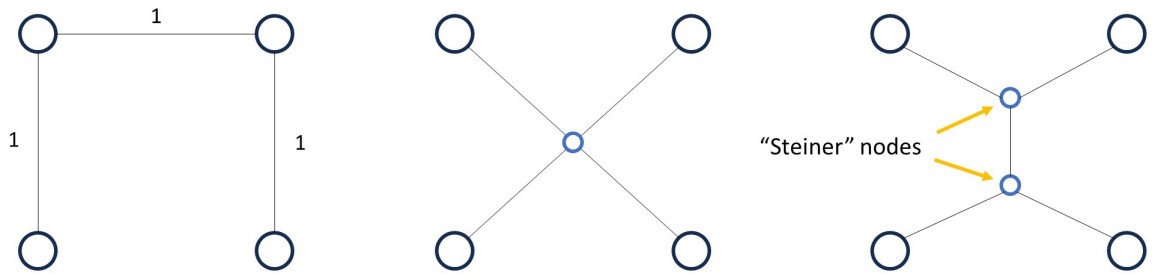

Figure 1.1: The four large circles are terminals arranged in a unit square in the plane. The smaller circles are Steiner vertices. While the cost of the left network is 3, the middle network has a reduced cost of $2\sqrt{2} \approx 2.83$ *and the right network has cost* $\frac{4}{\sqrt{3}} + (1 - \frac{1}{\sqrt{3}}) \approx 2.73$.

Hence, a given network of interest may involve many Steiner nodes. We may want to increase the edge-connectivity between terminal nodes, but we don't necessarily care about augmenting the connectivity of Steiner vertices. These non-global connectivity requirements bring forth new algorithmic challenges and necessitate the development of novel techniques to handle them.

We define two related Steiner generalizations of the Connectivity Augmentation Problem below.

Problem 1.1.6 (k-Steiner Connectivity Augmentation Problem)**.** We are given a graph $G = (V, E)$ and a subset of terminals $R \subseteq V$ such that $\lambda_G(u, v) \geq k$ for all $u, v \in R \times R$. We also have a collection of links $L \subseteq V$ with costs $c : L \to \mathbb{R}_{\geq 0}$. The goal is to choose $F \subseteq L$ of minimum cost so that the graph $G' = (V, E \cup F)$ has $\lambda_{G'}(u, v) \geq k + 1$ for all $u, v \in R \times R$.

The Steiner Connectivity Augmentation Problem (k-SCAP) is perhaps the most natural problem in Steiner connectivity augmentation. It simply asks for the cheapest way to augment the edge-connectivity between the terminals in a graph from k to $k + 1$. Notice that if $R = V$ in the above problem, we recover the standard CAP problem.

The second problem we define is easier than k-SCAP, while still generalizing CAP. Recall that in the standard CAP problem, we can only add links between pairs of vertices in the given graph. What if we are allowed to use intermediary Steiner nodes during the augmentation process? This may allow us to obtain cheaper augmentations. To capture this, we define the Steiner Augmentation of a Graph problem (k-SAG) as follows.

Problem 1.1.7 (k-Steiner Augmentation of a Graph)**.** We are given a k-edge-connected graph $H = (R, E)$ on terminals R, and a collection of Steiner nodes S. Denote the set of all vertices by

$V := R \cup S$. We have a collection of links $L \subseteq V$ with costs $c : L \to \mathbb{R}_{\geq 0}$. The goal is to choose $F \subseteq L$ of minimum cost so that the graph $G' = (V, E \cup F)$ has $\lambda_{G'}(u, v) \geq k+1$ for all $u, v \in R \times R$.

The nuances that distinguish these two problems can be subtle. Consider the pair of examples in Figure 1.2 in which we seek 4-edge-connectivity between the grey terminal nodes. Notice that in the left figure, the graph H to be augmented is already globally 3-edge-connected. The white Steiner vertices outside H can be utilized to achieve a cheaper augmentation, but are not embedded in the original graph. In contrast, the right figure presents a network which crucially uses Steiner vertices to establish 3-edge-connectivity between its terminals. We want to augment this network (possibly using additional Steiner vertices) to increase the connectivity between its terminals. Feasible solutions are shown in blue.

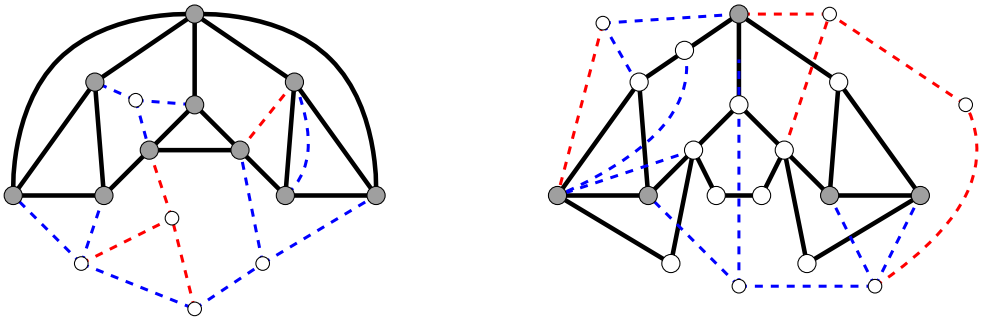

Figure 1.2: A 3-SAG instance is shown on the left, and a 3-SCAP instance is shown on the right. The shaded nodes are terminals R, the black edges denote the edges of E and the dashed edges represent the links L. In both pictures, the blue dashed links form a feasible solution.

Because these problems are special cases of SNDP, they both admit 2-approximations. *One of the major results of this thesis is that one can achieve better-than-2 approximations for k-SCAP when $k \in \{1, 2\}$.* Recall that, in global connectivity augmentation, these cases are sufficient to prove a claim for *any* k. However, the structure of cuts to be covered is much more complicated in the Steiner setting, and these reductions no longer hold. Despite this, we *are* able to achieve a better-than-2 approximation for the k-SAG problem for any k.

More details can be found in Chapters 2 and 3 where we prove these results. For the sake of conciseness, I summarize the above discussion and the results of these chapters in the following theorem.

Theorem 1.1.8 ([RZZ23] [HZ23])**.** *For any $\varepsilon > 0$, there is a polynomial time $(1.5+\varepsilon)$-approximation for 1-SCAP. There is a polynomial time $(1 + \ln 2 + \varepsilon)$-approximation for 2-SCAP. There is a polynomial time $(1.5 + \varepsilon)$-approximation for k-SAG for any k.*

The paper [RZZ23] is based on joint work with R. Ravi and Weizhong Zhang. The paper [HZ23] is based on joint work with Daniel Hathcock.

1.1.3 Integrality Gaps for TAP

The polyhedral approach has been a major theme in the development of improved algorithms for network design problems, and for combinatorial optimization in general. Often, these two areas go hand in hand, with a better understanding of the polyhedral structure of a problem leading to improved algorithms and vice-versa.

We now return to one of the most basic problems in connectivity augmentation: that of augmenting a tree to be a 2-edge-connected graph. In the search for a improved approximations for the Tree Augmentation Problem, an important and related goal is to find a linear relaxation for TAP with integrality gap less than 2. The existence of such a polytope, coupled with an associated rounding algorithm would be enough to breach the barrier of 2 for TAP. Of course, in order to yield an efficient algorithm, we also need to be able to optimize over this linear relaxation in an efficient way.

However, while the recent improvements of Traub and Zenklusen [TZ22b] [TZ22c] for TAP achieve improved approximation ratios, they shed no light on the question of integrality gaps. Loosely speaking, this is because their algorithms involve a Dynamic Programming step which does not lend itself to comparison to an optimal fractional solution.

Consider the most basic set-covering linear relaxation for TAP with tree $T = (V, E)$, links L and costs c. There is a variable for each link, and the constraints encode that that every 1-cut in the tree must be covered by a link we purchase in our solution. For a link $\ell = (u, v)$, let $P_\ell \subseteq E$ be the unique path in T from u to v.

$$\begin{aligned} \min \quad & \sum_{\ell \in L} c_\ell x_\ell & \\ \text{s.t} \quad & \sum_{\ell : e \in P_\ell} x_\ell \geq 1, & \text{for all } e \in E \\ & x_\ell \geq 0, & \text{for all } \ell \in L. \end{aligned}$$

Denote the polyhedron defined by the above program by P_{cut}. The integrality gap of P_{cut} is known to be at most 2 (this follows, e.g. from Jain's result [Jai01]). It is also at least $\frac{3}{2}$, due to a family of examples given in [CKKK08].

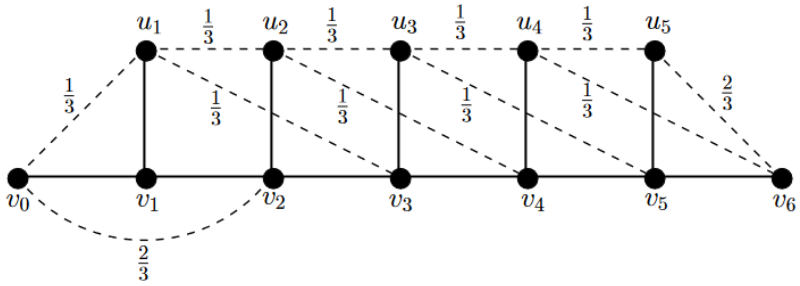

Figure 1.3: A $k = 5$ instance from the family of instances which demonstrates that the integrality gap of P_{cut} is at least $\frac{3}{2}$. The dashed edges represent links, all of which have unit cost. The fractions represent a feasible fractional solution of cost $\frac{2k}{3} + 1$, while the optimal integral solution has cost $k + 1$. This shows a gap of at least $\frac{3k+3}{2k+3}$. This diagram is from [CKKK08].

Hence, as far we currently know, the integrality gap of P_{cut} may be exactly 2. However, strengthened formulations for TAP have also been considered. In [FGKS18], the so-called ODD-LP for TAP is introduced. Similar to the cut-LP, there is a variable for each link $\ell \in L$, and all constraints which are present in the cut-relaxation remain. But now, an exponential number of constraints have been added corresponding to every odd subset of tree edges.

Denote the polyhedron defined by the following linear program as P_{odd}.

$$\begin{aligned}
\min \quad & \sum_{\ell \in L} c_\ell x_\ell & & \\
\text{s.t} \quad & \sum_{\ell : e \in P_\ell} x_\ell \geq 1, & & \text{for all } e \in E \\
& \sum_{\ell \in \delta(S)} x_\ell + \sum_{e \in \delta(S) \cap E} \left(\sum_{\ell : e \in P_\ell} x_\ell \right) \geq |\delta(S) \cap E| + 1, & & \text{for all } S \subseteq V \text{ with } |S \cap E| \text{ odd} \\
& x_\ell \geq 0, & & \text{for all } \ell \in L.
\end{aligned}$$

Because inequalities have been added, we have $P_{odd} \subseteq P_{cut}$. It is not hard to see that it is in fact a strict subset. Hence, the above system is a stronger formulation than the cut-LP for TAP.

The ODD-LP is derived by including a choice subset of "Chvátal cuts" for P_{cut}. Loosely speaking, Chvátal cuts are inequalities that may be added to a formulation while preserving integer solutions, and, if enough rounds of Chvátal cuts are performed, one recovers the convex hull of its integer points. Of course, this is not necessarily algorithmically useful, since the strengthened formulation may be intractable to solve.

However, in the case of the ODD-LP, the authors of [FGKS18] show that one can optimize a linear function over this polyhedron in polynomial time, despite the exponentially many constraints. They do this by providing an efficient separation oracle, which allows one to run the ellipsoid algorithm by generating violated constraints on the fly. They also show that the integrality gap of the ODD-LP is 1 on a subclass of TAP instances (those without "in-links").

In Chapter 4, we leverage this result to prove an upper bound on the integrality gap of the ODD-LP for any TAP instance, depending on the depth of the tree to be augmented. This is the first result which yields a formulation for TAP with integrality gap below 2, at least for instances of constant depth. In particular, we show the following theorem.

Theorem 1.1.9 ([PRZ22]). *The integrality gap of P_{odd} is at most $(2 - \frac{1}{2^{k-1}})$ on TAP instances of depth k. Furthermore, a solution with this approximation ratio can be obtained in polynomial time for any fixed k.*

This is based on joint work with Ojas Parekh and R. Ravi.

1.2 Integrality Gaps for Set Cover

Given a $m \times n$ matrix A with entries in $\{0, 1\}$, and a cost vector $c \in \mathbb{R}^n_{\geq 0}$, the Set Cover problem is:

$$\min\{cx : Ax \geq 1, x \in \{0, 1\}^n\}. \tag{1.1}$$

The Set Cover problem is so ubiquitous that it seems to require no introduction. Versions of this problem appear in myriad algorithmic decision-making tasks.

Unfortunately, in its full generality, the Set Cover problem is NP-complete. It's approximability is also very well understood. The best approximation ratio for this problem $O(\log m)$, which can be

obtained using a variety of classical methods in approximation algorithms, from greedy algorithms, to randomized rounding, to local search.

Consider the greedy protocol for weighted Set Cover, which iteratively picks the remaining set which covers the greatest number of uncovered elements relative to its cost. It is a standard homework exercise to show that this algorithm returns a feasible solution whose cost is at most H_m times the cost of the optimal cover, where H_m is the m^{th} Harmonic number. In fact, a tighter analysis shows that the approximation ratio of the greedy algorithm is exactly $\ln m - \ln \ln m + \theta(1)$ [Sla96]. In a series of works beginning with Feige [Fei98], it is now known that unless P = NP, there is no algorithm which obtains an approximation ratio of $(1-\varepsilon)\ln n$ for general Set Cover in polynomial time [DS14]).

However, because the Set Cover problem is so prevalent, it has attracted a much more fine-grained interest in its complexity under more structured circumstances. For example, it is no coincidence that the network design problems we study in Chapters 2, 3, and 4 of this thesis are all network *augmentation* problems. These problems can all be framed as instances of the Set Cover problem which arise from covering collections of cuts in a graph.

Consider the integer programming formulation in (1). By relaxing the integrality constraints on the variables, we obtain a significantly more tractable linear program. When do the optimal values of these two programs coincide? More generally, what can we we bound the gap between these two optimal values? These are very deep questions. In Chapters 5 and 6, we focus on two specific topics relevant to this context.

1.2.1 The Base Augmentation Problem

Recall that the Tree Augmentation Problem (TAP) is the problem of adding a minimum cost set of links to a tree so that it becomes 2-edge-connected. One apparent motivation for TAP is the desire to obtain a network which is robust to the failure of any single edge, in the sense that the graph will remain connected after this failure. In this section, we introduce the Base Augmentation Problem (BAP), a matroidal analogue of TAP, and show how it corresponds to some interesting classes of the Set Cover problem.

Suppose that $M = (E, \mathcal{I})$ is a matroid with ground set E and independent sets $\mathcal{I}$. Recall that the rank function $r_M : 2^E \to \mathbb{Z}_{\geq 0}$ indicates the size of the largest independent set which can be found within a set $S \subseteq E$. The rank of the matroid is defined as $r_M(E)$ and is denoted as r.

We seek to obtain a set which is somehow "robust" to the failure of any single element. To formalize this, we say that a is subset $S \subseteq E$ is 1-**robust** if $r_M(S \setminus e) = r$ for all $e \in S$. Notice that in order to be 1-robust, S itself must be rull rank. Hence, this definition essentially states that S is 1-robust if it is still a full rank set even after any single element is removed.

Now, we can define the Base Augmentation Problem.

Problem 1.2.1 (Base Augmentation Problem)**.** We are given a base $B \subseteq E$ of a matroid $M = (E, \mathcal{I})$, and a cost function $c : (E \setminus B) \to \mathbb{R}_{\geq 0}$. The goal is to choose a minimum cost set $F \subseteq E \setminus B$ so that $B \cup F$ is a 1-robust set.

Notice that if M is a graphic matroid, then a base of M corresponds to a spanning tree of the associated graph. Furthermore, a 1-robust set in this matroid exactly corresponds to a 2-edge-connected subgraph of G. Hence, the TAP problem is exactly captured by BAP for graphic matroids.

While TAP has been intensely studied, we are interested in characterizing the approximability of the BAP problem on other matroids as well. Furthermore, as we soon describe, the BAP problem can be modeled as an instance of Set Cover. By studying the integrality gaps of the associated linear relaxations we uncover interesting connections to Network matrices and total unimodularity.

The BAP problem is a set covering problem. At a high level, it can be seen as the problem of covering the elements of B with the fundamental circuits C_e which result from adding e to B. Let us formally define the constraint matrix. For a matroid M with base B, the base-representation matrix A is the $r \times (|E| - r)$ matrix whose rows are indexed by the elements of B, and whose columns are indexed by the elements of $E \setminus B$ such that $A_{e,f} = 1$ if $e \in C_f$ and 0 otherwise.

Then the BAP problem with matroid M and base B is exactly captured by the following integer program, where A is the base-respresentation matrix of M with respect to B.

$$\min\{cx : Ax \geq 1, x \in \{0,1\}^{(|E|-r)}\}.$$

We seek to understand the integrality gap of the above program for different matroids. We first show that for both binary and tranversal matroids, the BAP problem is actually general enough to capture arbitrary Set Cover instances. In other cases, such as for laminar matroids and their duals, we show that the above formulation is exact, hence BAP can be solved in polynomial time on these matroid classes.

Of course, when M is graphic, we know from the results on TAP that the integrality gap is at most 2. Indeed, in this case, one can show that the constraint matrix is a $\{0,1\}$ matrix which is signable[1] to be a Network matrix. When M is co-graphic, we show that the BAP problem becomes another well-studied problem called Multi-cut on Trees. The constraint matrix then corresponds to a matrix which is signable to be the transpose of a Network matrix. Again, one can show that these matrices have an integrality gap of at most 2 for Set Covering instances.

Theorem 1.2.2. *The integrality gap of the BAP problem is at most 2 for graphic and co-graphic matroids. For laminar matroids and their duals, the integrality gap is 1. For transversal and binary matroids, it is the same as that for general Set Cover, namely* $O(\log r)$.

The results on graphic and co-graphic matroids motivated us to study the BAP problem on *regular* matroids. Regular matroids are linear matroids which can be represented by a matrix over any field. However, an equivalent definition is that their base representation matrices are signable to be totally unimodular. Seymour showed that every totally unimodular matrix can be obtained by combining Network matrices, their transposes, and the two sporadic R_{10} matrices using the 1,2, and 3-sum operations [Sey80]. An analogous statement can easily be seen to hold for matrices which are signable to be totally unimodular.

As mentioned, both Network-signable matrices and their transposes have integrality gap at most 2 for Set Covering problems. Furthermore, it is easy to check that the two R_{10} matrices have integrality gaps at most 2. The two instances of R_{10} are displayed below:

[1] Signing a $\{0,1\}$-matrix is multiplying some subset of its entries by -1.

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Hence, we finish the chapter with a conjecture. If true, this conjecture would have many interesting implications even outside of the BAP context.

Conjecture 1.2.3. *If A is an $m \times n$ is a TU-signable matrix, then the integrality gap of*

$$\min\{cx : Ax \geq 1, x \in \{0,1\}^n\}$$

is at most 2. Equivalently, the integrality gap of the BAP problem for regular matroids is 2.

This is based on joint work with Madhusudhan Reddy Pittu.

1.2.2 Woodall's Conjecture

Dual to set covering problems are set packing problems. When both a set covering problem and its dual have integral linear relaxations, we obtain a min-max relation. Some of the most beautiful and classical theorems in combinatorial optimization are of this form, including the Max-flow Min-cut theorem and Edmond's branching theorem.

However, there is one min-max relation that has remained a question mark for almost 50 years. Posed in 1976, Woodall's conjecture posits that in any directed graph, the minimum size of a directed cut is equal to the maximum number of disjoint "dijoins".

Let us define these terms.

Definition 1.2.4. Given a directed graph $D = (V, A)$, a **dicut** is a cut $\delta^+(U) \subseteq A$ for some nonempty proper subset $U \subseteq V$ such that $\delta^-(U) = \emptyset$. A **dijoin** is a set of arcs $J \subseteq A$ whose contraction yields a strongly connected digraph.

A good way to think about a dicut is a certificate that the digraph is *not* strongly connected (there can be no path from a vertex outside the dicut shore to a vertex within it). In fact, the converse is also true: a directed graph is strongly connected if and only if it contains no dicut.

Since contracting a dijoin yields a strongly connected digraph, for an arcset $J \subseteq A$ to be a dijoin it is necessary and sufficient for its contraction to eliminate all dicuts. This yields the following two propositions.

Proposition 1.2.5. *Let $J \subseteq A$ be a dijoin and let $F \subseteq A$ be a dicut. Then $|J \cap F| \geq 1$.*

Proposition 1.2.6. *An arcset $J \subseteq A$ is a dijoin if and only if J intersects every dicut at least once. Similarly, an arcset $F \subseteq A$ is a dicut if and only if it intersects every dijoin at least once.*

Now suppose that $D = (V, A)$ is a fixed digraph. We define two parameters: let $\tau(D)$ be the minimum cardinality of a dicut of D, and let $\nu(D)$ be the maximum number of pairwise arc-disjoint dijoins.

In other words, $\tau(D)$ is defined as the optimal solution to the following integer program, with variables x_a for each arc $a \in A$:

$$\tau(D) = \min\{x \cdot \mathbf{1} : x(J) \geq 1 \text{ for all dijoins } J \subseteq A, \quad x \in \{0,1\}^{|A|}\}$$

Similarly, we can define ν with the following program whose variables correspond to the dijoins of D.

$$\nu(D) = \max\{1 \cdot y : \sum_{J \ni a} y_J \leq 1 \text{ for each arc } a \in A, \quad y \in \{0,1\}^{|A|}\}$$

Woodall's conjecture states that for any digraph, the optimal values of these two programs coincide.

Clearly, there is a dual relationship between the values $\tau(D)$ and $\nu(D)$. Consider the linear relaxations of both programs, and let their optimal values be denoted $\underline{\tau}(D)$ and $\bar{\nu}(D)$ respectively. Since these are relaxations, we have $\tau(D) \geq \underline{\tau}(D)$ and $\nu(D) \leq \bar{\nu}(D)$. Furthmore, by linear programming duality, we have $\underline{\tau}(D) = \bar{\nu}(D)$. Hence, we have shown that for any digraph, $\tau(D) \geq \nu(D)$. We're halfway to proving Woodall's conjecture already!

Of course, the reverse inequality is much more mysterious. There is surprisingly little understanding of the clutter of dijoins in digraphs. For example, while Woodall's conjecture is true for digraphs with $\tau(D) = 2$, it is open for $\tau = 3$, even on planar digraphs. Just this year, the first constant factor gap was shown [CLR23], demonstrating that in any digraph, we have $\tau(D) \leq 6\nu(D)$.

The starting point of our contribution to this problem is notion of an **admissible dijoin**. Informally, a dijoin is admissible if it can be packed into a digraph with mincut τ, while "leaving room" for the $\tau - 1$ remaining dijoins of an optimal packing. For this to be true, setting the weights on an admissible dijoin J to 0 should result in a digraph with minimum-weight dicut equal to $\tau - 1$. Clearly, if Woodall's conjecture holds, then there is an admissible dijoin. We are able to demonstrate the existence of this "good first step" independently of the conjecture.

Theorem 1.2.7. *Let $D = (V, A)$ be a digraph, and $\tau \geq 3$ the minimum cardinality of a dicut. Then there exists a dijoin J such that for every dicut $\delta^+(U)$, $|\delta^+(U) \setminus J| \geq \tau - 1$.*

With the above theorem in hand, one could hope to select the admissible dijoin into a packing, and then apply the theorem inductively to obtain τ integral dijoins. Unfortunately, after the first step, we are left with a $\{0,1\}$-weighted digraph, to which our theorem does not apply.

However, Theorem 6.0.2 is interesting in that it separates the collection of dijoins from other non-packing instances such as $\mathcal{Q}_6$. The ground set of $\mathcal{Q}_6$ has six elements, corresponding to the six edges of the graph K_4. Its subsets correspond to the four triangles in K_4. Since we can select two edges to hit all triangles, we have $\tau(\mathcal{Q}_6) = 2$, but $\nu(\mathcal{Q}_6) = 1$ since there are no two edge-disjoint triangles in K_4. Since $\tau(\mathcal{Q}_6) \neq \nu(\mathcal{Q}_6)$, we say that $\mathcal{Q}_6$ does not pack.

Yet, the analogue of Theorem 6.0.2 for $\mathcal{Q}_6$ does not hold, since integrally selecting any triangle leaves no room for any other triangle to be chosen, even fractionally. Hence, our result distinguishes dijoins from $\mathcal{Q}_6$ in this way. Furthermore, it is known that $\mathcal{Q}_6$ is a central obstruction to a large class of instances (the so-called "binary clutters"), as shown by Seymour [Sey77].

Let us discuss some of our techniques. On the way to proving Theorem 6.0.2, we develop a "lifting" procedure which takes as input any directed graph D with minimum dicut equal to τ, and outputs a new digraph D' such that if D' satisifes Woodall's conjecture, then so does D. Furthermore, D' has a special structure. It is a digraph in which every node is either a source or sink vertex, and

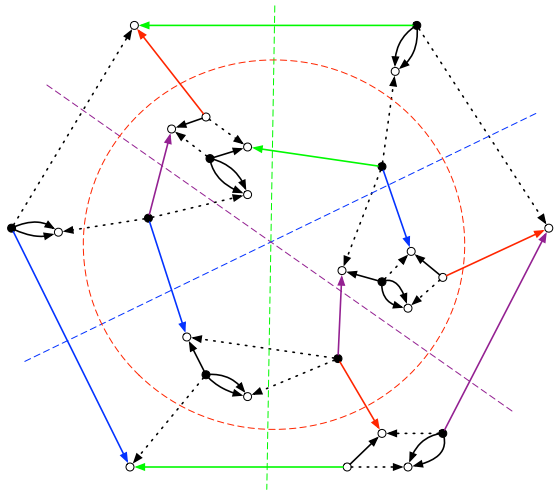

Figure 1.4: The digraph D_{27}: A $(3,4)$-semiregular digraph, in which each node is a source or sink. We have $\tau(D_{27}) = 3$, and four non-trivial (non-minimum) dicuts are colored green, blue, purple and red. The nodes of degree $\tau + 1$ are filled in. An admissible dijoin J is shown with dashed arcs, but the solid arcs cannot be partitioned into two disjoint dijoins. Hence, J cannot be part of an optimal integral packing of dijoins.

all degrees are either τ or $\tau + 1$. We work with these "semi-regular" digraphs and exploit their structure to prove Theorem 6.0.2, among other results.

The lifting procedure works by iteratively "uncontracting" any node which is not a source or sink, and then any node with incorrect degree, while preserving the minimum size of a dicut in the digraph. The uncontraction operation ensures that the presence of τ disjoint dijoins in the lifted graph can be projected down to τ dijoins in the original one.

Of course, our lifting procedure is of independent interest to anyone interested in studying Woodall's conjecture. Our results show that, for the purposes of proving or disproving Woodall's conjecture, it suffices to only consider semi-regular digraphs of the form we described.

Let us provide one more example of how we exploit these more structured instances to prove Woodall's conjecture in some special cases, depending on the "balancedness" of the digraph.

Suppose for a moment that we performed our lifting procedure, and we ended up with a digraph D' such that *all* vertices have degree τ (and none have degree $\tau + 1$). Then we have a τ-regular bipartite digraph, which, when viewed as an undirected graph, admits a packing of τ disjoint perfect matchings. We can show that in this case, each perfect matching is a dijoin, so we have found τ arc disjoint dijoins! Woodall's conjecture is satisfied for this digraph.

In general, the number of vertices of degree $\tau + 1$ in D' is equal to:

$$\sum (\mathrm{imb}(v) \mod \tau : v \in V),$$

where the *imbalance* of a vertex v is defined as $\mathrm{imb}(v) := |\delta^+(v)| - |\delta^-(v)|$, and $a \mod \tau$ is defined as the integer in $\{0, \ldots, \tau - 1\}$ which is congruent to a modulo τ. We call the above expression the *imbalance* of D. As noted above, if D has an imbalance of 0, then Woodall's conjecture holds. We also show that Woodall's holds for digraphs of imbalance τ and 2τ.

Theorem 1.2.8. *Let $D = (V, A)$ be a digraph without a cut-vertex where the minimum cardinality of a dicut is $\tau \geq 3$. If $\frac{1}{\tau} \sum (imb(v) \mod \tau : v \in V) \in \{0, 1, 2\}$, then there exist τ disjoint dijoins.*

Finally, we note that the results in this chapter have been siginificantly expanded, and the full paper, *Packing dijoins in digraphs and weighted digraphs*, has been published in the SIAM Journal of Discrete Math. This is joint work with Ahmad Abdi and Gérard Cornuéjols.

Chapter 2

The Steiner Tree Augmentation Problem

In the Steiner Tree Augmentation Problem (STAP), we are given a graph $G = (V, E)$, a set of terminals $R \subseteq V$, and a Steiner tree T spanning R. The edges $L := E \setminus E(T)$ are called links and have non-negative costs. The goal is to augment T by adding a minimum cost set of links, so that there are 2 edge-disjoint paths between each pair of vertices in R. This problem is a special case of the Survivable Network Design Problem, which can be approximated to within a factor of 2 using iterative rounding [Jai01].

In this chapter, we give the first polynomial time algorithm for STAP with approximation ratio better than 2. In particular, we achieve an approximation ratio of $(1.5 + \varepsilon)$. To do this, we employ the Local Search approach of [TZ22c] for the Tree Augmentation Problem and generalize their main decomposition theorem from links (of size two) to hyper-links.

This is based on joint work with Weizhong Zhang and R. Ravi, and appeared in SODA 2023 [RZZ23].

Formally, the STAP problem is defined as follows.

Problem 2.0.1 (STAP)**.** We are given as input a graph $G = (V, E)$, a set of terminals $R \subseteq V$, and a minimal Steiner tree T spanning R. The edges of G which are not in T are called links and are denoted by L. That is, $L := E(G) \setminus E(T)$. Note that L may have endpoints in $V(T)$ or $V \setminus V(T)$. Finally, we have a cost function $c : L \to \mathbb{R}_{\geq 0}$.

The goal is to augment T to be a 2-edge-connected Steiner subgraph spanning R. That is, we seek $S \subseteq V \setminus R$ and $F \subseteq L$ of minimum cost such that the graph $(V(T) \cup S, E(T) \cup F)$ has two edge-disjoint paths between every pair of terminals. This is equivalent to requiring that $(V(T) \cup S, E(T) \cup F)$ is a 2-edge-connected graph. Thus, we assume in the remainder that $V(T) = R$.

We show the following improved approximation ratio for STAP.

Theorem 2.0.2. *For any $\varepsilon > 0$, there is a $(1.5+\varepsilon)$-approximation algorithm for STAP which runs in polynomial time.*

2.1 Preliminaries

We consider a solution to STAP (S, F) where $S \subseteq V$ and $F \subseteq L$.

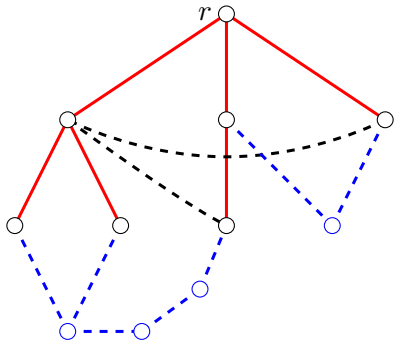

Figure 2.1: An instance of STAP. The red edges are the edges of the given tree. The dashed edges are the links, and the blue links form a feasible solution.

Definition 2.1.1. A **full component** of a STAP solution (S, F), is a subtree of the solution where each leaf is a terminal (that is, a vertex of R), and each internal node is in $V \setminus R$.

It is clear that any STAP solution can be uniquely decomposed into link-disjoint full components. We say that a full component "joins" the terminals that it contains.

Definition 2.1.2. Let (S, F) be a solution to STAP. We say that a set A is **joined** by (S, F) if there is a full component with leaves A.

In fact, the feasibility of a solution is determined only by which sets of nodes it joins.

Definition 2.1.3. We say that a tree edge $e \in E(T)$ is **covered** by a solution (S, F) if e lies on the unique path in the tree P_{uv} between two nodes u and v which are joined by a full component of (S, F).

Lemma 2.1.4. *A solution (S, F) is feasible for STAP iff all tree edges are covered.*

Proof. Suppose all tree edges are covered, and consider any cut in the tree induced by an edge $e \in E(T)$. The cut contains e and since e is covered, some link in F also crosses the cut. Thus, any cut separating terminals has a cardinality of at least 2. This implies two edge-disjoint paths between u and v for every u and v in R. the augmentation is a 2-edge-connected graph.

Conversely, if some edge $e = (u, v) \in E(T)$ is not covered, then the cut e induces in the tree contains no links and there cannot be two disjoint paths between u and v. □

In the WTAP problem, we must choose a set of links to cover all the edges of a given tree. In particular, each link joins exactly two tree vertices. The added difficulty in the case of STAP is that full components may join an arbitrary number of terminals. Thus, we introduce the Hyper-TAP problem as the natural generalization of WTAP to hyper-links, which join arbitrary subsets of tree vertices.

Problem 2.1.5 (Hyper-TAP)**.** In Hyper-TAP, we are given a tree $T = (V, E)$, and a collection of hyper-links $\mathcal{L} \subseteq 2^V$, with non-negative costs c_ℓ for $\ell \in \mathcal{L}$. The goal is to cover the edges of the given tree with the minimum cost subset of hyper-links.

Consider a hyper-link $\ell = \{a_1, \ldots, a_k\}$. We say that the vertices $a_1, \ldots, a_k$ are *joined* by ℓ. After fixing a root of T, denote the least common ancestor of $\{a_1, \ldots, a_k\}$ by $\mathrm{lca}(a_1, \ldots, a_k)$ and define $\mathrm{apex}(\ell) := \mathrm{lca}(a_1, \ldots, a_k)$. Let $P_{a,b}$ be the unique edge path in the tree from a to b.

Let T_ℓ be the subtree of T consisting of the union of all paths between vertices joined by ℓ. Equivalently, $T_\ell := \bigcup_{a \in \ell} P_{a,\text{apex}(\ell)}$. We say that the link ℓ covers the edges in T_ℓ.

Then, the Hyper-TAP problem is the following covering problem:

$$\min_{Z \subseteq \mathcal{L}} \left\{ \sum_{\ell \in Z} c(\ell) : \bigcup_{\ell \in Z} T_\ell = E \right\}.$$

Clearly, Hyper-TAP is an instance of Set Cover. However, they are in fact equivalent. Indeed, given any instance of Set Cover with ground set E and subsets $\mathcal{S}$, we can create an instance of Hyper-TAP in which T is a star, and the edges of T correspond to elements of E. Finally, we can create a hyper-link ℓ for each $S \in \mathcal{S}$ covering exactly the edges corresponding to the elements covered by S.

Thus, we cannot expect to achieve approximation algorithms for general Hyper-TAP better than those for general Set Cover. However, in proving Theorem 2.0.2, we exploit the structure of Hyper-TAP instances that come from instances of STAP to achieve improved approximations in this case.

As noted above, every STAP instance is equivalent to an instance of Hyper-TAP obtained as follows: for each subset of tree vertices $S \subseteq R$, find the cheapest full component joining S, and create a hyper-link ℓ_S joining S with this cost. However, if we allow full components of unbounded cardinality, this reduction cannot be carried out in polynomial time. To perform this reduction efficiently, we restrict the size of the full components we consider.

Definition 2.1.6. We say that a full component is **γ-restricted** if it joins at most γ terminals. We say that a solution to STAP is **γ-restricted** if it uses only γ-restricted full components. Analogously, we say an instance of Hyper-TAP is **γ-restricted** if each hyper-link has size at most γ.

In Section 2.3.3, we show that up to a factor of $(1+\varepsilon)$, it suffices to find the best γ-restricted solution to STAP for some constant $\gamma(\varepsilon)$. This allows us to reduce an instance of STAP to an instance of γ-restricted Hyper-TAP in polynomial time, while only losing a factor of $(1+\varepsilon)$ in the approximation ratio.

2.2 Our Techniques

Our algorithm for (edge-weighted) STAP is a local search algorithm which follows in the vein of the recent improved approximation algorithms for WTAP due to Traub and Zenklusen [TZ22b]. We find it helpful to first describe the relative greedy $(1+\ln(2)+\varepsilon)$-approximation algorithm for STAP, and then show how it can be modified to achieve a $(1.5+\varepsilon)$-approximation.

We begin by briefly describing the methods of [TZ22b]. The relative greedy algorithm for WTAP begins with an initial 2-approximate feasible solution. It then makes local moves to improve the current solution by adding carefully chosen subsets of links and dropping links in the initial solution which are rendered unnecessary for feasibility.

An up-link is a link which joins two nodes having an ancestor-descendant relationship in the tree. The initial 2-approximate solution for WTAP has a special structure: it consists of only up-links and each tree edge is covered exactly once. In the WTAP setting, this structured 2-approximate

solution can be found efficiently by replacing every link with two up-links to their least common ancestor and using dynamic programming to find the best up-links only solution [FJ81].

For a given up-link solution U and a subset of links $C \subseteq L$, $\text{drop}_U(C)$ is the set of up-links in U which can be removed from $U \cup C$ while preserving feasibility. In each iteration, the relative greedy algorithm in [TZ22b] seeks to choose a subset of links C minimizing the ratio between the cost of links in C, and the cost of up-links in $\text{drop}_U(C)$. It then adds these links to the current solution and removes all the links in $\text{drop}_U(C)$.

However, the minimization cannot be done efficiently over all subsets of links. Traub and Zenklusen introduce the notion of a k-thin subset of links, and show that one can compute the minimizer over all k-thin subsets in polynomial time using dynamic programming. Thus, these subsets of links are simple enough so that we can efficiently choose the best to add at each iteration.

Crucial for the analysis of the algorithm is the property that, as long as the current solution is expensive, there will always be an improving k-thin subset of links to add. This follows from the main decomposition theorem of Traub and Zenklusen in [TZ22b].

In order to apply the relative greedy approach to the STAP setting, we need to first find an initial structured 2-approximate up-link solution covering each tree edge exactly once. We show how to do this in Section 2.3.1 using Euler tours over the optimal solution components. In particular, we show the following.

Lemma 2.2.1. *Given an instance of STAP, let OPT denote the cost of the optimal solution. There is a polynomial time algorithm which returns a feasible up-link solution $U \subseteq L$, with $c(U) \leq 2OPT$ and where each edge $e \in E(T)$ is covered exactly once.*

Next, we need to prove a decomposition result analogous to the theorem for WTAP. We extend the decomposition theorem for WTAP to arbitrary hyper-links in Section 3.7.

Definition 2.2.2. Let $Z \subseteq \mathcal{L}$ be a collection of hyperlinks. We say that Z is **k-thin** if for each $v \in V(T)$, we have $|\{\ell \in Z : v \in T_\ell\}| \leq k$.

Let P_u be the edges covered by up-link u. Given a set of up-links U and a collection of hyper-links $Z \subseteq \mathcal{L}$, let

$$\text{drop}_U(Z) = \{u \in U : P_u \subseteq \bigcup_{\ell \in Z} T_\ell\}.$$

Theorem 2.2.3 (Decomposition Theorem)**.** *Given an instance of Hyper-TAP $(T, \mathcal{L})$, suppose U is an up-link solution such that the sets P_u are pairwise edge-disjoint for $u \in U$. Suppose $F \subseteq \mathcal{L}$ is any solution. Then for any $\varepsilon > 0$, there exists a partition $\mathcal{Z}$ of F into parts so that:*

- *For each $Z \in \mathcal{Z}$, Z is k-thin for $k = \lceil 1/\varepsilon \rceil$.*
- *There exists $Q \subseteq U$ with $c(Q) \leq \varepsilon \cdot c(U)$, such that for all $u \in U \setminus Q$, there is some $Z \in \mathcal{Z}$ with $u \in drop_U(Z)$. That is, $U \setminus Q \subseteq \bigcup_{Z \in \mathcal{Z}} drop_U(Z)$.*

Following the method of Traub and Zenklusen [TZ22b], these two results are enough to prove that the local greedy algorithm achieves an approximation ratio of $1 + \ln 2 + \varepsilon$. However, in order to attain a polynomial runtime, we cannot afford to search over arbitrary size hyper-links. This is where we use the notion of γ-restricted hyperlinks. We extend the result of Borchers and Du for Steiner trees [BD97], which shows a bounded ratio between optimal γ-restricted Steiner trees and

optimal unrestricted Steiner trees, to the case of STAP. This allows us to efficiently approximately reduce an instance of STAP to an instance of Hyper-TAP, where each hyper-link has size bounded by a constant.

Lemma 2.2.4 (Bounding loss through restriction)**.** *Given an instance of STAP, let OPT be the optimal value and OPT_γ be the optimal value over all γ-restricted solutions. Then for all $\varepsilon > 0$, there exists $\gamma(\varepsilon) = 2^{\lceil \frac{1}{\varepsilon} \rceil}$ such that*

$$\frac{OPT_\gamma}{OPT} \leq 1 + \varepsilon.$$

This ensures that we can generate Hyper-TAP instances with only constant-sized hyper-links, which, in particular, allows for computing the greedy local move used in each iteration of the algorithm in polynomial time.

To prove Theorem 2.0.2, we use the idea of [TZ22c] to convert the relative greedy algorithm into a non-oblivious local search algorithm which achieves a better approximation guarantee. The crux of the improvement is that the local search algorithm is able to leverage the gains by dropping links that are added during the course of the algorithm, rather than just those in the initial up-link solution.

However, the decomposition theorem (Theorem 3.2.3) only applies to up-links. To get around this, we can associate to each link f in a STAP solution F, a set of up-links W_f which are responsible for covering the same tree edges as f. These are called witness sets. Throughout the algorithm, we maintain a feasible STAP solution F and a collection of witness sets W_f for $f \in F$ such that $U := \bigcup_{f \in F} W_f$ is feasible. Whenever we add a collection of links to our solution F, we also add the associated witness sets to U, as well as drop up-links from witness sets which are unnecessary for U to be feasible. Finally, when the witness set of a link f becomes empty, we can remove f from our solution without disrupting feasibility.

The approximation ratio that this method provides is related to maximum size of a witness set. If the maximum size of a witness set is W, it yields an approximation ratio of $H_W + \varepsilon$, where $H_W = \sum_{i=1}^{W} \frac{1}{i}$.

This approach was used in the context of WTAP where the witness set of a link $\ell = (u, v)$ initially consists of the up-links $\{(u, \mathrm{lca}(u, v)), (\mathrm{lca}(u, v), v))\}$, which together cover the same tree edges as the original link ℓ. Since $|W_f| = 2$ this yields an approximation ratio of $\frac{3}{2} + \varepsilon$.

The challenge in the context of STAP is that it is unclear how to associate to each link f a witness set of at most two up-links. However, the proof of Lemma 2.2.1 yields a natural choice for the up-links in each witness set arising from the up-links generated by each subpath along the Euler tour. This choice allows us to prove Theorem 2.0.2.

2.3 An Improved Approximation for Edge Weighted STAP

We prove Theorem 2.0.2 in this section using the outline in Section 2.2. In Section 2.3.1, we show how to achieve a 2-approximation algorithm for STAP which returns a structured solution involving up-links with disjoint coverages. In Section 2.3.2, we define a relative greedy algorithm for STAP which achieves an approximation ratio of $1 + \ln(2) + \varepsilon$. In Section 2.3.3, we show that we can restrict our search for solutions to STAP to γ-restricted ones while only losing a factor of $1 + \varepsilon$ in the approximation ratio. In Section 3.7, we prove the decomposition theorem and in Section 3.8, we describe the dynamic program which allows us to implement our algorithms in polynomial time.

Finally, in Section 2.3.6, we show how to modify the relative greedy algorithm using witness sets to achieve an approximation ratio of $1.5 + \varepsilon$ and prove Theorem 2.0.2.

2.3.1 A Structured 2-Approximation for STAP

Consider an instance of STAP and let (S^*, F^*) be an optimal augmentation. We describe an approximation algorithm for STAP which yields a feasible solution (S, F) of cost at most $2c(F^*)$. This approximation ratio is already achievable using Jain's algorithm for SNDP. However, we show that we can find an approximate solution (S, F) with nice structural properties. In particular, we will have $S \subseteq V(T)$, F consisting of only up-links, and the coverage of each $\ell \in F$ being pairwise disjoint.

We will perform a metric completion step on the instance without loss of generality. For every pair of nodes u and v in R, consider the shortest path from u to v in the graph (V, L). We add a link (u, v) with cost equal to the shortest path length. This does not change the problem since any solution which uses one of the added links may instead use the shortest path instead, paying the same cost.

Next, we perform shadow completion. Let P_{uv} be the path in T between terminals u and v. If there exists a link $\ell = (u, v)$ of cost c, we will also add links $\ell' = (u', v')$ of cost c where $u', v' \in P_{uv}$. Again, this can be done without loss of generality since any solution which uses an added link may be converted to a solution to the original instance of the same cost.

It is easy to see that both of these preprocessing steps can be done in polynomial time.

For the following, we will consider the given tree T to be rooted at an arbitrary node r. See Figure 2.2 for an illustration of the proof of Lemma 2.3.1.

Lemma 2.3.1. *Given any feasible solution (S^*, F^*) to STAP, there exists a solution (S, F) with $c(F) \leq 2c(F^*)$ such that $S \subseteq R$. Furthermore, $F \subseteq L$ involves only up-links.*

Proof. Denote the full components of (S^*, F^*) by $(A_1^*, H_1^*), \ldots, (A_p^*, H_p^*)$. For each full component (A_i^*, H_i^*), we will take an Eulerian tour which traverses each link in H_i exactly twice. This yields an ordering of the terminals in $S^* \cap R$, say $r_1, \ldots, r_k$. Now, let $v_i := \text{lca}(r_i, r_{i+1})$ for $i = 1, \ldots, n$, where $r_{n+1} := r_1$.

We will consider the set of links $F := \{(r_1, v_1), (r_2, v_2), \ldots, (r_k, v_k)\}$. It is clear that F contains only up-links between nodes in R. We will show that $c(F) \leq 2c(F^*)$.

Note that the total cost of $F' := \{(r_1, r_2), (r_2, r_3), \ldots, (r_k, r_1)\}$ is at most $2c(F^*)$ because of the metric completion step, and the fact that the Eulerian tour traverses each link in F^* exactly twice. Furthermore, the link (r_i, v_i) is a shadow of (r_i, r_{i+1}), so it exists and has at most the cost of (r_i, r_{i+1}). Thus, $c(F) \leq c(F') \leq 2c(F^*)$.

We will now show that F is a feasible solution. We want to show that $G' = (R \cup S, E(T) \cup F)$ is a 2-edge-connected graph. It suffices to show that G' is connected after any edge $e \in E(T)$ is deleted.

Thus, we fix some $e \in E(T)$ and consider the cut $(W, \bar{W})$ it induces on the tree T. Notice that since F' is a feasible solution and e must be covered, there must be terminals r_i and r_j in the Euler tour with $r_i \in W$ and $r_j \in \bar{W}$. We assume $i < j$. In fact, since the tour is a cycle and therefore returns to its starting node, we also have a pair of vertices $r_{i'} \in \bar{W}$ and $r_{j'} \in W$ with $i' < j'$.

We claim that either link (r_i, v_i) or link $(r_{i'}, v_{i'})$ covers e. Indeed, if $r \in W$, then both v_i and $v_{i'}$ are in W as well. In this case, link $(r_{i'}, v_{i'})$ covers edge e. Otherwise, if $r \in \bar{W}$, then both v_i and $v_{i'}$ are in $\bar{W}$ and link (r_i, v_i) covers e.

Thus, F is a feasible solution with the desired properties. □

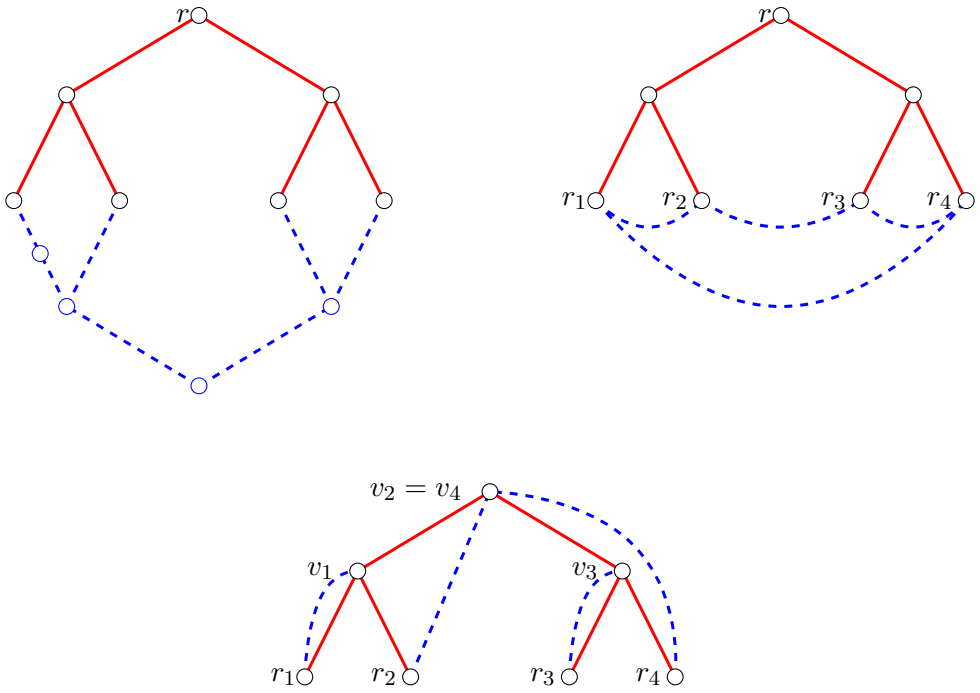

Figure 2.2: An illustration of the existence of a 2-approximate up-link solution for STAP. In the top left picture, the blue dashed edges are a full component of the optimal solution. In the top right picture, $r_1, \dots, r_4$ denotes the ordering of the terminals obtained from an Euler tour of the component in the left picture. Here, the blue links only go between terminals. In the bottom picture, shadows of the links in picture 2 are used to obtain a solution that only uses up-links.

Finally, we will need the following standard result, showing that we can shorten the up-link solution by replacing certain links with their shadows, so that each tree edge is covered exactly once.

Lemma 2.3.2. *Given an up-link solution U, we can in polynomial time find an up-link solution U' with $c(U') \leq c(U)$ and with $|\{\ell \in U' : e \in P_\ell\}| = 1$ for all $e \in E(T)$.*

By Lemma 2.3.1, if the optimal solution to a STAP instance has cost OPT, then there is an up-link solution of cost at most $2OPT$. Since the optimal up-link solution can be easily computed in polynomial time (see e.g. [FJ81]), we obtain Lemma 2.2.1.

2.3.2 Relative Greedy for STAP

We now give the algorithm for STAP which, given any $\varepsilon > 0$, computes a solution F with cost at most $(1 + \ln 2 + \varepsilon)OPT$ and runs in polynomial time. See Algorithm 3.3.

We assume we are given a shadow-complete, metric-complete instance of STAP with graph $G = (V, E)$, tree $T = (R, E(T))$, links $L := E \setminus E(T)$ with costs $c : L \to \mathbb{R}$.

The algorithm uses Lemma 2.2.1 to compute an up-link solution U that has cost at most 2OPT and can be chosen to have P_u disjoint for each $u \in U$.

By Lemma 2.2.4, we may restrict our attention to finding a γ-restricted solution to the STAP instance for sufficiently large γ. Our algorithm now constructs an equivalent instance of γ-restricted

Algorithm 1: Relative greedy algorithm for STAP

Input: A shadow-complete, metric-complete instance of STAP with graph $G = (V, E)$, tree $T = (R, E(T))$, links $L = E \setminus E(T)$, and $c : L \to \mathbb{R}$. Also an $\varepsilon > 0$.

Output: A solution $F \subseteq L$ with $c(F) \leq (1 + \ln(2) + \varepsilon)OPT$.

1. Compute a 2-approximate up-link solution U such that each edge $e \in E(T)$ is covered exactly once (Lemma 2.2.1).
2. Let $\varepsilon' := \frac{\varepsilon/2}{1+\ln 2+\varepsilon/2}$ and $\gamma := 2^{\lceil 1/\varepsilon' \rceil}$.
3. For each $S \subseteq R$ where $|S| \leq \gamma$, compute the cheapest full component joining S and denote the cost by c_S.
4. Create an instance of γ-restricted Hyper-TAP on tree $T = (R, E(T))$ with hyper-links $\mathcal{L} = \{\ell_S : S \subseteq R, |S| \leq \gamma\}$. Set the cost of hyper-link ℓ_S to be c_S.
5. Initialize $F := \emptyset$
6. Let $k := \lceil 4/\varepsilon \rceil$
7. While $U \neq \emptyset$:
   - Compute the k-thin subset of hyper-links $Z \subseteq 2^{\mathcal{L}}$ minimizing $\frac{c(Z)}{c(\text{drop}_U(Z))}$.
   - Let $F := F \cup Z$ and let $U := U \setminus \text{drop}_U(Z)$.
8. **Return** A STAP solution with full components corresponding to the hyper-links in F.

Hyper-TAP. It enumerates over all subsets $S \subseteq R$ of at most γ terminals, and computes the cheapest full component joining S. Notice that there are at most n^γ such sets. Furthermore, for each subset S, we can compute the cheapest full component joining S in polynomial time by solving an instance of Steiner tree with constantly many terminals using the following result.

Lemma 2.3.3 (Dreyfus and Wagner [DW71])**.** *There is an algorithm for Steiner tree which returns the optimal Steiner tree and runs in time $O(n^3 \cdot 3^p)$, where p is the number of terminals.*

For the remainder of the procedure the algorithm works with this Hyper-TAP instance and the previously computed up-link solution U.

We iteratively improve the current solution by finding the best k-thin subset of hyper-links to add. In particular, we find the k-thin subset of hyper-links Z which minimizes the ratio $\frac{c(Z)}{c(\text{drop}_U(Z))}$. This can be done in polynomial time via dynamic programming, see Lemma 2.3.10.

Finally, since at least one up-link is dropped in each iteration of the while-loop, this algorithm runs in polynomial time overall.

We now turn to analyzing the quality of the solution returned. This relies on the Decomposition Theorem 3.2.3 for Hyper-TAP which we prove in Section 3.7.

With this Decomposition Theorem, we can immediately conclude that the relative greedy procedure computes a $(1+\ln 2+\varepsilon)$-approximation of the optimal γ-restricted Hyper-TAP solution by leveraging the results of previous work (See [CN13b],[TZ22b] Theorem 6). This proves our main Theorem 2.0.2.

Proof of Theorem 2.0.2. We prove that for every $\varepsilon > 0$, Algorithm 7.1 returns a solution F of cost at most $(1 + \ln 2 + \varepsilon)OPT$.

As usual, OPT denotes the cost of the optimal solution to the original STAP problem. Let OPT_γ denote the cost of the optimal γ-restricted STAP solution. Notice that by taking $\varepsilon' = \frac{\varepsilon/2}{1+\ln 2+\varepsilon/2}$, and $\gamma = 2^{\lceil 1/\varepsilon' \rceil}$, we have that $OPT_\gamma \leq (1 + \varepsilon')OPT$ by Lemma 2.2.4.

By the Decomposition Theorem 3.2.3, the relative greedy procedure returns a solution F with cost at most $(1 + \ln 2 + \varepsilon/2)OPT_\gamma$.

Hence our overall cost is at most

$$c(F) \leq (1 + \ln 2 + \varepsilon/2)(1 + \frac{\varepsilon/2}{1 + \ln 2 + \varepsilon/2})OPT = (1 + \ln 2 + \varepsilon)OPT.$$

□

2.3.3 Effects of γ-restriction

In this section, we prove that, for any $\varepsilon > 0$, there is a large enough γ so that the cost of the optimal γ-restricted solution to STAP costs at most $(1+\varepsilon)$ times the optimal cost of an unrestricted STAP solution. This is an extension of the result of Borchers and Du [BD97] for Steiner trees.

Recall that a full component of a STAP solution (S, F), is a subtree of the solution where each leaf is a terminal (that is, a vertex in R), and each internal node is in $V \setminus R$. Also, recall that a STAP solution is γ-restricted if each of its full components joins at most γ terminals.

Notice that finding a minimum cost 2-restricted STAP solution is simply the Weighted Tree Augmentation Problem. We show that for γ large enough, the optimal γ-restricted STAP solution is close (in cost) to the optimal unrestricted solution.

First, we recall the following result for Steiner trees. Given a Steiner tree solution, a full component of this solution is a subtree whose leaves are terminals and whose non-leaves are non-terminals. A γ-restricted Steiner tree is a solution whose full components contain at most γ terminals.

Theorem 2.3.4 (Borchers and Du [BD97])**.** *Let $\varepsilon > 0$ and $\gamma = 2^{\lceil \frac{1}{\varepsilon} \rceil}$. Fix any instance of Steiner tree and let T^* be the optimal Steiner tree and T_γ be the optimal γ-restricted Steiner tree. Then we have*

$$\frac{c(T_\gamma)}{c(T^*)} \leq 1 + \varepsilon.$$

Now, we prove an analogous result for STAP. For an instance of STAP, let (S^*, F^*) be the optimal solution and (S_γ, F_γ) be the optimal γ-restricted solution.

Proof of Lemma 2.2.4. Let $\varepsilon > 0$. Then we show that for $\gamma = 2^{\lceil \frac{1}{\varepsilon} \rceil}$, we have

$$\frac{c(F_\gamma)}{c(F^*)} \leq 1 + \varepsilon.$$

Let (S^*, F^*) be the optimal STAP solution. Each of its full components (A_i^*, H_i^*) is a tree joining some set of terminals $R_i \subseteq R$.

Since (S^*, F^*) is optimal, and (A_i^*, H_i^*) is a full component, it must be the cheapest way to join the nodes in R_i. That is, (A_i^*, H_i^*) is an optimal solution to the Steiner tree instance on the graph $(R_i \cup (V \setminus R), L)$, with terminals R_i.

By Theorem 2.3.4 above, there is a γ-restricted Steiner tree solution of cost at most $(1+\varepsilon)c(H_i^*)$.

Applying this to each full component of (S^*, F^*) yields a solution of cost at most $(1+\varepsilon)c(F^*)$ which only has full components joining at most γ terminals. □

2.3.4 The Decomposition Theorem

In this section, we prove our main decomposition theorem (Theorem 3.2.3). In order to prove this result, we will follow the methods of [TZ22b], and extend them from WTAP to Hyper-TAP. At a high level, the argument is as follows. We will build a directed graph D whose vertices correspond to the hyper-links $\ell \in F$. For each up-link $u \in U$, we will choose a minimal set of hyper-links F_u such that $u \in \text{drop}_U(F_u)$. Each set F_u will correspond to a directed path A_u in D.

In [TZ22b], the authors show that based on the choices for the minimal sets F_u, the dependency graph satisfies the following key properties, which allow for the selection $Q \subseteq U$ and a partition of F into the desired k-thin components.

1. The dependency graph is a branching.
2. Let (Z, A) be a connected component of the dependency graph. If the arc set of every directed path in (Z, A) has a non-empty intersection with A_u for at most k up-links $u \in U$, then Z is $(k+1)$-thin.

In [TZ22b], Theorem 5 shows how one can use these two properties to select $Q \subseteq U$ and the partition of F into the desired k-thin components, proving the Decomposition Theorem.

We argue that the same properties hold in the hyper-link setting. The crucial property that allows us to extend the arguments from links to hyper-links is that if u is an up-link and ℓ is a hyper-link, then the intersection of P_u and T_ℓ is a subpath of P_u. Recall that $T_\ell := \bigcup_{a \in \ell} P_{a,\text{apex}(\ell)}$.

We now describe how to construct for each $u \in U$ a minimal set F_u such that $u \in \text{drop}_U(F_u)$. Suppose $u = (t, b)$ where t is an ancestor of b. We define v_u to be the lowest ancestor of t, i.e., the ancestor farthest away from the root r, such that P_u is covered by hyper-links in

$$B_{v_u} := \{\ell \in F : \text{apex}(\ell) \text{ is a descendant of } v_u\}$$

i.e., $P_u \subseteq \cup_{\ell \in B_{v_u}} P_\ell$. Then we choose $F_u \subseteq B_{v_u}$ minimal such that $P_u \subseteq \bigcup_{\ell \in F_u} T_\ell$.

There is a natural ordering on the hyper-links in F_u. First, we make some observations about how each hyper-link interacts with an up-link. Let $P_{u,\ell} := P_u \setminus \cup_{\bar{\ell} \in F_u \setminus \{\ell\}} T_{\bar{\ell}}$.

Lemma 2.3.5. *For any up-link u, let $\ell \in F_u$. Then $P_{u,\ell}$ is non-empty and the edge set of a path.*

For $\ell_1, \ell_2 \in F_u$, we define $\ell_1 \prec \ell_2$ if and only if the edges in P_{u,ℓ_1} appear before the edges of P_{u,ℓ_2} on the $t - b$ path in T.

The arcs of the dependency graph are determined by the orderings on F_u for $u \in U$. For every up-link $u \in U$, let $\ell_1 \prec \cdots \prec \ell_q$ be the links in F_u. Then

$$A_u := \{(\ell_1, \ell_2), \ldots, (\ell_{q-1}, \ell_q)\}$$

The arcs of the dependency graph consist of the union of A_u over all $u \in U$. We will now show that the dependency graph has the two key properties which were introduced in [TZ22b], which allow the selection of $Q \subseteq U$ and a partition of F into the desired k-thin components.

First, we consider property (1). This property simply follows from the minimality of F_u for each $u \in U$ and has been shown in [CN13b].

Lemma 2.3.6 (Cohen and Nutov [CN13b])**.** *The dependency graph D is a node-disjoint collection of arborescences.*

To prove property (2), we need the following lemma, which relies on the particular choice of minimal F_u. Fix any connected component of the dependency graph (Z, A).

Lemma 2.3.7. *Let $\ell_1, \ell_2 \in Z$ with $V_{\ell_1} \cap V_{\ell_2} \neq \emptyset$. Then ℓ_1 and ℓ_2 have an ancestry relationship in the arborescence (Z, A), i.e., either ℓ_1 is an ancestor of ℓ_2 or ℓ_2 is an ancestor of ℓ_1.*

Lemma 2.3.7 has been shown in the context of WTAP in [TZ22b]. The proof extends verbatim to the case of Hyper-TAP, so we don't rewrite it here. Just as in [TZ22b], Lemma 2.3.7, along with the property that F_u is 2-thin for each $u \in U$ (which follows from the minimality of F_u) imply property (2).

Lemma 2.3.8. *Let k be a positive integer, and (Z, A) be a connected component of the dependency graph. If every directed path in (Z, A) intersects at most k sets A_u, then Z is $(k+1)$-thin.*

Thus, the dependency graph satisfies the 2 properties described earlier in the section. We can use the identical technique as in [TZ22b] to select a set of up-links $Q \subseteq U$ with $c(Q) \leq \varepsilon$. We will delete the arc sets A_u for $u \in Q$ from the dependency graph, and the connected components of the remaining digraph will each be $(k+1)$-thin. We reproduce the proof below.

Theorem 2.3.9. *Given an instance of Hyper-TAP $(T, \mathcal{L})$, suppose U is an up-link solution such that the sets P_u are pairwise edge-disjoint for $u \in U$. Suppose $F \subseteq \mathcal{L}$ is any solution. Then for any $\varepsilon > 0$, there exists a partition $\mathcal{Z}$ of F into parts so that:*

- *For each $Z \in \mathcal{Z}$, Z is k-thin for $k = \lceil 1/\varepsilon \rceil$.*
- *There exists $Q \subseteq U$ with $c(Q) \leq \varepsilon \cdot w(U)$, such that for all $u \in U \setminus Q$, there is some $Z \in \mathcal{Z}$ with $u \in drop_U(Z)$. That is, $U \setminus Q \subseteq \bigcup_{Z \in \mathcal{Z}} drop_U(Z)$.*

Proof. Let $k := \lceil \frac{1}{\varepsilon} \rceil$. We will construct an arc labeling for each connected component (Z, A) of the dependency graph. The arcs in the same set A_u will receive the same label.

For each directed path (F_u, A_u) which begins at the root of the arborescence (Z, A), we set the labels of the arcs in this path to be 0. For a directed path (F_u, A_u) which does not begin at the root, let ℓ be its starting node and suppose the arc entering ℓ has label $j \in \mathbb{Z}_{\geq 0}$. We set the labels of arcs in A_u to be $j+1$. Since (Z, A) is an arborescence, this labeling is well-defined.

For $i \in \{0, \ldots, k-1\}$, let $Q_i \subseteq U$ be the set of up-links in U for which the arcs in A_u have a label j with $j \equiv i \pmod{k}$. Since $Q_0, Q_1, \ldots, Q_{k-1}$ is a partition of U, the average cost of the sets Q_i is $c(U)/k$. Hence, the cheapest set Q_i has cost at most $c(U) \leq c(U)/k$, and we set $Q := Q_i$.

Based on the choice of Q, we obtain a partition of F by removing from the dependency graph all arcs in A_u where $u \in Q$. Then, the links in the connected components of the resulting directed graph form the partition $\mathcal{Z}$ of F. By the choice of labeling, every directed path in the dependency graph after deleting these arcs intersects at most $k-1$ distinct sets A_u. Therefore, by Lemma 2.3.8, each part $Z \in \mathcal{Z}$ is k-thin as desired. □

2.3.5 Dynamic Programming to find the best k-thin component

In this section, we prove that we can find the k-thin subset of hyper-links $Z \subseteq \mathcal{L}$ minimizing $\frac{c(Z)}{c(\text{drop}_U(Z))}$ in polynomial time using dynamic programming. A similar result was needed in [TZ22b]. However, in general Hyper-TAP, there may be exponentially many hyper-links, so we cannot enumerate over all $\binom{|\mathcal{L}|}{k}$ sets efficiently. Thus, we again make use of the results in Section 2.3.3. In our algorithm, we work with an instance of γ-restricted Hyper-TAP for some constant γ. Therefore, there are at most $O(n^\gamma)$ hyper-links overall. This, along with the fact that we optimize over k-thin subsets for a constant k, will be necessary for the efficiency of the dynamic program.

Recall that we seek to find the minimizer ρ^* of $\frac{c(Z)}{c(\text{drop}_U(Z))}$ over all k-thin subsets $Z \subseteq \mathcal{L}$. Using binary search, we can reduce this problem to deciding whether a given ρ is greater or less than $\rho^* \in [0, 1]$.

For a given ρ and $Z \subseteq \mathcal{L}$, define

$$\text{slack}_\rho(Z) := \rho \cdot c(\text{drop}_U(Z)) - c(Z).$$

Notice that the question of whether $\frac{c(Z)}{c(\text{drop}_U(Z))} \leq \rho$ is equivalent to whether $\text{slack}_\rho(Z) \geq 0$.

Lemma 2.3.10. *The maximizer among all k-thin sets of hyper-links*

$$\max_{Z \subseteq \mathcal{L}} \{\textit{slack}_\rho(Z) : Z \textit{ is } k\textit{-thin}\}$$

can be found efficiently by dynamic programming.

Proof. The proof is an extension from [TZ22b] which proves the result for $\gamma = 2$. We denote by $D_v \subseteq V$ the set of all descendants of v in G, $X[D_v] \subseteq X$ the set of hyper-links in $X \subseteq \mathcal{L}$ with all endpoints in D_v, $\delta_X(D_v) \subseteq X$ the set of links with at least one endpoint in D_v and at least one not in D_v.

The dynamic program maintains a triple $\{v, Y, x\}$:

- $v \in V$ represents the subtree D_v we are considering.
- A set of hyper-links $Y \subseteq \delta_{\mathcal{L}}(D_v)$ with $|Y| \leq k$. These are the hyper-links that do not interact solely with D_v, but nevertheless affect the choices in the subproblem rooted at v. However, since we are seeking a k-thin set of hyper-links, and each member of $\delta_{\mathcal{L}}(D_v)$ goes through v, we have $|Y| \leq k$.
- $x \in \{+, -\}$. Note that since the sets P_u are disjoint for $u \in U$, there is at most one up-link in $\delta_U(D_v)$. If $x = +$, the k-thin set is required to cover the edges of P_u that are under v. If $x = -$, there is no requirement.

We create a table $\mathcal{T}$ with an entry for each such triple. The dimensions of this table are $\mathcal{T} \subseteq V \times 2^{\mathcal{L}} \times \{+, -\}$, and since $|\mathcal{L}| \leq O(n^\gamma)$, this table has polynomial size for any constant k.

We will proceed to fill this table from the leaves up to the root of the tree and use previously computed entries to ensure that we can fill each entry in polynomial time.

Let

$$\text{slack}_\rho(Z, Y, v) := \rho \cdot c(\text{drop}_{U[D_v]}(Z \cup Y)) - c(Z).$$

If $x = -$ then

$$\mathcal{T}[v, Y, x] := \max\{\text{slack}_\rho(Z, Y, v) : Z \subseteq \mathcal{L}[D_v],\ Z \cup Y \text{ is } k\text{-thin}\},$$

and if $x = +$, then $\mathcal{T}[v, Y, x]$ is the solution to the same optimization problem, with the additional constraint that $Z \cup Y$ must cover the edges of the unique up-link going through v, if it exists. Let $Z(v, Y, x)$ be the associated maximizer. Notice that the answer to the original problem is in the table entry $\mathcal{T}[r, \emptyset, -]$.

Fix an entry $\mathcal{T}[v, Y, x]$ and let the children of v be $v_1, v_2, \ldots, v_m$. We partition the problem into computing $Z[v_i, Y_i, x]$ for some choices of Y_i and x. We enumerate to find the correct Y_i for each $D[v_i]$. We use the following rule to partition $Z \cup Y$:

- $Z_i := Z \cap \mathcal{L}[D_{v_i}]$, which is the set of hyper-links that are contained fully in some D_{v_i}
- $\overline{Y} := Y \cup \{\ell \in Z : v \in V_\ell\}$, which is the set of hyper-links with at least one endpoint in some D_{v_i} and at least one endpoint outside of D_{v_i}.

Note that the set $Z \cup Y$ should be k-thin because we seek a k-thin solution. Since T_ℓ contains v for each $\ell \in \overline{Y}$ we have $|\overline{Y}| \leq k$. We consider the following set $\mathcal{Y}$ which is the set of all feasible $\overline{Y}$:

- $|\overline{Y}| \leq k$;
- $\overline{Y} \cup \delta_L(D_v) = Y$;
- Each hyperlink $\ell \in \overline{Y}$ goes through vertex v;
- If x equals $+$ and if the link $u \in \delta_U(D_v)$ interacts with some subtree D_{v_i}, i.e at least one endpoint of u is in some D_{v_i}, then we have $\overline{Y} \cap \delta_L(D_{v_i}) \neq \emptyset$.

Since $|\overline{Y}| \leq k$, we can bound the size of $\mathcal{Y}$. For γ-restricted hyper-links, the choice of one hyper-link is $\sum_{i=1}^{\gamma} \binom{n}{i} \leq \gamma n^\gamma$ for constant parameter γ. The size of $\mathcal{Y}$ satisfies $|\mathcal{Y}| \leq \binom{\gamma n^\gamma}{k} \leq \gamma^k n^{\gamma k} = O(n^{k\gamma})$, which is polynomially tractable. Thus, we can enumerate among all $\overline{Y}$ that satisfy the above four conditions and we obtain all information we need before breaking our dynamic program into sub-problems for D_{v_i}.

Let's fix some set $\overline{Y} \in \mathcal{Y}$, then we have

$$\text{slack}_\rho(Z_{\overline{Y}}, Y, v) = \sum_{i=1}^{m} \text{slack}_\rho(Z_i, \overline{Y} \cap \delta_L(D_{v_i}, v_i)) + \rho \cdot \sum_{u_i \in \text{Drop}_U(Z_i \cup \overline{Y})} c(u_i) - c(\overline{Y}/Y).$$

To compute Z_i, we need to determine whether $(v_i, Y_i, +)$ is feasible. There are three cases:

- $\delta_U(D_{v_i}) = \emptyset$: then $(v_i, Y_i, +)$ is infeasible due to the previous definition, we only need to compute $Z(v_i, Y_i, -)$;
- The up-link u_i only interacts with v in vertex set V/D_{v_i}: then we need to compare if we want to drop u_i or not, i.e: if $\text{slack}_\rho(Z_i^+, Y_i, v_i) + \rho \cdot w(u_i) \geq \text{slack}_\rho(Z_i^-, Y_i, v_i)$, then we will choose $Z(v_i, Y_i, +)$, otherwise we choose $Z(v_i, Y_i, -)$.
- The up-link u_i interacts with V/D_v, then we choose same sign for $Z(v_i, Y_i, x)$ as (v, Y, x).

We enumerate over all choices for $\overline{Y}$ and choices of x for each child v_i, and pick the best of these cases. Thus, we can compute $Z[v, Y, x]$ in polynomial time by relying on solutions to sub-problems on the children of v. By proceeding from the leaves to the root, we can compute the value of $\mathcal{T}[r, \emptyset, -]$ and the associated maximizer as desired. □

2.3.6 A $(1.5 + \varepsilon)$-Approximate Local Search Algorithm

In this section, we show how to achieve an approximation algorithm for STAP with approximation ratio $(1.5 + \varepsilon)$. The main idea behind the improvement is to consider dropping links that were added in previous iterations of the local search algorithm. Contrast this with Algorithm 3.3, which obtains a $1 + \ln(2) + \varepsilon$ approximation by merely dropping the up-links in the initial 2-approximate solution.

At a high level, the algorithm works as follows. Recall that we are given an instance of STAP involving a graph $G = (V, E)$ with a set of terminals $R \subseteq V$, a tree $T = (R, E(T))$ spanning R and non-negative costs $w : L \to \mathbb{R}_{\geq 0}$. The algorithm will at all times maintain a feasible solution $F \subseteq L$ and witness sets W_f for $f \in F$ such that $U := \bigcup_{f \in F} W_f$ is feasible. Each witness set consists of up-links and has size at most two.

Initially, we begin with an arbitrary feasible STAP solution F_0, and its associated witness sets. In each iteration, we add a collection of links to the solution along with their associated witness sets, drop any up-links from witness sets which are not necessary for the feasibility of U, and finally delete any links whose witness sets have become empty.

Since in Algorithm 3.3, we add a set of hyper-links to the current solution in each iteration, one might initially try to associate a witness set to each hyper-link. However, a hyper-link joining k terminals requires k up-links in its witness set, resulting in a worse approximation guarantee of $H_k + \varepsilon$.

Therefore, a key idea is to show how we can construct witness sets of size at most 2 for each link in a given STAP solution. We will use the idea behind the 2-approximate up-link solution in Lemma 2.3.1 to obtain a natural choice for the witness sets for each link, obtained by choosing the up-links responsible for covering the subpaths of the Euler tour containing f. Note that a key difference in our setting is that a single up-link may be contained in the witness set of several links, rather than just one as in the case of WTAP.

We now turn to defining how the witness sets are constructed. Given a STAP solution (S, F), we assume for simplicity that (S, F) consists of a single full component (otherwise, we enact the same procedure for each full component in the solution). Then (S, F) is a tree and admits an eulerian tour traversing each edge in F exactly twice. This tour induces an ordering on the terminals in $S \cap R$, say $\{r_1, \ldots, r_k\}$. Notice that a particular link $f \in F$ is traversed exactly twice; let's say it is used on the Euler subpath from r_i to r_{i+1}, and then again on the subpath from r_j to r_{j+1} (where we take $r_{k+1} := r_1$). We define the witness set W_f to consist of the two up-links $(r_i, \text{lca}(r_i, r_{i+1}))$ and $(\text{lca}(r_j, r_{j+1}), r_{j+1})$.

With this choice of witness set for each link $f \in F$, it is clear that each set has at most two up-links. For a set of links $C \subseteq F$, let U_C be the union of all up-links corresponding to links in C. One can show, following the same argument as the proof of Lemma 2.3.1, that the set of links $C \subseteq F$ and the set of up-links $U_C \subseteq L$ cover the same tree edges. This ensures that the union of all up-links in the sets W_f for $f \in F$ remains a feasible solution throughout the algorithm.

Now, we define a potential function Φ which maps a solution F and its witness sets to a non-negative

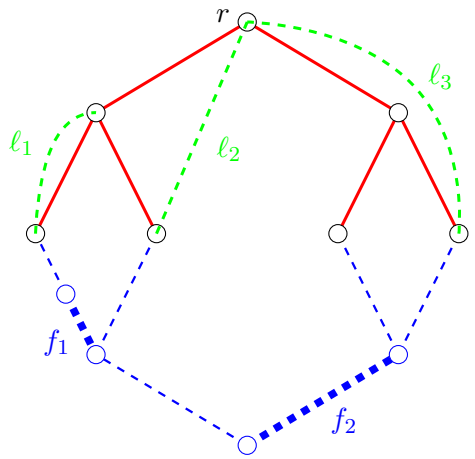

Figure 2.3: The above illustrates the construction of witness sets for each link in a given STAP solution. The red tree is the tree to be augmented and the blue dashed lines are links in a feasible STAP solution F. The witness sets corresponding to links f_1 and f_2 are $W_{f_1} = \{\ell_1, \ell_3\}$ and $W_{f_2} = \{\ell_2, \ell_3\}$ respectively.

real number.

$$\Phi(F) := \sum_{f:|W_f|=1} w(f) + \frac{3}{2} \sum_{f:|W_f|=2} w(f).$$

We also define a weight function for the up-links in witness sets. For an up-link u in any W_f, we define

$$\bar{w}(u) := \sum_{f:u\in W_f} \frac{w(f)}{|W_f|}.$$

We now turn to formally defining our algorithm (Algorithm 4.3). Recall that we are given an instance of STAP involving a graph $G = (V, E)$ with a set of terminals $R \subseteq V$, a tree $T = (R, E(T))$ spanning R and non-negative costs $w : L \to \mathbb{R}_{\geq 0}$.

We begin with an arbitrary STAP solution F_0 and its associated witness sets. Our algorithm iterates the following procedure. It finds the k-thin subset of hyper-links Z which maximizes $\bar{w}(\text{drop}_U(Z)) - 1.5w(Z)$, where $U := \bigcup_{f\in F} W_f$, and adds the links in these components to the current solution. It then updates the collection of witness sets by adding the witness sets corresponding to the new links, removing any up-links from witness sets which are covered by these new links, and shortening up-links in witness sets to ensure that their coverages are disjoint. Finally, it deletes any link from the solution F with an empty witness set. This procedure is iterated as long as the cost of the current STAP solution drops by a sufficient amount in each iteration.

Before we turn to the analysis of the algorithm, we first show that we can efficiently find a k-thin component maximizing $\bar{w}(\text{drop}_U(Z)) - 1.5w(Z)$.

Lemma 2.3.11. *Given an instance of γ-restricted Hyper-TAP and an up-link solution U, there is a polynomial time algorithm computing a k-thin collection of hyper-links maximizing $\bar{w}(\text{drop}_U(Z)) - 1.5w(Z)$.*

Proof. Define a new Hyper-TAP instance with the same tree T and hyper-links $\mathcal{L}$, but with a new weight function $\tilde{w}$. For a hyper-link ℓ, if $\ell \in U$, we define $\tilde{w}(\ell) := w(\ell)$ and $\tilde{w}(\ell) := 1.5w(\ell)$

Algorithm 2: Local search algorithm for STAP

Input: A shadow-complete, metric-complete instance of STAP with graph $G = (V, E)$, tree $T = (R, E(T))$, links $L = E \setminus E(T)$, and $c : L \to \mathbb{R}$. Also a constant $1 \geq \varepsilon > 0$.

Output: A solution $F \subseteq L$ with $c(F) \leq (1.5 + \varepsilon)OPT$.

1. Compute an arbitrary STAP solution $F \subseteq L$. Construct witness sets W_f for each $f \in F$.
2. Let $\varepsilon' := \frac{\varepsilon/2}{1.5+\varepsilon/2}$ and $\gamma := 2^{\lceil 1/\varepsilon' \rceil}$.
3. For each $S \subseteq R$ where $|S| \leq \gamma$, compute the cheapest full component joining S and denote the cost by c_S.
4. Create an instance of γ-restricted Hyper-TAP on tree $T = (R, E(T))$ with hyper-links $\mathcal{L} = \{\ell_S : S \subseteq R, |S| \leq \gamma\}$. Set the cost of hyper-link ℓ_S to be c_S.
5. Let $k := \lceil 8/\varepsilon \rceil$
6. Iterate the following as long as $\Phi(F)$ decreases in each iteration by at least a factor $(1 - \frac{\varepsilon}{12n})$:
   - Compute the k-thin subset of hyper-links $Z \subseteq 2^{\mathcal{L}}$ maximizing $\bar{w}(\text{drop}_U(Z)) - 1.5 \cdot w(Z)$, where $U := \bigcup_{f \in F} W_f$.
   - Update the witness sets by replacing each W_f with $W_f \setminus \text{drop}_U(Z)$.
   - Update F by adding in all links contained in each full component of Z.
   - Shorten up-links in W_f to ensure that their coverage is disjoint. If $W_f = \emptyset$ for some $f \in F$, then remove f from F.
7. **Return** Return F.

otherwise. Now, applying Lemma 2.3.10 with $\rho = 1$ on this new instance Hyper-TAP returns the desired k-thin maximizer. □

Now we prove the correctness of the algorithm, i.e., that the returned solution is feasible for STAP.

Lemma 2.3.12. *Both F and $U := \bigcup_{f \in F} W_f$ are feasible STAP solutions before and after each iteration of Algorithm 2. In particular, when the algorithm terminates, it returns a feasible STAP solution.*

Proof. By definition, F is initially a feasible solution. Also, U is a feasible solution initially by the proof of Lemma 2.2.1.

A link is only removed from U when it is contained in $\text{drop}_U(Z)$ for some set of hyper-links $Z \subseteq \mathcal{L}$ whose corresponding links $C \subseteq L$ were added to the solution with their accompanying witness sets W_f for $f \in C$. By definition of $\text{drop}_U(Z)$, the only up-links dropped from U are those not necessary for the feasibility of U after the links in C are added. Since U_C covers the same tree edges that C does, the feasibility of U remains intact after each iteration of the algorithm.

For a fixed up-link $u \in U$, let $X_u \subseteq L$ denote the set of links containing u in their witness set. Notice that the set of tree edges covered by X_u is always a superset of the tree edges covered by u. Indeed, initially this is true by construction, and no link in X_u is deleted so long as $u \in U$. Since U is feasible throughout the algorithm, this implies that F is feasible as well. □

We can apply the decomposition theorem to lower bound the progress made by the algorithm in each iteration.

Lemma 2.3.13. *In every iteration of Algorithm 4.3, there exists a $\lceil \frac{8}{\varepsilon} \rceil$-thin collection of hyper-links $Z \subseteq \mathcal{L}$ such that*

$$\bar{w}(drop_U(Z)) - 1.5w(Z) \geq \frac{1}{n}\Big((1 - \frac{\varepsilon}{8})w(F) - 1.5w(OPT_\gamma)\Big).$$

Proof. Recall that $U = \bigcup_{f \in F} W_f$ consists of only up-links, and throughout the algorithm we maintain that the coverage of these up-links is disjoint. Hence, we can apply Theorem 3.2.3 to U using weights $\bar{w}$ and the hyper-links in OPT_γ to obtain a partition $\mathcal{Z}$ of OPT_γ such that each part is $\lceil \frac{8}{\varepsilon} \rceil$ - thin, and

$$\sum_{Z \in \mathcal{Z}} \bar{w}(\text{drop}_U(Z)) \geq (1 - \frac{\varepsilon}{8})\bar{w}(U) = (1 - \frac{\varepsilon}{8})w(F).$$

We show a lower bound on the average value of $\bar{w}(\text{drop}_U(Z)) - 1.5w(Z)$ over all parts $Z \in \mathcal{Z}$. Note that $\sum_{Z \in \mathcal{Z}} w(Z) = OPT_\gamma$ since $\mathcal{Z}$ is a partition, and $n \geq |\mathcal{Z}|$ since . Using these and the above, we have

$$\frac{1}{|\mathcal{Z}|} \sum_{Z \in \mathcal{Z}} \bar{w}(\text{drop}_U(Z)) - 1.5w(Z) \geq \frac{1}{n}\Big((1 - \varepsilon/8)w(F) - 1.5w(OPT_\gamma)\Big).$$

Since the average value is lower bounded as above, there must be some subset Z of hyper-links satisfying

$\bar{w}(\text{drop}_U(Z)) - 1.5w(Z) \geq \frac{1}{n}\Big((1 - \varepsilon/8)w(F) - 1.5w(OPT_\gamma)\Big)$, as desired.

□

This allows us to bound the number of iterations performed by the algorithm in terms of the initial and final potentials, yielding a polynomial runtime since $w(F_0) \leq w(L)$.

Lemma 2.3.14. *Algorithm 4.3 runs for at most $\ln(\frac{3/2 \cdot w(F_0)}{w(OPT)}) \cdot (\frac{12n}{\varepsilon})$ iterations.*

Proof. The potential of the solution F initially is at most $\Phi(F) \leq \frac{3}{2}w(F_0)$. The potential of F never decreases below $w(OPT)$. Hence, since the potential decreases in each iteration by at least a $(1 - \frac{\varepsilon}{12n})$ factor, we have that the number of iterations is bounded above by

$$\log_{(1-\frac{\varepsilon}{12n})^{-1}}\left(\frac{3w(F_0)}{2w(OPT)}\right) = \frac{\ln\left(\frac{3w(F_0)}{2w(OPT)}\right)}{\ln(1 - \frac{\varepsilon}{12n})^{-1}} \leq \ln\left(\frac{3w(F_0)}{2w(OPT)}\right) \cdot \frac{12n}{\varepsilon}$$

where we used $\ln(1 + x) \leq x$ for $x > -1$. □

Finally, we show that the cost of the returned solution is small relative to the optimum.

Lemma 2.3.15. *Algorithm 2 returns a feasible STAP solution which costs at most $(1.5 + \varepsilon)$ times the cost of the optimal STAP solution.*

Proof. Denote by OPT the optimal STAP solution and by OPT_γ the optimal γ-restricted STAP solution.

Using Lemma 2.3.13, we can show that $c(F) \leq (1.5 + \frac{\varepsilon}{2})OPT_\gamma$. Indeed, upon termination, there must be no local move which decreases the potential by at least a factor $(1 - \frac{\varepsilon}{12n})$. Thus, $\bar{w}(\text{drop}_U(Z)) - 1.5w(Z) < \frac{\varepsilon}{12n}\Phi(F)$ for any $\lceil 8/\varepsilon \rceil$-thin set of hyper-links $\mathcal{Z} \subseteq \mathcal{L}$, and by Lemma 2.3.13, this implies

$$\big(1 - \frac{\varepsilon}{8}\big)w(F) - 1.5w(OPT_\gamma) < \big(\frac{\varepsilon}{12}\big)\Phi(F) \leq \big(\frac{\varepsilon}{8}\big)w(F).$$

Rearranging, this becomes $(1 - \frac{\varepsilon}{4})w(F) \leq 1.5w(OPT_\gamma)$. Hence $w(F) \leq (1.5)(\frac{1}{1-\varepsilon/4})w(OPT_\gamma)$. For $\varepsilon < 1$, this is at most $(1.5 + \varepsilon/2)w(OPT_\gamma)$ as desired.

Now, by Theorem 2.2.4 and our choice of γ, we have $c(OPT_\gamma) \leq (1 + \varepsilon')c(OPT)$. Combining these inequalities, we have

$$c(F) \leq (1.5 + \frac{\varepsilon}{2})(1 + \varepsilon')c(OPT) = (1.5 + \frac{\varepsilon}{2})(1 + \frac{\varepsilon/2}{1.5 + \varepsilon/2})c(OPT) = (1 + \varepsilon)c(OPT).$$

□

Chapter 3

The Steiner Connectivity Augmentation Problem

In this chapter, we focus on two related problems. This is based on joint work with Daniel Hathcock.

Problem 3.0.1 (k-Steiner Connectivity Augmentation Problem)**.** We are given a graph $G = (V, E \cup L)$ and a subset of terminals $R \subseteq V$ such that $H := (V, E)$ has k edge-disjoint paths between every pair of vertices in R. We are also given a cost function $c : L \to \mathbb{R}_{\geq 0}$.

The goal is to select $S \subseteq L$ of cheapest cost so that the graph $(V, E \cup S)$ has $k + 1$ pairwise edge-disjoint paths between all pairs of nodes in R.

There is another, simpler problem which is a Steiner generalization of WCAP. In the standard WCAP problem, we can only add links between pairs of vertices in the given graph. What if we are allowed to use links which join nodes outside the graph to be augmented? Utilizing these Steiner vertices can yield cheaper augmentations. As such, we define the Steiner Augmentation of a Graph problem (k-SAG) as follows.

Problem 3.0.2 (k-Steiner Augmentation of a Graph)**.** We are given a k-edge-connected graph $H = (R, E)$, which is a subgraph of $G = (V, E \dot\cup L)$. The links L have non-negative costs $c : L \to \mathbb{R}_{\geq 0}$.

The goal is to select $S \subseteq L$ of cheapest cost so that the graph $H' = (V, E \cup S)$ has $k + 1$ pairwise edge-disjoint paths between u and v for all $u, v \in R$.

The STAP problem studied in Chapter 2 corresponds to the $k = 1$ case of k-SAG. As discussed there, this yields a $1.5 + \varepsilon$ approximation for k-SAG for all odd k.

It also yields an improved approximation for 1-SCAP since k-SAG and k-SCAP are equivalent when $k = 1$. This unification ceases for higher k, making the higher connectivity setting much more interesting.

In this chapter, we give the first approximation algorithm with approximation ratio better than 2 for 2-SCAP, and for k-SAG for any k. To do this we introduce and solve the Steiner Ring Augmentation Problem (SRAP).

Problem 3.0.3 (Steiner Ring Augmentation Problem)**.** We are given a cycle $H = (V(H), E)$, which is a subgraph of $G = (V, E \dot\cup L)$. The links L have non-negative costs $c : L \to \mathbb{R}_{\geq 0}$. Furthermore, we are given a set of terminals $R \subseteq V(H)$.

The goal is to select $S \subseteq L$ of minimum cost so that the graph $H' = (V, E \cup S)$ has 3 pairwise edge-disjoint paths between u and v for all $u, v \in R$.

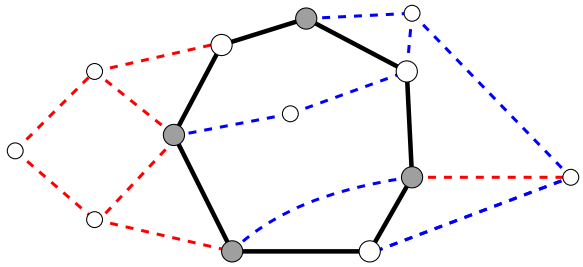

Figure 3.1: A SRAP instance where the black edges denote the given cycle, the dashed edges are the links, and the blue links form a feasible solution. The shaded nodes are the terminals R.

In terms of approximability, the Steiner Ring Augmentation Problem captures both CAP and STAP as special cases, in the sense that an α-approximation for SRAP implies the same guarantee for both CAP and STAP. We show that it also implies improved approximation algorithms for 2-SCAP and k-SAG.

Lemma 3.0.4. *If there is an α-approximation for SRAP, then there is an α-approximation for 2-SCAP. If there is an α-approximation for SRAP when $R = V(H)$, then there is an α-approximation for k-SAG.*

Our main theorem is an improved approximation algorithm for SRAP.

Theorem 3.0.5. *There is a $(1 + \ln 2 + \varepsilon)$-approximation algorithm for SRAP.*

The ultimate goal of this line of work would be to achieve a better-than-2 approximation algorithm for k-SCAP for general k. Our result on SRAP implies an improved approximation for this problem when $k = 2$.

Corollary 3.0.6. *There is a $(1 + \ln 2 + \varepsilon)$-approximation algorithm for the 2-Steiner Connectivity Augmentation Problem.*

The challenge of obtaining a result for k-SCAP when $k \geq 3$ is that the cuts to be covered are no longer necessarily minimum cuts in the given graph. Thus, there is no way to represent these in a cactus or ring structure, and new techniques would have to be developed to deal with this case.

However, in the case of k-SAG, we can replace H with a cactus, and subsequently a ring without changing the structure of its minimum cuts. Hence, the k-SAG problem ultimately reduces to the special case of SRAP where all cycles nodes are terminals. In this case, we can employ the local search methodology introduced by Traub and Zenklusen [TZ22c] to achieve an improved approximation ratio of $(1.5 + \varepsilon)$.

Theorem 3.0.7. *There is a $(1.5 + \varepsilon)$-approximation algorithm for SRAP when $R = V(H)$. Hence there is a $(1.5 + \varepsilon)$-approximation algorithm for the Steiner Augmentation of a Graph problem.*

3.1 Preliminaries

Suppose $G = (V, E)$ is a graph with vertex set V and edge set E. For a non-empty subset of vertices $C \subsetneq V$, the cut $\delta(C)$ consists of all edges in E with one endpoint in C. For a subset $X \subseteq E$, we denote by $\delta_X(C) := \delta(C) \cap X$. A cut C is a k-cut if $|\delta(C)| = k$.

The edge-connectivity $\lambda(u, v)$ between a pair of vertices $u, v \in V$ is the maximum number of edge-disjoint paths between u and v in G. Equivalently, $\lambda(u, v) = \lambda(v, u)$ is the minimum cardinality of a cut $\delta(C)$ with $u \in C$ and $v \notin C$. A graph is said to be k-edge-connected if $\lambda(u, v) \geq k$ for all pairs $u, v \in V$. Given a subset of terminals $R \subseteq V$, we say that G is Steiner k-edge-connected on R if $\lambda(u, v) \geq k$ for all pairs of terminals $u, v \in R$.

Given a Steiner k-edge-connected graph $G = (V, E)$ on terminals R, let $r \in R$, and define

$$\mathcal{C}'' = \{C \subseteq V \setminus r : |\delta(C)| = k, C \cap R \neq \emptyset\}$$

to be the family of k-cuts of G which separate some terminal from r. We call the cuts in $\mathcal{C}$ **dangerous cuts**.

The k-SCAP problem is a hitting set problem where the ground set is the collection of links L, and the sets are $\delta(C)$ where $C \in \mathcal{C}''$. That is, a set of links which is a solution to this hitting set problem will cause the graph to become Steiner $(k + 1)$-edge-connected. The link ℓ "covers" the dangerous cuts C with $\ell \in \delta(C)$.

Problem 3.0.3 (Steiner Ring Augmentation Problem)**.** We are given a cycle $H = (V(H), E)$, which is a subgraph of $G = (V, E \dot\cup L)$. The links L have non-negative costs $c : L \to \mathbb{R}_{\geq 0}$. Furthermore, we are given a set of terminals $R \subseteq V(H)$.

The goal is to select $S \subseteq L$ of minimum cost so that the graph $H' = (V, E \cup S)$ has 3 pairwise edge-disjoint paths between u and v for all $u, v \in R$.

We refer to the nodes in R as **terminals**, and the nodes in $V \setminus R$ as **Steiner nodes**. We will also refer to H as the **ring** to distinguish it from a generic cycle. It will be convenient to fix a root $r \in R$ of the ring and an edge $e_r \in E$ incident on r.

We now introduce some terminology which allows us to be phrase the SRAP problem as a covering problem on a collection of ring-cuts only. Indeed, given the ring $H = (V(H), E)$, denote the set of min-cuts of H as:

$$\mathcal{C}' = \{C \subseteq V(H) \setminus r : |\delta_E(C)| = 2\}.$$

Since we are only interested in connectivity between the terminals, we only need to cover the subfamily of $\mathcal{C}'$ which separates terminals.

Let

$$\mathcal{C} = \{C \subseteq V(H) \setminus r : |\delta_E(C)| = 2, C \cap R \neq \emptyset\}.$$

We call the cuts in $\mathcal{C}$ **dangerous ring-cuts**.

We will use a similar notion of "full components" as introduced in Ravi, Zhang and Zlatin [RZZ23] for STAP. Consider any solution $S \subseteq L$ to SRAP.

Definition 3.1.1. A **full component** of a SRAP solution S, is a maximal subtree of the solution where each leaf is a ring node (that is, a vertex of $V(H)$), and each internal node is in $V \setminus V(H)$.

Any link-minimal SRAP solution can be uniquely decomposed into link-disjoint full components. We say that a full component "joins" the ring nodes that it contains.

Definition 3.1.2. Let S be a solution to SRAP. We say that a set $A \subseteq V(H)$ is **joined** by S if there is a full component with leaves A.

Definition 3.1.3. We say that a cut $C \in \mathcal{C}$ is **covered** by a solution S if $A \cap C \neq \emptyset$ and $A \cap \bar{C} \neq \emptyset$ for some subset of ring nodes A which are joined by a full component of S.

Hence, we can think of the SRAP problem as the problem of hitting the dangerous ring-cuts with full components.

Lemma 3.1.4. *A solution S is feasible for SRAP iff all dangerous ring-cuts are covered by S.*

This motivates the definition of the Hyper-SRAP problem: We are given a ring $H = (V(H), E)$ with terminals $R \subseteq V(H)$, a root vertex $r \in R$, and a collection of hyper-links $\mathcal{L} \subseteq 2^{V(H)}$ with non-negative costs $c : \mathcal{L} \to \mathbb{R}_{\geq 0}$.

Let $\mathcal{C} = \{C \subseteq V(H) \setminus r : |\delta_E(C)| = 2, C \cap R \neq \emptyset\}$ be the set of dangerous ring-cuts, i.e. the set of min-cuts of the ring H which separate some terminal from r. A cut $C \in \mathcal{C}$ is **covered** by a hyper-link ℓ if $\ell \cap C \neq \emptyset$ and $\ell \cap \bar{C} \neq \emptyset$. The Hyper-SRAP problem is to find a minimum cost subset of hyper-links so that all cuts in $\mathcal{C}$ are covered.

We will use the notion of the hyper-link intersection graph. First, we define what it means for two hyper-links to be intersecting.

Definition 3.1.5. Let ℓ and ℓ' be a pair of hyper-links. Let $(v_1, \ldots, v_k)$ be a sequence of vertices of $\ell \cup \ell'$ obtained by traversing the ring (in either direction), where we take $v_{k+1} := v_1$. Then ℓ and ℓ' are **intersecting** if there are vertices $v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4}$ with $i_1 < i_2 < i_3 < i_4$ and $v_{i_1}, v_{i_3} \in \ell$ and $v_{i_2}, v_{i_4} \in \ell'$.

Given an instance of Hyper-SRAP with ring $H = (V(H), E)$ and hyper-links $\mathcal{L}$, we define the hyper-link intersection graph Γ as follows. For each hyper-link $\ell \in \mathcal{L}$ there is a node v_ℓ. Two nodes v_{ℓ_1} and v_{ℓ_2} are adjacent in the hyper-link intersection graph if and only if ℓ_1 and ℓ_2 are intersecting hyper-links. For ring vertices $u, v \in V(H)$, we say that there is a path from u to v in Γ if there is a path in Γ from a hyper-link containing u to a hyper-link containing v.

It turns out that a solution S to Hyper-SRAP is feasible if and only if there is a path between every pair of terminals in the hyper-link intersection graph when restricted to S. The following lemma is proved in § 3.6.

Lemma 3.1.6. *Suppose $(H, \mathcal{L}, R)$ is an instance of Hyper-SRAP with root r and intersection graph Γ. Then $S \subseteq \mathcal{L}$ is feasible iff for each terminal $r' \in R$, there is a path between r and r' in the hyper-link intersection graph restricted to S.*

Definition 3.1.7. We say that a full component is **γ-restricted** if it joins at most γ ring nodes. We say that a solution to SRAP is **γ-restricted** if it uses only γ-restricted full components. Analogously, we say an instance of Hyper-SRAP is **γ-restricted** if each hyper-link has size at most γ.

Similar to the approach developed in [RZZ23] for STAP, we can work with γ-restricted solutions to SRAP while losing an arbitrarily small constant in the approximation ratio. This follows from a result of Borchers and Du for Steiner trees [BD97].

Lemma 3.1.8. *For an instance of SRAP, let S^* be the optimal solution and S_γ be the optimal γ-restricted solution, where $\gamma(\varepsilon) = 2^{\lceil \frac{1}{\varepsilon} \rceil}$ for some $\varepsilon > 0$. Then $\frac{c(S_\gamma)}{c(S^*)} \leq 1 + \varepsilon$.*

Recall that the minimum cost Steiner tree problem can be solved in polynomial time when the number of terminals is constant.

Theorem 3.1.9 (Dreyfus and Wagner [DW71])**.** *The minimum Steiner tree problem can be solved in time $O(n^3 \cdot 3^p)$ where p is the number of terminals.*

Because Lemma 3.1.4 shows that the feasibility of a solution only depends on the nodes that are joined by full components of S, we can effectively disregard the Steiner nodes outside the ring and observe that any instance of SRAP is equivalent to an instance of Hyper-SRAP in which the cost of a hyper-link A is the minimum cost Steiner tree connecting A in the graph $(A \cup (V \setminus V(H)), L)$. By Lemma 3.1.8, and Theorem 3.1.9, we can perform this reduction from an arbitrary SRAP instance to an instance of γ-restricted Hyper-SRAP in polynomial time while only losing a factor of $(1 + \varepsilon)$ in the approximation ratio.

Finally, we will make use of directed solutions to the SRAP problem. If $\vec{F}$ is a collection of directed links between pairs of vertices of the ring, then a dangerous ring-cut $C \in \mathcal{C}$ is covered by $\vec{F}$ if $\delta^-_{\vec{F}}(C) \neq \emptyset$, i.e. if there is an arc in $\vec{F}$ which enters C. Then $\vec{F}$ is a feasible **directed solution** if all dangerous ring-cuts are covered by $\vec{F}$. Analogously, if S is a set of undirected links and $\vec{F}$ is a set of directed links then we will say that $S \cup \vec{F}$ is a feasible **mixed solution** if every dangerous ring-cut is covered by S or by $\vec{F}$.

3.2 Technical Overview

The main contribution of this article is to prove Theorem 3.0.5.

Theorem 3.0.5. *There is a $(1 + \ln 2 + \varepsilon)$-approximation algorithm for SRAP.*

We show this implies improved approximation guarantees for both 2-SCAP and k-SAG. Recall that in the k-SCAP problem, we are given a graph H which is Steiner k-edge-connected on the terminal set R. We want to augment this graph by including additional links of minimum cost so that there exists $k + 1$ pairwise edge-disjoint paths between every pair of terminals. This is equivalent to covering the dangerous cuts of H with a minimum cost set of links.

Note that if k is larger than the global edge-connectivity of H, then there may be exponentially many dangerous cuts. Indeed even when $|R| = 2$, the number of minimum $\{s, t\}$-cuts in a graph G on n nodes may be exponential in n. This shows that, unlike in global connectivity augmentation, the set of cuts to be covered in the k-SCAP problem cannot be represented by a cactus structure which is efficiently computable. Although it is true that there are polynomially many dangerous cuts up to containing the same subset of terminals [DV94], this is not sufficient for our purposes.

However, in the case of $k = 2$, we show that we may take H to be a globally 2-edge-connected graph rather than merely Steiner 2-edge-connected. This allows us to compactly represent the cuts to be covered by a cactus on a set of nodes, some of which are terminals. We then use the technique introduced by Gálvez et. al. [GGJAS21] to replace the cactus with a ring by adding in links of 0 cost. This bring the problem into the SRAP framework. In the case of k-SAG, we are guaranteed that H is k-edge-connected so we can directly replace H with a cactus with the same cut structure. This yields:

Lemma 3.0.4. *If there is an α-approximation for SRAP, then there is an α-approximation for 2-SCAP. If there is an α-approximation for SRAP when $R = V(H)$, then there is an α-approximation for k-SAG.*

In order to prove Theorem 3.0.5, we give a relative greedy algorithm which follows the methodology developed in [TZ22a] for the Weighted Ring Augmentation Problem. We first summarize this method and then illustrate the challenges that arise in the Steiner setting, and how we deal with them.

First we provide an overview of the algorithm which achieves an approximation ratio of $(1+\ln 2+\varepsilon)$ for WRAP. The algorithm is a relative greedy algorithm which begins by computing an initial directed solution which is highly structured, then iteratively improves upon this solution through local moves. In each local move, a collection of links is added to the solution, and a collection of directed links are dropped from the initial solution. The algorithm repeats this process until the initial solution becomes empty.

For the initial structured solution, Traub and Zenklusen show that, after performing a "shadow completion" of the instance, it is possible to compute in polynomial time an initial 2-approximate directed solution to WRAP which is a planar r-out arborescence $\vec{F}$ reaching all other vertices in the ring. This arborescence structure gives a condition for when a directed link (u, v) is allowed to be dropped from this solution such that a) feasibility of the mixed solution is maintained throughout the algorithm, and b) in each iteration of the algorithm, progress is made in terms of reducing the overall cost.

To achieve the latter property, Traub and Zenklusen prove a decomposition theorem with respect to $\vec{F}$ and the optimal undirected solution S^* which allows them to show that progress is made in each step.

Our relative greedy algorithm for SRAP follows this framework. However, there are several key ingredients which are needed to obtain the result in the Steiner setting.

First, we need an initial structured 2-approximate solution to SRAP. Note that in the SRAP problem, unlike in the standard Weighted Ring Augmentation Problem, the optimal solution may use links between Steiner nodes outside the ring in order to increase the connectivity between pairs of terminals in the ring. Nonetheless, using analagous techniques to [RZZ23] for STAP, we can obtain a 2-approximate solution consisting of directed links only between ring nodes. Essentially, after performing a metric completion step, taking a directed cycle on the nodes joined by each full component of the optimal solution yields a directed solution of at most 2 times the cost. Since, the optimal *directed* solution to SRAP can be found in polynomial time, such a 2-approximate directed solution can be found efficiently. Then, by iteratively shortening this solution as in [TZ22a], we can find a 2-approximate directed arborescence solution $\vec{F}_1$ for SRAP in polynomial time.

However, this is not sufficient because we cannot prove a decomposition theorem with respect to $\vec{F}_1$ in our Steiner setting. Instead, we show in § 3.4 that, after a finite sequence of metric completion and shadow completion steps are performed to obtain a "complete" instance, there is always a directed 2-approximate arboresence solution which is *only incident on terminals*.

Theorem 3.2.1. *There is a polynomial time algorithm for SRAP which yields a directed solution $\vec{F}$ of cost at most $2OPT$ such that:*

1. *$\vec{F}$ is only incident on the terminals R*
2. *$(R, \vec{F})$ is an r-out arborescence.*

3. *$(R, \vec{F})$ is planar when $V(H)$ is embedded as a circle in the plane.*

4. *For any $v \in V$, no two directed links in $\delta^+_{\vec{F}}(v)$ go in the same direction along the ring.*

We call a directed solution which satisfies the conditions in Theorem 3.2.1 an ***R*-special** directed solution. The proof of Theorem 3.2.1 is significantly more involved than the directed 2-approximation for standard WRAP. At a high level, we begin with the 2-approximate directed solution consisting of a collection of directed cycles on the ring nodes, inspired by [RZZ23] and discussed above. Then, we prove a "cycle merging lemma" which allows us to iteratively merge these cycles to eventually obtain a feasible directed solution which is a *single* directed cycle of at most the cost. Once we have a solution of this form, we can shortcut over the Steiner nodes in the ring to obtain a directed cycle solution which only touches terminals. We can then iteratively shorten this solution as in [TZ22a] to yield the arborescence structure guaranteed in Theorem 3.2.1.

An R-special directed solution of this form is necessary because it allows us to leverage a decomposition result on directed solutions with respect to the optimum. Because of the decomposition result, it can be argued that every iteration of the algorithm will find an improving local move as long as the current solution is expensive. Traub and Zenklusen prove a decomposition result of this kind to bound the approximation ratio of the relative greedy algorithm for WRAP [TZ22a]. In our setting, we need to extend the decomposition theorem to hyper-links which may join an arbitrary number of vertices in the ring. First we need to define the notion of an α-thin collection of hyper-links. Recall that $\mathcal{C}'$ is the set of minimum cuts of the ring H which do not include the root.

Definition 3.2.2. A collection of hyper-links $K \subseteq \mathcal{L}$ is α-thin if there exists a maximal laminar subfamily $\mathcal{D}$ of $\mathcal{C}'$ such that for each $C \in \mathcal{D}$, the number of hyper-links in K which cover C is at most α.

Theorem 3.2.3 (Decomposition Theorem)**.** *Given an instance of Hyper-SRAP $(H = (V(H), E), R, \mathcal{L})$, suppose $\vec{F}_0$ is an R-special directed solution and $S \subseteq \mathcal{L}$ is any solution. Then for any $\varepsilon > 0$, there exists a partition $\mathcal{Z}$ of S into parts so that:*

- *For each $Z \in \mathcal{Z}$, Z is α-thin for $\alpha = 4\lceil 1/\varepsilon \rceil$.*
- *There exists $Q \subseteq \vec{F}_0$ with $c(Q) \leq \varepsilon \cdot c(\vec{F}_0)$, such that for all $f \in \vec{F}_0 \setminus Q$, there is some $Z \in \mathcal{Z}$ with $f \in drop_{\vec{F}_0}(Z)$. That is, $\vec{F}_0 \setminus Q \subseteq \bigcup_{Z \in \mathcal{Z}} drop_{\vec{F}_0}(Z)$.*

In the above theorem, for a collection of hyper-links K and an R-special directed solution $\vec{F}_0$, the notation $\text{drop}_{\vec{F}_0}(K)$ denotes a set of directed links from $\vec{F}_0$ which can be dropped while preserving feasibility of $\vec{F}_0 \cup K$. For each directed link $f \in \vec{F}$, we describe a collection of dangerous ring-cuts for which f is "responsible", denoted $\mathcal{R}(f)$ and defined formally in § 3.6.

Then

$$\text{drop}_{\vec{F}_0}(K) := \{f \in \vec{F} : |\delta_K(C)| \geq 1 \text{ for all } C \in \mathcal{R}_{\vec{F}_0}(f)\}$$

is the set of all directed arcs such that the cuts they are responsible for are covered by K.

This definition was used by Traub and Zenklusen [TZ22a] for WRAP. They also prove an equivalent characterization showing that a directed link (u, v) can be dropped if and only if v is connected to a "v-good" vertex in link-intersection graph restricted to K. In [TZ22a], a v-good vertex is a vertex which is not a descendant of v with respect to the initial 2-approximate arborescence.

Because our initial directed solution $\vec{F}_0$ only touches terminals, we can prove a similar characterization, where we have adapted the definitions of v-good and v-bad to our setting.

Definition 3.2.4. Let $v \in R$ and consider the maximal interval $I_v \subseteq V(H)$ containing v such that I_v does not contain a terminal which is a non-descendant of v in $(R, \vec{F})$. We say that the nodes in I_v are ***v*-bad**, and all nodes in $V(H) \setminus I_v$ are ***v*-good**.

We use these new definitions to prove the following.

Lemma 3.2.5. *For a collection of hyper-links K, a directed link (u, v) is in $\text{drop}_{\vec{F}}(K)$ if and only if $\Gamma(K)$ contains a path from a hyper-link containing v to a hyper-link containing a v-good vertex w.*

Note that Lemma 3.6.6 does not necessarily hold if $\vec{F}_0$ is not R-special, since a directed link entering a Steiner node could be droppable even if K has no hyper-links containing it. We also show that an R-special solution $\vec{F}_0$ can be augmented with artificial links to obtain a $V(H)$-special solution $\vec{F}'$ so that the set of v-bad nodes in $\vec{F}_0$ correspond to the set of descendants of v in the arborescence $\vec{F}'$.

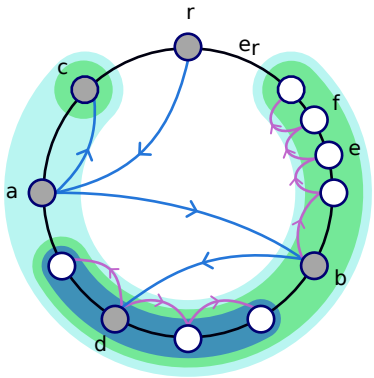

Figure 3.2: An example of an R-special solution with $R = \{r, a, b, c, d\}$ and its extension to an artificial $V(H)$-special solution. The artificial links are purple. The r-bad interval is always $V(H)$. The a-bad interval is shown in cyan. The b-bad and c-bad intervals are green, and the d-bad interval is dark blue.

This characterization of when a directed arc is droppable is used to prove Theorem 3.2.3. We follow the approach in [TZ22a] which proves the result when all hyper-links have size 2 and $R = V(H)$. We construct a "dependency graph" which allows us to partition the links of S into the desired α-thin pieces. In our hyper-link setting, the nodes of the dependency graph correspond to "festoons" composed of hyper-links rather than festoons of links of size 2. See § 3.7.

Lemma 3.1.8 and Theorem 3.1.9 allow us to convert a given instance of SRAP into an equivalent Hyper-SRAP instance efficiently. It also ensures that each local move of the algorithm runs in polynomial time. In each local move, the algorithm chooses amongst all α-thin collections of hyper-links K, the choice which minimizes the ratio between the cost of the hyper-links in K, and the cost of the directed links which will be dropped as a result of adding K to the solution. We show that this operation can be performed in polynomial time as long as all hyper-links have size at most γ.

Theorem 3.2.6. *Given an instance of γ-restricted Hyper-SRAP, an R-special directed solution $\vec{F}_0$ and an integer $\alpha \geq 1$, there is a polynomial time algorithm which finds a collection of hyper-links*

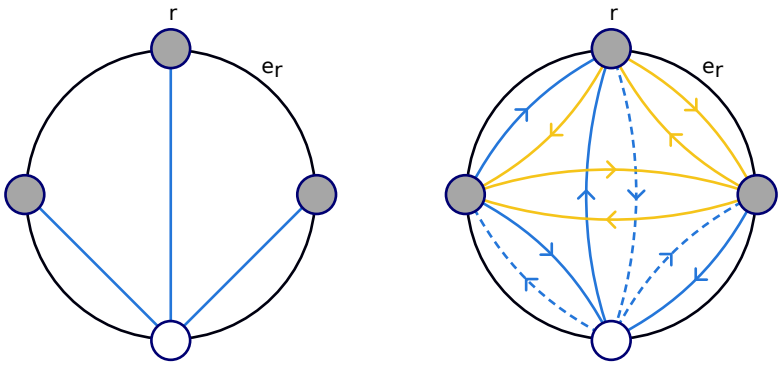

Figure 3.3: An example of a SRAP instance on the left, where all three undirected links have cost 1. The grey nodes are terminals. The completed instance is shown on the right, where the blue links have cost 1 and the orange links have cost 2 (only the directed links are shown). The dashed blue arcs form a feasible directed solution of cost 3, but there is no R-special solution of cost at most 3.

K minimizing

$$\frac{c(K)}{c(drop_{\vec{F}_0}(K))}$$

over all α-thin subsets of hyper-links.

Theorems 3.2.1, 3.2.3, and 3.2.6 together are essentially enough to employ a relative greedy strategy to obtain a $(1 + \ln 2 + \varepsilon)$-approximation for SRAP. Beginning with an R-special 2-approximate directed solution, we iteratively add to this solution a greedily chosen α-thin collection of hyper-links in each step. When a particular collection of hyper-links K is chosen to be added, the directed links in $\text{drop}_{\vec{F}_0}(K)$ are dropped from $\vec{F}_0$, and this process is repeated until $\vec{F}_0$ becomes empty. The relative greedy algorithm is described in § 3.5.

Additionally, in the case that all ring nodes are terminals, we can use the local search framework introduced in [TZ22c] to give an algorithm with an improved approximation guarantee of $(1.5+\varepsilon)$. Together with Lemma 3.0.4, this yields:

Theorem 3.0.7. *There is a $(1.5+\varepsilon)$-approximation algorithm for SRAP when $R = V(H)$. Hence there is a $(1.5+\varepsilon)$-approximation algorithm for the Steiner Augmentation of a Graph problem.*

The key idea in the improvement is to not only consider dropping links from the initial directed solution, but to drop undirected links which were added in previous iterations by associating to each undirected link a witness set of directed links which indicate when it can be dropped.

We are able to do this in the case that $V(H) = R$ because in this case, *any* feasible directed solution can be transformed into an R-special solution of at most the cost. However, this is not the case for general SRAP (see Figure 3.7).

This illustrates why it may be surprising that Theorem 3.2.1, our key technical contribution, holds. While it is not true that any feasible directed solution can be made R-special, the proof of Theorem 3.2.1 shows that a directed solution which consists of a collection of directed cycles on the ring *can* be made R-special, and this is enough to prove the desired guarantees with respect to the optimum.

3.3 Reductions to SRAP

In this section, we prove Lemma 3.0.4.

Lemma 3.0.4. *If there is an α-approximation for SRAP, then there is an α-approximation for 2-SCAP. If there is an α-approximation for SRAP when $R = V(H)$, then there is an α-approximation for k-SAG.*

Consider an instance of the 2-SCAP problem, where we are given a graph H which is Steiner 2-edge-connected for a set of terminals R, and recall that $\mathcal{C}'' = \{C \subseteq V \setminus r : |\delta(C)| = 2, C \cap R \neq \emptyset\}$ is the set of dangerous cuts to be covered.

We will show that the connected component of H which contains R can be assumed to be genuinely 2-edge-connected, rather than merely Steiner 2-edge-connected.

Lemma 3.3.1. *We may assume without loss of generality that the connected components of H consist only of isolated Steiner nodes, and a single 2-edge-connected loopless component H' containing all of the terminals R.*

Proof. First, since H contains a path between every pair of terminals, there must be a single connected component containing R, call it H'. So all other connected components contain only Steiner nodes.

Now consider a connected component K of Steiner nodes. Observe that any dangerous cut C of size $|\delta_E(C)| = 2$ separating terminals must either contain K, or be disjoint from K. Otherwise, the cut $C \cup K$ is still dangerous, but $|\delta_E(C \cup K)| < 2$, contradicting the Steiner 2-edge-connectivity of H. Hence, K may be contracted into a single isolated Steiner node.

Now, if H' contains any cut K of size $|\delta_E(K)| = 1$, it cannot separate terminals, so assume $K \cap R = \varnothing$. Denote the edge crossing this cut as $\delta_E(K) = \{(u, v)\}$ with $u \in K$, $v \notin K$. Similarly to above, we observe that any dangerous cut C containing v must contain K. Otherwise, the cut $C \cup K$ is still dangerous, but $|\delta_E(C \cup K)| < 2$. So, again, we may contract $K \cup v$ without changing the structure of cuts to be covered.

Such a contraction removes at least one cut of size 1, so we may repeat this iteratively until no cuts of size 1 remain. This leaves us with H' being a 2-edge-connected graph (and observe that neither of the contraction operations we performed could have introduced a loop). □

With the above lemma, it is easy to see that any dangerous cut C of H is obtained by taking a dangerous cut of H' (that is, a 2-cut $C \subseteq V(H')$ which separates terminals), and adding in some Steiner nodes in $V \setminus V(H')$. At a high level, this allows us to replace H' with a different graph with the same cut structure without meaningfully changing the problem.

We can now apply an easy extension of a theorem of Dinits, Karzanov, and Lomonosov on the cactus representation of the min-cuts of a graph.

Theorem 3.3.2 (Cactus representation of min-cuts, [DKL76])**.** *Let $G = (V, E)$ be a loopless graph. There is a cactus $\widehat{G} = (U, F)$ and a map $\phi : V \to U$ such that for every 2-cut of $\widehat{G}$ with shores U_1 and $U \setminus U_1$, the preimages $\phi^{-1}(U_1)$ and $\phi^{-1}(U \setminus U_1)$ are the two shores of a min-cut in G. Moreover, every min-cut of G arises in this way.*

The following strengthening of Theorem 3.3.2 immediately follows by letting $u \in U$ be in R' if and only if $\phi^{-1}(u)$ contains a node of R.

Corollary 3.3.3. *Let $G = (V, E)$ be a loopless graph with terminals $R \subseteq V$. There is a cactus $\widehat{G} = (U, F)$, a set $R' \subseteq U$, and a map $\phi : V \to U$ such that for every 2-cut of $\widehat{G}$ separating nodes of R', with shores U_1 and $U \setminus U_1$, the preimages $\phi^{-1}(U_1)$ and $\phi^{-1}(U \setminus U_1)$ are the two shores of a min-cut in G which separates terminals. Also, every min-cut in G separating terminals arises in this way.*

Applying Corollary 3.3.3 to H', we may replace H' by a cactus $\widehat{H'}$ such that there is a correspondence between 2-cuts which separate terminals in H' and 2-cuts in which separate nodes of R' in $\widehat{H'}$. All links in G incident to a node v of H' will now be incident to the corresponding node $\phi(v)$ in $\widehat{H'}$. Thus, there is a correspondence between feasible solutions to 2-SCAP and the problem of 2-SCAP where the connected component of H containing R is a cactus. In particular, we have shown that 2-SCAP reduces to the following problem:

Problem 3.3.4 (Steiner Augmentation of a Cactus)**.** We are given a cactus $H = (V(H), E)$, which is a subgraph of $G = (V, E \dot{\cup} L)$. The links L have non-negative costs $c : L \to \mathbb{R}_{\geq 0}$. There is also a set of terminals $R \subseteq V(H)$.

The goal is to select $S \subseteq L$ of cheapest cost so that the graph $H' = (V, E \cup S)$ has 3 pairwise edge-disjoint paths between u and v for all $u, v \in R$.

By the addition of zero-cost links (c.f Theorem 4 in [GGJAS21]), we can unfold the cactus further so that H is a cycle. This is precisely the Steiner Ring Augmentation Problem, which we restate here:

Problem 3.0.3 (Steiner Ring Augmentation Problem)**.** We are given a cycle $H = (V(H), E)$, which is a subgraph of $G = (V, E \dot{\cup} L)$. The links L have non-negative costs $c : L \to \mathbb{R}_{\geq 0}$. Furthermore, we are given a set of terminals $R \subseteq V(H)$.

The goal is to select $S \subseteq L$ of minimum cost so that the graph $H' = (V, E \cup S)$ has 3 pairwise edge-disjoint paths between u and v for all $u, v \in R$.

Each of these reductions can be performed in polynomial time. And, since the structure of dangerous cuts to be covered remains the same, any solution to the 2-SCAP instance gives rise a solution of the same cost in the SRAP instance, and every solution in the SRAP instance arises in such a way. This proves the first half of Lemma 3.0.4.

The second half of Lemma 3.0.4, the reduction from k-SAG, follows in an almost identical (albeit simpler) fashion. We replace the k-edge-connected graph H with a cactus $\widehat{H}$ using Theorem 3.3.2, such that there is a correspondence between the min-cuts of H and the 2-cuts of $\widehat{H}$. This time, all nodes in $\widehat{H}$ are terminals. We can again add zero cost links to replace $\widehat{H}$ with a cycle, yielding an instance of SRAP in which $R = V(H)$.

3.4 A structured 2-approximate solution for SRAP

SRAP can be approximated to within a factor of 2 using Jain's algorithm for Survivable Network Design [Jai01]. Recall that for a rooted SRAP instance on ring H, the set of cuts to be covered is

$$\mathcal{C} = \{C \subseteq V(H) \setminus r : |\delta_E(C)| = 2, C \cap R \neq \emptyset\}.$$

Also recall that a subset of directed links $\vec{F}$ is feasible if $\delta^-_{\vec{F}}(C) \geq 1$ for all $C \in \mathcal{C}$. In other words, a directed link covers only those cuts in $\mathcal{C}$ that it enters. In this section, we show that we can find

a 2-approximate directed-link solution to SRAP which is only incident on terminals, and is highly structured.

Theorem 3.2.1. *There is a polynomial time algorithm for SRAP which yields a directed solution $\vec{F}$ of cost at most $2OPT$ such that:*

1. *$\vec{F}$ is only incident on the terminals R*
2. *$(R, \vec{F})$ is an r-out arborescence.*
3. *$(R, \vec{F})$ is planar when $V(H)$ is embedded as a circle in the plane.*
4. *For any $v \in V$, no two directed links in $\delta^+_{\vec{F}}(v)$ go in the same direction along the ring.*

We call a directed solution which satisfies the above conditions an ***R*-special** directed solution.

3.4.1 Complete instances

In order to prove Theorem 3.2.1, we need to perform several preprocessing steps on our instance to ensure that the necessary links are available. This will result in an equivalent "completed" instance which can be efficiently computed.

To bring our instance into the desired form, we will perform three preprocessing steps: metric completion, shadow completion, and then a second metric completion on the resulting directed links.

For the first metric completion, we place a new undirected link between each pair of vertices of the ring $u, v \in V(H)$, with cost equal to the shortest path from u to v which uses only links (if there is no link-path from u to v the cost is infinity). This does not affect the cost of the optimal solution, so we may assume without loss of generality that the instance is metric complete. After this step, there is an undirected link between every pair of vertices in the ring.

For shadow completion, for each link $\ell = (u, v)$ with $u, v \in V(H)$, we will add to the instance a collection of directed links known as the **shadows** of ℓ. These will all have the same cost as ℓ. The shadows of ℓ consist of the two directed links (u, v) and (v, u) as well as all shortenings of these two directed links. If (u, v) is a directed link, then (s, t) is a **shortening** of (u, v) if $v = t$ and s is a vertex on the path from u to v in $E \setminus e_r$.

With this definition, it is not hard to see that a shadow of ℓ covers a subset of the cuts in $\mathcal{C}$ that ℓ covers. Hence, we may perform this step without affecting the cost of the optimal solution.

Finally, we do a second round of metric completion on the newly added directed links, similar to the first. For every pair of vertices $u, v \in V(H)$, if there is a directed path from u to v, we add a directed link (u, v) with cost equal to the cost of the shortest such path.

After these operations, we are left with an instance of SRAP such that any solution to this instance can be converted into a solution to the original SRAP instance with the same cost. Hence, we may assume that these operations have been performed on the given SRAP instance without loss of generality. See Figure 3.4 for an example of this preprocessing.

Furthermore, we claim that any further iterations of these two preprocessing operations (metric completion and shadow completion) will not change the instance.

Definition 3.4.1. An instance of SRAP is called a **complete instance** if for every $u, v \in V(H)$, there is a directed link (u, v) whose cost is equal to the shortest directed path from u to v, and for every directed link (u, v), the instance contains all shortenings of (u, v) with at most cost $c((u, v))$.

Lemma 3.4.2. *Any SRAP instance can be made complete by performing metric completion, then shadow completion, then a second metric completion.*

Proof. Let L denote the set of links in the original instance, and L^1, L^2, L^3 denote the links after each of the three preprocessing steps, respectively. We want to show that L^3 is a complete instance. Clearly the instance is already metrically complete, since the final preprocessing operation is a metric completion. So it suffices to show that all shortenings of links in L^3 already exist in L^3.

Consider any link $(u, v) \in L^3$. If (u, v) did not arise from the second metric completion step, then it is in L^2 and all of its shortenings were added in the shadow completion step.

So we focus on the case in which (u, v) arose as a metric completion of some path of links $(u = w_0, w_1), (w_1, w_2), \ldots, (w_{k-1}, w_k = v)$, where all links on this path are in L^2. Without loss of generality, suppose that u lies to the left of v. Consider an arbitrary shortening of (u, v), which takes the form (s, v) for some s lying between u and v. There must be some link (w_i, w_{i+1}) on the path such that w_i lies to the left of s, and w_{i+1} to the right of s. Then (s, w_{i+1}) is a shortening of (w_i, w_{i+1}). Moreover, since $(w_i, w_{i+1}) \in L^2$, then so is (s, w_{i+1}).

In particular, L^2 contains the path of links $(s, w_{i+1}), \ldots, (w_{k-1}, v)$ whose total cost is at most the cost of the path from u to v. And hence, L^3 contains the shortening (s, v) of (u, v). □

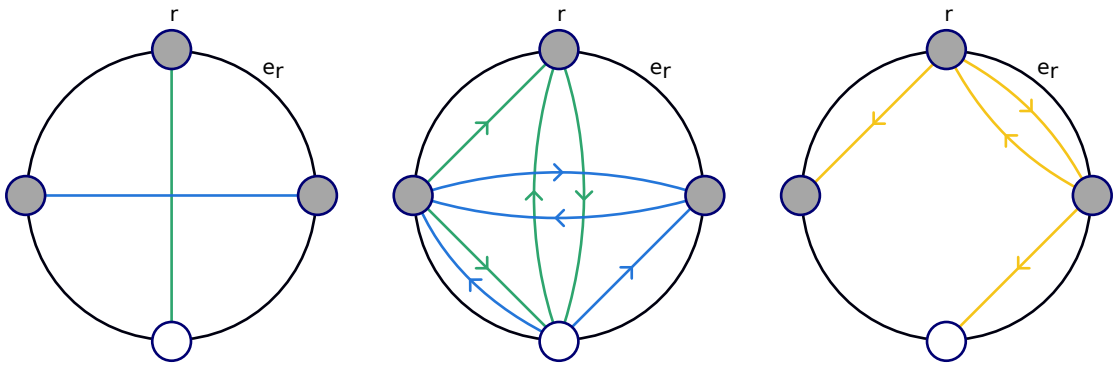

Figure 3.4: An example of a SRAP instance undergoing preprocessing steps to obtain a complete instance. The leftmost SRAP instance has two undirected links ℓ_1 (green) and ℓ_2 (blue) in L. There are no links added in L^1. The middle picture shows the directed links added in L^2, where the green arcs are shadows of ℓ_1 and have cost $c(\ell_1)$, and the blue arcs are shadows of the ℓ_2 with cost $c(\ell_2)$. Finally, the third picture shows the (undominated) directed links added in L^3 in yellow. Each of these arcs have cost $c(\ell_1) + c(\ell_2)$. The final completed SRAP instance contains all of these links.

3.4.2 An R-special 2-approximate solution

We now proceed with the 2-approximate R-special solution for SRAP. First, we make an existential claim about the existence of a 2-approximate directed solution which only touches terminals, then we show that we can find one efficiently.

In the proof of Lemma 3.4.3, we will first show that there is a 2-approximate directed solution which consists of a collection of directed cycles on nodes of the ring. We will then merge these cycles to obtain a directed solution which is a single directed cycle.

It is helpful to notice that an undirected cycle on nodes $A \subseteq V(H)$ covers the same set of dangerous ring-cuts that a directed cycle on A covers, which is also the same set of dangerous ring-cuts that

the hyper-link A covers. This allows us to use Lemma 3.1.6 to understand when a collection of directed cycles is feasible.

For a node set $A \subseteq V(H)$, denote by $I_A \subseteq V(H)$ the nodes in the minimal interval containing A which does not contain e_r.

Lemma 3.4.3. *Given an instance of SRAP, suppose the optimal solution has cost OPT. Then there exists a directed solution of cost at most 2OPT whose links consist of a single directed cycle including r.*

Proof. Let (U^*, F^*) be a full component of the optimal SRAP solution. Then (U^*, F^*) is a tree. Starting from an arbitrary vertex in $U^* \cap V(H)$, we take an Euler tour of this tree traversing each link exactly twice. This induces an ordering of the ring nodes which are visited during the tour $a_1, \ldots, a_k$, such that the undirected link set $F = \{(a_1, a_2), \ldots, (a_{k-1}, a_k), (a_k, a_1)\}$ has cost at most $2c(F^*)$ by metric completion.

Now, consider the directed link set $\vec{F} = \{(a_1, a_2), \ldots, (a_{k-1}, a_k), (a_k, a_1)\}$. The cost of $\vec{F}$ is at most $c(F)$ since it consists of shadows of links in F. Hence $c(\vec{F}) \leq 2c(F^*)$. Furthermore, since $\vec{F}$ forms a directed cycle on the nodes joined by F^*, any cut covered by F^* is also covered by $\vec{F}$.

Repeating this for each full component in the optimal solution yields a directed solution with cost at most 2OPT. Furthermore, this directed solution consists of a collection of directed cycles on the nodes of the ring.

We now exploit the following lemma to obtain a directed solution with cost at most 2OPT consisting of a single directed (not necessarily simple) cycle.

Lemma 3.4.4 (Cycle Merging Lemma)**.** *Consider a SRAP instance on the ring $H = (V(H), E)$. Let $\vec{F}_S$ and $\vec{F}_A$ be two directed cycles of links on nodes $S, A \subseteq V(H)$, respectively, where $r \in S$. If S and A are intersecting as hyper-links, then there exists a directed link set $\vec{F}_{S \cup A}$ which forms a directed cycle on $S \cup A$ of cost at most $c(\vec{F}_S) + c(\vec{F}_A)$.*

Proof. First observe that if $A \cap S \neq \varnothing$, we simply define $\vec{F}_{S \cup A} = \vec{F}_S \cup \vec{F}_A$, and the statement of the lemma is satisfied.

So, suppose that $A \cap S = \varnothing$. Denote the cycle links as $\vec{F}_S = \{(s_1, s_2), \ldots, (s_{k-1}, s_k), (s_k, s_1)\}$, with $s_1 = r$, and $\vec{F}_A = \{(a_1, a_2), \ldots, (a_{\ell-1}, a_\ell), (a_\ell, a_1)\}$. Consider the interval I_A of A. Since S and A are intersecting (as hyper-links), and S contains the root r while $r \notin I_A$, it must be the case that the cycle formed by $\vec{F}_S$ leaves I_A at least once. Let $(s_i, s_{i+1}) \in \vec{F}_S$ be a directed link leaving I_A.

Now consider the interval $I^* = I_{\{s_i, s_{i+1}\}}$. We claim that $\vec{F}_A$ leaves I^* at least once. Indeed, since (s_i, s_{i+1}) leaves I_A, some $a \in A$ is in I^*. But all of A cannot be contained in I^*, since then (s_i, s_{i+1}) leaving I_A would imply that $s_i \in A$, contradicting our assumption that $A \cap S = \varnothing$. Let $(a_j, a_{j+1}) \in \vec{F}_A$ be a directed link leaving I^*.

We now define $\vec{F}_{S \cup A}$ to be the directed links along the cycle

$$s_1, \ldots, s_i, a_{j+1}, a_{j+2}, \ldots, a_j, s_{i+1}, \ldots, s_1.$$

See Figure 3.5 for an example. Observe that (s_i, a_{j+1}) is a shortening of (a_j, a_{j+1}) and (a_j, s_{i+1}) is a shortening of (s_i, s_{i+1}). Therefore $c(\vec{F}_{S \cup A}) \leq c(\vec{F}_S) + c(\vec{F}_A)$ as desired. Moreover, $\vec{F}_{S \cup A}$ forms a cycle on $S \cup A$. □

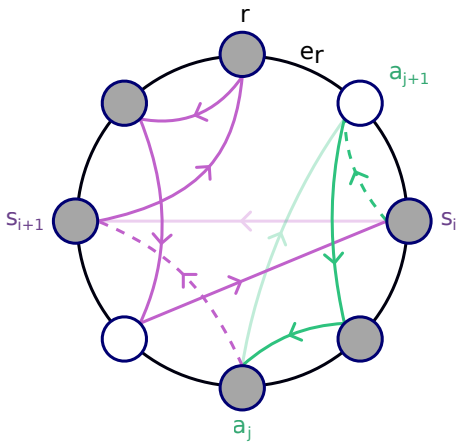

Figure 3.5: An example of cycle merging, with $\vec{F}_S$ in purple and $\vec{F}_A$ in green. Note that (s_i, s_{i+1}) leaves the interval I_A, and (a_j, a_{j+1}) leaves the interval $I_{\{s_i,s_{i+1}\}}$. Dropping these two links, and adding the shortenings drawn as dotted arcs creates the single cycle $\vec{F}_{S\cup A}$.

We can now use Lemma 3.4.4 to transform a directed solution that is a collection of directed cycles on ring nodes $A_1, \ldots, A_p$ into a single directed cycle on $A := \bigcup_{i=1}^{p} A_i$. Let $\vec{J}$ denote the directed solution consisting of a collection of directed cycles on the nodes in the ring. Since r is a terminal, it must be contained in some cycle $\vec{F}_{S_0}$ on $S_0 \subseteq V(H)$. We build a new directed solution starting with $\vec{F}_{S_0}$ by repeatedly apply Lemma 3.4.4, cycle merging. In each step, choose a new cycle $\vec{F}_A$ from $\vec{J}$ that has not yet been merged and such that A is intersecting with S_i, and update $\vec{F}_{S_i}$ by merging: $\vec{F}_{S_{i+1}} \leftarrow \vec{F}_{S_i\cup A}$. If all cycles from $\vec{J}$ can be merged in this way, this yields a single directed cycle. It is feasible, since it is incident to all terminals, and its cost is at most that of $\vec{J}$, which is at most 2OPT.

It remains only to show that at every step, if there are un-merged cycles from $\vec{J}$, then one of them is on a set of nodes which is intersecting with S_i. First, observe that $\vec{F}_{S_i}$ is a cycle on all of the nodes belonging to the cycles merged so-far, so it suffices to find some un-merged $\vec{F}_A$ in $\vec{J}$ such that A is intersecting with the nodes in one of the cycles merged so-far.

Since $\vec{J}$ is feasible, the corresponding collection of undirected cycles is also feasible. Hence, by Lemma 3.1.6, the hyper-links on the vertices of the cycles in $\vec{J}$ are connected in the hyper-link intersection graph. Now for any un-merged cycle $\vec{F}_{A'}$, look at the path in the hyper-link intersection graph from the hyper-link on S_0 to the hyper-link on A'. The first hyper-link along this path corresponding to an un-merged cycle $\vec{F}_A$ must be intersecting with a merged cycle, so is intersecting with S_i. □

With the above lemma in hand, and due to our second metric completion step, we can now shortcut over the non-terminals in the cycle to obtain a 2-approximate directed solution which only touches terminals.

Lemma 3.4.5. *Given an instance of SRAP, if the optimal solution has cost OPT, then there is a directed solution of cost at most 2OPT whose links consist of a single directed cycle with node set R.*

Proof. This follows by taking the directed cycle solution $\vec{F}$ from Lemma 3.4.3 and short-cutting through all non-terminal nodes. Since the third preprocessing step ensures that the directed links

are metrically complete, replacing a path of directed links with a single link cannot increase the cost. Hence, the resulting solution has cost at most $c(\vec{F}) \leq 2\text{OPT}$. □

Having shown that there exists a directed 2-approximate solution to any SRAP instance which is incident only on the terminals R, we now proceed to show how to compute one in polynomial time. In particular, we will prove that the optimal such solution can be found efficiently. And moreover, the solution can be assumed to have additional structure.

Recall that the Weighted Ring Augmentation Problem (WRAP) is a special case of SRAP in which all nodes are terminals, and there are no nodes outside of the ring H. A directed solution $\vec{F}$ to WRAP is **non-shortenable** if it is feasible but deleting or strictly shortening any link $f \in \vec{F}$ results in an infeasible solution. Traub and Zenklusen proved that a non-shortenable directed solution to WRAP has a planar arborescence structure, and can be computed in polynomial time.

Lemma 3.4.6 (Lemma 2.5 in [TZ22a])**.** *A non-shortenable optimal directed solution to the WRAP problem can be found in polynomial time.*

Reducing our problem to a WRAP instance and applying the above lemma yields the main theorem of this section:

Theorem 3.2.1. *There is a polynomial time algorithm for SRAP which yields a directed solution $\vec{F}$ of cost at most $2OPT$ such that:*

1. *$\vec{F}$ is only incident on the terminals R*
2. *$(R, \vec{F})$ is an r-out arborescence.*
3. *$(R, \vec{F})$ is planar when $V(H)$ is embedded as a circle in the plane.*
4. *For any $v \in V$, no two directed links in $\delta^+_{\vec{F}}(v)$ go in the same direction along the ring.*

Proof. Consider a SRAP instance on the ring $H = (V(H), E)$ with terminals $R \subseteq V(H)$, root $r \in R$, and links L. We construct a new ring $H' = (R, E')$ on the terminals by iteratively replacing each non-terminal node in $V(H)$ with an edge between its two neighbors in H. We now create a WRAP instance on H' by taking the subset L' of links which are directed links incident only to R.

We apply Lemma 3.4.6 (Lemma 2.5 from [TZ22a]) to get a non-shortenable optimal directed solution $\vec{F}$ to this WRAP instance. Since Lemma 3.4.5 guarantees the existence of a directed 2-approximation which only touches terminals, this implies that $c(\vec{F}) \leq 2\text{OPT}$.

Now consider $\vec{F}$ as a solution to the SRAP instance on H. Since $\vec{F}$ is non-shortenable when viewed as a solution to the WRAP instance on H', Theorem 2.6 from [TZ22a] shows that it satisfies the three conditions when viewed as a solution to H. It remains only to argue that $\vec{F}$ is actually feasible. This is immediate from the arborescence structure of $\vec{F}$: there is a path from r to every terminal. □

3.5 A $(1 + \ln 2 + \varepsilon)$-approximation for SRAP

In this section, we describe a polynomial time algorithm for SRAP which achieves an approximation ratio of $(1 + \ln 2 + \varepsilon)$. Before writing the algorithm, we comment on a slight departure of our algorithm from the relative greedy algorithm for standard WRAP.

Algorithm 3: Relative greedy algorithm for SRAP

Input: A complete instance of SRAP with graph $G = (V, E \dot\cup L)$, ring $H = (V(H), E)$, terminals $R \subseteq V(H)$ and $c : L \to \mathbb{R}$. Also an $\varepsilon > 0$.

Output: A solution $S \subseteq L$ with $c(S) \leq (1 + \ln(2) + \varepsilon) \cdot c(\mathrm{OPT})$.

1. Compute a 2-approximate R-special directed solution $\vec{F}_0$ (Theorem 3.2.1).
2. Let $\varepsilon' := \frac{\varepsilon/2}{1+\ln 2+\varepsilon/2}$ and $\gamma := 2^{\lceil 1/\varepsilon' \rceil}$.
3. For each $A \subseteq V(H)$ where $|A| \leq \gamma$, compute the cheapest full component joining A and denote the cost by c_A.
4. Create an instance of γ-restricted Hyper-SRAP on the ring $H = (V(H), E)$ with hyper-links $\mathcal{L} = \{\ell_A : A \subseteq V(H), |A| \leq \gamma\}$. Set the cost of hyper-link ℓ_A to be c_A.
5. Initialize $S_0 := \emptyset$
6. Let $\alpha := 4\lceil 4/\varepsilon \rceil$
7. While $\vec{F}_i \neq \emptyset$:
   - Increment i by 1.
   - Compute the α-thin subset of hyper-links $Z_i \subseteq \mathcal{L}$ minimizing $\frac{c(Z_i)}{c(\mathrm{drop}_{\vec{F}_0}(Z_i) \cap \vec{F}_{i-1})}$.
   - If $\frac{c(Z_i)}{c(\mathrm{drop}_{\vec{F}_0}(Z_i) \cap \vec{F}_{i-1})} > 1$, then update $Z_i = \kappa_f$ for some $f \in \vec{F}_{i-1}$.
   - Let $S_i := S_{i-1} \cup Z_i$ and let $\vec{F}_i := \vec{F}_{i-1} \setminus \mathrm{drop}_{\vec{F}_0}(Z_i)$.
8. **Return** A SRAP solution with full components corresponding to the hyper-links in $S := S_i$.

Given any SRAP instance, we may convert it into an equivalent complete instance (see § 3.4.1). We observe that for any directed link $f = (u, v)$ on the ring, there is a collection of undirected links whose coverage is at least that of f, and whose total cost is at most $c(f)$. We call this set κ_f. Indeed, if f is a shadow of an undirected link ℓ, then we may take $\kappa_f = \{\ell\}$. Otherwise, f arises from the metric completion of some directed path of links $(u = s_0, s_1), (s_1, s_2), \ldots, (s_{k-1}, s_k = v)$. Each of these links in the directed path may themselves be shortenings of undirected links $\{\ell_1, \ldots, \ell_k\}$. In this case, $\kappa_f = \{\ell_1, \ldots, \ell_k\}$. Note in particular that κ_f covers all of the dangerous ring-cuts that f covers, so if f is in some R-special solution $\vec{F}_0$, then $f \in \mathrm{drop}_{\vec{F}_0}(\kappa_f)$ (see § 3.6 for a formal definition of drop).

In order to show that our algorithm makes sufficient progress at each step, we must have that the cost of our mixed solution does not ever increase over the course of the algorithm. In the case of WRAP, this is immediate, since any directed link is a shadow of some single undirected link of the same cost, an α-thin set. However, in our case, κ_f may not be α-thin, so we explicitly consider this as a separate case in each step of our algorithm. See Algorithm 3.

The proof of the following theorem follows from a standard analysis of the relative greedy algorithm as in [CN13b], [TZ22b], and [TZ22a]. We include it here for completeness.

Theorem 3.5.1. *Algorithm 3 is a $(1+\ln 2+\varepsilon)$-approximation algorithm for SRAP.*

Proof. First, we note that in each iteration of the algorithm, $|\vec{F}|$ will be reduced by at least 1. The initial value of $|\vec{F}|$ is at most $|R|-1$, so there are polynomially many iterations. By Theorem 3.8.1, each iteration can be executed in polynomial time. Hence Algorithm 3 runs in polynomial time.

The returned solution S is feasible since the invariant that $S_i \cup \vec{F}_i$ is a feasible mixed solution is maintained throughout the algorithm.

To complete the proof, we show that S has cost at most $(1+\ln 2+\varepsilon)\cdot c(\mathrm{OPT})$. Denote by OPT_γ the optimal γ-restricted solution. Apply the decomposition theorem, Theorem 3.2.3, to OPT_γ and the R-special solution $\vec{F}_0$. It gives a partition $\mathcal{Z}$ of OPT_γ into α-thin parts, and some $Q \subseteq \vec{F}_0$ with $c(Q) \le \varepsilon/4 \cdot c(\vec{F}_0) \le \varepsilon/2 \cdot c(\mathrm{OPT})$. Then observe that

$$\frac{c(Z_i)}{c(\mathrm{drop}_{\vec{F}_0}(Z_i)\cap \vec{F}_{i-1})} \le \min_{Z\in\mathcal{Z}} \frac{c(Z)}{c(\mathrm{drop}_{\vec{F}_0}(Z)\cap \vec{F}_{i-1})} \le \frac{\sum_{Z\in\mathcal{Z}} c(Z)}{\sum_{Z\in\mathcal{Z}} c(\mathrm{drop}_{\vec{F}_0}(Z)\cap \vec{F}_{i-1})} \le \frac{c(\mathrm{OPT}_\gamma)}{c(\vec{F}_{i-1})-c(Q)},$$

where the first inequality is by the choice of Z_i in the algorithm, and the second and third follow from the statement of Theorem 3.2.3. Since $f \in \mathrm{drop}_{\vec{F}_0}(\kappa_f)$, our choice of Z_i implies that, $c(Z_i)/c(\mathrm{drop}_{\vec{F}_0}(Z_i)\cap \vec{F}_{i-1}) \le 1$. Therefore, since $\mathrm{drop}_{\vec{F}_0}(Z_i)\cap \vec{F}_{i-1} = \vec{F}_{i-1}\setminus \vec{F}_i$, we have

$$c(Z_i) \le \min\left\{1, \frac{c(\mathrm{OPT}_\gamma)}{c(\vec{F}_{i-1})-c(Q)}\right\}\cdot c(\vec{F}_{i-1}\setminus \vec{F}_i) \le \int_{c(\vec{F}_i)}^{c(\vec{F}_{i-1})} \min\left\{1, \frac{c(\mathrm{OPT}_\gamma)}{x-c(Q)}\right\} dx.$$

Finally, sum over all iterations of the algorithm to get the cost of the output S:

$$\begin{aligned}
c(S) &= \sum_i c(Z_i) \\
&\le \int_0^{c(\vec{F}_0)} \min\left\{1, \frac{c(\mathrm{OPT}_\gamma)}{x-c(Q)}\right\} dx \\
&= \int_0^{c(\mathrm{OPT}_\gamma)+c(Q)} 1\, dx + \int_{c(\mathrm{OPT}_\gamma)+c(Q)}^{c(\vec{F}_0)} \frac{c(\mathrm{OPT}_\gamma)}{x-c(Q)}\, dx \\
&\le \left(1+\frac{\varepsilon}{2}\right)\cdot c(\mathrm{OPT}_\gamma) + \ln\left(\frac{c(\vec{F}_0)-c(Q)}{c(\mathrm{OPT}_\gamma)}\right)\cdot c(\mathrm{OPT}_\gamma) \\
&\le \left(1+\ln(2)+\frac{\varepsilon}{2}\right)\cdot c(\mathrm{OPT}_\gamma)
\end{aligned}$$

And applying Lemma 3.1.8 for our choice of ε' and γ from the algorithm, we have $c(\mathrm{OPT}_\gamma) \le (1+\varepsilon')\cdot c(\mathrm{OPT})$. Hence, we get the desired bound

$$c(S) \le \left(1+\ln(2)+\frac{\varepsilon}{2}\right)\cdot(1+\varepsilon')\cdot c(\mathrm{OPT}) = (1+\ln(2)+\varepsilon)\cdot c(\mathrm{OPT}). \qquad \square$$

3.6 Dropping Directed Links

The main reason that we work with structured R-special directed solutions to SRAP is that it allows us to cleanly characterize when a directed link can be dropped after a collection of hyperlinks are added to the solution. In this section, we recall the properties of an R-special directed solution and use them give such a characterization.

Given an instance of Hyper-SRAP we can use the root r and root-edge e_r to define a notion of right and left along the ring. In particular, we imagine deleting the edge e_r from the ring and consider the root to be the left-most node on the remaining path. The other node incident to e_r is the right-most node in the ring.

Consider a R-special directed solution $\vec{F}$ for the SRAP problem. Recall that $\vec{F}$ has the following properties:

1. $\vec{F}$ is only incident on terminals R.
2. $(R, \vec{F})$ is an r-out arborescence.
3. $(R, \vec{F})$ is planar when $V(H)$ is embedded as a circle in the plane.
4. For any $v \in R$, no two directed links in $\delta^+_{\vec{F}}(v)$ go in the same direction along the ring.

Given an R-special directed SRAP solution $\vec{F}$, we associate to each cut $C \in \mathcal{C}$ a single link which is responsible for covering it, as in [TZ22a]. In particular, an arc $\ell = (u, v) \in \vec{F}$ is **responsible** for covering a cut $C \in \mathcal{C}$ if ℓ enters C and there is no other arc on the unique r–u path in $(R, \vec{F})$ which enters C. We denote the set of cuts for which a link $\ell \in \vec{F}$ is responsible by $\mathcal{R}_{\vec{F}}(\ell)$. Notice that $\mathcal{R}_{\vec{F}}(\ell) \neq \emptyset$ for all $\ell \in \vec{F}$, since if some directed link were not responsible for any cuts, then it could be deleted without affecting the feasibility of $\vec{F}$, but this is not true of any arc in an R-special directed solution.

We will show that every ring-dangerous cut $C \in \mathcal{C}$ has exactly one directed link responsible for it in an R-special solution $\vec{F}$. First we will need the following lemma which follows from the properties of $\vec{F}$.

Lemma 3.6.1. *Let $\vec{F}$ be an R-special directed solution for SRAP.*

(i) For any $v \in R$, the set of descendants of v in $(R, \vec{F})$ is of the form $R \cap I$ for some ring interval $I \subseteq V(H)$.

(ii) For $v_1, v_2 \in R$, the least common ancestor of v_1 and v_2 in $(R, \vec{F})$ lies between v_1 and v_2.

This is an analogue to Lemma 4.8 from [TZ22a]. In fact, this lemma follows immediately from Lemma 4.8 from [TZ22a] by simply removing the Steiner nodes in the ring and viewing $\vec{F}$ as an R-special solution on a ring with only the terminals R (c.f. the proof of Theorem 3.2.1).

Lemma 3.6.2. *For each cut $C \in \mathcal{C}$, there is exactly one directed link $\ell \in \vec{F}$ responsible for it.*

Proof. A cut $C \in \mathcal{C}$ clearly has at least one link responsible for it, since $\vec{F}$ enters all such cuts. On the other hand, no two links can both be responsible for C. This is because C is an interval, so if $(u_1, v_1) \neq (u_2, v_2)$ are both responsible for C, then the least common ancestor v of v_1 and v_2 is in C, by Lemma 3.6.1(ii). Since $r \notin C$, there must be some link in the r–v path in $(R, \vec{F})$ (and hence also on the r–v_1 path) entering C, contradicting that (u_1, v_1) is responsible for C. □

We can now define when a directed link will be dropped from an R-special directed solution $\vec{F}$. In particular, if a collection of hyper-links K is added to the solution, then we will drop a directed link if and only if all cuts it is responsible for are covered by the hyper-links in K. Let $\delta_K(C)$ be the set of hyper-links in K which cover $C \in \mathcal{C}$.

Formally, denote

$$\mathrm{drop}_{\vec{F}}(K) = \{f \in \vec{F} : |\delta_K(C)| \geq 1 \text{ for all } C \in \mathcal{R}_{\vec{F}}(f)\}.$$

With this definition, if a collection of hyper-links K is added to an R-special directed solution $\vec{F}$ to SRAP, and $\mathrm{drop}_{\vec{F}}(K)$ is removed, then the solution remains feasible as a mixed SRAP solution.

Our ultimate goal of this section is an alternate characterization of the set $\mathrm{drop}_{\vec{F}}(K)$. For this, we will use the notion of the hyper-link intersection graph. Recall that a pair of hyper-links ℓ_1 and ℓ_2 are **intersecting** if they share a vertex or there is a vertex in ℓ between two vertices in ℓ' and a vertex of ℓ' lies between two vertices of ℓ.

Given an instance of Hyper-SRAP with ring $H = (V(H), E)$ and hyper-links $\mathcal{L}$, we define the hyper-link intersection graph Γ as follows. For each hyper-link $\ell \in \mathcal{L}$ there is a node v_ℓ. Two nodes v_{ℓ_1} and v_{ℓ_2} are adjacent in the hyper-link intersection graph if and only if ℓ_1 and ℓ_2 are intersecting hyper-links. For $K \subseteq \mathcal{L}$, $\Gamma(K)$ is the hyper-link intersection graph restricted to the hyper-links in K.

The following lemma and its proof are similar to the analogous statements for standard links (Lemma 5.4 in [TZ22a]).

Lemma 3.6.3. *Let $K \subseteq \mathcal{L}$ be a collection of hyper-links, and $C \in \mathcal{C}$ a dangerous ring-cut. Then $\delta_K(C) \neq \varnothing$ if and only if there is a path in $\Gamma(K)$ from a hyper-link containing some $v \in C$ to a hyper-link containing some $w \notin C$.*

Proof. If $\delta_K(C) \neq \varnothing$, then clearly such a path in $\Gamma(K)$ exists. In particular, any single hyper-link in $\delta_K(C)$ forms a path. Conversely, suppose such a path exists in $\Gamma(K)$, but that $\delta_K(C) = \varnothing$ for contradiction. That is, any hyper-link in K is either fully contained C or fully contained in $V(H) \setminus C$. Since the path starts with a hyper-link containing $v \in C$, and ends with one containing $w \notin C$, the path must contain some pair of intersecting hyper-links ℓ_1 and ℓ_2 with ℓ_1 contained in C, while ℓ_2 is contained in $V(H) \setminus C$. Hence, clearly ℓ_1 and ℓ_2 do not share a vertex. Furthermore, since C is an interval, any vertex lying between two vertices in ℓ_1 must also lie in C, and hence not in ℓ_2. This contradicts that ℓ_1 and ℓ_2 are intersecting. □

Lemma 3.6.3 yields Lemma 3.1.6 as an immediate corollary, but we will not need Lemma 3.1.6 in this section.

The notions of v-bad and v-good were used in [TZ22a], but the definitions need to be modified for our setting.

Definition 3.6.4. Let $v \in R$ and consider the maximal interval $I_v \subseteq V(H)$ containing v such that I_v does not contain a terminal which is a non-descendant of v in $(R, \vec{F})$. We say that the nodes in I_v are ***v*-bad**, and all nodes in $V(H) \setminus I_v$ are ***v*-good**.

Observe that by Lemma 3.6.1(i), the interval of v-bad nodes actually contains all descendants of v (but may also contain some Steiner nodes in the ring).

The following lemmas are analogous to Lemmas 5.6 and 2.10, respectively, in [TZ22a]. The proofs are similar, but we include them here as our definitions of v-good and v-bad have changed.

Lemma 3.6.5. *A directed link $(u, v) \in \vec{F}$ is responsible for a cut $C \in \mathcal{C}$ if and only if $v \in C$ and all of the nodes in C are v-bad.*

Proof. If $v \in C$ and all nodes of C are v-bad, then $C \cap R$ contains only descendants of v. In particular, $u \notin C$, so (u, v) enters C, and no other link on the r–v path in $\vec{F}$ can enter C.

Conversely, if (u, v) is responsible for cut C, then it enters C, so $v \in C$. Moreover, for every $t \in C \cap R$, the r–t path in $\vec{F}$ must have a link entering C. By Lemma 3.6.2 we know that (u, v) is the only link responsible for C, so every such t must be a descendant of v. That is, $C \cap R$ are all descendants of v, and in particular, C contains no terminals which are non-descendants of v. Hence, since C is an interval, it is contained in the maximal interval containing no terminal non-descendants of v, which is precisely the set of v-bad nodes. □

This yields the main result of this section, which provides a characterization of when a directed link (u, v) can be dropped, namely when v is connected to a v-good vertex through the hyper-link intersection graph. This is the criterion we will work with in § 3.7.

Lemma 3.6.6. *For a collection of hyper-links K, a directed link (u, v) is in $\text{drop}_{\vec{F}}(K)$ if and only if $\Gamma(K)$ contains a path from a hyper-link containing v to a hyper-link containing a v-good vertex w.*

Proof. First suppose that there is such a path in $\Gamma(K)$. Then any cut $C \in \mathcal{R}_{\vec{F}}((u, v))$ for which (u, v) is responsible cannot contain w, since by Lemma 3.6.5 it only contains v-bad nodes. Hence, by Lemma 3.6.3, we have that $\delta_K(C) \neq \varnothing$.

Conversely, if v is not connected to a v-good vertex by K in the hyper-link intersection graph, then consider the set $K_v \subseteq V(H)$ of nodes reachable from v via the hyper-link intersection graph of K. Let I_{K_v} be the interval of these nodes. Then $I_{K_v} \in \mathcal{C}$, since $v \in I_{K_v}$, $r \notin I_{K_v}$ (since r is v-good) and it is a 2-cut. Moreover, since K_v are all v-bad, then I_{K_v} are as well. By Lemma 3.6.5, this implies that (u, v) is responsible for the cut I_{K_v}. We finish the proof by arguing that K does not cover I_{K_v}, implying that $(u, v) \notin \text{drop}_{\vec{F}}(K)$. This is clear from the definition of the hyper-link intersection graph: if some hyperlink in K covers I_{K_v}, then it must be intersecting with a hyperlink reachable from v in the hyper-link intersection graph, thus contradicting that K_v are all of the nodes reachable from v in the hyper-link intersection graph of K. □

3.7 The Decomposition Theorem for Hyper-SRAP

In this section, we prove the following theorem.

Theorem 3.2.3 (Decomposition Theorem)**.** *Given an instance of Hyper-SRAP $(H = (V(H), E), R, \mathcal{L})$, suppose $\vec{F}_0$ is an R-special directed solution and $S \subseteq \mathcal{L}$ is any solution. Then for any $\varepsilon > 0$, there exists a partition $\mathcal{Z}$ of S into parts so that:*

- *For each $Z \in \mathcal{Z}$, Z is α-thin for $\alpha = 4\lceil 1/\varepsilon \rceil$.*
- *There exists $Q \subseteq \vec{F}_0$ with $c(Q) \leq \varepsilon \cdot c(\vec{F}_0)$, such that for all $f \in \vec{F}_0 \setminus Q$, there is some $Z \in \mathcal{Z}$ with $f \in \text{drop}_{\vec{F}_0}(Z)$. That is, $\vec{F}_0 \setminus Q \subseteq \bigcup_{Z \in \mathcal{Z}} \text{drop}_{\vec{F}_0}(Z)$.*

We follow the approach in [TZ22a] which proves the result when all hyper-links have size 2 and $R = V(H)$. We will first partition the hyper-links in S into a collections of festoons whose spans form a laminar family. We will then construct a dependency graph whose nodes correspond to festoons, which will allow us to partition the links of S into the desired α-thin pieces. In the general hyper-link setting, the festoons are composed of hyper-links rather than links of size 2.

We begin by defining festoons in the context of hyper-links. The interval of a hyper-link ℓ is denoted I_ℓ and is the set of vertices in the interval between the leftmost vertex and the rightmost vertex of ℓ. We say that hyper-links ℓ and ℓ' are **crossing** if their intervals intersect and neither is a subset of the other. Recall that two hyper-links are **intersecting** if they share a vertex, or there is a vertex in ℓ between two vertices in ℓ' and a vertex of ℓ' lies between two vertices of ℓ (when the vertices are viewed in the left to right order along the ring). Notice that if two hyper-links are crossing, then they also are intersecting.

Definition 3.7.1. A **festoon** is a set of hyper-links $X \subseteq \mathcal{L}$ which can be ordered $\ell_1, \ldots, \ell_p$ such that ℓ_i and ℓ_{i+1} are crossing for $i \in \{1, \ldots, p-1\}$ and I_{ℓ_i} and I_{ℓ_j} are disjoint unless $|i-j| = 1$.

Notice that whether a set of hyper-links is a festoon only depends on their hyper-link intervals. Let $\mathcal{C}' = \{C \subseteq V(H) : |\delta_E(C)| = 2, r \notin C\}$ be the family of minimum cuts of the ring H. One of the key properties of festoons is that no minimum cut is covered by more than 4 hyper-links in a festoon.

Lemma 3.7.2. *Let X be a festoon and $C \in \mathcal{C}'$. Then $|\{\ell \in X : \ell \text{ covers } C\}| \leq 4$.*

Proof. Suppose $X = \ell_1, \ldots, \ell_p$ is a festoon of hyper-links and $C \in \mathcal{C}'$. Then C is an interval between vertices say u and v in $V(H)$, where u is the left endpoint of C and v is the right endpoint.

We will show that there are at most two hyper-links in $\delta_X(C)$ which contain vertices to the left of u. A symmetric argument shows that there can be at most two hyper-links in $\delta_X(C)$ containing a vertex to the right of v. Together, these imply that there are at most 4 hyper-links in $\delta_X(C)$.

Suppose there are 3 hyper-links covering C which contain vertices to to the left of u. Since they cover C, each of these hyper-links must also contain some vertex in C. Thus, their hyper-link intervals all contain the vertex u. But this is impossible, as the definition of festoons requires that I_{ℓ_i} and I_{ℓ_j} are disjoint unless $|i-j| = 1$. □

Definition 3.7.3 (α-thin set of hyper-links)**.** A set of hyper-links K is α-thin if there exists a maximal laminar subfamily $\mathcal{D}$ of $\mathcal{C}'$ such that for every cut $C \in \mathcal{D}$, we have $|\{\ell \in K : \ell \text{ covers } C\}| \leq \alpha$.

Given a festoon X, let I_X denote the festoon interval which is the interval from the left-most vertex in the festoon to the right-most. We can partition S into a collection of festoons $\mathcal{X}$, so that the festoon intervals form a laminar family. To do this, we iteratively construct the partition by choosing a festoon with the largest interval in each iteration and adding it to the partition. We now prove that if the set S is partitioned in this way, it yields a partition $\mathcal{X}$ such that the set of festoon intervals form a laminar family.

Lemma 3.7.4. *The festoon intervals $\{I_X : X \in \mathcal{X}\}$ form a laminar family.*

Proof. Suppose that $X, Y \in \mathcal{X}$ are such that I_X and I_Y cross. We also assume without loss of generality that X was added the festoon family before Y was.

Let $\ell_1, \ldots, \ell_p$ be the hyper-links in the festoon X, numbered according to the festoon order. We will assume X is to the left of Y. Since they cross, there is some hyper-link in Y whose left endpoint is in I_X. But then this hyper-link would have been included in X to form a longer interval. □

Given the laminar structure of the festoon intervals, we can define a partial order on the festoons in $\mathcal{X}$. In particular, we will say that $X \prec Y$ if $I_X \subseteq I_Y$.

Definition 3.7.5. We say that two festoons X and Y are **tangled** if some hyper-link in X is intersecting with some hyper-link of Y.

Suppose that $X_1, \ldots, X_p$ is a sequence of festoons such that X_1 contains a vertex $v \in R \setminus r$, X_i and X_j are tangled iff $|i-j| = 1$, and X_p contains a v-good vertex, but no festoons X_i for $i < p$ contain a v-good vertex. Then by Lemma 3.6.6, $\{X_1, \ldots, X_p\}$ is a minimal collection of festoons such that the directed link $(u, v) \in \vec{F}$ can be dropped. We will choose for each $v \in R$, a minimal collection of festoons $\mathcal{X}_v = \{X_1, \ldots, X_p\}$ as above so that $|I_{X_p}|$ is minimized. This will allow us to construct the dependency graph with the desired properties.

We now turn to defining the dependency graph $(\mathcal{X}, A)$. It will have a vertex for each festoon in $\mathcal{X}$, and its arcs are obtained by inserting for each directed link $(u, v) \in \vec{F}$ a directed path corresponding to a minimal set of festoons as above. Formally, for each $v \in R \setminus r$, there will be a directed path P_v consisting of the festoons in $\mathcal{X}_v$ where $(X_i, X_{i-1}) \in A$ for $i \in \{2, \ldots p\}$.

It will be helpful to consider a subgraph of the dependency graph which is constructed only from paths P_v where $v \in U$ for some set of terminals $U \subseteq R \setminus r$. This is called the **$U$-dependency graph**.

We will use that the dependency graph is a branching, which follows from the minimality in its construction and the partial order defined on festoons. Clearly, this implies that the U-dependency graph is also a branching for any $U \subseteq R \setminus r$.

Lemma 3.7.6. *The dependency graph $(\mathcal{X}, A)$ is a branching. That is, the in-degree of every node $X \in \mathcal{X}$ is at most 1 and it contains no cycles.*

Proof. Since $(X, Y) \in A$ implies $Y \prec X$, it is clear that $(\mathcal{X}, A)$ contains no directed cycles. We now show that each festoon X has in-degree at most 1. Suppose for the sake of contradiction that $(Y, X) \in A$ and $(Z, X) \in A$ where $Y \neq Z$. Since the arcs in A arise from a collection of directed paths, it must be the case that $(Y, X) \in P_v$ and $(Z, X) \in P_w$ for distinct vertices $v, w \in R \setminus r$.

We will use the following key property, used in [TZ22a], which continues to hold in our setting: for any pair of terminals $v, w \in R \setminus r$, either w is v-good or v is w-good. Hence, we assume without loss of generality that w is v-good.

Since $(Z, X) \in P_w$, we have that w is contained in the festoon interval I_X of X. Since the v-good vertices form an interval, X must contain a hyper-link which contains a v-good vertex. But by Lemma 3.6.6, this means that Y is not necessary for a directed link entering v to be dropped, contradicting $(Y, X) \in P_v$. □

We can bound the thinness of the components of the dependency graph with the following lemmas, which are direct extensions of analogous lemmas in [TZ22a]. Let α be some positive integer, and $U \subseteq R \setminus r$.

Lemma 3.7.7. *If $\mathcal{X}'$ is a connected component of the U-dependency graph and the set*

$$\{Z \in \mathcal{X}' : X \preceq Z,\ X \text{ and } Z \text{ are tangled}\}$$

has cardinality at most α for every festoon $X \in \mathcal{X}'$, then $\mathcal{X}'$ is $4(\alpha+1)$-thin.

Proof. The proof is the same as the proof of Lemma 7.16 in [TZ22a]. Note that Lemma 7.8 in [TZ22a] holds for festoons of hyper-links as well as standard festoons. □

Lemma 3.7.8. *Suppose X and Y are festoons in the same connected component of the U-dependency graph. If X and Y are tangled, then they have an ancestor-descendant relationship in the U-dependency graph.*

Proof. Again, the proof is a direct extension of Lemma 7.17 in [TZ22a]. □

Lemma 3.7.9. *Let $(\mathcal{X}', A')$ be a connected component of the U-dependency graph. If*

$$|\{u \in V : P \cap P_u \neq \emptyset\}| \leq \alpha$$

for every directed path $P \subseteq A'$ then the collection of links which are contained in festoons of $\mathcal{X}'$ is $4(\alpha+1)$-thin.

Proof. First, notice that for every $X \in \mathcal{X}$, there are at most α festoons in $\mathcal{X}'$ which are at least X in the partial ordering. This is because X can be tangled with at most one festoon from each P_u, which follows from minimality as well as Lemma 3.7.8. Hence, by Lemma 3.7.7, $\mathcal{X}'$ is $4(\alpha+1)$-thin as desired. □

Having shown that the dependency graph is a branching, the property given in Lemma 3.7.9 is enough to replicate the same argument which was introduced by Traub and Zenklusen in [TZ22b] for the Weighted Tree Augmentation Problem to prove the decomposition theorem. We break the dependency graph up into pieces, such that each corresponds to a collection of hyper-links which is α-thin, while only destroying a small fraction of the sets P_u. We reproduce the proof below.

Theorem 3.2.3 (Decomposition Theorem)**.** *Given an instance of Hyper-SRAP $(H = (V(H), E), R, \mathcal{L})$, suppose $\vec{F}_0$ is an R-special directed solution and $S \subseteq \mathcal{L}$ is any solution. Then for any $\varepsilon > 0$, there exists a partition $\mathcal{Z}$ of S into parts so that:*

- *For each $Z \in \mathcal{Z}$, Z is α-thin for $\alpha = 4\lceil 1/\varepsilon \rceil$.*
- *There exists $Q \subseteq \vec{F}_0$ with $c(Q) \leq \varepsilon \cdot c(\vec{F}_0)$, such that for all $f \in \vec{F}_0 \setminus Q$, there is some $Z \in \mathcal{Z}$ with $f \in drop_{\vec{F}_0}(Z)$. That is, $\vec{F}_0 \setminus Q \subseteq \bigcup_{Z \in \mathcal{Z}} drop_{\vec{F}_0}(Z)$.*

Proof. Let $q := \lceil \frac{1}{\varepsilon} \rceil$. We will construct an arc labeling for each connected component $(\mathcal{X}', A')$ of the $(R \setminus r)$-dependency graph $(\mathcal{X}, A)$, which is a branching by Lemma 3.7.6. The arcs in the same set P_u will receive the same label.

We define the labeling inductively as follows. For each directed path P_u which begins at the root of the arborescence $(\mathcal{X}', A')$, we set the labels of the arcs in this path to be 0. For a directed path P_u which begins at a node X which is not the root, we set the label of the arcs in P_u to be $j+1$, where j is the label of the unique arc entering X. We perform the labeling in this fashion for each connected component of the dependency graph to obtain a labeling of all arcs in A

Now, let $Q_i \subseteq \vec{F}$ be the set of directed links (u, v) such that P_v received label i, where $i \in \{0, \ldots, q-1\}$. This is a partition of $\vec{F}$ into q parts. Hence, the average cost of the sets Q_i is $c(\vec{F})/q$, implying that the cheapest set among $Q_0, \ldots, Q_{q-1}$ has cost at most $c(\vec{F})/q$. Let this cheapest part be denoted by $Q \subseteq \vec{F}$.

We define $U \subseteq R$ to be those terminals which are not entered by a directed link in Q, and consider the U-dependency graph. The U-dependency graph is obtained by deleting from $(\mathcal{X}, A)$ all arcs with label i for some $i \in \{0, \ldots, q-1\}$. Hence, each of its connected components satisfies the hypothesis of Lemma 3.7.9 where $\alpha = q - 1$, implying that the set of hyper-links in the festoons of

each component is $4q$-thin. This yields our partition of the hyper-links of S into parts where each part is $4\lceil\frac{1}{\varepsilon}\rceil$-thin.

Finally, for each directed link $(u,v) \in \vec{F} \backslash Q$ there is some connected component of the U-dependency graph which contains all the arcs in P_v. Hence, (u,v) is droppable by adding all the hyper-links in the festoons of this component. Since, $c(Q) \leq \varepsilon \cdot c(\vec{F})$, we have the desired property. □

3.8 Dynamic Programming to find the best α-thin component

In this section, we prove that, given an R-special directed solution $\vec{F}_0$, and a subset $\vec{F} \subseteq \vec{F}_0$, the α-thin collection of γ-restricted hyper-links K which minimizes the ratio

$$\frac{c(K)}{c(\text{drop}_{\vec{F}_0}(K) \cap \vec{F})}$$

can be found in polynomial time.

As is now standard following the results of [TZ22b][TZ22a][RZZ23], we will focus on maximizing $\text{slack}_\rho(K)$ defined as

$$\text{slack}_\rho(K) := \rho \cdot c(\text{drop}_{\vec{F}_0}(K) \cap \vec{F}) - c(K).$$

Notice that deciding whether the ratio $\frac{c(K)}{c(\text{drop}_{\vec{F}_0}(K) \cap \vec{F})}$ is smaller than a fixed ρ^* is equivalent to deciding whether $\text{slack}_{\rho^*}(K)$ is greater than 0. Thus, if we can efficiently find a maximizer of the slack function for any given ρ, then we can use binary search to obtain our desired result.

It will be convenient for the results in § 3.9 to prove a more general result allowing different cost functions on the two terms of the slack function.

Theorem 3.8.1. *Given an instance of SRAP, let $\vec{F}_0$ be an R-special directed solution. Let $\tilde{c} : \vec{F}_0 \to \mathbb{R}_{\geq 0}$ be a cost function on $\vec{F}_0$. Then a maximizer of $\tilde{c}(\text{drop}_{\vec{F}_0}(K)) - c(K)$ over all α-thin subsets of γ-restricted hyper-links can be computed in polynomial time.*

Given Theorem 3.8.1, we can maximize slack_ρ by setting $\tilde{c}(\ell) = \rho \cdot c(\ell)$ for $\ell \in \vec{F} \cap \vec{F}_0$ and 0 otherwise. We will prove the above theorem by dynamic programming.

Traub and Zenklusen prove this optimization theorem for the WRAP problem in [TZ22a]. The proof in our setting is an extension of their methods. The key differences in our context are twofold. First, we are working with an R-special directed solution $\vec{F}_0$ which is incident only on the terminals R, whereas in WRAP, the arborescence contains all the nodes of the ring. Secondly, we must extend their techniques to hyper-links, rather than standard undirected links containing pairs of vertices of the ring.

To handle this, we will add artificial links to the R-special solution $\vec{F}_0$ to obtain a $V(H)$-special solution $\vec{F}'$, which allows us to extend the least common ancestor function to all subsets of ring nodes. This allows us to leverage Lemma 3.8.4, which is the analogue of Lemma 2.11 in [TZ22a], and is crucial in computing new table entries from already computed ones. Finally, we exploit the fact that we are only working with hyper-links of size at most γ, implying that there are polynomially many hyper-links available.

To state Lemma 3.8.4, we will need to extend the notion of the least common ancestor to a set of ring nodes, some of which may not be involved in the arborescence $\vec{F}_0$. We first prove some useful

properties of v-bad intervals. Recall that for a terminal $v \in R$, its v-bad interval is the maximal interval containing v which does not include a terminal non-descendant of v in $\vec{F}_0$. We show that this collection of intervals is a laminar family.

Lemma 3.8.2. *Suppose $\vec{F}_0$ is an R-special solution, and let $I_v \subseteq V(H)$ denote the v-bad interval. Then the family of all v-bad intervals $\mathcal{F} = \{I_v : v \in R\}$ is laminar.*

Proof. Consider two terminals u and v with u-bad interval I_u and v-bad interval I_v. By Lemma 3.6.1(ii), the least common ancestor (in $\vec{F}_0$) w of u and v, lies between them. If w is not u or v, then it is a non-descendant of both of them, so neither I_u nor I_v can contain w. Since $u \in I_u$ and $v \in I_v$, these intervals do not intersect.

Otherwise, w is equal to either u or v; suppose $w = u$ without loss of generality. But then all non-descendants of u are non-descendants of v, so $I_v \subseteq I_u$. Thus, the family is laminar. □

We now describe how we add artificial directed links to the R-special solution $\vec{F}_0$ to obtain a $V(H)$-special solution $\vec{F}'$. We will then use $\vec{F}'$ to define a notion of least common ancestor in $\vec{F}_0$ that is defined for any subset of ring nodes.

For each terminal $v \in R$, define the set of nodes $S_v \subseteq V(H)$ to consist of all nodes u such that I_v is the minimal interval of $\mathcal{F}$ which contains u. Notice that S_v is an interval containing exactly one terminal, namely v, and potentially Steiner nodes to the left and right of v. Suppose $S_v = S_v^L \cup S_v^R \cup \{v\}$ where S_v^L and S_v^R are the (possibly empty) sets of Steiner nodes in S_v to the right and left of v respectively. Suppose $S_v^L = \{s_0^L, \ldots, s_k^L\}$ ordered right to left along the ring, and $S_v^R = \{s_0^R, \ldots, s_m^R\}$ ordered left to right along the ring. We now add the set of artificial directed links $\{(v, s_0^L), (s_0^L, s_1^L), \ldots, (s_{k-1}^L, s_k^L)\} \cup \{(v, s_0^R), (s_0^R, s_1^R), \ldots, (s_{m-1}^R, s_m^R)\}$. See Figure 3.6.

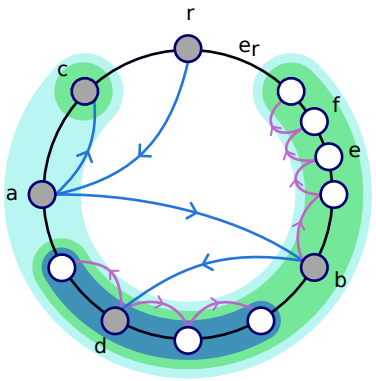

Figure 3.6: An example of an R-special solution with $R = \{r, a, b, c, d\}$ and its extension to an artificial $V(H)$-special solution. The artificial links are purple. The r-bad interval is always $V(H)$. The a-bad interval is shown in cyan. The b-bad and c-bad intervals are green, and the d-bad interval is dark blue. Note that $\overline{lca}(\{e, f\}) = e$ and $\overline{lca}(\{e, f, d\}) = b$.

When these artificial links are added to $\vec{F}_0$, we obtain an associated $V(H)$-special solution $\vec{F}'$. Observe that for a terminal v, the set of descendants of v in $\vec{F}'$ (including v itself) is exactly its v-bad interval I_v.

Definition 3.8.3 (Extended Least Common Ancestor Function)**.** Suppose $\vec{F}_0$ is an R-special solution and $A \subseteq V(H)$. Then the least common ancestor of A with respect to $\vec{F}_0$ is

$$\overline{\text{lca}}(A) := \text{lca}_{\vec{F}'}(A),$$

where $\text{lca}_{\vec{F}'}(A)$ is the least common ancestor of A in the artificial solution $\vec{F}'$.

Notice that the above definition coincides with the standard least common ancestor function with respect to $\vec{F}_0$ on sets A which consist only of terminals.

Now we can state the key Lemma 3.8.4. For a set of hyper-links S, let $V(S) := \bigcup_{\ell \in S} \ell$.

Lemma 3.8.4. *If $\vec{F}_0$ is an R-special directed solution to SRAP and $S \subseteq \mathcal{L}$ is a collection of hyper-links which form a connected component in the hyper-link intersection graph, then*

$$drop_{\vec{F}_0}(S) = \Big(\bigcup_{v \in V(S)} \delta^-_{\vec{F}_0}(v) \Big) \setminus \delta^-_{\vec{F}_0}(\overline{lca}(V(S))).$$

Proof. First, observe that if $V(S) \cap R = \varnothing$, then both sets in the lemma statement are empty, and the lemma is true. So assume that $V(S)$ contains some terminal.

Since $\delta^-_{\vec{F}_0}(v) = \emptyset$ whenever $v \notin R$, we consider some vertex $v \in V(S) \cap R$. Suppose that $v \neq \overline{\text{lca}}(V(S))$. Since $v \in V(S)$, there must be some vertex u in $V(S)$ which is not a descendant of v with respect to $\vec{F}'$. Since the set of descendants of v in $\vec{F}'$ is exactly the v-bad interval I_v, we have that u is a v-good vertex. Thus, by Lemma 3.6.6, we have that $\delta^-_{\vec{F}_0}(v) \in \text{drop}_{\vec{F}_0}(S)$, since S connects v to a v-good vertex via the hyper-link intersection graph.

On the other hand, if $v = \overline{\text{lca}}(V(S))$, then all other vertices in $V(S)$ are descendants of v in $\vec{F}'$. Again, the set of descendants of v in $\vec{F}'$ is its v-bad interval I_v, so all vertices in $V(S)$ are v-bad, and so by Lemma 3.6.6, $\delta^-_{\vec{F}_0}(v)$ is not contained in $\text{drop}_{\vec{F}_0}(S)$. □

In the following, we will build up a solution by considering subproblems defined on intervals of the ring. As such we need a definition of a hyper-link set which is an α-thin with respect to an interval $C \subseteq V(H)$.

For a ring $H = (V(H), E)$, let the set of 2-cuts not containing the root be denoted by $\mathcal{C}' := \{C \subseteq V(H) : |\delta_E(C)| = 2, r \notin C\}$. Recall that a collection of hyper-links K is α-thin if there exists a maximal laminar subfamily $\mathcal{D}$ of $\mathcal{C}'$ such that there are at most α hyper-links in K which cover C for each $C \in \mathcal{D}$.

Definition 3.8.5. A collection of hyper-links K is (α, C)-thin if there exists a maximal laminar subfamily $\mathcal{D}$ of $\mathcal{C}'$ on ground set C, such that for each $\bar{C} \in \mathcal{D}$, there are at most α hyper-links from K which cover $\bar{C}$.

We will use the notation $\delta(C)$ to denote the set of hyper-links which cover the cut $C \in \mathcal{C}'$. We maintain a table T of polynomial size with a table entry $T[C, B, \mathcal{T}, \phi, \psi]$ where:

- $C \in \mathcal{C}'$,
- $B \subseteq \delta(C)$ is of size at most α,
- $\mathcal{T}$ is a partition of B,
- $\phi : \mathcal{T} \to V$,
- $\psi : \mathcal{T} \to \{0, 1\}$.

Note that there are $|V(H)|^2$ choices for C. Since there are at most $|V(H)|^\gamma$ possible γ-restricted hyper-links, there are at most $|V(H)|^{\gamma\alpha}$ choices for B. Finally, if α is a constant then there are

constantly many choices for $\mathcal{T}$ and ψ, and polynomially many choices for ϕ. Thus, the overall dimensions of the table are polynomial.

We now describe how to interpret a subproblem corresponding to a table entry $T[C, B, \mathcal{T}, \phi, \psi]$. A subset $S \subseteq \mathcal{L}$ of hyper-links realizes this table entry if:

- The hyper-links in S contain some vertex of C,
- $\delta_S(C) = B$,
- $\mathcal{T}$ consists of the non-empty sets $S_i \cap B$, where $S_1, \ldots, S_q$ are the connected components of S in the hyper-link intersection graph,
- For each set $S_i \cap B \in \mathcal{T}$, we have $\phi(S_i \cap B) = \overline{\text{lca}}(V(S_i))$,
- For each set $S_i \cap B \in \mathcal{T}$, we have $\psi(S_i \cap B) = 1$ if and only if $\phi(S_i \cap B) \in V(S_i)$.

Thus, the table entry $T[C, B, \mathcal{T}, \phi, \psi]$ contains the maximizer and maximum value of

$$\tilde{c}\left(\text{drop}_{\vec{F}_0}(S) \cap \bigcup_{v \in C} \delta^-_{\vec{F}_0}(v)\right) - c(S)$$

over all (α, C)-thin collections of hyper-links S which realize this table entry.

Traub and Zenklusen show how to compute the table entry $T[C, B, \mathcal{T}, \phi, \psi]$ from previously computed ones in polynomial time. At a high level, we will enumerate over all possible table entries whose solutions can be combined to yield a solution to $T[C, B, \mathcal{T}, \phi, \psi]$. This is done by guessing a partition of C into two neighboring cuts C_1 and C_2 from $\mathcal{C}'$ (notice that there are at most $|V(H)|$ choices for this partition), and choices of B_1 and B_2 which are compatible with each other and also respect the (α, C)-thinness. In particular, we must have $\delta_{B_2}(C_1) = \delta_{B_1}(C_2)$ and $|\delta_{B_1 \cup B_2}(C)| \leq \alpha$. Finally, we must have $\mathcal{T}_i$, ϕ_i and ψ_i interact in such a way as to yield a solution to $T[C, B, \mathcal{T}, \phi, \psi]$ when their solutions are combined.

Suppose $(C_1, B_1, \mathcal{T}_1, \phi_1, \psi_1)$ and $(C_2, B_2, \mathcal{T}_2, \phi_2, \psi_2)$ are table entries such that C_1 and C_2 are adjacent intervals, $\delta_{B_2}(C_1) = \delta_{B_1}(C_2)$ and $|\delta_{B_1 \cup B_2}(C_1 \cup C_2)| \leq \alpha$. Then the merger of these table entries is defined as $(C, B, \mathcal{T}, \phi, \psi)$, where $C = (C_1 \cup C_2)$, $B = \delta_{B_1 \cup B_2}(C)$, and $\mathcal{T}$, ϕ, and ψ are defined as follows:

Consider the graph with vertex set $B_1 \cup B_2$ where x and y are adjacent if either: x and y are intersecting hyper-links, x and y are in B_1 and in the same set in the partition $\mathcal{T}_1$, or x and y are in B_2 and in the same set of the partition $\mathcal{T}_2$. Then $\mathcal{T}$ is the partition of $B_1 \cup B_2$ corresponding to the connected components of this graph. For any $T \in \mathcal{T}$ which by definition is equal to the union of some parts $(T^1_1, T^1_2, \ldots, T^1_{q_1})$ from $\mathcal{T}_1$ and some parts $(T^2_1, T^2_2, \ldots, T^2_{q_2})$ from $\mathcal{T}_2$, define $\phi(T) = \overline{\text{lca}}(\phi_1(T^1_1), \ldots, \phi_1(T^1_{q_1}), \phi_2(T^2_1), \ldots, \phi_2(T^2_{q_2}))$. Finally, let $\psi(T) = 1$ if there exists some T^i_j with $\psi_i(T_j) = 1$ and $\phi_i(T^i_j) = \phi(T)$. With this definition, if S_1 is a set of hyper-links which realizes $(C_1, B_1, \mathcal{T}_1, \phi_1, \psi_1)$ and S_2 realizes $(C_2, B_2, \mathcal{T}_2, \phi_2, \psi_2)$, then $S_1 \cup S_2$ realizes their merger.

Thus, to compute the optimal value for table entry $T[C, B, \mathcal{T}, \phi, \psi]$, we can enumerate over pairs $(C_1, B_1, \mathcal{T}_1, \phi_1, \psi_1)$ and $(C_2, B_2, \mathcal{T}_2, \phi_2, \psi_2)$ whose merger is $(C, B, \mathcal{T}, \phi, \psi)$. Let $\pi(C, B, \mathcal{T}, \phi, \psi)$ denote the optimal value for this table entry. Then

$$\pi(C, B, \mathcal{T}, \phi, \psi) = \max \Bigg\{ \pi(Q_1) + \pi(Q_2) + c(B_1 \cap B_2) + \sum_{u \in U_{Q_1, Q_2}} \tilde{c}(\delta^-_{\vec{F}_0}(u)) :$$

$$Q_1 = (C_1, B_1, \mathcal{T}_1, \phi_1, \psi_1), \text{ and}$$
$$Q_2 = (C_2, B_2, \mathcal{T}_2, \phi_2, \psi_2) \text{ have merger } (C, B, \mathcal{T}, \phi, \psi)\Big\},$$

where $U_{Q_1,Q_2} = \{\phi_i(T) : T \in \mathcal{T}_i \text{ with } \psi_i(T) = 1 \text{ and } \phi_i(T) \in C_i, \text{ for } i = 1, 2\} \setminus \{\bigcup_{T\in\mathcal{T}} \phi(T)\}$. Furthermore, the optimal solution S to $(C, B, \mathcal{T}, \phi, \psi)$ will be $S_1 \cup S_2$ where S_1 is the optimizer of Q_1^* and S_2 is the optimizer of Q_2^* for the best choice of Q_1^* and Q_2^* in the above maximization.

The overall solution to the problem will be found in some table entry with $C = V(H) \setminus r$. Since the table has polynomially many entries, and each can be filled in polynomial time, this proves Theorem 3.8.1.

3.9 A $(1.5 + \varepsilon)$-approximation for k-SAG

In this section, we show how to use the local search framework introduced in [TZ22c] to give a $(1.5 + \varepsilon)$-approximation algorithm for SRAP, in the case that $R = V(H)$. By the arguments in § 3.3, this yields a $(1.5 + \varepsilon)$ approximation algorithm for k-SAG. See Algorithm 4.

The algorithm begins by computing an arbitrary SRAP solution S. For each link $f \in S$, we will construct a witness set W_f which initially consists of two directed links. In each iteration of the algorithm, we add a collection of links to the solution along with their associated witness sets, and drop directed links from other witness sets. If the witness set of a link f becomes empty then f is removed from S. Throughout the algorithm, we maintain that $\vec{F} := \bigcup_{f\in S} W_f$ is a feasible directed solution. The algorithm terminates when there is no local move which substantially improves the potential function.

We now define how the initial witness sets are constructed from our arbitrary starting solution S. Suppose a link f is in a full component (U', S'), where $U \subseteq V$ and $S' \subseteq S$. Consider the eulerian tour traversing each link in S' exactly twice. This tour induces an ordering on the ring nodes in $U' \cap V(H)$, say $\{a_1, \ldots, a_k\}$. Suppose that $f \in S'$ is traversed on the Euler subpath from a_i to a_{i+1} and on the subpath from a_j to a_{j+1} (where we take $a_{k+1} := a_1$). Then the witness set W_f will consist of the two directed links (a_i, a_{i+1}) and (a_j, a_{j+1}).

Note that $\vec{F} := \bigcup_{f\in S} W_f$ is a feasible directed solution. We now update the directed links in the witness sets by iteratively shortening links as long as $\vec{F}$ remains feasible. By Theorem 2.6 in [TZ22a], at the end of this shortening process, $\vec{F}$ is an R-special directed solution.

Now, we define a potential function Φ defined on a solution S along with its witness sets.

$$\Phi(S) := \sum_{f:|W_f|=1} c(f) + \frac{3}{2} \sum_{f:|W_f|=2} c(f).$$

We also define a weight function for the directed links in witness sets. For a directed link u in some W_f, we define

$$\bar{c}(u) := \sum_{f:u\in W_f} \frac{c(f)}{|W_f|}.$$

Our algorithm then proceeds in step 6 by choosing an α-thin collection of hyper-links maximizing $\bar{c}(\text{drop}_{\vec{F}}(Z)) - 1.5 \cdot c(Z)$. The links in the full components corresponding to these hyper-links are added to our solution S, while the directed links in $\text{drop}_{\vec{F}}(Z)$ are removed from witness sets.

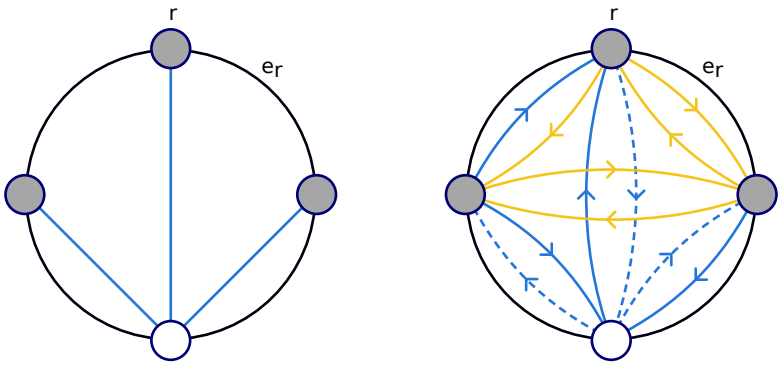

Figure 3.7: An example of a SRAP instance on the left, where all three undirected links have cost 1. The grey nodes are terminals. The completed instance is shown on the right, where the blue links have cost 1 and the orange links have cost 2 (only the directed links are shown). The dashed blue arcs form a feasible directed solution of cost 3, but there is no R-special solution of cost at most 3.

Finally, witness sets are constructed for the new undirected links which were added to the solution, and all directed links in witness sets are shortened so that $\vec{F}$ becomes an R-special solution once more.

Remark 3.9.1. In part (iv) of step 6 of Algorithm 4, it is crucial that $V(H) = R$, since in this case, any feasible directed solution can be iteratively shortened to obtain an R-special solution of at most the cost. This does not hold if $R \neq V(H)$, see Figure 3.7.

Notice that Theorem 3.8.1 implies that the maximization step in step 6 of Algorithm 4 can be performed in polynomial time. The following lemma, whose proof is identical to Lemma 9.3 in [TZ22a], shows that the algorithm will terminate after polynomially many iterations. Let OPT denote the cost of the optimal augmentation.

Lemma 3.9.2. *Algorithm 4 runs for at most*

$$\ln\left(\frac{1.5 \cdot c(S)}{OPT}\right) \cdot \frac{6|R|}{\varepsilon}$$

iterations where S is the initial SRAP solution.

Finally, the solution returned has cost at most $(1.5 + \varepsilon)$OPT.

Lemma 3.9.3. *At the end of Algorithm 4, we have $c(S) \leq (1.5 + \varepsilon)OPT$.*

The proof of the above lemma uses the same potential function analysis which is has been used in [RZZ23][TZ22a][TZ22c], so we do not reproduce it again here. These yield the main result of this section:

Theorem 3.9.4. *Algorithm 4 is a $(1.5 + \varepsilon)$-approximation algorithm for SRAP when $R = V(H)$.*

Corollary 3.9.5. *There is a polynomial time $(1.5 + \varepsilon)$-approximation algorithm for k-SAG.*

Algorithm 4: Local search algorithm for SRAP

Input: A complete instance of SRAP with graph $G = (V, E \dot{\cup} L)$, ring $H = (R, E)$, and $c : L \to \mathbb{R}$. Also a constant $1 \geq \varepsilon > 0$.

Output: A solution $S \subseteq L$ with $c(S) \leq (1.5 + \varepsilon)\mathrm{OPT}$.

1. Compute an arbitrary SRAP solution $S \subseteq L$. Construct witness sets W_f for each $f \in S$ so that $\vec{F} := \bigcup_{f \in S} W_f$ is an R-special directed solution.
2. Let $\varepsilon' := \frac{\varepsilon/2}{1.5+\varepsilon/2}$ and $\gamma := 2^{\lceil 1/\varepsilon' \rceil}$.
3. For each $A \subseteq R$ where $|A| \leq \gamma$, compute the cheapest full component joining A and denote the cost by c_A.
4. Create an instance of γ-restricted Hyper-SRAP on ring $H = (R, E)$ with hyper-links $\mathcal{L} = \{\ell_A : A \subseteq R, |A| \leq \gamma\}$. Set the cost of hyper-link ℓ_A to be c_A.
5. Let $\alpha := \lceil 8/\varepsilon \rceil$
6. Iterate the following as long as $\Phi(S)$ decreases in each iteration by at least a factor $(1 - \frac{\varepsilon}{12n})$:
   (i) Compute the α-thin subset of hyper-links $Z \subseteq \mathcal{L}$ maximizing $\bar{c}(\mathrm{drop}_{\vec{F}}(Z)) - 1.5 \cdot c(Z)$, where $\vec{F} := \bigcup_{f \in S} W_f$.
   (ii) Update the witness sets by replacing each W_f with $W_f \setminus \mathrm{drop}_{\vec{F}}(Z)$.
   (iii) Update S by adding all links of full components corresponding to the hyper-links in Z.
   (iv) Shorten directed links in $\vec{F}$ to obtain an R-special solution. If $W_f = \emptyset$ for some $f \in S$, then remove f from S.
7. **Return** S.

Chapter 4

On Small Depth Tree Augmentations

In this chapter, we study the integrality gap of the ODD-LP for Weighted Tree Augmentation Problem for general link costs. We show that the integrality gap of the ODD-LP relaxation for the (weighted) Tree Augmentation Problem for a k-level tree instance is at most $2 - \frac{1}{2^{k-1}}$. For 2- and 3-level trees, these ratios are $\frac{3}{2}$ and $\frac{7}{4}$ respectively. Our proofs are constructive and yield polynomial-time approximation algorithms with matching guarantees.

This chaper is based on joint work with Ojas Parekh and R. Ravi, and appeared in Operations Research Letters [PRZ22].

While TAP is well studied in both the weighted and unweighted case [FJ81, KT93, Rav94, CN13a, CG15, KN16, Adj17, FGKS18, GKZ18], it is NP-hard even when the tree has diameter 4 [FJ81] or when the set of available links form a single cycle on the leaves of the tree T [CJR99], and is also APX-hard [KKL04]. Weighted TAP was one of the simplest network design problems without a better than 2-approximation in the case of general (unbounded) link costs and arbitrary depth trees, until very recently [TZ21a, TZ21b]. For the case of n-node trees with height k, Cohen and Nutov [CN13a] gave a $(1 + \ln 2) \simeq 1.69$-approximation algorithm that runs in time $n^{3^k} \cdot poly(n)$ using an idea of Zelikovsy for approximating Steiner trees. Very recently, this approach has been extended to provide an approximation to the general case of the problem with the same performance guarantee by Traub and Zenklusen [TZ21a]. A follow-up paper by the same authors [TZ21b] improved the approximation ratio to nearly 1.5. However, these papers do not provide any new results on the integrality gap of some natural LP relaxations for the problem that we discuss next.

4.1 Preliminaries

4.1.1 The cut-LP Relaxation

TAP can also be viewed as a set covering problem. The edges of the tree T define a laminar collection of cuts that are the elements to be covered using sets represented by the links. A link ℓ is said to *cover* an edge e if the unique cycle of $\ell + T$ contains e. Here we use $\text{cov}(e)$ for a tree edge e to denote the set of links which cover e. The natural covering linear programming relaxation for the problem, cut-LP, is a special instance of a set covering problem with one requirement (element) corresponding to each cut edge in the tree. Since the tree edges define subtrees under them (after rooting it at an arbitrary node) that form a laminar family, this is also equivalent to a laminar

cover problem [CJR99].

$$\min \sum_{\ell \in E} c_\ell x_\ell$$

$$x(\text{cov}(e)) \geq 1 \qquad \forall e \in E(T) \tag{4.1}$$

$$x_\ell \geq 0 \qquad \forall \ell \in E \tag{4.2}$$

Fredrickson and Jájá showed that the integrality gap for cut-LP can not exceed 2 [FJ81] and also studied the related problem of augmenting the tree to be two-node-connected (biconnectivity versus bridge-connectivity augmentation) [FJ82]. Cheriyan, Jordán, and Ravi, who studied half-integral solutions to cut-LP and proved an integrality gap of $\frac{4}{3}$ for such solutions, also conjectured that the overall integrality gap of cut-LP was at most $\frac{4}{3}$ [CJR99]. However, Cheriyan et al. [CKKK08] later demonstrated an instance for which the integrality gap of cut-LP is at least $\frac{3}{2}$.

4.1.2 ODD-LP Relaxation

Fiorini et al. studied the relaxation consisting of all $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts of the cut-LP [FGKS18]. We call their extended linear program the ODD-LP.

We define $\delta(S)$ for $S \subset V$ as the set of all links and edges with exactly one endpoint in S, and recall that $\text{cov}(e)$ for a tree edge e is the set of links that cover e. We use $E(T)$ to refer to the set of tree edges, and L is the set of links, $E(G) \setminus E(T)$.

$$\min \sum_{\ell \in E} c_\ell x_\ell$$

$$x(\delta(S) \cap L) + \sum_{e \in \delta(S) \cap E(T)} x(\text{cov}(e)) \geq |\delta(S) \cap E(T)| + 1 \qquad \forall S \subseteq V, |\delta(S) \cap E(T)| \text{ is odd} \tag{4.3}$$

$$x_\ell \geq 0 \qquad \forall \ell \in E$$

We describe here the validity of the constraints in ODD-LP using a proof due to Robert Carr. Consider a set of vertices S such that $|\delta(S) \cap E(T)|$ is odd. By adding together the edge constraints for $\delta(S) \cap E(T)$ we get:

$$\sum_{e \in \delta(S) \cap E(T)} x(\text{cov}(e)) \geq |\delta(S) \cap E(T)|$$

Now we can add any non-negative terms to the left hand side and still remain feasible. Therefore

$$x(\delta(S) \cap L) + \sum_{e \in \delta(S) \cap E(T)} x(\text{cov}(e)) \geq |\delta(S) \cap E(T)|$$

is also feasible. Now consider any link ℓ. If x_ℓ appears an even number of times in $\sum_{e \in \delta(S) \cap E(T)} x(\text{cov}(e))$ then ℓ is not in $\delta(S)$. Similarly, if x_ℓ appears an odd number of times in $\sum_{e \in \delta(S) \cap E(T)} x(\text{cov}(e))$ then ℓ is in $\delta(S)$. So, the coefficient of every x_ℓ on the left hand side of this expression is even. In particular, for any integer solution the left hand side is even and the right hand side is odd. Therefore, we can strengthen the right hand side by increasing it by one, and the resulting constraint will still be feasible for any integer solution. The constraint,

$$x(\delta(S) \cap L) + \sum_{e \in \delta(S) \cap E(T)} x(\text{cov}(e)) \geq |\delta(S) \cap E(T)| + 1$$

is thus valid for any integer solution to TAP as desired.

We prove a lemma about how a transformation of a feasible solution to the ODD-LP that we call a link subdivision continues to preserve feasibility, that we use later in our integrality gap proofs.

Lemma 4.1.1. *Let (T, L) be a TAP instance and let $\bar{x}$ be a feasible solution to ODD-LP(T, L). Fix some link $\ell = (a, b) \in L$, and let $(a, v_1, \ldots, v_k, b)$ be an ordered sequence of distinct nodes on the path from a to b in T. Let $\Lambda = \{(a, v_1), (v_1, v_2), \ldots (v_{k-1}, v_k), (v_k, b))\}$ and let $L' = (L \setminus (a, b)) \cup \Lambda$ (Call this a link subdivision). Then the solution $\bar{y}$ obtained from $\bar{x}$ by setting $\bar{y}_\ell = \bar{x}_{(a,b)}$ for all $\ell \in \Lambda$ is feasible for ODD-LP(T, L').*

Proof. Fix a set of vertices $S \subseteq V$ such that $|\delta(S) \cap E(T)|$ is odd. Since $\bar{x}$ is feasible, we have

$$\bar{x}(\delta(S) \cap L) + \sum_{e \in \delta(S) \cap E(T)} \bar{x}(\text{cov}(e)) \geq |\delta(S) \cap E(T)| + 1.$$

Since the links in Λ cover exactly the set of edges along the path between a and b in T, we have

$$\sum_{e \in \delta(S) \cap E(T)} \bar{x}(\text{cov}(e)) = \sum_{e \in \delta(S) \cap E(T)} \bar{y}(\text{cov}(e)).$$

We now consider two cases. If $a, b \in S$, then $(a, b) \notin \delta(S) \cap L$, and so clearly $\bar{y}(\delta(S) \cap L) \geq \bar{x}(\delta(S) \cap L))$. Otherwise, suppose without loss of generality that $a \in S$ and $b \notin S$. Then the path between a and b in T contains an odd number of edges in $\delta(S) \cap E(T)$. Each link in Λ which is not $\delta(S) \cap L$ covers an even number of edges in $\delta(S) \cap E(T)$. Therefore, there must be some link ℓ in Λ contained in $\delta(S) \cap L$. This link has value $\bar{y}_\ell = \bar{x}_{(a,b)}$, hence we have

$$\bar{y}(\delta(S) \cap L) \geq \bar{x}(\delta(S) \cap L),$$

and the claim follows. □

We will use the following theorem about the ODD-LP [FGKS18]. For a choice of a root r, we call links which connect two different components of $T - r$ as cross-links, and those that go from a node of T to its ancestor as up-links.

Theorem 4.1.2 ([FGKS18], Theorem 1.1)**.** The ODD-LP is integral for weighted TAP instances that contain only cross- and up-links.

The integrality of the formulation is shown by demonstrating that the constraint matrix is an example of a binet matrix [AK06, AKPP07], a generalization of network matrices that are a well-known class of totally unimodular matrices. Moreover, while general Chvátal-Gomory closures are NP-hard to optimize over, these restricted versions over half-integral combinations can be optimized in polynomial time [CF96]. Such instances with only cross- and up-links are informally called "star-shaped" with the center of the star being the chosen root, so we will refer to the above result as saying that the ODD-LP for star-shaped instances centered at a root have integrality gap 1 and solutions to such instances can be obtained in polynomial time.

Without loss of generality, we may consider TAP instances where all links go between two leaves [IR17]. We reproduce the proof here for completeness.

Lemma 4.1.3. *Given an instance (T, L, c) of weighted TAP, there is a corresponding, polynomial-sized instance (T', L', c') with all links having both endpoints as leaves, such that there is a cost-preserving bijection between the solutions to the two instances.*

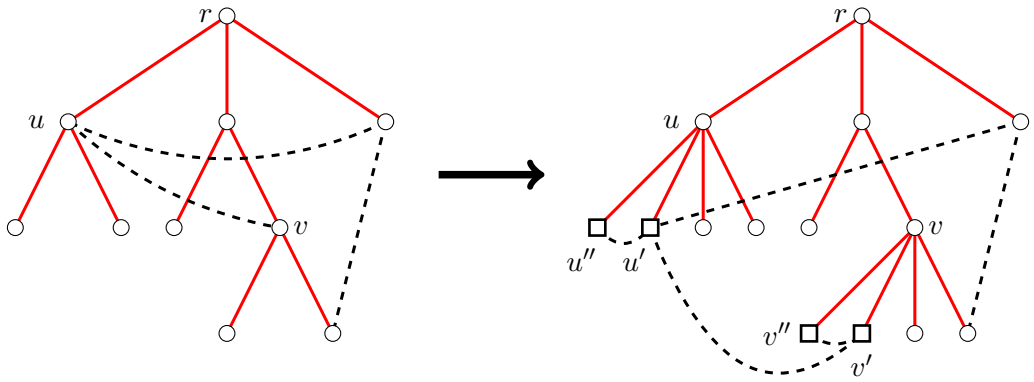

Figure 4.1: Transformation to a leaf to leaf instances

Proof. The proof proceeds by a simple graph reduction. Suppose we are given an instance defined by a graph G with associated tree T for the weighted TAP. We create a new instance of the leaf-to-leaf version as follows: For every internal node u in the original tree T, we add two new leaf nodes u' and u'' both adjacent to u to get a new tree T'. For every link $f = (v, u)$ in the original instance, we reconnect the link to now end in the leaf u' rather than the internal node u in the tree T'. Thus, if both v and u are internal nodes, the new link is (v', u'); if only u is internal, the new link is (v, u') and if both are leaves, the new link is the same (u, v) as in G. Note that the new graph G' is a leaf-to-leaf instance. In addition, for every internal node u in the original tree T, we add a new link of zero cost between u' and u'' - this will serve to cover the newly added edges (u, u') and (u, u'') without changing the coverage of any of the edges in the original tree T. See Figure 4.1.

□

Remark 4.1.4. The cost-preserving bijection described above can be extended to map fractional solutions of odd-LP(T, L) to odd-LP(T', L'). In other words, every weighted TAP problem can be reduced to an instance where all links go between a pair of leaves without loss of generality for investigating approximation ratios for the problem and integrality gaps of the odd-LP.

Note that given a rooted tree of k levels (i.e., the maximum distance of any leaf from the root is k), the above transformation results in a leaf-to-leaf instance also with k levels.

4.2 Improved Integrality Gaps for Trees of depth 2 and 3

Theorem 4.2.1. The integrality gap of the ODD-LP for a two-level tree instance is at most $\frac{3}{2}$. A solution with this approximation ratio for these instances can be obtained in polynomial time.

Proof. Given a two-level TAP instance on tree T and links L, we first show how to transform the instance into two instances that we can solve exactly. We then apply the same transformation to the components of any integral solution A to obtain feasible solutions to the two new instances, the better of which has value at most $\frac{3}{2} \cdot c(A)$. We will then use the same reductions on fractional feasible solutions to the ODD-LP, and use Lemma 4.1.1 to show that the resulting star-shaped instances are feasible for the ODD-LP and then use Theorem 4.1.2 to arrive at the stated integrality gap and resulting algorithm.

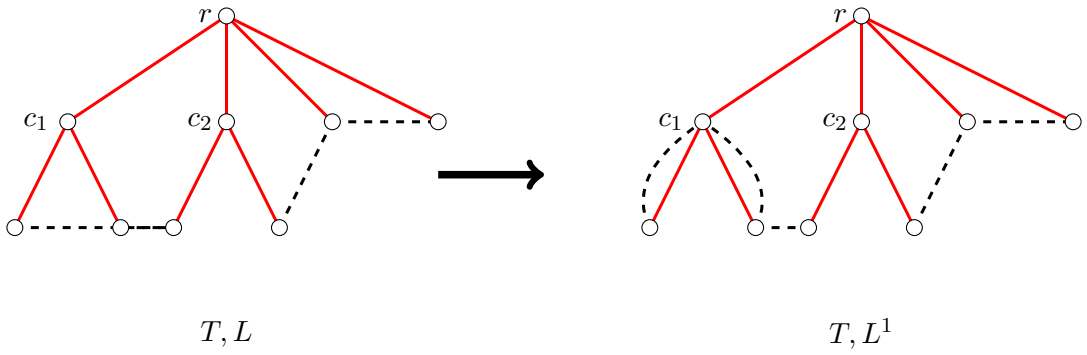

Figure 4.2: Transformation to a star-shaped instance for the root

We say that the root r is at level 1 and its children $\{c_1, c_2, \ldots, c_d\}$ are internal nodes at level 2, where d is the number of non-leaf children of the root. First using Lemma 4.1.3, we assume that all links go between a pair of leaves. We partition L into $L_1 \dot\cup L_2$, where L_i is the set of links whose least common ancestor (henceforth lca) is a node at level i in the tree.

We will create two new instances (T, L^1) and (T, L^2). The first instance (T, L^1) is obtained by replacing every link (u, v) in L_2 with lca c say, with two up-links (u, c) and (v, c) of the same cost. Notice that (T, L^1) is a star-shaped instance with root r. The second instance (T, L^2) is obtained by replacing every link (u, v) in L_1 with lca the root r, with two up-links (u, r) and (r, v) of the same cost. (T, L^2) can be decomposed into $d + 1$ disjoint star shaped instances: d of these are defined by the star around each non-leaf child of the root, and the last is the star defined by the root and its leaf-children. Thus, (T, L^2) is a disjoint collection of star-shaped instances which can be solved by solving each of these $d + 1$ induced star-shaped instances separately and taking the union of their solutions. See Figures 4.2 and 4.3.

Given an optimal solution A to (T, L), we now construct two solutions A^1 and A^2, feasible to (T, L^1) and (T, L^2) respectively, by applying the same reduction as above to the corresponding components in A. Let $A = A_1 \dot\cup A_2$ where A_i the set of links whose lca is a node in level i of the tree. To create A^1, we replace each link (u, v) in A_2 with the two up-links (u, c) and (v, c) in L^1. Note that this set of links along with A_1 gives a feasible solution to (T, L^1) and has cost $c(A_1) + 2c(A_2)$. To create A^2, we replace each link (u, v) in A_1 with the two up-links (u, r) and (v, r) in L^2. Note that this set of links along with A_2 gives a feasible solution to (T, L^2) and has cost $2c(A_1) + c(A_2)$.

As described above, each of A^1 and A^2 can be transformed into a feasible solution to (T, L) with the same cost. Therefore, by taking the better of the two, there is a solution of cost at most $\min\{c(A_1) + 2c(A_2), 2c(A_1) + c(A_2)\} \leq \frac{3}{2}c(A)$.

Note that the link replacements in both transformations are link subdivisions. Hence, by Lemma 4.1.1, we can apply the same transformation to any fractional solution feasible to ODD-LP(T, L) to obtain two fractional solutions feasible to ODD-LP(T, L^1) and ODD-LP(T, L^2) respectively. Since the resulting star shaped instances have integrality gap 1 by Theorem 4.1.2, the claims about the integrality gap and a polynomial-time approximation algorithm with this ratio also follow.

□

Theorem 4.2.2. The integrality gap of the odd-LP for a three-level tree instance is at most $\frac{7}{4}$. A solution with this approximation ratio for these instances can be obtained in polynomial time.

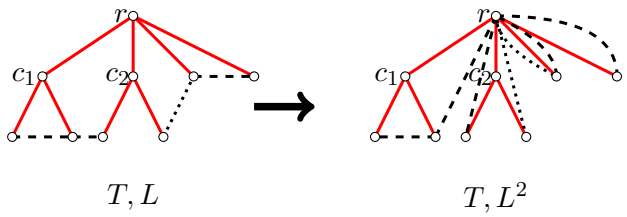

Figure 4.3: Transformation to three star-shaped instances around the root and its two internal children

Proof. As before, we first show how to transform the instance into new instances, and show that the same applied to any integral solution A gives a solution of value at most $\frac{7}{4} \cdot c(A)$. Again, the same reduction will also apply to fractional solutions that obey the ODD-LP constraints to prove the claim.

Using Lemma 4.1.3, we assume that all links go between a pair of leaves. We partition L into $L_1 \dot\cup L_2 \dot\cup L_3$, where L_i is the set of links whose lca is a node at level i in the tree. We say that the root r is at level 1 and its non-leaf children $\{c_1, c_2, \ldots, c_d\}$ are at level 2, and the children of these nodes that are internal nodes are in level 3 of the tree. We will create three new instances (T, L^1), (T, L^2) and (T, L^3) as follows.

- The first instance (T, L^1) is obtained by replacing every link (u, v) in $L_2 \cup L_3$ with lca c say, with two up-links (u, c) and (v, c) of the same cost. See Figure 4.4. Notice that (T, L^1) is a star-shaped instance with root r.
- The second instance (T, L^2) is obtained by replacing every link (u, v) in $L_1 \cup L_3$ with lca c say, with two up-links (u, c) and (c, v) of the same cost. Notice, that (T, L^2) can be decomposed into $d + 1$ different star shaped instances. One of these star-shaped instances is a one-level instance centered at the root and consisting of the root and all of its leaf children. Each of the remaining d instances corresponds to a non-leaf child of the root v_i, consisting of the whole subtree T_i rooted at v_i and its edge to the root (v_i, r). Notice that these instances are pairwise disjoint and are star-shaped with centers v_i. See Figure 4.5.
- Finally, the third instance (See Figure 4.6), is obtained as follows. For each link $(a, b) \in L_2$, with lca c say, we replace it with two up-links (a, c) and (b, c) of the same cost. The interesting transformation is for links in L_1, where we now make up to *three* copies. For every link $(a, b) \in L_1$, let c_a and c_b denote the ancestor of a and b respectively in level 2. (if either a or b is in level 2 itself, then its ancestor in level 2 is itself). We now add three links $(a, c_a), (c_a, c_b), (c_b, b)$ of the same cost as (a, b). Again, this instance can be solved exactly by decomposing it into pairwise disjoint star-shaped instances: one for each stars around the internal nodes, say $v_1, \ldots, v_q$ in level 3, and one more tree around the root consisting of the set of all tree edges not in the stars around the v_i's.

Given an optimal solution A to (T, L), we now construct three solutions A^1, A^2 and A^3, feasible to (T, L^1), (T, L^2) and (T, L^3) respectively. Let $A = A_1 \dot\cup A_2 \dot\cup A_3$ where A_i the set of links whose lca is a node in level i of the tree. The solutions A^1, A^2 and A^3 are obtained from by transforming the links in A exactly as we transformed the corresponding type of links in L to obtain L^1, L^2 and L^3. Finally, each of A^1, A^2 and A^3 can be transformed into a feasible solution to (T, L) with the same cost. Therefore, by taking the best of the three, we can find a solution in polynomial time with cost at most $\min\{c(A_1) + 2c(A_2) + 2c(A_3), 2c(A_1) + c(A_2) + 2c(A_3), 3c(A_1) + 2c(A_2) + c(A_3)\} \leq \frac{7}{4}c(A)$.

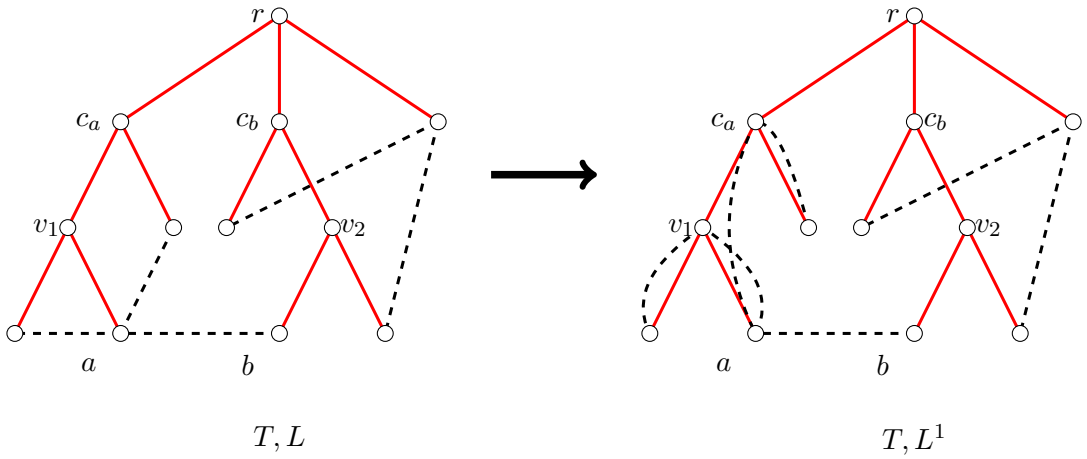

Figure 4.4: Transformation to a star-shaped instance centered at the root

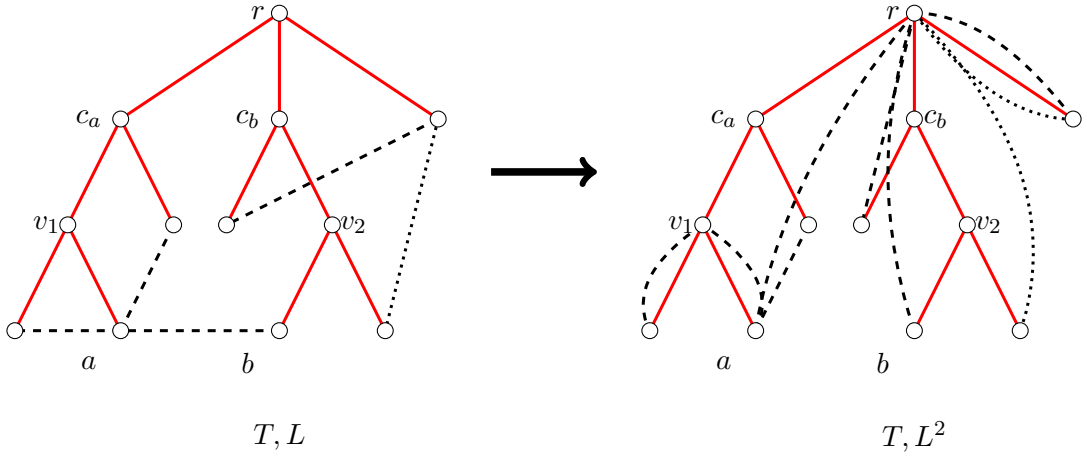

Figure 4.5: Transformation to three star-shaped instances centered at the root and its two internal children

Note that the link replacements in both transformations are link subdivisions. Hence, by Lemma 4.1.1, we can apply the same transformation to any fractional solution feasible to ODD-LP(T, L) to obtain three fractional solutions feasible to ODD-LP(T, L^1), ODD-LP(T, L^2), and ODD-LP(T, L^3) respectively. Since the resulting star shaped instances have integrality gap 1 by Theorem 4.1.2, the claims about the integrality gap and a polynomial-time approximation algorithm with this ratio also follow.

□

4.3 Integrality gap for k-level trees

With the above cases, we can now calculate an upper bound on the value of the integrality gap for general k-level trees where the depth of any leaf from the root is k.

Theorem 4.3.1. The integrality gap of the ODD-LP for a k-level tree instance is at most $2 - \frac{1}{2^{k-1}}$. A solution with this approximation ratio for these instances can be obtained in polynomial time for any fixed k.

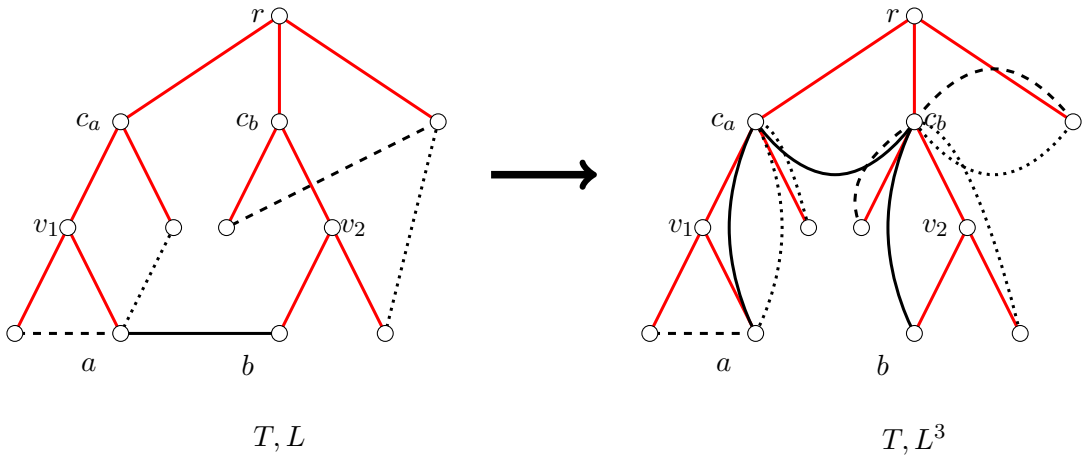

Figure 4.6: Transformation to three star-shaped instances centered at the root and the stars around the two internal nodes in level 3

Proof. We will show how to transform a given k-level TAP instance (T, L) into k new instances, and show that the same applied to any integral solution A gives a solution of value at most $(2 - \frac{1}{2^{k-1}}) \cdot c(A)$. Again, the same reduction will also apply to fractional solutions that obey the ODD-LP constraints to prove the claim.

Partition the links in L into subsets of links $L = L_1 \dot\cup L_2 \dots \dot\cup L_k$ where L_ℓ is the subset whose lca is a node in level l of the tree for $\ell = 1, \dots, k$. As before, we set up k new TAP instances $(T, L^1), \dots, (T, L^k)$, where in (T, L^ℓ) we preserve the links in L_ℓ.

For $\ell = 1$, we replace each link (u, v) in $L_2, \dots, L_k$ with two links (u, c) and (v, c), where c is the lca of u and v, yielding a star-shaped instance (T, L^1).

Suppose $1 < \ell \leq k$. For any link $(a, b) \in L_{\ell-1} \cup \bigcup_{p>\ell} L_p$, we replace it with the two links $(a, lca(a, b))$ and $(b, lca(a, b))$. For links $(a, b) \in L_q$ for $1 < q < \ell - 1$, let the ancestors of a and b in level $\ell - 1$ be u_a and u_b respectively, if they exist. We replace (a, b) with one of the following sets, with at most four links: $\{(a, u_a), (u_a, lca(u_a, u_b)), (lca(u_a, u_b), u_b), (u_b, b)\}$, or $\{(a, lca(a, u_b)), (lca(a, u_b), u_b), (u_b, b)\}$, or $\{(a, u_a), (u_a, lca(u_a, b)), (lca(u_a, b), b)\}$, or $\{(a, lca(a, b)), (lca(a, b), b)\}$, depending on which of u_a and u_b exist. Analogously, for $q = 1$, we instead use the following sets, with at most three links: $\{(a, u_a), (u_a, u_b), (u_b, b)\}$, $\{(a, u_b), (u_b, b)\}$, $\{(a, u_a), (u_a, b)\}$, $\{(a, b)\}$. Denote the obtained instance by (T, L^ℓ).

We now show that (T, L^ℓ) can be decomposed into pairwise disjoint star-shaped instances, thereby allowing us to solve these instances exactly. For every internal node v at level ℓ, we consider the instance on the subtree below it, along with the edge to its parent. These will be star-shaped instances with centers v. In addition, we consider a final instance, which will be a star-shaped instance around the root, whose tree edges are disjoint from the others.

We will show that each link is a cross-link or an up-link in one of the star-shaped instances. First consider the instances around the internal nodes v in level ℓ. Links in L_ℓ are already cross-links in these. Links in L_p for $p > \ell$ have been replaced with two links that become up links in these instances. Consider a link $(a, b) \in L_{l-1}$, such that v_a and v_b are the ancestors of a and b respectively that are in level l. We replaced this link with the two links $(a, lca(a, b))$ and $(b, lca(a, b))$. Now $lca(a, b)$ is a parent of v_a and v_b since $(a, b) \in L_{l-1}$ so these links form cross links for the star-shaped instances around v_a and v_b.

All the tree edges not in any of these star-shaped instances are considered in a final star-shaped instance rooted at r. For links $(a, b) \in L_q$ for $1 < q < \ell - 1$, let the ancestors of a and b in level $\ell - 1$ be u_a and u_b respectively, if they exist. For $q > 1$, all the links in these links become cross links for the star-shaped instances around the level ℓ internal nodes or up links for the instance rooted at r. The same holds for links in L_1, except that these sets also include cross links for the instance rooted at r.

Given an optimal solution A to (T, L), we can construct k solutions A^1, A^2, ..., A^k, feasible to (T, L^1), (T, L^2), ..., (T, L^k) respectively. To get A^k, we simply apply the same transformations to the links of A which were applied to L to obtain L^k. Let $A = A_1 \dot\cup A_2, \cdots, \dot\cup A_k$, where A_i is the set of links in A with lca at level i, and let c_i denote the cost of the links in A_i. Each A^ℓ corresponds to a solution to (T, L) with at most the cost.

Based on the above construction, an upper bound on the cost of this set of candidate solutions is

$$\begin{aligned} C_1 &= c_1 + 2c_2 + 2c_3 + 2c_4 + \ldots + 2c_k, \text{ if } \ell = 1 \\ C_2 &= 2c_1 + c_2 + 2c_3 + 2c_4 + \ldots + 2c_k, \text{ if } \ell = 2 \\ C_3 &= 3c_1 + 2c_2 + c_3 + 2c_4 + \ldots + 2c_k, \text{ if } \ell = 3 \\ C_\ell &= 3c_1 + 4c_2 + \ldots + 4c_{\ell-2} + 2c_{\ell-1} + c_\ell + 2c_{\ell+1} + \ldots + 2c_k, \; if \;\; 3 < \ell \leq k. \end{aligned}$$

The best of these solutions has cost at most $\min\{C_1, \ldots, C_k\} \leq \sum_{i=1}^k \lambda_i C_i$ for any setting of weights such that $\lambda_i \geq 0$ for all i and $\sum_{i=1}^k \lambda_i = 1$.

Let $J(n) := \frac{2^n - (-1)^n}{3}$. Let $\lambda_i = \frac{1}{2^k - 1} J(k - i + 1)$ for $i \geq 2$. If k is even, let $\lambda_1 = \lambda_2$. If k is odd, let $\lambda_1 = \lambda_2 + 1$. With this choice of $\lambda = (\lambda_1, \ldots, \lambda_k)$, it is straightforward to verify that

$$\sum_{i=1}^k \lambda_i C_i = \left(\frac{2^k - 1}{2^{k-1}} \right) (c_1 + \ldots + c_2).$$

The key observation to verify this is that $J(n)$ satisfies the recurrence $J(n) = J(n-1) + 2J(n-2)$ for $n \geq 2$, which reflects the changes in the coefficients of $c_\ell, c_{\ell-1}$ and $c_{\ell-2}$ in going from $C_{\ell-1}$ to C_ℓ. With $J(1) = 1$ and $J(0) = 0$, this defines the Jacobsthal sequence, see (https://oeis.org/A001045).

Thus, we have $\min\{C_1, \ldots, C_k\} \leq \left(2 - \frac{1}{2^{k-1}}\right)(c_1 + \ldots + c_k) = \left(2 - \frac{1}{2^{k-1}}\right) c(A)$.

As before, all link replacements correspond to link subdivisions. Thus, by Lemma 4.1.1, we can apply the same transformation to any fractional solution feasible to ODD-LP(T, L) to obtain k fractional solutions feasible to ODD-LP(T, L^ℓ) for each $\ell = 1, \ldots, k$. Since the resulting star shaped instances have integrality gap 1 by Theorem 4.1.2, the claims about the integrality gap and a polynomial-time approximation algorithm with this ratio also follow. □

While the above analysis shows integrality gaps of the ODD-LP converging to 2 as the depth of the tree grows, the main open question in our opinion is to show that the integrality gap of $\frac{3}{2}$ that we showed for 2-level trees is indeed the upper bound for all trees.

Chapter 5

The Base Augmentation Problem

In this chapter, we study the Base Augmentation Problem. This is based on joint work with Madhusudhan Reddy Pittu. Let $M = (E, \mathcal{I})$ be a matroid with ground set E and independent sets $\mathcal{I}$. Recall that the rank function $r_M : 2^E \to \mathbb{Z}_{\geq 0}$ indicates the size of the largest independent set which can be found within a set $S \subseteq E$. The rank of the matroid is defined as $r_M(E)$ and is denoted as r.

A subset $S \subseteq E$ is 1-**robust** if $r_M(S \setminus e) = r$ for all $e \in S$. Notice that in order to be 1-robust, S itself must be rull rank. Hence, this definition essentially states that S is 1-robust if it is still a full rank set even after any single element is removed.

Problem 5.0.1 (Base Augmentation Problem)**.** We are given a base $B \subseteq E$ of a matroid $M = (E, \mathcal{I})$, and a cost function $c : (E \setminus B) \to \mathbb{R}_{\geq 0}$. The goal is to choose a minimum cost set $F \subseteq E \setminus B$ so that $B \cup F$ is a 1-robust set.

We can formulate BAP as a covering problem. When an element $e \in (E \setminus B)$ is added to base B, a unique circuit is present in $B \cup e$. This is called the **fundamental circuit** of e with respect to B. We define an analogous **fundamental path** of an element e with respect to B to be the restriction of its fundamental circuit to B. For a fixed base, we denote the fundamental path of element e by P_e.

With these definitions, we can phrase the BAP problem as follows. Let A be the $\{0,1\}$-matrix whose columns are indexed by the elements of rows are indexed by the elements of B, and whose columns are indexed by the elements of $E \setminus B$ such that $A_{e,f} = 1$ if $e \in P_f$ and 0 otherwise. This is called the base-representation matrix of the matroid M with base B.

Then the BAP problem is

$$\min\{cx : Ax \geq 1, x \in \{0,1\}^{(|E|-r)}\}.$$

We now turn to studying the approximability and integrality gap of this program for common classes of matroids.

5.1 Set-Cover Hardness for Binary Matroids

A matroid is a "linear" or "representable" matroid if its independent sets are the subsets of linearly independent columns of some matrix over a field $\mathbb{F}$. A **binary matroid** is a matroid which is

representable over the field with two elements $\mathbb{F}_2$.

We will show that there is a Set-Cover hardness for the Base Augmentation Problem in the case that the matroid is a binary matroid. To do this, we show that given an arbitrary instance of Set Cover, we can provide an equivalent instance of BAP on a binary matroid with a specified base B.

Theorem 5.1.1. *The Base Augmentation Problem for binary matroids is equivalent to Set Cover.*

Proof. Given an instance of set cover with ground set $\mathcal{U} = \{u_1, \ldots, u_n\}$ and subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$, we create a linear matroid with vectors in $\mathbb{F}_2^n$ as follows. For each $u_i \in \mathcal{U}$, there is the standard basis vector e_i. These vectors will form a basis for the rank n matroid we will create. For each subset $S_j \in \mathcal{S}$, we will form the vector χ_S which is the characteristic vector of the subset S. See Figure 5.1.

□

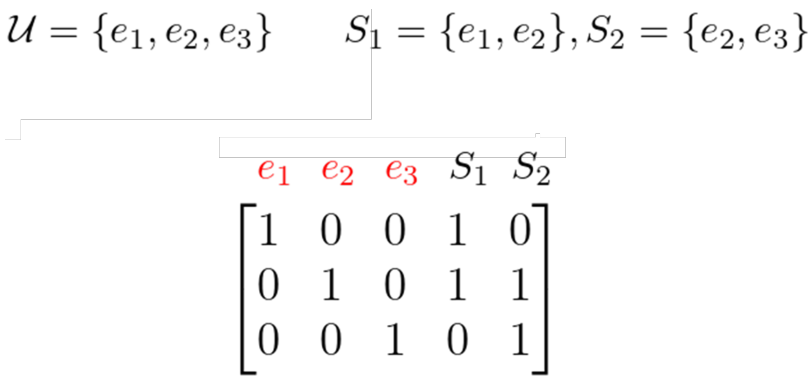

Figure 5.1: A figure illustrating the reduction from general Set Cover to BAP on a linear matroid.

5.2 Set-Cover Hardness for Transversal Matroids

In this section, we show that the Base Augmentation Problem is Set Cover hard for transversal matroids (even those whose rank is equal to $|R|$, where $G = (L \cup R, E)$ is the bipartite graph).

We recall the definition of a transversal matroid. Let $G = (L \cup R, E)$ be a bipartite graph with a maximum matching of cardinality r. The subsets of L which can be saturated by a matching form the independent sets of a matroid M with ground set L. Thus, the rank function $r(A)$ is the maximum size of a matchable subset of A. This is called a transversal matroid.

In the BAP problem for a transversal matroid, we are given a base $B \subseteq L$ which can be saturated by a matching. Let $r := |B|$ be the rank of the matroid. We are also given weights $w_u \geq 0$ for $u \in (L \setminus B)$.

Theorem 5.2.1. *The Base Augmentation Problem for transversal matroids is equivalent to Set Cover.*

Proof. Given an instance of Set Cover with ground set $\mathcal{U} = \{u_1, \ldots, u_n\}$ and subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$, we create a transversal matroid as follows. We create a bipartite graph $G = (L \cup R, E)$, where $|L| = n+m$ and $|R| = n$. There is a matching of size n between vertices $\{\ell_1, \ldots, \ell_n\}$ and $\{r_1, \ldots, r_n\}$.

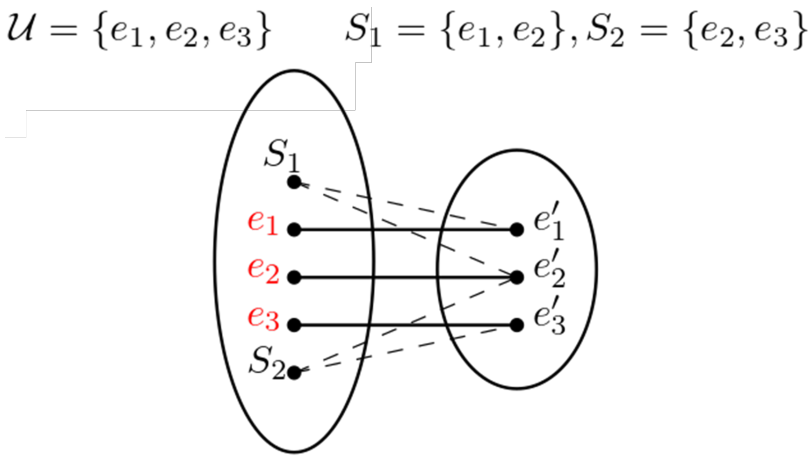

Figure 5.2: A figure illustrating the reduction from general Set Cover to BAP on a linear matroid

Finally, each vertex ℓ_{n+i} for $1 \leq i \leq m$ is adjacent to the elements covered by set $S_i \in \mathcal{S}$. Clearly, feasible solutions to the BAP instance correspond exactly to feasible covers of $\mathcal{U}$. See Figure 5.2.

□

5.3 Laminar Matroids and their Duals

In this section, we show that the Base Augmentation Problem is polynomial time solvable for laminar matroids.

There are, as is typical in matroid theory, several equivalent ways to define laminar matroids. The most common way to define them is to specify a tree, with non-negative integer capacities on it's internal nodes. The ground set of the matroid corresponds to the leaves of this tree, and a subset of leaves is independent if the number of leaves chosen respects the capacities at each internal node. That is, A is independent if for every internal node v, we have $|T_v \cap A|$ is at most c_v where c_v is the capacity of internal node v and T_v is the subtree rooted at v.

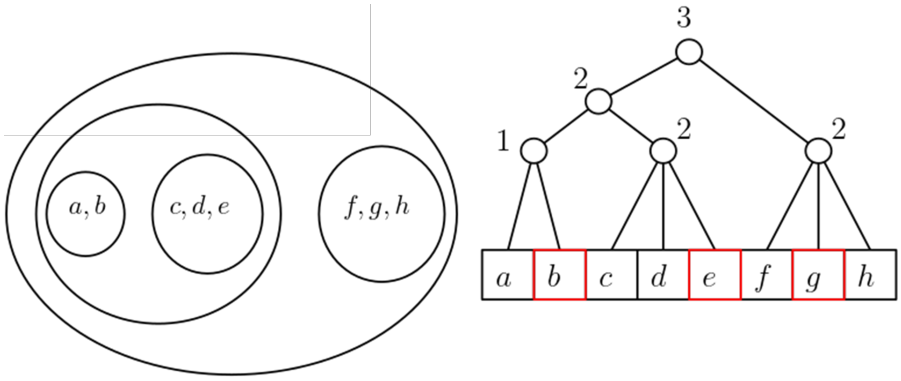

Figure 5.3: A laminar matroid and its associated laminar family.

Recall that the Base Augmentation Problem induces a Set Cover instance which involves covering B with the fundamental paths of elements in $E \setminus B$. We show that in the case of a laminar matroid, the constraint matrix is totally unimodular, so the BAP problem on this class is tractable.

Theorem 5.3.1. *The integrality gap of the set covering relaxation for BAP on laminar matroids is 1. Hence BAP on laminar matroids can be solved in polynomial time.*

Proof. Given a laminar matroid with base B, the collection of fundamental paths of elements in $E \setminus B$ themselves form a laminar family. Thus, the base-representation matrix is the incidence matrix of a laminar family and hence is totally unimodular. In this case, the Set Cover instance is a laminar family, and hence is solvable in polynomial time. □

We also show a relationship between the BAP problem on a matroid M and its dual matroid M^*. If A is the base-representation matrix for a matroid M with base B, then the constraint matrix for the dual matroid M^* with base $E \setminus B$ is equal to A^T.

How can we understand the duals of laminar matroids? For this purpose, it is most enlightening to consider a different definition of laminar matroids. All laminar matroids are obtained by beginning with a trivial matroid (where every subset is independent), and repeatedly applying the operations of **truncation** and **direct sum**. Hence, the duals of laminar matroids can be defined as any matroid which can be obtained from a trivial matroid by applying the operations of direct sums, and **matroid extensions**.

Because the tranposes of totally unimodular matrices are again totally unimodular, we obtain the following result.

Corollary 5.3.2. *The integrality gap of the set covering relaxation for BAP on duals of laminar matroids is 1. Hence BAP on these matroids can be solved in polynomial time.*

5.4 Network Matroids

A network matrix is a totally unimodular matrix which arises from directed paths in an oriented tree. We define a network matroid to be the matroids whose base representation matrices are signable to be network matrices or their transposes. These matroids are exactly the graphic and co-graphic matroids.

We are already familiar with the Base Augmentation Problem for graphic matroids. This is the Tree Augmentation Problem. The integrality gap of the Set Covering relaxation is not known exactly, but it is at least $\frac{3}{2}$ and at most 2. The current best approximation ratio for weighted TAP is $1.5 + \varepsilon$ due to [TZ22c]. See Figure 5.4

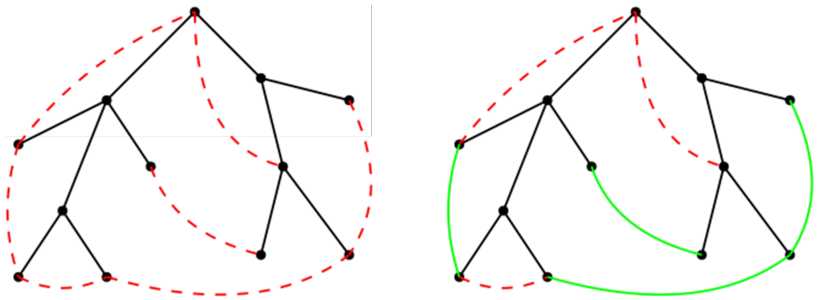

Figure 5.4: The BAP problem for a graphic matroid is the well-studied Tree Augmentation Problem. The green links form a feasible solution.

We show that the Base Augmentation Problem for co-graphic matroids corresponds to the Multicut problem on trees. This problem can be approximation to within a factor of 2, and this is the

best known ratio. Indeed, since vertex cover is a special case of multi-cut on trees (in which the tree is a star), we cannot hope to beat an approximation ratio of 2 assuming the unique games conjecture. Thus, the approximability of this problem for the co-graphic case is well understood.

The constraint matrix corresponds to a matrix which is signable to be a network matrix. The integrality gap of the linear relaxation for this problem is also known to be exactly 2. See Figure 5.5.

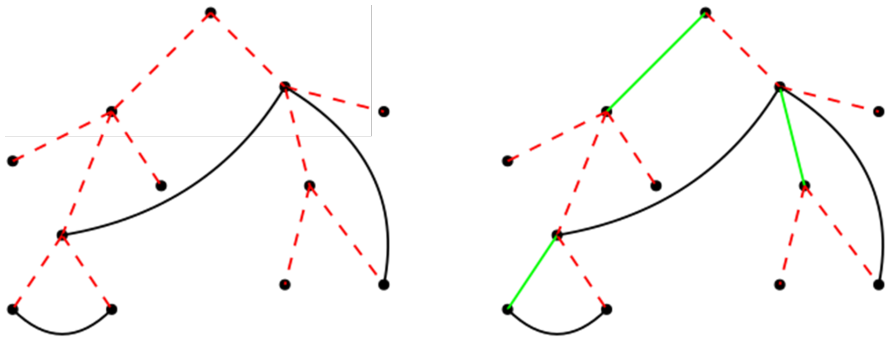

Figure 5.5: The BAP problem for a co-graphic matroid is the Multi-cut problem on trees. The green links form a feasible solution

5.5 Regular Matroids

We say that a $\{0,1\}$-matrix is **TU-signable** if some of its entries can negated so that is becomes totally unimodular. We say that a $\{0,1\}$-matrix is **Network-signable** if some of its entries can negated so that is becomes a Network matrix.

A regular matroid is a matroid which is representable over any field. An alternative definition of regular matroids are those matroids whose base-representation matrices are TU-signable.

We recall Seymour's famous decomposition theorem which characterizes all totally unimodular matrices as those which can be obtained by recursively by applying the k-sum operation for $k \in \{1,2,3\}$ to Network matrices, tranposes of Network matrices, and the two R_{10} matrices shown below [Sey80].

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Notice that both R_{10} matrices are totally unimodular.

The k-sum operations are general operations that can be applied to matroids, but since we are working with the base-representation matrices, we will define them on matrices, and only for $k \in \{1,2,3\}$. We use the definition from [PPAK09].

Definition 5.5.1. If A, B are matrices and a, d, b, c are column and row vectors of appropriate size with entries in $\mathbb{R}$, then

$$\text{1-sum: } A \oplus_1 B := \left[\begin{array}{c|c} A & 0 \\ \hline 0 & B \end{array}\right]$$

$$\text{2-sum: } \left[\begin{array}{c|c} A & a \end{array}\right] \oplus_2 \left[\begin{array}{c} b \\ \hline B \end{array}\right] := \left[\begin{array}{c|c} A & ab \\ \hline 0 & B \end{array}\right]$$

$$\text{3-sum: } \left[\begin{array}{c|c|c} A & a & a \\ \hline c & 0 & 1 \end{array}\right] \oplus_3 \left[\begin{array}{c|c|c} 1 & 0 & b \\ \hline d & d & B \end{array}\right] := \left[\begin{array}{c|c} A & ab \\ \hline dc & B \end{array}\right] \text{ or}$$

$$\left[\begin{array}{c|c} A & 0 \\ \hline b & 1 \\ \hline c & 1 \end{array}\right] \oplus_3 \left[\begin{array}{c|c|c} 1 & 1 & 0 \\ \hline a & d & B \end{array}\right] := \left[\begin{array}{c|c} A & 0 \\ \hline D & B \end{array}\right].$$

The analogous statement for TU-signable matrices is easily seen as a simple corollary of Seymour's theorem.

Corollary 5.5.2. *Let A be a matrix which is TU-signable. Then A can be obtained by recursively applying the k-sum operation for $k \in \{1, 2, 3\}$ to Network-signable matrices, their transposes, and the two R_{10} matrices.*

What this means is that the constraint matrix for the BAP problem on regular matroids is constructed from the ingredients in the above three classes. We have seen that the Set Cover integrality gaps of Network-signable matrices and transposes of Network-signable matrices are at most 2. It is a straightforward exercise to check that the Set Cover integrality gaps for both R_{10} constraint matrices are also at most 2.

Hence, as long as Set Cover integrality gaps are preserved under 1,2, and 3-sum operations, we would have that the integrality gap for all regular matroids would be at most 2. This motivates the following conjecture.

Conjecture 5.5.3. *If A is an $m \times n$ is a TU-signable matrix, then the integrality gap of*

$$\min\{cx : Ax \geq 1, x \in \{0, 1\}^n\}$$

is at most 2. Equivalently, the integrality gap of the BAP problem for regular matroids is 2.

Chapter 6

Woodall's Conjecture

We start by recalling Woodall's conjecture. Let $D = (V, A)$ be a digraph. A *dicut* is a cut $\delta^+(U) \subseteq A$ for some nonempty proper vertex subset U such that $\delta^-(U) = \emptyset$. A *dijoin* is an arc subset that intersects every dicut at least once.

Conjecture 6.0.1 (Woodall's Conjecture [Woo78]). For any directed graph $D = (V, A)$,if τ is the minimum cardinality of a dicut, then there exist τ dijoint dijoins.

In this chapter, we describe a lifting operation which allows us to reduce Woodall's conjecture to a more structured class of bipartite, semi-regular instances. Then we use this reduction to prove the following theorem, demonstrating the existence of an admissible dijoin.

Theorem 6.0.2. *Let $D = (V, A)$ be a digraph, and $\tau \geq 3$ the minimum cardinality of a dicut. Then there exists a dijoin J such that for every dicut $\delta^+(U)$, $|\delta^+(U) \setminus J| \geq \tau - 1$.*

This is based on joint work with Ahmad Abdi and Gérard Cornuéjols, and appeared in the SIAM Journal of Discrete Mathematics [ACZ23].

6.1 Lifting

In this section, we prove that Woodall's conjecture is equivalent to the following.

Conjecture 6.1.1. Let $D = (V, A)$ be a digraph, and let τ be the minimum cardinality of a dicut. Suppose every vertex is either a sink of degree τ, or a source of degree $\in \{\tau, \tau + 1\}$. Then there exist τ dijoint dijoins.

We begin by defining the notion of the imbalancedness of vertices in a directed graph. For each vertex $u \in V$, the *imbalance of* u is

$$\mathrm{imb}(u) := \deg^+(u) - \deg^-(u) = |\delta^+(u)| - |\delta^-(u)|$$

and for each $U \subseteq V$, the *imbalance of* U is

$$\mathrm{imb}(U) := \sum_{u \in U} \mathrm{imb}(u) = |\delta^+(U)| - |\delta^-(U)|.$$

Theorem 6.1.2. *Let $D = (V, A)$ be a digraph without a cut-vertex where the minimum cardinality of a dicut is $\tau \geq 3$. Then there exists a digraph $D' = (V', A')$ without a cut-vertex such that*

1. *every vertex is a source or a sink of degree τ or $\tau+1$,*
2. *every vertex of degree $\tau+1$ is a source, and the number of such vertices is equal to*
$$\sum (imb(u) \mod \tau : u \in V)$$
3. *$A \subseteq A'$, every dicut in D corresponds to a dicut in D', and the minimum size of a dicut in D' is τ, and*
4. *if J' is a dijoin in D', then $J' \cap A$ is a dijoin in D.*

Here, $m \mod n$ is the integer $d \in \{0, 1, \ldots, n-1\}$ such that $n \mid m + d$.

To prove this theorem, we replace non-compliant vertices with certain gadgets, designed in the next subsection, by means of un-contraction. After building the gadgets, we prove the theorem.

6.1.1 Gadgets

Lemma 6.1.3. Given integers $r \geq 2$ and $k \geq r$, there is an r-regular r-edge-connected bipartite graph $G = (L \cup R, E)$ with $|L| = |R| = k$, where G has no cut-vertex, and if $r \geq 3$, then every minimum cut of G is trivial.

Proof. Let $L = \{\ell_0, \ldots, \ell_{k-1}\}$ and $R = \{r_0, \ldots, r_{k-1}\}$. We will form r edge-disjoint perfect matchings: For $i \in \{0, \ldots, r-1\}$, let
$$M_i := \{(\ell_j, r_{i+j}) : j \in \{0, \ldots, k-1\}\},$$
where $i + j$ is taken modulo k.

We take $G = \bigcup_{i=0}^{r-1} M_i$. Clearly G is r-regular, and has no cut-vertex. Furthermore, as $k \geq r$, each $M_i \cup M_{i+1}$ is isomorphic to a Hamilton circuit on $2k$ vertices. So you could say that if all edges in G were doubled, G would be an edge-disjoint union of r Hamilton circuits, implying that every cut has at least $2r$ edges crossing it. In fact, however, every edge of G is counted twice, so we get that G is r-edge-connected.

Next assume that $r \geq 3$. We prove that every minimum cut is trivial. Suppose otherwise. Let $\delta(U)$ be a nontrivial minimum cut of G, of size r. Then each $M_i \cup M_{i+1}$ crosses U exactly twice, implying in turn that the edges of $M_i \cup M_{i+1}$ in $G[U]$ form a path.

Since $M_0 \cup M_1$ forms a path in $G[U]$, and $|U|, |\overline{U}| \geq 2$, we may assume that $U = \{r_0, \ell_0, \ldots\}$ and $\overline{U} = \{\ldots, r_{k-1}, \ell_{k-1}\}$. In particular, $(\ell_{k-1}, r_0) \in M_1 \cap \delta(U)$. If $r_1 \in U$ and $\ell_{k-2} \in \overline{U}$, then $(\ell_{k-1}, r_1), (\ell_{k-2}, r_0) \in M_2 \cap \delta(U)$, so $|\delta(U) \cap (M_1 \cup M_2)| \geq 3$, a contradiction. Otherwise, either $U = \{r_0, \ell_0\}$ or $\overline{U} = \{r_{k-1}, \ell_{k-1}\}$. Since $k \geq r$, G has no parallel edges, so in both cases, $|\delta(U)| = 2r - 1 > r$, a contradiction. □

Remark 6.1.4. Adding a vertex v of degree at least d to a d-edge-connected graph G yields a d-edge-connected graph G'. Moreover, if G has no cut-vertex, every cut of size d in G is trivial, and v has at least two distinct neighbors in G', then G' has no cut-vertex, and every cut of size d in G' is trivial.

Lemma 6.1.5 (Gadget 1)**.** Given integers a, b, r such that $r \geq 2$, $b > r$, and $a(r+1) = br$, there is a bipartite graph $G = (A \cup B, E)$ with $|A| = a$, $|B| = b$, such that the vertices in A have degree $r+1$, the vertices in B have degree r, G has no cut-vertex, is r-edge-connected, and every minimum cut is trivial.

Proof. Assume in the first case that $r = 2$. Let $G' = (A \cup B', E')$ be the circuit on $2a$ vertices, with bipartition A, B'. Then connect every node in A to the opposite node of the circuit. Since $3a = 2b$, a is even, so every node of A is paired with another node in A. Now subdivide every new edge, add the new vertices to B' to create B, and call the new graph $G = (A \cup B, E)$, which is bipartite with bipartition A, B. It can be readily verified G is the desired graph.

Assume in the remaining case that $r \geq 3$. We begin by using Lemma 6.1.3 with parameters r and $k = a = \frac{b}{r+1} \cdot r \geq r$ to obtain an r-edge-connected bipartite graph $G' = (A \cup B', E')$ with $|A| = |B'| = a$, where G' has no cut-vertex, and every minimum cut is trivial. Now, create $b - a$ extra nodes, put them into B' to create B, and attach each of them to r distinct nodes of A so that every vertex in A now has degree $r + 1$. Call this new graph $G = (A \cup B, E)$, which is bipartite with bipartition A, B, where every vertex in A has degree $r + 1$, every vertex in B has degree r, and $|A| = a, |B| = b$. Moreover, by a repeated application of Remark 6.1.4, we see that G has no cut-vertex, is r-edge-connected, and every minimum cut is trivial, as desired. □

Lemma 6.1.6 (Gadget 2)**.** Take an integer $\tau \geq 3$, and another integer $d \geq \tau$ such that $\tau \mid d$. Then there exists a graph G without a cut-vertex such that

- G is a $(\tau - 1)$-edge-connected bipartite graph with bipartition $L \cup R$,
- L has d vertices of degree $\tau - 1$, and all its other vertices, of which there are at least $\tau - 1$ many, have degree τ, and
- every vertex in R has degree τ.

Proof. Suppose $d = \tau c$ for some integer $c \geq 1$. Let $k \geq 2$ be an arbitrary integer. Let H be a $(\tau-1)$-regular $(\tau - 1)$-edge-connected bipartite graph on $2kc\tau$ vertices and without a cut-vertex, whose existence is guaranteed by Lemma 6.1.3. Let $L' \cup R'$ be the bipartition of H, where $|L'| = |R'| = kc\tau$. Pick a subset $S \subseteq L'$ of size $(k - 1)c\tau$. Create $(k - 1)c$ new vertices of degree τ, put them into R' to create R, and attach them to the vertices in S so that the vertices in S now have degree τ. Note that the $c\tau$ vertices in $L' \setminus S$ have degree $\tau - 1$, and there are still $kc\tau$ vertices of degree $\tau - 1$ in R.

Next, introduce kc new vertices $\ell_1, \ldots, \ell_{kc}$, and add them to L' to create L. Each new vertex will have degree τ and attach to distinct vertices of degree $\tau - 1$ in R. Thus, all $kc\tau$ many degree $\tau - 1$ vertices in R become degree τ.

Let G be the new graph, which is bipartite with bipartition $L \cup R$. Clearly, G satisfies the desired degree conditions. Moreover, since H is $(\tau - 1)$ edge-connected and has no cut-vertex, and every vertex added has (simple) degree τ, it follows that G is $(\tau - 1)$-edge-connected and has no cut-vertex. □

6.1.2 Proof of the Reduction

To create D', we shall go through three phases.

Phase 1: In this phase, we replace every vertex that is neither a source nor a sink by an appropriate gadget. By the end of this phase, D is turned into a digraph $D_1 = (V_1, A_1)$ without a cut-vertex such that

1.1 every vertex is either a source or a sink,

1.2 $\sum (\text{imb}_D(u) \mod \tau : u \in V) = \sum (\text{imb}_{D_1}(u) \mod \tau : u \in V_1)$,

1.3 $A \subseteq A_1$, every dicut in D corresponds to a dicut in D_1, and the minimum size of a dicut in D_1 is τ, and

1.4 if J_1 is a dijoin in D_1, then $J_1 \cap A$ is a dijoin in D.

To this end, for every v that is neither a sink nor a source:

> We replace v by two nodes v^+, v^-, where all the arcs previously leaving (resp. entering) v now leave (resp. enter) v^+ (resp. v^-). Furthermore, we place $\tau + (-\deg^-(v) \mod \tau)$ parallel arcs from v^+ to v^-.

At the end of this process, we create a digraph $D_1 = (V_1, A_1)$ without a cut-vertex that satisfies 1.1-1.2. To see that 1.3-1.4 are satisfied, note that for $I := A_1 - A$, we have $D = D_1/I$. As a result, 1.4 is satisfied, and every dicut in D remains a dicut in D_1. The latter, combined with the fact that at least τ parallel arcs are placed from v^+ to v^-, guarantees that 1.3 is satisfied, too.

Phase 2: In this phase, we replace every sink of degree $c\tau - k$ where $0 \leq k < \tau$, by a gadget involving only sources and sinks of degree τ, and exactly k sources of degree $\tau + 1$. By the end of this phase, D_1 is turned into a digraph $D_2 = (V_2, A_2)$ without a cut-vertex such that

2.1 every vertex is either a source or a sink, and every vertex of degree $\neq \tau$ is a source,

2.2 $\sum (\text{imb}_{D_1}(u) \mod \tau : u \in V_1) = \sum (\text{imb}_{D_2}(u) \mod \tau : u \in V_2)$, and the number of new sources of degree $\tau + 1$ in D_2 is equal to $\sum (\text{imb}(v) \mod \tau : v \text{ is a sink of } D_1)$,

2.3 $A_1 \subseteq A_2$, every dicut in D_1 corresponds to a dicut in D_2, and the minimum size of a dicut in D_2 is τ, and

2.4 if J_2 is a dijoin in D_2, then $J_2 \cap A_1$ is a dijoin in D_1.

To this end, for every sink v of degree $c\tau - k$, where $0 \leq k < \tau$:

> We create $c\tau - k$ new sinks which will each take exactly one of the incoming arcs going into v. Now, we place down k many sources of degree $\tau + 1$, such that their neighborhoods are pairwise disjoint, creating $k(\tau + 1)$ additional sinks. Thus in total, we have created $k(\tau+1) + c\tau - k$ sinks of degree 1, as well as k sources of degree $\tau + 1$.
>
> To finish this gadget, we just need the sinks to have degree τ. So we add an appropriate orientation of the graph H obtained by applying Lemma 6.1.5 with parameters $r = \tau - 1 \geq 2$, $b = k(\tau+1) + c\tau - k = (c+k)\tau > r$ and $a = br/(r+1)$. In this orientation, the b many, old vertices remain sinks, while the a many, new vertices become sources.

In the construction just described, note that H has no cut-vertex, is $(\tau - 1)$-edge-connected, and every minimum cut is trivial. Note that the underlying undirected graph of the resulting gadget, call it H', is obtained from H by adding k vertices of degree $\tau+1$; denote this set by $K = V(H') - V(H)$. In particular, H' has no cut-vertex. Thus, the resulting digraph obtained by replacing v with the gadget, call it D_2', has no cut-vertex.

We now show that the size of the smallest dicut in D_2' is at least τ. Consider a dicut $C = \delta^+(U)$ in D_2'. If U does not separate $V(H)$, then C would be a dicut of D_2'/H, which is obtained from

D_1 by adding, for each $u \in K$, $\tau+1$ parallel arcs from u to v. As every dicut of D_1 has size at least τ, so does every dicut of D_2'/H, so $|C| \geq \tau$. Otherwise, U separates $V(H)$, implying in turn that $|C| \geq \tau - 1$, as H is $(\tau-1)$-edge-connected. If C contains an arc outside $A(H)$, then this additional arc gives $|C| \geq (\tau-1)+1 = \tau$. Otherwise, C is a dicut of the oriented H. In particular, since v is not a cut-vertex of D_1 (as D_1 has no cut-vertex), it follows that U does not separate $V(D_2' \setminus H') = V(D_1 \setminus v)$. It can now be readily checked that C does not correspond to a trivial dicut of H, implying in turn that $|C| \geq \tau$, as every nontrivial cut of H has size at least τ, as required.

At the end of this process, we create a digraph $D_2 = (V_2, A_2)$ without a cut-vertex, one that satisfies 2.1-2.4 for similar reasons as Phase 1.

Phase 3: In this final phase, we replace every source of degree $\neq \tau, \tau+1$, and more precisely of degree $c\tau+k$ where $0 \leq k < \tau$, by a gadget involving only sources and sinks of degree τ, and exactly k sources of degree $\tau+1$. By the end of this phase, D_2 is turned into a digraph $D' = (V', A')$ without a cut-vertex such that

3.1 every vertex is either a source or a sink of degree τ or $\tau+1$, and every vertex of degree $\neq \tau$ is a source,

3.2 $\sum(\text{imb}_{D_2}(u) \mod \tau : u \in V_2) = \sum(\text{imb}_{D'}(u) \mod \tau : u \in V')$, and the number of sources of degree $\tau+1$ in D' is equal to $\sum(\text{imb}(v) \mod \tau : v \text{ is a source of } D_2)$,

3.3 $A_2 \subseteq A'$, every dicut in D_2 corresponds to a dicut in D', and the minimum size of a dicut in D' is τ, and

3.4 if J' is a dijoin in D', then $J' \cap A_2$ is a dijoin in D_2.

To this end, for every source u of degree $\neq \tau, \tau+1$, and more precisely of degree $c\tau+k$, where $0 \leq k < \tau$:

> Let $d := c\tau$. Let G_u be the gadget from Lemma 6.1.6 with respect to parameters τ, d, and with bipartition $L \cup R$. Then orient the edges of G_u from L to R in order to obtain D_u. Then replace vertex u by the digraph D_u, where the $\deg(u) = d+k$ arcs previously leaving u now leave $d+k$ distinct vertices in L including all the vertices of degree $\tau-1$ in L. Let L' be the set of these $d+k$ vertices in L.

Observe that every vertex of D_u is a source or a sink of degree τ or $\tau+1$, that all the degree $\tau+1$ vertices are contained in L, and there is k many such vertices.

Let D'' be the digraph obtained from D_2 by replacing u with the gadget D_u. Since G_u, and therefore D_u, has no cut-vertex, it follows that D'' has no cut-vertex.

We claim that every dicut in D'' has size at least τ. Let $C = \delta^+(U)$ be a dicut of D''. If U does not separate $V(D_u)$, then C is a dicut of $D''/D_u = D_2$, so $|C| \geq \tau$. Otherwise, U separates $V(D_u)$. In particular, since G_u is $(\tau-1)$-edge-connected, it follows that $|C| \geq \tau - 1$. If U separates L', then since u is not a cut-vertex of D_2 (as D_2 has no cut-vertex), we would have $|C| \geq (\tau-1)+1 = \tau$. Otherwise, U does not separate L'. If U separates $V(D' \setminus D_u)$, then $|C| \geq (\tau-1)+1 = \tau$. Otherwise, U does not separate $V(D' \setminus D_u)$. If U separates L' from $V(D' \setminus D_u)$, then $|C| \geq (\tau-1) + |L'| \geq \tau$. Otherwise, U does not separate $L' \cup V(D' \setminus D_u)$. Consequently, either U or $\overline{U}$ is equal to $V(D_u) - L'$.

Since every vertex in $V(D_u) - L'$ is a source or a sink of degree τ, and since $C = \delta^+(U)$, it follows that $|C| \equiv 0 \pmod{\tau}$, implying in turn that $|C| \geq \tau$, as required.

At the end of this process, we create a digraph $D' = (V', A')$ without a cut-vertex, one that satisfies 3.1-3.4 for similar reasons as Phase 1.

The digraph D' that we obtain from Phase 3 is the desired digraph, as can be readily verified by the reader. □

The above lifting procedure allows us to reduce Woodall's conjecture to these structured digraphs.

Theorem 6.1.7. *Take an integer $\tau \geq 3$. Then the following statements are equivalent:*

1. *Conjecture 6.0.1 (Woodall's Conjecture) for τ,*
2. *Conjecture 6.1.1 for τ,*

Proof. **(1)** $\Rightarrow$ **(2)** is clear. **(2)** $\Rightarrow$ **(1)** Let D be a digraph where the minimum cardinality of a dicut is τ. We need to prove that D has τ disjoint dijoins. We may assume that D has no cut-vertex. Thus, by Theorem 6.1.2, there exists a digraph D' where

- every vertex is a source or a sink of degree τ or $\tau + 1$,
- every vertex of degree $\tau + 1$ is a source,
- $A(D) \subseteq A(D')$, every dicut in D corresponds to a dicut in D', and the minimum size of a dicut in D' is τ,
- if J' is a dijoin in D', then $J' \cap A(D)$ is a dijoin D.

Consequently, it suffices to find τ disjoint dijoins in D', which follows from (2). □

6.2 An Admissible Dijoin

Let $\tau \geq 2$ be an integer. A $(\tau, \tau+1)$-*biregular bipartite digraph* is a digraph $D = (V, A)$ where every vertex is either a sink of degree τ or a source of degree τ or $\tau + 1$, and every dicut has cardinality at least τ. A node u is *active* if it has degree $\tau + 1$. Given a subset $U \subseteq V$, define

$$\begin{aligned} \text{sources}(U) &:= \{u \in U : u \text{ is a source of } D\} \\ \text{sinks}(U) &:= \{u \in U : u \text{ is a sink of } D\} \\ \text{depth}(U) &:= |\text{sinks}(U)| - |\text{sources}(U)| \\ a(U) &:= \{u \in U : u \text{ is an active node}\}. \end{aligned}$$

A subset $P \subseteq V$ satisfying

$$\begin{aligned} P &\subseteq a(V) \\ |P| &= \text{depth}(V) \\ |P \cap U| &\geq \rho + \text{depth}(U) \qquad \forall \text{ dicuts } \delta^+(U) \text{ of } D \end{aligned}$$

for $\rho = 0$ is called *weakly admissible*; if P satisfies the conditions above for $\rho = 1$, then it is called *admissible*. A *balanced edge cover* is a subset $F \subseteq A$ such that

$$\left\lfloor \frac{|\delta(v)|}{\tau} \right\rfloor \leq |J \cap \delta(v)| \leq \left\lceil \frac{|\delta(v)|}{\tau} \right\rceil \qquad \forall v \in V.$$

The set of *dyad centers of* F is $\mathrm{dc}(F) := \{u \in V : \deg_F(v) = 2\}$. We will show that $P \subseteq a(V)$ is weakly admissible if, and only if, $P = \mathrm{dc}(F)$ for some balanced edge cover $F \subseteq A$. Moreover, $P \subseteq a(V)$ is admissible if, and only if, $P = \mathrm{dc}(F)$ for some balanced edge cover $F \subseteq A$ that is a dijoin.

Given that $d := \mathrm{depth}(V)$, the number of active nodes is $d\tau$. The *depth of a dicut* is the depth of its shore.

Remark 6.2.1. Let U, W be subsets of V. Then $\mathrm{depth}(U \cup W) + \mathrm{depth}(U \cap W) = \mathrm{depth}(U) + \mathrm{depth}(W)$.

6.2.1 Balanced edge covers

An *edge cover* is an $F \subseteq A$ where $F \cap \delta(u) \neq \emptyset$ for all $u \in V$. An edge cover F is *balanced* if for each $u \in V$,

$$\left\lceil \frac{\deg(u)}{\tau} \right\rceil \geq |F \cap \delta(u)| \geq \left\lfloor \frac{\deg(u)}{\tau} \right\rfloor.$$

Observe that every balanced edge cover is the vertex disjoint union of d dyads, whose centers are active nodes, and $|\mathrm{sources}(V)| - d$ edges.

Proposition 6.2.2. Let $F_1, \ldots, F_\tau$ be disjoint dijoins of D. Then the dijoins partition A, and each is a balanced edge cover.

Proof. For each $u \in \mathrm{sinks}(V)$, $\delta^-(u)$ is a minimum dicut, so the arcs incident with u are evenly distributed amongst the τ disjoint dijoins. As every arc in A is incident with a vertex in $\mathrm{sinks}(V)$, it follows that $F_1, \ldots, F_\tau$ partition A, and that for each $u \in \mathrm{sinks}(V)$ and $i \in [\tau]$, $|F_i \cap \delta^-(u)| = 1$. Now let $v \in \mathrm{sources}(V)$. If $\deg(v) = \tau$, then the arcs incident with v are also evenly distributed amongst the τ disjoint dijoins, so that $|F_i \cap \delta^+(v)| = 1$ for each $i \in [\tau]$. Otherwise, $\deg(v) = \tau + 1$, i.e. v is an active node. In this case, $\tau - 1$ of the dijoins have exactly one arc incident with v, and the remaining dijoin has exactly two arcs incident with v.

The observations above imply that each $F_i, i \in [\tau]$ is a balanced edge cover, as required. □

Remark 6.2.3. Let F be a balanced edge cover. Then for every dicut $\delta^+(U)$,

$$|F \cap \delta^+(U)| = |\mathrm{dc}(F) \cap U| - \mathrm{depth}(U).$$

In particular, F is a dijoin if, and only if, $\mathrm{dc}(F)$ is admissible, i.e., for every non-trivial dicut $\delta^+(U)$,

$$|\mathrm{dc}(F) \cap U| \geq \mathrm{depth}(U) + 1.$$

Proposition 6.2.4. Let $P \subseteq a(V)$. Then P is weakly admissible if, and only if, there exists a balanced edge cover F such that $\mathrm{dc}(F) = P$.

Proof. ($\Leftarrow$) Let F be a balanced edge cover such that $\mathrm{dc}(F) = P$. By definition, $|P| = d$. Let $\delta^+(U)$ be a dicut. Then

$$0 \leq |\delta^+(U) \cap F| = |\mathrm{sources}(U)| + |U \cap P| - |\mathrm{sinks}(U)| = |U \cap P| - \mathrm{depth}(U).$$

As the inequality holds for every dicut, it follows that P is weakly admissible. ($\Rightarrow$) Let $b := \mathbf{1} + \chi_P \in \mathbb{Z}_+^V$. Then $F \subseteq E$ is a balanced edge cover such that $\mathrm{dc}(F) = P$ if, and only if, F is a perfect b-matching. Thus, it suffices to prove that a perfect b-matching exists. Since $b(\mathrm{sources}(V)) = b(\mathrm{sinks}(V))$, it suffices to show that for each non-trivial vertex cover $K \subseteq V$ of G, $b(K) \geq \frac{1}{2}b(V)$. To this end, let K be a non-trivial vertex cover, that is, K is a proper nonempty subset of $\mathrm{sources}(V)$, and a proper nonempty subset of $\mathrm{sinks}(V)$. Let $U := K \triangle \mathrm{sinks}(V)$. Then $\delta^+(U)$ is a dicut, so $|U \cap P| \geq \mathrm{depth}(U)$. Then

$$\begin{aligned} b(K) - \frac{1}{2}b(V) &= b(K) - b(\mathrm{sinks}(V)) \\ &= b(\mathrm{sources}(U)) - b(\mathrm{sinks}(U)) \\ &= |\mathrm{sources}(U)\,| + |U \cap P| - |\mathrm{sinks}(U)\,| \\ &= |U \cap P| - \mathrm{depth}(U) \\ &\geq 0, \end{aligned}$$

as required. □

Lemma 6.2.5. The following statements hold:

1. A can be partitioned into τ balanced edge covers.
2. Let $F_1, \ldots, F_\tau$ be a partition of A into balanced edge covers. Then for every dicut $\delta^+(U)$, and for $i, j \in [\tau]$,

$$|F_i \cap \delta^+(U)| - |F_j \cap \delta^+(U)| = |\mathrm{dc}(F_i) \cap U| - |\mathrm{dc}(F_j) \cap U|.$$

Proof. **(1)** Let G be a τ-regular bipartite graph obtained from D as follows:

- introduce $d = |\mathrm{sinks}(V)\,| - |\mathrm{sources}(V)\,|$ new vertices $v_1, \ldots, v_d$,
- for each active node v, take a single edge e_v incident with v,
- for each e_v, replace the active end of e_v with one of the new vertices $v_1, \ldots, v_d$, in a way so that by the end of the procedure, each v_i has degree τ.

Observe now that any perfect matching of G corresponds to a balanced edge cover of D. Since G is τ-edge-colourable, it follows that A can be partitioned into τ balanced edge covers. **(2)** follows immediately from Remark 6.2.3. □

6.2.2 Admissible partitions

Recall that $a(V) \subseteq \mathrm{sources}(V)$, the set of active nodes of D, has cardinality $d\tau$. Recall that a subset $P \subseteq V$ is admissible if it has cardinality d, and

$$|U \cap P| \geq 1 + \mathrm{depth}(U) \qquad \text{for each dicut } \delta^+(U).$$

An *admissible partition* of $a(V)$ is a partition of it into τ admissible parts. Observe that if D has τ disjoint dijoins, then by Proposition 6.2.2 and Remark 6.2.3, $a(V)$ has an admissible partition. Here, we prove this consequence as a first step towards proving Woodall's Conjecture.

A polyhedron P has the *integer decomposition property* if it is integral, and for every integer $k \geq 2$, every integer point in P written as the sum of k points in P can be written as the sum of k integer points in P.

Theorem 6.2.6 (see [Sch03], Corollary 42.1e)**.** *The base polytope of a matroid has the integer decomposition property.*

Let $\mathcal{C}$ be a family of subsets of a finite ground set V. A pair of sets $U, W \in \mathcal{C}$ are *crossing* if $U \cap W \neq \emptyset$ and $U \cup W \neq V$. We say $\mathcal{C}$ is a *crossing family* if $U \cap W, U \cup W \in \mathcal{C}$ for all crossing pairs U, W.

Theorem 6.2.7 ([FT84])**.** *Let $\mathcal{C}$ be a crossing family over ground set V, and let $g : \mathcal{C} \to \mathbb{Z}$ be a crossing submodular function. Then for any integer k, the set $\{B \subseteq V : |B| = k, |B \cap U| \leq g(U)\ \forall U \in \mathcal{C}\}$, if nonempty, is the set of bases of a matroid.*

Corollary 6.2.8. $\{P \subseteq V : P \text{ is admissible}\}$, if nonempty, is the set of bases of a matroid.

Proof. Let $\mathcal{C} := \{U \subseteq V : \delta^+(U) \text{ is a dicut}\}$, which is a crossing family over ground set V. Then the family of admissible sets is described as

$$\{P \subseteq V : |P| = d, |P \cap U| \geq 1 + \text{depth}(U)\ \ \forall U \in \mathcal{C}\},$$

which we assume is nonempty. The family of complements of admissible sets can be described as

$$\{\overline{P} \subseteq V : |\overline{P}| = |V| - d, |\overline{P} \cap U| \leq |U| - 1 - \text{depth}(U)\ \ \forall U \in \mathcal{C}\}.$$

Since $g(U) := |U| - 1 - \text{depth}(U)$ is a modular, hence submodular, function, and since the family above is nonempty, it follows from Theorem 6.2.7 that the complements of admissible sets form the bases of a matroid, implying in turn that admissible sets form the bases of the dual matroid. □

Theorem 6.2.9 ([Sch03], Theorem 49.7)**.** *Let $\mathcal{C}$ be a crossing family over ground set V, let $g : \mathcal{C} \to \mathbb{Z}$ be a crossing submodular function, and let k be an integer. Then the system $x(V) = k,\ x(U) \leq g(U)\ \forall U \in \mathcal{C}$ is box-TDI.*

Theorem 6.2.10. *Let $D = (V, A)$ be a $(\tau, \tau + 1)$-biregular bipartite digraph. Then $a(V)$ can be partitioned into τ admissible sets.*

Proof. Consider the system $x(V) = d, x(U) \geq 1 + \text{depth}(U)\ \ \forall U \in \mathcal{C}$. By Theorem 6.2.9, this system is box-TDI, so the polytope

$$B(M) := \{x \in [0,1]^V : x(V) = d, x(U) \geq 1 + \text{depth}(U)\ \ \forall U \in \mathcal{C}\}$$

has $0 - 1$ vertices. Now let $x := \chi_{a(V)} \in \{0,1\}^V$, the incidence vector of the active nodes. Then $x(V) = a(V) = \tau d$. Moreover, for every dicut $\delta^+(U)$, $x(U) = a(U) \geq \tau(1 + \text{depth}(U))$. Thus, $\frac{1}{\tau}x \in B(M)$, so $B(M)$ is nonempty, implying in turn that an admissible set exists. In particular, by Corollary 6.2.8, $B(M)$ is the base polytope of a matroid, call it M. By Theorem 6.2.6, $B(M)$ has the integer decomposition property. Note that x is the sum of τ points in $B(M)$, namely τ identical copies of $\frac{1}{\tau}x$. Thus, by the integer decomposition property, x is the sum of τ integer points in $B(M)$. That is, $a(V)$ admits a partition into τ admissible sets, so there exists an admissible partition. □

6.2.3 Finding an admissible dijoin

Proposition 6.2.11. Take an integer $\tau \geq 3$. Let $D = (V, A)$ be a $(\tau, \tau + 1)$-biregular bipartite digraph, and let $P_1, \ldots, P_\tau$ be an admissible partition. Then the following statements hold:

1. There exists a balanced edge cover F such that $\mathrm{dc}(F) = P_\tau$.
2. Let $w := \mathbf{1} - \chi_F \in \{0,1\}^A$. Then for every dicut $\delta^+(U)$, $w(\delta^+(U)) \geq \tau - 1$.

Proof. **(1)** follows from the weak admissibility of P_τ. **(2)** Let $\delta^+(U)$ be a dicut. If $\delta^+(U)$ is a trivial dicut, then $w(\delta^+(U)) \in \{\tau - 1, \tau\}$, so we are done. Otherwise, we have

$$\begin{aligned} w(\delta^+(U)) &= |\delta^+(U)| - |F \cap \delta^+(U)| \\ &= a(U) - \tau \cdot \mathrm{depth}(U) - |P_\tau \cap U| + \mathrm{depth}(U) \\ &= \sum_{i=1}^{\tau-1} |P_i \cap U| - (\tau - 1) \cdot \mathrm{depth}(U) \\ &\geq (\tau - 1)(1 + \mathrm{depth}(U)) - (\tau - 1) \cdot \mathrm{depth}(U) \\ &= \tau - 1, \end{aligned}$$

as required. □

Proof of Theorem 6.0.2. This follows from Theorem 6.1.2 and Proposition 6.2.11. Let us elaborate. Let $D = (V, A)$ be a digraph without a cut-vertex where the minimum cardinality of a dicut is $\tau \geq 3$. We need to find a dijoin J such that for $w = \mathbf{1} - \chi_J$, $w(\delta^+(U)) \geq \tau - 1$ for every dicut $\delta^+(U)$ of D.

By Theorem 6.1.2, there exists a digraph $D' = (V', A')$ without a cut-vertex such that

- every vertex is a source of degree τ or $\tau + 1$, or a sink of degree τ,
- $A \subseteq A'$, every dicut in D corresponds to a dicut in D', and the minimum size of a dicut in D' is τ, and
- if J' is a dijoin in D', then $J' \cap A$ is a dijoin in D.

By Theorem 6.2.10, D' has an admissible partition $P_1, \ldots, P_\tau$. Let $J' \subseteq A'$ be a balanced edge cover of D' such that $\mathrm{dc}(J') = P_\tau$. Let $w' := \mathbf{1} - \chi_{J'} \in \{0,1\}^{A'}$. Then by Proposition 6.2.11, every dicut of D' has w'-weight at least $\tau - 1$. Let $J := J' \cap A$. By construction, J is a dijoin of D. Let $w := \mathbf{1} - \chi_J \in \{0,1\}^A$. We claim that every dicut of D has w-weight at least $\tau - 1$, thereby finishing the proof. To this end, let $\delta^+(U)$ be a dicut of D. Then $\delta^+(U)$ is also a dicut of D', so $w'(\delta^+(U)) \geq \tau - 1$. As $\delta^+(U) \subseteq A$, it follows that

$$w(\delta^+(U)) = |\delta^+(U)| - |\delta^+(U) \cap J| = |\delta^+(U)| - |\delta^+(U) \cap J'| = w'(\delta^+(U)) \geq \tau - 1,$$

as required. □

6.2.4 An important example

Denote by D_{27} the $(3,4)$-biregular bipartite digraph presented in Figure 6.1.

This digraph shows that after finding one admissible dijoin, it may be the case that the remaining weighted digraph is a counterexample to the Edmonds-Giles conjecture. This means that in order to use the admissible dijoin in an optimal packing, it must be chosen carefully somehow, rather than arbitrarily.

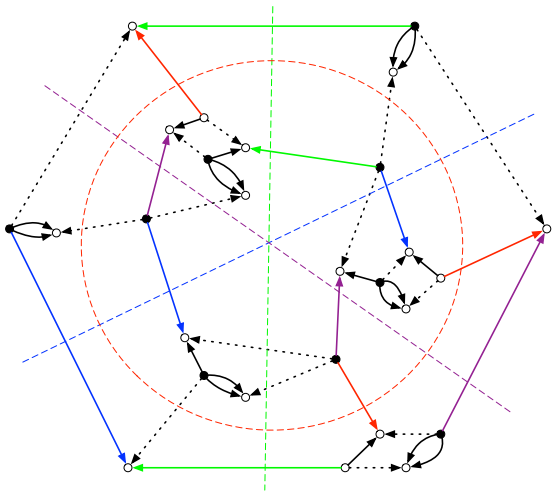

Figure 6.1: The digraph D_{27}: A $(3,4)$-biregular bipartite digraph. The active nodes are filled-in.

Chapter 7

Conclusion

When I was an undergraduate student at Rutgers, I did research for a summer under Professor Oliver Pechenik at the DIMACS REU. Towards the end of the summer, as we were writing up the results, I asked him if I should write a conclusion section for our paper. He told me that in math, people don't write conclusions, they just finish the last proof, put a little square at the bottom and end.

In homage to my first ever research experience, at least in mathematics (we won't discuss the fiascos that occured when I was a materials science student), I will follow Professor Pechenik's advice. □

Bibliography

[ACZ23] Ahmad Abdi, Gérard Cornuéjols, and Michael Zlatin. On packing dijoins in digraphs and weighted digraphs. *SIAM Journal on Discrete Mathematics*, 37(4):2417–2461, 2023.

[Adj17] David Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2384–2399, 2017.

[Adj18] David Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. *ACM Transactions on Algorithms (TALG)*, 15(2):1–26, 2018.

[AK06] Gautam Appa and Balázs Kotnyek. A bidirected generalization of network matrices. *Networks: An International Journal*, 47(4):185–198, 2006.

[AKPP07] Gautam Appa, Balázs Kotnyek, Konstantinos Papalamprou, and Leonidas Pitsoulis. Optimization with binet matrices. *Operations research letters*, 35(3):345–352, 2007.

[BD97] Al Borchers and Ding-Zhu Du. Thek-steiner ratio in graphs. *SIAM Journal on Computing*, 26(3):857–869, 1997.

[CF96] Alberto Caprara and Matteo Fischetti. {0, 1/2}-chvátal-gomory cuts. *Mathematical Programming*, 74(3):221–235, 1996.

[CG15] Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP. *CoRR*, abs/1508.07504, 2015.

[CJR99] Joseph Cheriyan, Tibor Jordán, and R Ravi. On 2-coverings and 2-packings of laminar families. *Algorithms-ESA'99*, pages 72–72, 1999.

[CKKK08] Joseph Cheriyan, Howard Karloff, Rohit Khandekar, and Jochen Könemann. On the integrality ratio for tree augmentation. *Operations Research Letters*, 36(4):399–401, 2008.

[CLR23] Gérard Cornuéjols, Siyue Liu, and R. Ravi. Approximately packing dijoins via nowhere-zero flows, 2023.

[CN13a] Nachshon Cohen and Zeev Nutov. A (1+ ln2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theoretical Computer Science*, 489:67–74, 2013.

[CN13b] Nachshon Cohen and Zeev Nutov. A $(1 + \ln 2)$-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theoret. Comput. Sci.*, 489/490:67–74, 2013.

[DKL76] Efim A Dinitz, Alexander V Karzanov, and Michael V Lomonosov. On the structure of the system of minimum edge cuts in a graph. *Issledovaniya po Diskretnoi Optimizatsii*, pages 290–306, 1976.

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 624–633, 2014.

[DV94] Yefim Dinitz and Alek Vainshtein. The connectivity carcass of a vertex subset in a graph and its incremental maintenance. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 716–725, 1994.

[DW71] Stuart E Dreyfus and Robert A Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[ET76] Kapali P Eswaran and R Endre Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976.

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[FF09] T. Fleiner and A. Frank. A quick proof for the cactus representation of mincuts. 2009.

[FGKS18] Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. Approximating weighted tree augmentation via chvátal-gomory cuts. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 817–831. SIAM, 2018.

[FJ81] Greg N Frederickson and Joseph Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM Journal on Computing*, 10(2):270–283, 1981.

[FJ82] Greg N Fredrickson and Joseph Jájá. On the relationship between the biconnectivity augmentation and traveling salesman problem. *Theoretical Computer Science*, 19(2):189–201, 1982.

[FT84] A. Frank and É. Tardos. Matroids from crossing families. In A. Hajnal, L. Lovász, and V.T. Sós, editors, *Finite and Infinite Sets*, pages 295–304. North-Holland, 1984.

[GGJAS21] Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Krzysztof Sornat. On the cycle augmentation problem: hardness and approximation algorithms. *Theory of Computing Systems*, 65(6):985–1008, 2021.

[GKZ18] Fabrizio Grandoni, Christos Kalaitzis, and Rico Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 632–645. ACM, 2018.

[HZ23] Daniel Hathcock and Michael Zlatin. Improved approximation algorithms for steiner connectivity augmentation problems, 2023.

[IR17] Jennifer Iglesias and R. Ravi. Coloring down: 3/2-approximation for special cases of the weighted tree augmentation problem, 2017.

[IR22] Jennifer Iglesias and R. Ravi. Coloring down: 3/2-approximation for special cases of the weighted tree augmentation problem. *Operations Research Letters*, 50(6):693–698, 2022.

[Jai01] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[KKL04] Guy Kortsarz, Robert Krauthgamer, and James R Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM Journal on Computing*, 33(3):704–720, 2004.

[KN16] Guy Kortsarz and Zeev Nutov. A simplified 3/2 ratio approximation algorithm for the tree augmentation problem. *Transaction on Algorithm*, 12(2):23, 2016.

[KT93] Samir Khuller and Ramakrishna Thurimella. Approximation algorithms for graph augmentation. *Journal of Algorithms*, 14(2):214–225, 1993.

[Nut10] Zeev Nutov. Approximating steiner networks with node-weights. *SIAM Journal on Computing*, 39(7):3001–3022, 2010.

[PPAK09] Leonidas Pitsoulis, Konstantinos Papalamprou, Gautam Appa, and Balázs Kotnyek. On the representability of totally unimodular matrices on bidirected graphs. *Discrete Mathematics*, 309(16):5024–5042, 2009.

[PRZ22] Ojas Parekh, Ramamoorthi Ravi, and Michael Zlatin. On small-depth tree augmentations. *Operations Research Letters*, 50:667–673, 11 2022.

[Rav94] R. Ravi. *Steiner Trees and Beyond: Approximation Algorithms for Network Design.* PhD thesis, Brown University, 1994.

[RZZ23] R Ravi, Weizhong Zhang, and Michael Zlatin. Approximation algorithms for steiner tree augmentation problems. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2429–2448. SIAM, 2023.

[Sch03] A. Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency.* Springer, Berlin, Heidelberg, 2003.

[Sey77] P.D. Seymour. The matroids with the max-flow min-cut property. *Journal of Combinatorial Theory, Series B*, 23(2):189–222, 1977.

[Sey80] Paul D Seymour. Decomposition of regular matroids. *Journal of combinatorial theory, Series B*, 28(3):305–359, 1980.

[Sla96] Petr Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 435–441, 1996.

[TZ21a] Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation, 2021.

[TZ21b] Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree, 2021.

[TZ22a] Vera Traub and Rico Zenklusen. A (1.5+\epsilon)-approximation algorithm for weighted connectivity augmentation. *arXiv preprint arXiv:2209.07860*, 2022.

[TZ22b] Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–12, 2022.

[TZ22c] Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3253–3272. SIAM, 2022.

[Woo78] D.R. Woodall. Menger and König systems. In Y. Alavi and D.R. Lick, editors, *Theory and Applications of Graphs.*, volume 642 of *Lecture Notes in Mathematics*. Springer, Berlin, Heidelberg, 1978.