Cover Small Cuts and Flexible Graph Connectivity Problems

by

Miles Simmons

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

Waterloo, Ontario, Canada, 2025

© Miles Simmons 2025

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Statement of Contributions

Section 2.1 was adapted from a paper co-authored with Ishan Bansal and Joseph Cheriyan [25].

Sections 3.1 and 3.2 were adapted from a paper co-authored with Ishan Bansal, Joseph Cheriyan, and Sanjeev Khanna [6] [5].

The programming in Section 4.2 and experiments in Section 4.3 were joint work with Keven Qiu. I solely wrote these sections of the thesis, but Keven was closely involved in the development of the programs.

All other sections are solely authored by Miles Simmons, under the supervision of Joseph Cheriyan.

Abstract

Given a graph with capacitated edges and a set of links on the same vertex set, the *Cover Small Cuts* problem aims to choose a minimum-cost link set such that for each cut with total edge capacity below a given threshold, at least one selected link covers that cut.

We present examples and analysis of the Cover Small Cuts problem, and cut covering problems on *strongly pliable families*, a subfamily of pliable families. Pliable set families require that for any two sets A, B in the family, at least two of $A - B, B - A, A \cup B, A \cap B$ are also in the family. Strongly pliable set families require that for any two sets A, B in the family that cross, at least one of $A - B, B - A$ is also in the family, *and* at least one of $A \cup B, A \cap B$ is also in the family. Shortly before the submission of this thesis, we learned about prior work on what we call strongly pliable families; see Section 1.4.1 for further details. We decided to stay with the term "strongly pliable families" since it is consistent with the notion of "pliable families" introduced recently.

We present a family of Cover Small Cuts problems such that key property used to prove the approximation ratio of Jain's iterative rounding approximation algorithm does not hold. We show that families of "small cuts" are always strongly pliable, but not all strongly pliable families can be realized as families of small cuts. We prove a 5-approximation algorithm for cut covering problems on strongly pliable families using the primal-dual method of Williamson, Goemans, Vazirani, and Mihail. We compare strongly pliable families to other subfamilies of pliable families, such as uncrossable families. We also study some other cut covering problems.

In *Flexible Graph Connectivity (FGC)* problems, the edges of the input graph are partitioned into safe and unsafe edges. In real-world applications, safe edges represent reliable connections and unsafe edges represent connections that could break. We aim to select a minimum-cost set of edges that achieve given connectivity requirements despite the limitations of the unsafe edges. We present a constant-factor approximation algorithm for the $(1, q)$-FGC problem. We explore relaxations of the (p, q)-FGC problem that allow multiple copies of an edge to be selected, and construct approximation algorithms with better approximation ratios than those currently known for the (unrelaxed) (p, q)-FGC problem.

We end with a summary of Williamson et al.'s primal-dual approximation algorithm, a versatile approximation algorithm for cut covering problems including Cover Small Cuts; and our results from computational experiments on this algorithm.

Acknowledgements

I extend many thanks to my supervisor, Joseph Cheriyan, for his immense support working with me throughout my time as a graduate student.

Thanks to everyone on the Statement of Contributions page, who I worked with on some of these results and much more: Joseph Cheriyan, Ishan Bansal, Sanjeev Khanna, and Keven Qiu.

Thanks to Chaitanya Swamy and Ricardo Fukasawa for being my official thesis readers, and thanks to everyone who attended my thesis presentation and asked thought-provoking new questions.

Additional thanks to Ishan Bansal for reading sections of my thesis and suggesting new ideas.

I am thankful for the entire C&O department at the University of Waterloo for fostering a supportive community.

Final thanks to my family and friends for their never-ending kindness.

Dedication

To my wife; I hope it's as interesting as Marxist theory.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgements v

Dedication vi

List of Figures x

List of Tables xii

1 Introduction 1

1.1 Network Design and Cut Covering Problems 1

1.1.1 Overview 1

1.1.2 Cover Small Cuts 3

1.1.3 WGMV Primal-Dual Algorithm 4

1.2 Capacitated Network Design 6

1.3 Flexible Graph Connectivity 6

1.4 Our Results and Structure 7

1.4.1 Cover Small Cuts and Strongly Pliable Families . . . 7
1.4.2 Flexible Graph Connectivity . . . 8
1.4.3 WGMV Computational Results . . . 9
1.5 Notational Remarks and Definitions . . . 9

2 Cover Small Cuts and Strongly Pliable Families **10**
2.1 Bad Example for Jain's Iterative Rounding . . . 10
2.1.1 Graph Construction . . . 12
2.1.2 Links Construction . . . 12
2.1.3 The rank of the incidence matrix of small cuts and links . . . 15
2.1.4 Example with $k = 4$ . . . 18
2.2 Special Case of Small Cut Families . . . 19
2.3 Strongly Pliable Families . . . 31
2.3.1 Overview . . . 31
2.3.2 Strongly Pliable Non-Cover Small Cuts Example . . . 33
2.3.3 5-Approximation for Cut Covering Problems on Strongly Pliable Families . . . 41
2.3.4 Tightness of Primal-Dual Ratio for Strongly Pliable Cut Covering Problems Using WGMV . . . 46
2.3.5 Subfamilies of Pliable Families . . . 50
2.4 Other Cut Covering Problems . . . 53
2.4.1 Cover Large Cuts . . . 53
2.4.2 Cover Bounded Cuts . . . 55
2.4.3 k-Cover Small Cuts . . . 56
2.4.4 Intersection of Small Cuts . . . 57

3 Flexible Graph Connectivity **59**
3.1 7-Approximation for (1,q)-FGC . . . 59
3.2 Relation Between FGC and CND . . . 64
3.3 2-Approximation for "Soft" (p,q)-FGC . . . 66
3.4 Approximation for "Semi-Hard" (p,q)-FGC . . . 68

4 Primal-Dual Computational Results **73**
4.1 WGMV Algorithm Summary . . . 73
4.1.1 Phase 1: Determine active sets . . . 74
4.1.2 Phase 2: Augment dual solution . . . 74
4.1.3 Phase 3: Reverse delete . . . 75
4.1.4 Algorithm . . . 75
4.2 WGMV Maple Code . . . 75
4.2.1 getOrder: determines minimum-capacity order of vertices . . . 76
4.2.2 contr: contracts an edge-set or link-set . . . 76
4.2.3 cutCap: determines cut capacity . . . 76
4.2.4 activeSets: determines the active sets for any iteration . . . 77
4.2.5 primalDualIt: runs one iteration of primal-dual algorithm . . . 77
4.2.6 primalDualFull: the full algorithm . . . 78
4.3 WGMV Algorithm Experiments . . . 79
4.4 Randomized Algorithm for Small Cuts . . . 79
4.4.1 Accuracy Tests . . . 81

5 Conclusion **84**
5.1 Overview . . . 84
5.2 Open Problems . . . 85

References **87**

Index **91**

List of Figures

2.1 The capacitated graph for $k = 4$. The small cuts are given by the nested sets $N_1, \ldots, N_7$ (indicated by dashed lines) and the sets Q_1, Q_2, Q_3 (indicated by ovals). The edges are indicated by blue lines, labelled with their capacities. 13

2.2 The links graph for $k = 4$. The links partition into s,t-paths $P_1, \ldots, P_4$, and the links of each of these s,t-paths is indicated by a distinct colour. Thus, the links of P_1 are ℓ_1, ℓ_5, ℓ_7, the links of P_2 are ℓ_2, ℓ_8, ℓ_9, the links of P_3 are $\ell_3, \ell_6, \ell_{10}$, and P_4 has the link ℓ_4. The small cuts are given by the nested sets $N_1, \ldots, N_7$ (indicated by dashed lines) and the sets Q_1, Q_2, Q_3 (indicated by ovals). 14

2.3 A set family that satisfies the properties of Theorem 13, and its capacitated edge set. The red sets are $\mathcal{F}_1$ and the blue set is $\mathcal{F}_2$. The orange edges are E_1 and the green edge is E_2. Note that the non-trivial cuts with total capacity less than 5 are induced by these sets and no other sets. 25

2.4 Visualization of the subcases of Claim 20, Case 1 28

2.5 Visualization of Claim 20 Case 2, Subcase 1. Nodes that are coloured with a black-white gradient could be either colour. The blue edges are guaranteed to be in $\delta(R)$. 29

2.6 Visualization of Claim 20 Case 2, Subcase 2. Nodes that are coloured with a black-white gradient could be either colour. The blue edges are guaranteed to be in $\delta(R)$. 30

2.7 Visualization of Property γ for a strongly pliable family 32

2.8 A visualization of the partitioning of $\mathcal{F}$, the strongly pliable family returned by Algorithm 1, into iterations. In each iteration, crossing pairs from the previous iterations are examined, and corner cuts are added to $\mathcal{F}$ if they are not already present. 35

2.9 Initial setup for Algorithm 1 when $k = 3$. 35
2.10 Visualization of the subcases of Claim 28, Case 2. 38
2.11 Visualization of Property γ^*. 44
2.12 Visualization of Fact 4. 44
2.13 Visualization of Lemma 35, Case 4. 46
2.14 Visualization of the initial sets and links for the $k = 3$ instance in the family of examples with primal-dual ratio 5 using the WGMV primal-dual algorithm. The red sets are the active sets. 48
2.15 Visualization of the subfamilies of pliable families. The figure is not "to scale". 52
2.16 One example of Nutov's family of cut covering problems with primal-dual ratio approaching 6. These examples are sparse γ-pliable, but not strongly pliable. 52
2.17 Non-pliable Cover Bounded Cuts instance. Edges are depicted and have capacity 1. 56
2.18 Non-pliable Intersection of Small Cuts instance 57

List of Tables

4.1 Results of implementing the WGMV primal-dual algorithm on various graphs. 80
4.2 Results of random algorithm Test 1 81
4.3 Results of random algorithm Test 2 82
4.4 Results of random algorithm Test 3 83
4.5 Results of random algorithm Test 4 83

Chapter 1

Introduction

1.1 Network Design and Cut Covering Problems

1.1.1 Overview

Network design is a major field in optimization. Graphs can represent a wide variety of practical applications. Scenarios are modelled using graphs $G = (V, E)$ with nodes and edges. Network design optimization problems aim to manipulate the graph (such as choosing a subset of edges) that satisfy desired properties for a minimized cost. Depending on the problem, cost could be a monetary amount, an amount of time, processing power on a computer, or the like. One of the simplest network design problems is the minimum spanning tree problem, which aims to construct a connected spanning subgraph on a graph for minimum cost. However, there are many more complex problems with interesting applications.

Given a graph $G(V, E)$, the **cut** induced by a subset $S \subseteq V$ is $\delta(S) = \{uv \in E : u \in S, v \notin S\}$. An edge e **covers** a cut $\delta(S)$ if $e \in \delta(S)$. A set $S \subseteq V$ is **non-trivial** if it is not equal to $\emptyset$ or V. A cut is also called **non-trivial** if it is induced by a non-trivial set.

We introduce a key problem in network design.

Cut Covering Problem
Input: a graph $G = (V, E)$ such that each edge $e \in E$ has a cost $c_e \in \mathbb{Q}_{\geq 0}$; a family $\mathcal{F}$ containing non-trivial subsets of V.
Goal: find a subset $F \subseteq E$ of minimum c-cost such that for each $S \in \mathcal{F}$, the solution satisfies $|F \cap \delta(S)| \geq 1$.

We can represent cut covering problems as integer programs, as shown below. Let x be a variable such that x_e denotes the inclusion of edge e in the solution set if and only if $x_e = 1$.

$$\text{minimize} \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta(S)} x_e \geq 1, \quad \forall\, S \in \mathcal{F}$$

$$x \in \{0, 1\}^{|E|}$$

Many significant problems in combinatorial optimization can be formulated as cut covering problems. For example, if $\mathcal{F}$ is every non-trivial vertex set, then the problem is equivalent to the minimum spanning tree problem. If we have a set of terminals $s_1, ..., s_j$ such that $\mathcal{F} = \{S \subset V : s_i \in S \text{ for at least one } i \in [j], \{s_1, ..., s_j\} \not\subseteq S\}$, then the problem is equivalent to the minimum Steiner tree problem. If we have a set of terminal pairs $(s_1, t_1), ..., (s_j, t_j)$, such that $\mathcal{F} = \{S \subset V : s_i \in S, t_i \notin S \text{ for at least one } i \in [j]\}$, then the problem is equivalent to the minimum Steiner forest problem.

Cut covering problems are also a special case of **f-connectivity problems**.

f-Connectivity Problem
Input: a graph $G = (V, E)$ such that each edge $e \in E$ has a cost $c_e \in \mathbb{Q}_{\geq 0}$; a function $f : 2^V \to \mathbb{Z}_{\geq 0}$.
Goal: find a subset $F \subseteq E$ of minimum c-cost such that for each $S \subseteq V$, the solution satisfies $|F \cap \delta(S)| \geq f(S)$.

Jain [21] has a 2-approximation algorithm using **iterative rounding** for cut covering problems and f-connectivity problems that satisfy certain properties.

Definition 1. *A family $\mathcal{F}$ of subsets of V is **uncrossable** if for any $A, B \in \mathcal{F}$, at least one of the following holds:*

- $A \cap B, A \cup B \in \mathcal{F}$
- $A - B, B - A \in \mathcal{F}$

Definition 2. *A function $f : 2^V \to \mathbb{Z}$ is **weakly supermodular** if for any $A, B \subseteq V$, at least one of the following holds:*

- $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$
- $f(A) + f(B) \leq f(A - B) + f(B - A)$

Theorem 1. *[21, Theorem 3.4] There exists an iterative rounding 2-approximation algorithm for f-connectivity problems where f is integral, weakly supermodular, and has at least one positive value. Moreover, the same algorithm achieves approximation ratio 2 for cut covering problems on uncrossable families.*

1.1.2 Cover Small Cuts

Given a graph with capacitated edges, the graph's **small cuts**, cuts with total capacity below a given threshold (in particular, cuts with total capacity within a factor of the minimum cut capacity) have been studied for some time (see [23]). The cut covering problem using families of small cuts was introduced by Bansal et al. [4]. Note that the input below has two edge sets, which we denote "edges" and "links" for clarity. The capacitated edges dictate which cuts are small, and the links are used in the covering optimization problem.

Cover Small Cuts
Input: an instance consists of $G(V, E), u, \lambda$, and L, c.

- $G = (V, E)$ is a graph.
- $u \in \mathbb{Q}_{\geq 0}^{|E|}$ is a vector assigning a capacity u_e to each $e \in E$.
- $\lambda \geq 0$ is a threshold value.
- $L \subseteq \binom{V}{2}$ is a set of links.
- $c \in \mathbb{Q}_{\geq 0}^{|L|}$ is a vector assigning a cost c_e to each $e \in L$.

Goal: find a subset $F \subseteq L$ of minimum c-cost such that for each $\delta(S)$ where $S \subset V, S \neq \emptyset$, and $\sum_{e \in \delta_E(S)} u_e < \lambda$, the solution satisfies $|F \cap \delta_L(S)| \geq 1$.

Definition 3. *Given a graph $G(V, E)$ with capacities u on the edges and threshold λ, a* ***small cut*** *is a cut $\delta(S), S \subset V, S \neq \emptyset$, such that $u(\delta_E(S)) = \sum_{e \in \delta_E(S)} u_e < \lambda$.*

The ***family of small cuts*** *for the graph instance is $\{S : S \subset V, S \neq \emptyset, \sum_{e \in \delta_E(S)} u_e < \lambda\}$*

1.1.3 WGMV Primal-Dual Algorithm

When an optimization problem can be formulated by a linear program, it is often useful to examine the **dual** as well. Many optimization problems admit **primal-dual algorithms**, which construct a primal solution and a dual solution together using the property of complementary slackness. There are exact algorithms (for example, for the shortest path problem) that output primal and dual solutions of equal objective value, which confirms their optimality.

For some NP-hard optimization problems, the best we can do with current knowledge is a **primal-dual approximation algorithm**, which returns a primal and dual solution of possibly non-equal objective values. We then justify that the outputted solution's objective value is within a given factor of the real optimal value.

Williamson, Goemans, Vazirani, and Mihail [28] (henceforth "WGMV") constructed a 2-approximation algorithm for cut covering problems on uncrossable families. This was an extension of approximation algorithms for the minimum Steiner forest problem by Goemans and Williamson [18] and Agarwal et al. [2]. The WGMV approximation algorithm utilizes the LP relaxation of the cut covering problem and its dual. The LP relaxation of a cut covering problem on $\mathcal{F}$ is

$$\begin{aligned} &\text{minimize} \sum_{e \in E} c_e x_e \\ &\text{subject to} \\ &\qquad \sum_{e \in \delta(S)} x_e \geq 1, \quad \forall\, S \in \mathcal{F} \\ &\qquad\qquad x \geq 0 \end{aligned}$$

The dual LP is

$$\begin{aligned} &\text{maximize} \sum_{S \in \mathcal{F}} y_S \\ &\text{subject to} \end{aligned}$$

$$\sum_{e \in \delta(\mathcal{S})} y_S \leq c_e, \quad \forall\, e \in E$$
$$y \geq 0$$

Definition 4. *Let $\mathcal{F}$ be a set family for a cut covering problem and $F \subseteq E$. A set $S \in \mathcal{F}$ is **violated** with respect to F if $|F \cap \delta(S)| < 1$.*

*The **residual family** of $\mathcal{F}$ with respect to F is the subfamily of all violated sets.*

Note that an edge set is a feasible solution to a cut covering problem if it results in no violated sets.

Definition 5. *An **active set** (also called a **core**) is a violated set in $\mathcal{F}$ that is inclusion-wise minimal with respect to the subfamily of all violated sets.*

To briefly summarize the WGMV approximation algorithm, in each iteration, the dual variables associated with the active sets are increased uniformly until the dual constraint for an edge becomes tight. That edge is added to a candidate solution set F. The algorithm continues until there are no violated sets with respect to F. A key feature of the WGMV algorithm is the **reverse delete phase**. When the algorithm finishes the previous phase, the edges are re-examined in reverse order of inclusion in F, and an examined edge e is removed from F if $F - \{e\}$ is still a feasible solution. The algorithm is described in full in Section 4.1.

WGMV's primal-dual approximation algorithm can be used for other cut covering problems beyond uncrossable families, although the approximation ratio may be worse than 2. For example, Bansal et al. [4] showed that the WGMV algorithm is a 16-approximation for the Cover Small Cuts problem. This was later improved to a 6-approximation by Bansal [3] (see also [24]).

Our analysis of the approximation algorithm will use the following definition.

Definition 6. *Given the primal integer solution x^* and dual solution d^* outputted by the WGMV primal-dual algorithm for a cut covering problem on set family $\mathcal{F}$, the **primal-dual ratio** is*

$$\frac{\sum_{e \in E} c_e x_e^*}{\sum_{S \in \mathcal{F}} d_S^*}$$

.

The analysis of the approximation ratio for the WGMV approximation algorithm derives an upper bound for the primal-dual ratio. To show the tightness of the analysis, we can construct an example such that the primal-dual ratio matches the derived upper bound.

1.2 Capacitated Network Design

Capacitated network design (CND) is a long-studied topic. Connecting a graph while maintaining a universal connectivity requirement is useful in many network design applications. In the minimum spanning tree problem, we require one edge to cover each non-trivial cut. In k-edge-connected-spanning subgraph problems, we require k edges to cover each non-trivial cut. In both of these examples, each edge contributes one "unit" of coverage to each cut it intersects. The Capacitated k-Edge-Connected Spanning Subgraph problem generalizes this, allowing edges to contribute defined **capacities**.

Capacitated k-Edge-Connected Spanning Subgraph (Cap-k-ECSS)
Input: a graph $G(V, E)$ where each edge $e \in E$ has capacity $u_e \geq 0$ and cost $c_e \geq 0$; a threshold $k \geq 0$.
Goal: find a minimum c-cost subset of edges $F \subseteq E$ such that for every non-trivial cut $\delta(S), S \subset V$, the total capacity of edges in $\delta(S) \cap F$ is at least k.

This problem admits the following integer program.

$$\text{minimize} \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta(S)} u_e x_e \geq k, \quad \forall\, S \subset V, S \neq \emptyset$$

$$x \in \{0, 1\}$$

The Cap-k-ECSS problem has several approximation algorithms. One of the earliest is a $\min\{2k, |E|\}$-approximation algorithm by Goemans et al. [17]. The most recent as of this thesis is a $O(\log(k/u_{min}))$-approximation algorithm (where u_{min} is the minimum edge capacity in the graph) by Bansal et al. [6]. This thesis will not be studying the Cap-k-ECSS problem directly, but it will be useful in some of our other analysis.

1.3 Flexible Graph Connectivity

Flexible Graph Connectivity (henceforth "FGC") problems were first introduced by Adjiashvili et al. [1]. In the (p, q)-**Flexible Graph Connectivity problem**, generalized

by Boyd et al. [7], we have a graph $G(V, E)$ such that the edges are partitioned into two sets $E = \mathcal{S} \cup \mathcal{U}$. $\mathcal{S}$ is the set of **safe edges**, representing edges in a network that are consistently reliable. $\mathcal{U}$ is the set of **unsafe edges**, representing edges that could break down. We aim to determine a minimum-cost set of edges that satisfy given connectivity requirements, even when some unsafe edges fail.

(p, q)-Flexible Graph Connectivity ((p, q)-FGC)
Input: a graph $G(V, E)$ such that each edge $e \in E$ has cost $c_e \geq 0$; and the edges are partitioned into $E = \mathcal{S} \cup \mathcal{U}$.
Goal: find a minimum c-cost $F \subseteq E$ such that the subgraph $G(V, F)$ is p-edge-connected, and remains p-edge-connected when any q or fewer unsafe edges are removed from the graph.
Equivalently, F is feasible if every non-trivial cut $\delta(S), S \subset V$ satisfies either $|F \cap \delta_{\mathcal{S}}(S)| \geq p$ or $|F \cap \delta_E(S)| \geq p + q$.

1.4 Our Results and Structure

1.4.1 Cover Small Cuts and Strongly Pliable Families

Families of small cuts are not uncrossable, so Theorem 1 does not apply to the Cover Small Cuts problem. It was an open problem whether the Cover Small Cuts problem could be approximated within ratio $O(1)$ using Jain's iterative rounding algorithm. In Section 2.1, we construct an example of the Cover Small Cuts problem such that the property of extreme point solutions used to prove an $O(1)$-approximation for Jain's iterative rounding method is not satisfied.

While they are not uncrossable, families of small cuts satisfy a weaker property.

Definition 7. *A family $\mathcal{F}$ of subsets of V is **pliable** if for any $A, B \in \mathcal{F}$, at least two of $A \cap B, A \cup B, A - B, B - A$ are also in $\mathcal{F}$.*

Diestel et al. [11], [12, Chapter 12],[13], [14],[15],[16] studied abstract separation systems satisfying **structural submodularity** (we refer the reader to these citations for relevant definitions and results). In this thesis, we explore this property with respect to cut covering problems. We were not aware of the previous work until near the end of writing this thesis; we therefore gave the property a different name, strong pliability. We continue to use this name, which is consistent with the terminology of [4].

Definition 8. *A set family $\mathcal{F}$ is **strongly pliable** if for any $A, B \in \mathcal{F}$ that cross, then **both** of the following hold:*

- *At least one of $A \cup B$ or $A \cap B$ is in $\mathcal{F}$.*
- *At least one of $A - B$ or $B - A$ is in $\mathcal{F}$.*

We prove that families of small cuts are strongly pliable, but the converse does not necessarily hold, in Sections 2.3.1 and 2.3.2, respectively. We present some strongly pliable families that can be realized as a family of small cuts in Section 2.2. As mentioned, WGMV's primal-dual approximation algorithm was previously proven to be a 6-approximation for the Cover Small Cuts problem ([4], [24], [3]). In Section 2.3.3, we prove that the WGMV primal-dual algorithm is a 5-approximation algorithm for strongly pliable families, and in Section 2.3.4 we present an example with primal-dual ratio 5, confirming that this analysis is tight. In Section 2.3.5, we compare some subfamilies of pliable families, such as strongly pliable families and uncrossable families.

Definition 9. *A non-trivial set family $\mathcal{F}$ over universe V can be **realized as a family of small cuts** if there exists an edge set $E \subseteq \binom{V}{2}$, capacity assignment $u \in \mathbb{Q}_{\geq 0}^{|E|}$, and threshold $\lambda > 0$ such that $\{S : S \subset V, S \neq \emptyset, \sum_{e \in \delta_E(S)} u_e < \lambda\} = \mathcal{F}$.*

We end Chapter 2 with an overview of four other cut covering problem in Section 2.4. The first two have integrality ratio $\Omega(n)$, where n is the number of vertices in the graph.

1.4.2 Flexible Graph Connectivity

In Chapter 3, we explore the Flexible Graph Connectivity problem. We design and analyze a 7-approximation algorithm for the $(1, q)$-FGC problem. This improves the previous best approximation ratio of $(q+1)$ [7]. There are similarities between the FGC and Cap-k-ECSS problems, which are discussed in [6] and Section 3.2 of this thesis.

We also analyze two relaxed versions of the (p, q)-FGC problem:

- "soft" (p, q)-FGC, where any edge can be selected multiple times; we design a 2-approximation algorithm for the soft problem in Section 3.3.
- "semi-hard" (p, q)-FGC, where unsafe edges can be selected multiple times; we design an $O(p^2)$-approximation algorithm for the semi-hard problem in Section 3.4.

1.4.3 WGMV Computational Results

Finally, in Chapter 4, we explore WGMV's primal-dual approximation algorithm in more detail, specifically for the Cover Small Cuts problem. We developed a program in Maple that implements the algorithm, and ran it on a variety of instances. One of the goals of implementing the algorithm was to find instances of the problem with large primal-dual ratio. It was challenging to find graphs with primal-dual ratio more than 2.

1.5 Notational Remarks and Definitions

Throughout this thesis, we omit the word "problem" from named problems. For example, when we say "Cover Small Cuts", we mean "the Cover Small Cuts problem".

When a set family possesses a property, we will also say that the cut covering problem associated with that family possesses the property. For example, when we say "a pliable cut covering problem", we mean "a cut covering problem for which the set family $\mathcal{F}$ to be covered is a pliable family".

Definition 10. *Two sets $A, B \subseteq V$ **cross** if $A - B, B - A, A \cap B$, and $V - (A \cup B)$ are all non-empty.*

Definition 11. *When two sets $A, B \subseteq V$ cross, their **corner cuts** are $A-B, B-A, A\cap B$, and $A \cup B$.*

Definition 12. *A family of sets is **laminar** if for any pair of sets A, B in the family, either $A \subseteq B, B \subseteq A$, or A and B are disjoint.*

Definition 13. *Let $\mathcal{F}$ be a laminar family. A set $A \in \mathcal{F}$ is the **parent** of a set $B \in \mathcal{F}$, and B is a **child** of A, if A is the smallest set in $\mathcal{F}$ that properly contains B. We say the parent of B is V if no other set in $\mathcal{F}$ properly contains it.*

Definition 14. *A solution to a linear program satisfies an inequality constraint **tightly** if it satisfies the constraint with equality.*

Definition 15. *A set $A \subseteq V$ is a **witness set** for an edge $e \in E$ if $\delta_E(A) = \{e\}$.*

Chapter 2

Cover Small Cuts and Strongly Pliable Families

2.1 Bad Example for Jain's Iterative Rounding

Jain [21] constructed a significant iterative rounding 2-approximation algorithm for the Generalized Steiner Network problem. In this problem, pairs of vertices have a **connectivity requirement**, such that a feasible edge set must contain at least as many paths between any two vertices as their connectivity requirement. This can be represented as an f-connectivity problem where for any $S \subset V, S \neq \emptyset$, $f(S)$ is the largest connectivity requirement between any pair that is separated by S. Jain's result can approximate other problems as long as certain properties are met.

Definition 16. *A family $\mathcal{F}$ of subsets of V is **uncrossable** if for any $A, B \in \mathcal{F}$, at least one of the following holds:*

- $A \cap B, A \cup B \in \mathcal{F}$
- $A - B, B - A \in \mathcal{F}$

A property that relaxes uncrossability is pliability.

Definition 17. *A family $\mathcal{F}$ of subsets of V is **pliable** if for any $A, B \in \mathcal{F}$, at least two of $A \cap B, A \cup B, A - B, B - A$ are also in $\mathcal{F}$.*

It is immediate that any uncrossable family is also pliable, but the converse is not true.

Jain's iterative rounding algorithm is a 2-approximation for any cut covering problem on an **uncrossable** set family. The proof of the algorithm's approximability relies on the following result.

Theorem 2. *[21, Theorem 3.1] Let x be a basic solution to the LP relaxation of the cut covering problem on uncrossable set family $\mathcal{F}$. Then there exists at least one edge $e \in E$ such that $x_e \geq 1/2$.*

The algorithm is then conducted as follows:

1. Calculate a basic optimal solution x^* to the linear relaxation of the cut covering problem on $\mathcal{F}$.
2. For each $e \in E$ such that $x_e^* = 0$, permanently remove e from the graph.
3. For each $e \in E$ such that $x_e^* \geq 1/2$ (and at least one is guaranteed by Theorem 2), add a constraint to the LP that permanently sets $x_e^* = 1$.
4. Repeat the steps until the solution output is integer.

If we weaken the bound of $1/2$, it is easy to modify Jain's iterative rounding algorithm as follows.

Lemma 3. *Suppose that $\mathcal{F}$ is a family of sets such that for any basic feasible solution x to the LP relaxation of the cut covering problem for any residual family of $\mathcal{F}$, there exists at least one edge $e \in E$ with $x_e \geq 1/k$.*

Then Jain's iterative rounding algorithm is a k-approximation for that instance of the cut covering problem.

We were motivated to find out if Jain's iterative rounding algorithm can be used to approximate Cover Small Cuts within a constant factor. In this section, we will show that for arbitrarily small values, we can construct a Cover Small Cuts instance with a basic solution such that all values are less than that bound. Therefore, there is no constant k such that Lemma 3 holds for all Cover Small Cuts problems.

Theorem 4. *Let $k \geq 4$ be an even integer. There is an instance of Cover Small Cuts $G = (V, E), u, \lambda$ and L, c such that the LP relaxation has a basic solution x^* where every positive variable of x^* has value $1/k$.*

2.1.1 Graph Construction

Let $k \geq 4$ be an even integer. The capacitated graph $G = (V, E)$ has $n = 2 + \binom{k}{2}$ nodes.

We denote the nodes of G by $v_1, v_2, \ldots, v_n$, as well as by the indices $1, 2, \ldots, n$. For notational convenience, let $s = v_1$ and let $t = v_n$.

The family of small cuts consists of two sub-families. One sub-family corresponds to the node-sets $N_1 = \{v_1\}, N_2 = \{v_1, v_2\}, \ldots, N_{n-1} = \{v_1, v_2, \ldots, v_{n-1}\}$. In other words, for each of the sets $N_i = \{v_1, v_2, \ldots, v_i\}, i = 1, \ldots, n-1$, $\delta_E(N_i)$ is a small cut. We call the sets N_i $(i = 1, \ldots, n-1)$ the *nested sets*; note that this is a laminar subfamily.

The other subfamily corresponds to the $(k-1)$ node-sets $Q_1 = \{v_2, v_3, \ldots, v_{1+\frac{k}{2}}\}, Q_2 = \{v_{2+\frac{k}{2}}, v_{3+\frac{k}{2}}, \ldots, v_{1+2\cdot\frac{k}{2}}\}, \ldots, Q_{(k-1)} = \{v_{2+(k-2)\cdot\frac{k}{2}}, v_{3+(k-2)\cdot\frac{k}{2}}, \ldots, v_{1+(k-1)\cdot\frac{k}{2}}\}$. Note that these sets form a partition of $V(G) - \{s, t\}$ into $(k-1)$ sets, each of cardinality $k/2$. In other words, for each of the $(k-1)$ "intervals" Q_j $(j = 1, \ldots, k-1)$ formed by picking the node with index $2 + (j-1) \cdot \frac{k}{2}$ and the next $(\frac{k}{2} - 1)$ consecutively indexed nodes, $\delta_E(Q_j)$ is a small cut. We call the sets Q_j $(j = 1, \ldots, k-1)$ the *Q-sets*.

This set family satisfies the properties that will be described in Theorem 13 with $\mathcal{F}_1$ as the nested sets and $\mathcal{F}_2$ as the Q-sets. Thus, it can be realized as a family of small cuts. In particular, the capacitated edge set is constructed as follows.

We have one edge (v_i, v_{i+1}) for all $i \in [n-1]$. These edges have capacity 2 if $i = 1$ or $i + 1 = n$, 3 if v_i and v_{i+1} are in the same Q-set, and 1 otherwise. We have an additional edge between v_1 and the first node of Q_2; an edge between the last node of Q_{k-2} and v_n; and for all $j \in [k-3]$, an edge between the last node of Q_j and the first node of Q_{j+2}. These edges have capacity 1. Using threshold $\lambda = 5$, the family of small cuts is exactly the union of nested sets and Q-sets.

Figure 2.1 illustrates the construction of the capacitated graph for $k = 4$.

2.1.2 Links Construction

Let (V, L) denote the graph of the links; thus L is the set of links, and we denote the links by the symbol ℓ, e.g., ℓ_i, ℓ', etc. The number of links is $m = n + k - 2 = \binom{k}{2} + k$. The links are partitioned into k internally node-disjoint s, t-paths that we denote by $P_1, \ldots, P_{k-1}, P_k$. The s, t-path P_k has one link $st = v_1 v_n$. Each of the s, t-paths $P_1, \ldots, P_{k-1}$ has $k/2$ internal nodes and $1 + \frac{k}{2}$ links. Moreover, the sequence of node indices of each of these s, t-paths is an increasing sequence; that is, the node sequence has the form $v_1, v_{i_2}, v_{i_3}, \ldots, v_{i_{(\frac{k}{2}+1)}}, v_n$,

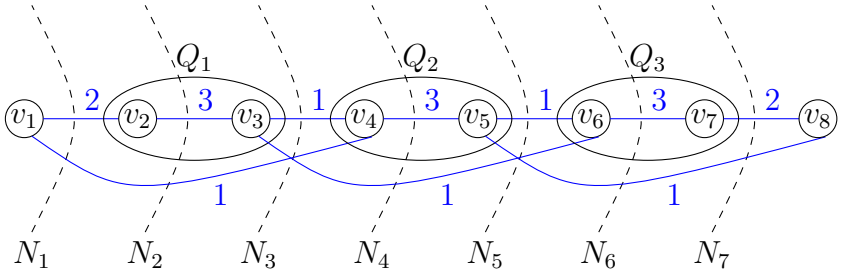


Figure 2.1: The capacitated graph for $k = 4$. The small cuts are given by the nested sets $N_1, \ldots, N_7$ (indicated by dashed lines) and the sets Q_1, Q_2, Q_3 (indicated by ovals). The edges are indicated by blue lines, labelled with their capacities.

where $1 < i_2 < i_3 < \cdots < i_{(\frac{k}{2}+1)} < n$. Each of the nodes $v_2, \ldots, v_{n-1}$ is in one of the s,t-paths $P_1, \ldots, P_{k-1}$; thus, each of these nodes is incident to two links.

Recall that we have $(k-1)$ Q-sets, each with $k/2$ nodes, and $(k-1)$ s,t-paths $P_1, \ldots, P_{k-1}$ each with $k/2$ internal nodes. Each s,t-path P_i $(i = 1, \ldots, k-1)$ is constructed such that it is incident to precisely one node from $k/2$ (of the $(k-1)$) Q-sets, and it is disjoint from the the other $(k-1-\frac{k}{2})$ Q-sets. The s,t-path P_i $(i = 1, \ldots, k-1)$ is incident to the Q-set with index i and the next $(k/2)-1$ consecutively indexed Q-sets (with "wrap around"); thus, for $i = 1, \ldots, k-1$ and $j = 1, \ldots, k-1$, the s,t-path P_i and the set Q_j have a node in common iff

$$j \geq i \quad \textbf{and} \quad j \leq \min(i + \frac{k}{2} - 1,\ k-1), \quad \textbf{or}$$
$$j < i \quad \textbf{and} \quad 1 \leq j \leq i - \frac{k}{2}.$$

If an s,t-path P_i and a set Q_j have a node in common, then any one node of Q_j could be assigned to P_i, as long as it is not the same node assigned to a different path. In other words, our construction does not specify the internal nodes of P_i, it only specifies the Q-sets that intersect P_i.

Let A^{PQ} denote the $(k-1) \times (k-1)$ zero-one matrix that has a row for each s,t-path P_i $(i = 1, \ldots, k-1)$ and a column for each set Q_j $(i = 1, \ldots, k-1)$ such that the entry $A^{PQ}_{i,j}$ is one iff P_i and Q_j have a node in common. Then, by construction, A^{PQ} has entries of zero or one such that each row has $k/2$ consecutive ones (with "wrap around") and the remaining $(k-1-\frac{k}{2})$ entries of the row are zero. Each row is identical to the row above it "shifted" by one column; this is a circulant matrix [27]. Observe that the number of ones in

each row, $k/2$, and the size of the matrix, $k-1$, are relatively prime (i.e., $\gcd(\frac{k}{2}, k-1) = 1$). For an illustration, see the example at the end of Section 2.1.3.

Lemma 5. *The matrix A^{PQ} has rank $(k-1)$. Moreover, $\det(A^{PQ}) = k/2$.*

One can prove this lemma directly, by applying row operations (and applying the matrix determinant lemma to a square matrix with diagonal entries 2 and all other entries one). Alternatively, the lemma follows directly from a result of Hariprasad [19, Theorem 2.1].

Figure 2.2 illustrates the construction of the links graph for $k = 4$. The graph has $\binom{k}{2} + 2 = 8$ nodes and $\binom{k}{2} + k = 10$ links. The links partition into $k = 4$ (internally node disjoint) s,t-paths $P_1, \ldots, P_4$; the links of $P_1, \ldots, P_4$ are indicated by distinct colours.

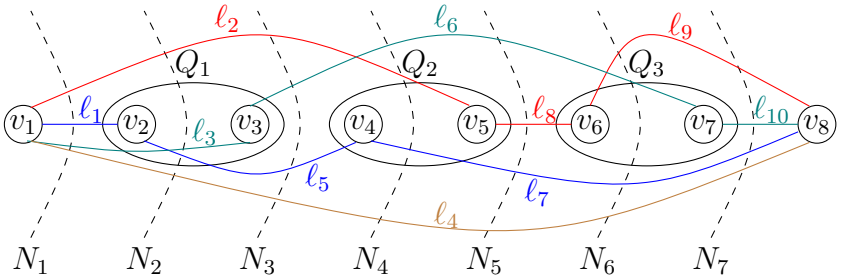


Figure 2.2: The links graph for $k = 4$. The links partition into s,t-paths $P_1, \ldots, P_4$, and the links of each of these s,t-paths is indicated by a distinct colour. Thus, the links of P_1 are ℓ_1, ℓ_5, ℓ_7, the links of P_2 are ℓ_2, ℓ_8, ℓ_9, the links of P_3 are $\ell_3, \ell_6, \ell_{10}$, and P_4 has the link ℓ_4. The small cuts are given by the nested sets $N_1, \ldots, N_7$ (indicated by dashed lines) and the sets Q_1, Q_2, Q_3 (indicated by ovals).

We construct a basic feasible solution x^* by assigning the value $1/k$ to each link, thus, $x^*_\ell = 1/k, \ \forall \ \ell \in L$. Informally speaking, we are assigning a value $1/k$ to each of the s,t-paths $P_1, \ldots, P_k$. It is easy to verify that each of the small cuts, namely, $\delta(N_i)$ $(i = 1, \ldots, n-1)$ and $\delta(Q_j)$ $(j = 1, \ldots, k-1)$, are "tightly covered" by x^*. First, consider a cut $\delta(N_i)$ $(i = 1, \ldots, n-1)$, and observe that each of the s,t-paths $P_1, \ldots, P_k$ has exactly one of its links in $\delta(N_i)$, hence, $x^*(\delta(N_i)) = (k)\frac{1}{k} = 1$. Next, consider a cut $\delta(Q_j)$ $(j = 1, \ldots, k-1)$, and observe that $k/2$ of the s,t-paths $P_1, \ldots, P_k$ have exactly one internal node in Q_j, hence, each of these s,t-paths has exactly two links in $\delta(Q_j)$, therefore, $x^*(\delta(Q_j)) = 2(\frac{k}{2})(\frac{1}{k}) = 1$.

The basis matrix A of x^* has one row for each small cut and one column for each link; the entry of A corresponding to a small cut $\delta(S)$ and a link ℓ is one if $\ell \in \delta(S)$, otherwise,

the entry is zero. Thus, A is an $m \times m$ matrix with entries of zero or one; recall that $m = n + k - 2 = \binom{k}{2} + k$. In the next subsection, we show that matrix A has full rank, and that will prove our claim that x^* is a basic feasible solution of the LP relaxation of Cover Small Cuts. This then proves Theorem 4.

2.1.3 The rank of the incidence matrix of small cuts and links

Let A denote the zero-one incidence matrix of the small cuts and the links; A is an $m \times m$ matrix, where $m = n + k - 2 = \binom{k}{2} + k$. The matrix A has rows corresponding to the cuts of the Q-sets followed by rows corresponding to the cuts of the nested sets; the columns of A correspond to the links. In this section, the main goal is to show that $\det(A)$ is non-zero; equivalently, A has full rank. We start by fixing a convenient indexing for the rows and columns of A. The rows $1, \ldots, (k-1)$ of A correspond to the cuts of $Q_1, \ldots, Q_{(k-1)}$, and the rows $k, (k+1), \ldots, m$ of A correspond to the cuts of $N_1, \ldots, N_{(n-1)}$. The links $\ell \in L$ are indexed as follows: The link of the s,t-path P_i $(i = 1, \ldots, k)$ incident to node $s = v_1$ is denoted ℓ_i; thus, each of $\ell_1, \ldots, \ell_k$ is in $\delta(s)$, and there is a bijection between these links and the s,t-path $P_1, \ldots, P_k$. Note that each of the remaining links is incident to one of the nodes $v_2, \ldots, v_{n-1}$, and, for each $i = 2, \ldots, (n-1)$, the unique link that is incident to node v_i and is in the cut $\delta(N_i)$ is assigned the index $(k + i - 1)$. The columns $1, \ldots, m$ of A correspond to the links $\ell_1, \ldots, \ell_m$.

Let us view the matrix A as a 2×2 block matrix $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ where A_{11} is the $(k-1) \times (k-1)$ sub-matrix of A whose rows correspond to the cuts of $Q_1, \ldots, Q_{(k-1)}$ and whose columns correspond to the links $\ell_1, \ldots, \ell_{(k-1)}$, and A_{22} is the $(m-k+1) \times (m-k+1)$ sub-matrix of A whose rows correspond to the cuts of $N_1, \ldots, N_{(n-1)}$ and whose columns correspond to the links $\ell_k, \ell_{(k+1)}, \ldots, \ell_m$.

Claim 6. *The sub-matrix A_{22} is lower-triangular and all entries on the diagonal are one; thus,* $\det(A_{22}) = 1$.

Proof. The link ℓ_k (with end nodes s and t) is in each of the cuts $\delta(N_i)$, $i = 1, \ldots, (n-1)$. For each of the links $\ell_{(k+i-1)}$ where $i = 2, \ldots, (n-1)$, note that the link is in the cut $\delta(N_i)$ and it is in none of the cuts $\delta(N_1), \ldots, \delta(N_{(i-1)})$. □

Our plan is to apply row operations to the first $(k-1)$ rows of A in order to transform the submatrix $\begin{pmatrix} A_{11} & A_{12} \end{pmatrix}$ to a submatrix of the form $\begin{pmatrix} (A^{PQ})^\top & \mathbf{0}_{(k-1)\times(m-k+1)} \end{pmatrix}$, where $(A^{PQ})^\top$ denotes the transpose of the matrix A^{PQ} and $\mathbf{0}_{(k-1)\times(m-k+1)}$ denotes a

$(k-1) \times (m-k+1)$ matrix of zeros. Therefore, using row operations, we will rewrite the matrix A as the 2×2 block matrix $\begin{pmatrix} (A^{PQ})^\top & \mathbf{0}_{(k-1)\times(m-k+1)} \\ A_{21} & A_{22} \end{pmatrix}$; then, by a result on the determinant of 2×2 block matrices, we have $\det(A) \neq 0$ if and only if $\det(A^{PQ}) \det(A_{22}) = \det(A^{PQ}) \neq 0$; the latter is indeed true by Lemma 5.

See the example in Section 2.1.4 for an illustration.

Let us introduce some notation to describe and analyze our row operations. For any set of links L' let $\chi^{L'}$ denote the zero-one row incidence vector of L'; thus, for any link ℓ, $\chi^{L'}_\ell = 1$ iff $\ell \in L'$. Note that rows $1, \ldots, (k-1)$ of A correspond to $\chi^{\delta(Q_1)}, \ldots, \chi^{\delta(Q_{(k-1)})}$. For any link ℓ, let $\phi(\ell)$ denote the index i of the s,t-path P_i that contains ℓ. For each of the links ℓ_i, $i = 1, \ldots, (k-1)$, note that $\phi(\ell_i) = i$, since we have a bijection between these links and the s,t-paths $P_1, \ldots, P_{(k-1)}$. For any set of links L', let $\phi(L')$ denote the set of indices of the s,t-paths $P_1, \ldots, P_{(k-1)}$ that contain at least one of the links of L'; thus, $\phi(L') \subseteq \{1, \ldots, k\}$, and index i is in $\phi(L')$ iff the s,t-path P_i contains one of the links of L'. For example, if L' is a subset of $\{\ell_1, \ldots, \ell_{(k-1)}\}$, then $\phi(L') = \{i \ : \ \ell_i \in L'\}$.

Consider any row j, $j = 1, \ldots, (k-1)$ of the matrix A; this row-vector is the same as $\chi^{\delta(Q_j)}$. Let g denote the largest index i such that the nested set N_i is disjoint from Q_j, and let h denote the smallest index i such that the nested set N_i contains Q_j. (By our construction, the $k/2$ nodes of Q_j are in the interval $[(g+1), (g+2), \ldots, h]$.)

Claim 7. $\left(\chi^{\delta(Q_j)} - \chi^{\delta(N_h)} + \chi^{\delta(N_g)}\right) = 2\chi^{L'}$, *where* L' *is the set of* $k/2$ *links in* $\delta(Q_j) \cap \delta(N_g)$.

Proof. By construction, every link has its two end nodes in two different Q-sets. Hence, every link $\ell \in \delta(Q_j)$ that has its lower-index node in Q_j is in $\delta(N_h)$; similarly, every link $\ell \in \delta(Q_j)$ that has its higher-index node in Q_j is in $\delta(N_g)$. Moreover, $\delta(N_g) - \delta(Q_j) = \delta(N_h) - \delta(Q_j)$ is the set of links that have their lower-index node in N_g and their higher-index node in $V - N_h$. The equation in the claim follows from these statements. □

In the above claim, note that L' is contained in the cut of a Q-set, hence, the link $\ell_k = st$ cannot be in L'.

Our plan is to (separately) apply the row operations described in the above claim to each of the rows $j = 1, \ldots, (k-1)$ of A. The next lemma explains how to apply row operations to $\chi^{L'}$, where L' is any set of $k/2$ links contained in some nested cut $\delta(N_i)$ and $\ell_k \notin L'$, to transform $\chi^{L'}$ to the vector $\chi^{\widehat{L}}$, where $\widehat{L} = \{\ell_i \ : \ i \in \phi(L')\}$. Informally speaking, we can apply row operations to "map" any row vector $\chi^{L'}$, where $L' \subset \delta(N_i) - \{\ell_k\}$ and $|L'| = k/2$, to the row incidence vector (padded with zeros on the right) of the $k/2$ s,t-paths among

$P_1, \ldots, P_{(k-1)}$ that contain (at least) one of the links of L'. Thus, by applying the above claim and the next lemma, we "map" each row of the submatrix $\begin{pmatrix} A_{11} & A_{12} \end{pmatrix}$ to a distinct row of the submatrix $\begin{pmatrix} (A^{PQ})^\top & \mathbf{0}_{(k-1)\times(m-k+1)} \end{pmatrix}$.

Lemma 8. *Let $i = 1, \ldots, n-1$, and let $L' \subset \delta(N_i) - \{\ell_k\}$ be any set of $k/2$ links. By applying row operations, one can transform the vector $\chi^{L'}$ to the vector $\chi^{\widehat{L}}$, where $\widehat{L} = \{\ell_i \ : \ i \in \phi(L')\}$.*

Proof. By induction on the index of the nested set whose cut contains L'.

Let h be the smallest index such that the nested cut $\delta(N_h)$ contains L'. If $h = 1$, then we have $\widehat{L} = L'$ and the lemma is proved.

Now, suppose $h \geq 2$; thus, L' contains a link ℓ_j with $j > k$. Observe that L' contains the link $\ell_{(h+k-1)}$ which is incident to node v_h and is in the cut $\delta(N_h)$; informally speaking, the presence of $\ell_{(h+k-1)}$ in L' certifies that L' is contained in $\delta(N_h)$ and L' is not a subset of $\delta(N_j)$ for any $j < h$. Let $\overline{L}$ denote the set of links $\delta(N_h) - L'$; $\overline{L}$ is a "complementary set" of $k/2$ links w.r.t. $\delta(N_h)$ and L'. For every link $\ell_j \in \overline{L}$, observe that $j < (h+k-1)$ (since $\ell_j \in \delta(N_h)$ and $\ell_{(h+k-1)} \notin \overline{L}$).

Let g be the smallest index such that the nested cut $\delta(N_g)$ contains $\overline{L}$. Again, observe that $\overline{L}$ contains the link $\ell_{(g+k-1)}$ which is incident to node v_g and is in the cut $\delta(N_g)$. Clearly, $g < h$ and $\delta(N_g) \neq \delta(N_h)$, since $\ell_{(h+k-1)} \in \delta(N_h)$ and $\ell_{(h+k-1)} \notin \delta(N_g)$.

Claim 9. *There exists a set of $k/2$ links of $\delta(N_g) - \{\ell_k\}$, call it L'', such that*
(i) $\chi^{L''} = \chi^{L'} - \chi^{\delta(N_h)} + \chi^{\delta(N_g)}$, *and*
(ii) $\phi(L'') = \phi(L')$.

Proof. Observe that $\overline{L} = \delta(N_h) - L'$, and that $\overline{L}$ is contained in $\delta(N_g)$; also, note that ℓ_k is in $\overline{L}$. Let $L'' = \delta(N_g) - \overline{L}$; observe that L'' is a set of $k/2$ links and $\ell_k \notin L''$. Hence, we have $\chi^{\delta(N_g)} - \chi^{\delta(N_h)} = \chi^{L''} - \chi^{L'}$. This proves part (i) of the claim.

Moreover, note that each of the nested cuts contains exactly one link from each of the s,t-paths $P_1, \ldots, P_k$; that is, for each $i = 1, \ldots, n-1$, $\phi(\delta(N_i)) = \{1, \ldots, k\}$. Hence, $\phi(L') = \phi(\delta(N_h) - \overline{L}) = \{1, \ldots, k\} - \phi(\overline{L}) = \phi(\delta(N_g) - \overline{L}) = \phi(L'')$. This proves part (ii) of the claim. □

In the induction step (of the proof of the lemma), we apply the above claim to replace the set L' by the set L''; note that L'' has the properties of L' (listed in the lemma statement) and $\phi(L'') = \phi(L')$. Moreover, L'' is contained in the cut of a nested set of smaller index than the index of any nested set whose cut contains L'.

Thus, the lemma follows by induction. □

2.1.4 Example with $k = 4$

In this subsection, we fix $k = 4$, and we illustrate our row operations, in particular, the row operations of Claims 7 and 9, on the links graph shown in Figure 2.2. The links partition into s, t-paths $P_1, \ldots, P_4$, and the links of P_1 are ℓ_1, ℓ_5, ℓ_7, the links of P_2 are ℓ_2, ℓ_8, ℓ_9, the links of P_3 are $\ell_3, \ell_6, \ell_{10}$, and P_4 has the link ℓ_4.

The incidence matrix A^{PQ} of the s, t-paths P_1, P_2, P_3 and the Q-sets is given below.

$$\begin{array}{c} \begin{array}{ccc} Q_1 & Q_2 & Q_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{array}{c} P_1 \\ P_2 \\ P_3 \end{array} \end{array}$$

The incidence matrix A of the small cuts and the links is given below. Note that the submatrix A_{22} (of the nested cuts and the links $\ell_4, \ldots, \ell_{10}$) is lower-triangular with diagonal entries of one.

$$\begin{array}{c} \begin{array}{ccc|ccccccc} \ell_1 & \ell_2 & \ell_3 & \ell_4 & \ell_5 & \ell_6 & \ell_7 & \ell_8 & \ell_9 & \ell_{10} & \end{array} \\ \left(\begin{array}{ccc|ccccccc} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & Q_1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & Q_2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & Q_3 \\ \hline 1 & 1 & 1 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & 0 & N_1 \\ 0 & 1 & 1 & 1 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & N_2 \\ 0 & 1 & 0 & 1 & 1 & \textcircled{1} & 0 & 0 & 0 & 0 & N_3 \\ 0 & 1 & 0 & 1 & 0 & 1 & \textcircled{1} & 0 & 0 & 0 & N_4 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & \textcircled{1} & 0 & 0 & N_5 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \textcircled{1} & 0 & N_6 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & \textcircled{1} & N_7 \end{array}\right) \end{array}$$

We describe the row operations for $\delta(Q_1), \delta(Q_2), \delta(Q_3)$, i.e., rows 1,2,3 of A; in particular, we describe the row operations of Claims 7,9.

Q_1: $\chi^{\delta(Q_1)} = \chi^{\{\ell_1, \ell_3, \ell_5, \ell_6\}}$. In Claim 7, for $j = 1$, we have $h = 3, g = 1$. $\left(\chi^{\delta(Q_1)} - \chi^{\delta(N_3)} + \chi^{\delta(N_1)}\right) = 2\chi^{\{\ell_1, \ell_3\}}$.

Note that $\phi(\delta(Q_1)) = \{1, 3\}$.

Q_2: $\chi^{\delta(Q_2)} = \chi^{\{\ell_2,\ell_5,\ell_7,\ell_8\}}$. In Claim 7, for $j = 2$, we have $h = 5, g = 3$. $\left(\chi^{\delta(Q_2)} - \chi^{\delta(N_5)} + \chi^{\delta(N_3)}\right) = 2\chi^{\{\ell_2,\ell_5\}}$.

Next, Claim 9 is applied with $L' = \{\ell_2, \ell_5\}$, and we have $h = 2, g = 1$. $\chi^{\{\ell_2,\ell_5\}} - \chi^{\delta(N_2)} + \chi^{\delta(N_1)} = \chi^{\{\ell_1,\ell_2\}}$.

Note that $\phi(\delta(Q_2)) = \{1, 2\}$.

Q_3: $\chi^{\delta(Q_3)} = \chi^{\{\ell_6,\ell_8,\ell_9,\ell_{10}\}}$. In Claim 7, for $j = 3$, we have $h = 7, g = 5$. $\left(\chi^{\delta(Q_3)} - \chi^{\delta(N_7)} + \chi^{\delta(N_5)}\right) = 2\chi^{\{\ell_6,\ell_8\}}$.

Next, Claim 9 is applied with $L' = \{\ell_6, \ell_8\}$, and we have $h = 5, g = 4$. $\chi^{\{\ell_6,\ell_8\}} - \chi^{\delta(N_5)} + \chi^{\delta(N_4)} = \chi^{\{\ell_2,\ell_6\}}$.

Next, Claim 9 is applied with $L' = \{\ell_2, \ell_6\}$, and we have $h = 3, g = 2$. $\chi^{\{\ell_2,\ell_6\}} - \chi^{\delta(N_3)} + \chi^{\delta(N_2)} = \chi^{\{\ell_2,\ell_3\}}$.

Note that $\phi(\delta(Q_3)) = \{2, 3\}$.

2.2 Special Case of Small Cut Families

Families of small cuts are always pliable, but they are not necessarily uncrossable (this is confirmed by Section 2.1, since otherwise families of small cuts would satisfy Theorem 2). We will also show in Section 2.3.1 that the families of small cuts are strongly pliable, but in Section 2.3.2 that not all strongly pliable families correspond to families of small cuts.

We are motivated by the question: what families of sets can be realized as families of small cuts?

Definition 18. *A non-trivial set family $\mathcal{F}$ over universe V can be **realized as a family of small cuts** if there exists an edge set $E \subseteq \binom{V}{2}$, capacity assignment $u \in \mathbb{Q}_{\geq 0}^{|E|}$, and threshold $\lambda \geq 0$ such that $\{S : S \subset V, S \neq \emptyset, \sum_{e \in \delta_E(S)} u_e < \lambda\} = \mathcal{F}$.*

We will present one such group of set families, which is useful because it contains the set families used in Section 2.1, confirming that the cut covering problems presented there are indeed Cover Small Cuts instances. We first define some new terms.

Let $\mathcal{F}$ be a laminar family over a node set V, which does not contain $\emptyset$ or V.

Definition 19. *given a set $S \in \mathcal{F} \cup \{V\}$, we can partition its nodes into **sectors** as follows.*

- *If S has no children, then it has one sector: S. This is its* ***key sector****.*
- *If S has children $D_1, ..., D_i$, then its sectors are $D_1, ..., D_i$ and $S - (D_1 \cup ... \cup D_i)$. Its* ***key sector*** *is $S - (D_1 \cup ... \cup D_i)$.*

Any edge with both endpoints in the same key sector is not part of any cut induced by a set in $\mathcal{F}$. Thus, if we try to realize $\mathcal{F}$ as a family of small cuts, if a key sector contains two nodes or more, we can add a high-capacity clique between these nodes, and contract them for convenience. Thus, we will henceforth assume that every key sector has at most one vertex.

Definition 20. *The* ***key node*** *of a set is the node within its key sector, if it exists.*

Also note that every node is a key node for some set in $\mathcal{F} \cup \{V\}$, and it is possible for a set to not have a key node.

Lemma 10. *If every set in a laminar family $\mathcal{F} \cup \{V\}$ has a key node, then there exists an edge set E that forms a rooted spanning tree such that a set $S \subseteq V$ satisfies $|\delta_E(S)| = 1$ if and only if $S \in \mathcal{F}$.*

Proof. The set of edges E will be constructed as follows. For each $S \in \mathcal{F} - \{V\}$, add an edge between the key node of S and the key node of the parent of S. Call this edge the **key edge** of S. Laminar families can be represented as trees such that each set is represented by a node, the root is the node representing V, and an edge joins two nodes if one node's associated set is the parent of the other's. This tree exactly corresponds to E; thus the edges form a rooted spanning tree.

Note that trivially, $\delta(V)$ has no edges. Since E forms a spanning tree, non-trivial cuts have at least one edge.

The rest of the proof requires two parts.

Claim 11. *For any $S \in \mathcal{F}, |\delta_E(S)| = 1$.*

Proof. Let $S \in \mathcal{F}$ with parent $S' \in \mathcal{F} \cup \{V\}$. The key edge of S is in $\delta(S)$. Assume for contradiction that there is another edge uv in $\delta(S), u \notin S, v \in S$. However, v cannot be the key node of S, because by construction, each key node only has one edge incident with a key node of a larger set. v also cannot be in a child of S; when an edge "starts" there, its other endpoint is the key node of that set's parent, which is either S itself or contained within S. This leaves no options for where v can exist; a contradiction.

Claim 12. *For any $S \subset V$ such that $|\delta_E(S)| = 1$, $S \in \mathcal{F}$.*

Proof. Let S be a cut such that $\delta_E(S) = \{uv\}$, such that u is the key node of $S' \in \mathcal{F}$ and v is the key node of the parent of S'. By symmetry of cuts, assume $u \in S$ (that is, we can replace S with $V - S$ if $u \notin S$). Removing uv from the tree disconnects the graph into two components. One of the components contains u and everything below u in the tree, which is equivalent to everything contained in a child of S'. S must contain all of these vertices as well, otherwise there would be a second edge in $\delta_E(S)$. Thus $S = S'$ as desired. □

We introduce a special case of set families that can be realized as a family of small cuts.

Theorem 13. *Let $\mathcal{F}$ be a set family on universe V such that $\mathcal{F}$ can be partitioned into $\mathcal{F}_1 \cup \mathcal{F}_2$ such that:*

- *$\mathcal{F}_1$ is a maximal laminar subfamily.*
- *$\mathcal{F}_2$ consists of pairwise-disjoint sets.*
- *For any $S \in \mathcal{F}_1$ and $D \in \mathcal{F}_2$ that cross, then $S \cup D, S - D \in \mathcal{F}_1$.*
- *The key node of each set in $\mathcal{F}_1 \cup \{V\}$ exists in V.*

Then $\mathcal{F}$ can be realized as a family of small cuts.

We see that $\mathcal{F}$ is a strongly pliable family (see Section 2.3.1). Sets in $\mathcal{F}_1$ and sets in $\mathcal{F}_2$ do not cross each other, and sets in $\mathcal{F}_1$ and $\mathcal{F}_2$ that cross have their corner cuts accounted for in the definition. Let's examine some other properties of $\mathcal{F}$ to prove this Theorem.

Lemma 14. *Let $\mathcal{F}, \mathcal{F}_1, \mathcal{F}_2$ be as described in Theorem 13. Suppose $D \in \mathcal{F}_2$ crosses $S_1, ..., S_k \in \mathcal{F}_1, k \geq 1$. Then one can reorder/rename the sets such that S_i is the child of S_{i+1} for all $i \in [i-1]$.*

Proof. **Case 1:** Assume for contradiction that D crosses S_1 and S_2 that are disjoint. Then we require that $D \cup S_1 \in \mathcal{F}_1$. However, since S_1 and S_2 are disjoint and D crosses S_2, this results in $D \cup S_1$ crossing S_2, which contradicts the fact that $\mathcal{F}_1$ is a laminar family.

Case 2: Suppose that D crosses S_1 and S_3, and there exists S_2 such that $S_1 \subset S_2 \subset S_3$. Then D crosses S_2, since all four corner cuts associated with them are non-empty.

1. $D \cap S_2$: there exists an element in $D \cap S_1 \subseteq D \cap S_2$.
2. $D - S_2$: there exists an element in $D - S_3 \subseteq D - S_2$.
3. $S_2 - D$: there exists an element in $S_1 - D \subseteq S_2 - D$.
4. $V - (S_2 \cup D)$: there exists an element in $V - (S_3 \cup D) \subseteq V - (S_2 \cup D)$.

□

Lemma 15. *Let $\mathcal{F}, \mathcal{F}_1, \mathcal{F}_2$ be as described in Theorem 13. Suppose $D \in \mathcal{F}_2$ crosses $S_1, ..., S_k \in \mathcal{F}_1, k \geq 1$ such that S_i is the child of S_{i+1} for all $i \in [i-1]$ as shown in Lemma 14. Then:*

1. *S_1 either has only one child, or all children but one are contained within D.*
2. *The parent of S_k either has only one child (S_k), or all children other than S_k are contained within D.*
3. *For all $i \in [i-1]$, either S_i is the only child of S_{i+1}, or all children but S_i are contained within D.*
4. *For all i, the key node of S_i is in D.*

Proof.

1. By definition, $S_1 - D \in \mathcal{F}_1$. We claim that is a child of S_1. If not, there is another set $S' \in \mathcal{F}_1$ such that $S_1 - D \subset S' \subset S_1$. However, this would mean that D crosses S', which would contradict the order established for the sets that cross D.

   Thus $S_1 - D$ is the child of S_1. If S_1 has any other children, by laminarity, they must be within $S_1 \cap D \subset D$.

2. By definition, $S_k \cup D \in \mathcal{F}_1$. We claim that this is the parent of S_k. If not, there is another set $S' \in \mathcal{F}_1$ such that $S_k \subset S' \subset S_k \cup D$. However, this would mean that D crosses S', which would contradict the order established for the sets that cross D.

   Thus $S_k \cup D$ is the parent of S_k. If $S_k \cup D$ has any other children, by laminarity, they must be within $D - S_k \subset D$.

3. Assume for contradiction that S_{i+1} has another child S' that is disjoint from both D and S_i. By definition, $S_{i+1} - D \in \mathcal{F}_1$. We claim that this set crosses S_i, contradicting laminarity. Note that all four elements required for crossing are present.

- $(S_{i+1} - D) - S_i$: this includes S'.
- $S_i - (S_{i+1} - D)$: since S_i crosses D, there is an element in $S_i \cap D \subseteq S_i - (S_{i+1} - D)$.
- $V - ((S_{i+1} - D) \cup S_i)$: Since D crosses S_{i+1}, there is an element in $D - S_{i+1} \subseteq V - ((S_{i+1} - D) \cup S_i)$
- $(S_{i+1} - D) \cap S_i \equiv S_i - D \neq \emptyset$ since S_i crosses D.

4. If the key node of S_i is not in $S_i \cap D$, then it is in $S_i - D$, which is in $\mathcal{F}_1$ by construction. Thus, the key node of S_i is part of a child of S_i, which contradicts the definition of key nodes.

□

Lemma 16. *Let $\mathcal{F}, \mathcal{F}_1, \mathcal{F}_2$ be as described in Theorem 13, and let E be edge set described in Lemma 10 with respect to $\mathcal{F}_1$. Let D be a set in $\mathcal{F}_2$. Then $\delta_E(D) = 2$.*

Proof. Since E forms a spanning tree, all non-trivial cuts have at least one edge. Recall that each set in $\mathcal{F}_1$ has exactly one edge of the tree in its cut. Consider a set $S \in \mathcal{F}_1$. Let u be the key node of S and v be the key node of its parent (call it S'). We'll consider all four cases.

- $u, v \notin D$. Then $uv \notin \delta(D)$.
- $u, v \in D$. Then $uv \notin \delta(D)$.
- $u \in D, v \notin D$. D contains the key node of S, but not the key node of S'. This means that D does not cross S' by Lemma 15 Property 4. We assume that D crosses at least one set in $\mathcal{F}_1$ (since otherwise we could move it to $\mathcal{F}_1$, which is assumed to be maximal). We cannot have $S \subset D$ such that D crosses some $S'' \subseteq S' - S$. By definition, $S \subset S'' \cup D \subset S'$. This contradicts the fact that S' is the parent of S.

  There are two valid cases.

  - D crosses S. Then S is the largest set in the chain of sets crossed by D.
  - $D \subset S$, and D crosses some other $S'' \subseteq S$. By definition $S'' \cup D \in \mathcal{F}_1$, but since $u \in S'' \cup D$, then $S'' \cup D = S$, otherwise we would contradict the definition of key nodes. Then S'' is the largest set in the chain of sets crossed by D.

- $u \notin D, v \in D$. D contains the key node of S', but not the key node of S. This means that D does not cross S by Lemma 15 Property 4. We assume that D crosses at least one set in $\mathcal{F}_1$. We cannot have $S' \subset D$ since $u \notin D$. It is also not possible that $D \subset S'$ and crosses some $S'' \subseteq S' - S$. By definition $S'' \cup D \in \mathcal{F}_1$, but since $v \in S'' \cup D$, then $S'' \cup D = S'$, otherwise we would contradict the definition of key nodes. However, this leads to another contradiction, since $u \in S'$, but $u \notin D, S''$.

  Only one valid case remains: D crosses S'. Then $S' - D \in \mathcal{F}_1$. Since D does not cross S and $u \notin D$, $S \subseteq S' - D$, which means that $S = S' - D$ since otherwise this would contradict S' being the parent of S. This also means that there are no children of S' outside of D other than S. Thus, S' is the smallest set in the chain of sets crossed by D.

There are only two situations in the analysis above that lead to an edge being part of $\delta_E(D)$: when examining the largest set or the smallest set crossed by D. By Lemma 14, D can only have one largest and smallest set it crosses, so these cases only happens once each. Thus, $\delta_E(D)$ has two edges incident to it. □

Now let's modify E such that $\mathcal{F}_1 \cup \mathcal{F}_2$ is the family of small cuts. Let E_1 be the set of edges described in Lemma 10 with respect to $\mathcal{F}_1$. Let E_2 be an alternate set of edges defined as follows: each edge in E_2 has unit capacity and corresponds to one set in $\mathcal{F}_2$. Let $D \in \mathcal{F}_2$. Lemma 16 proved that $|\delta_{E_1}(D)| = 2$; let $\delta_{E_1}(D) = \{d_1v_1, d_2v_2\}$ where $d_1, d_2 \in D$. The E_2 edge corresponding to D is v_1v_2.

Another way to interpret the E_2 edge corresponding to $D \in \mathcal{F}$ is as follows: let S' be the smallest set that D crosses and let S'' be the largest set that D crosses (these can be the same set if D only crosses one set). The endpoints of the E_2 edge associated with D are the key node of $S' - D$ and the key node of the parent of $S'' \cup D$.

The edges of E_1 are given capacity as follows.

- If both endpoints are contained within the same set in $\mathcal{F}_2$, its capacity is 3.
- If its endpoints are contained within two different sets in $\mathcal{F}_2$, its capacity is 1.
- If one endpoint is contained within a set in $\mathcal{F}_2$ and the other is not contained within any set in $\mathcal{F}_2$, its capacity is 2.
- If both endpoints are not in any set in $\mathcal{F}_2$, its capacity is 4.

Let $E = E_1 \cup E_2$. An example is visualized in Figure 2.3.

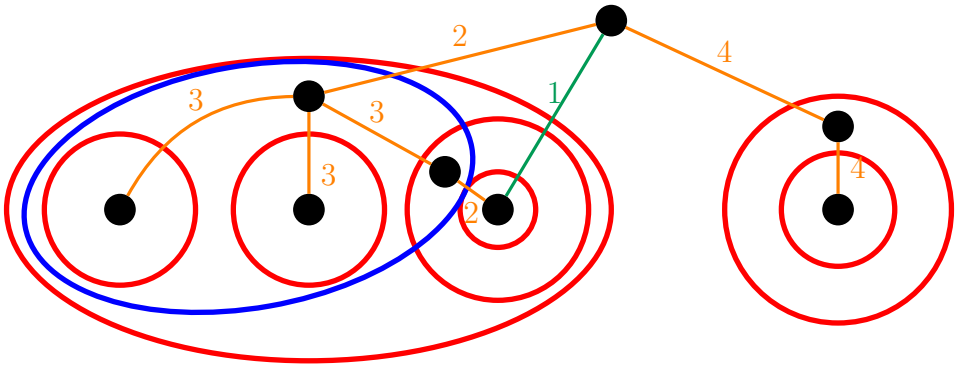


Figure 2.3: A set family that satisfies the properties of Theorem 13, and its capacitated edge set. The red sets are $\mathcal{F}_1$ and the blue set is $\mathcal{F}_2$. The orange edges are E_1 and the green edge is E_2. Note that the non-trivial cuts with total capacity less than 5 are induced by these sets and no other sets.

Proposition 17. *The non-trivial sets with cut-capacity less than 5 with respect to E (described above) are exactly the sets in $\mathcal{F}$, as defined in Theorem 13.*

Claim 18. *Sets in $\mathcal{F}_1$ have capacity less than 5 with respect to E.*

Proof. Let S be any set in $\mathcal{F}_1$. We know that there is one edge in $\delta_{E_1}(S)$ by construction of the tree, with capacity 4 or less. If this is the only edge in $\delta_E(S)$, we are done.

Let's consider all four possible capacities for the key edge of S.

1. If the key edge is contained within some $D \in \mathcal{F}_2$, its capacity is 3. The E_2 edge associated with D can also be incident to $\delta(S)$, adding 1 to the total capacity.

2. If the endpoints of the key edge are in two different sets $D, D' \in \mathcal{F}_2$, its capacity is 1. The E_2 edges associated with D, D' can also be incident to $\delta(S)$, adding 2 to the total capacity.

3. If one endpoint of the key edge is contained within a set $D \in \mathcal{F}_2$ and the other is not, its capacity is 2. The E_2 edge associated with D can also be incident to $\delta(S)$, adding 1 to the total capacity.

4. If neither of the key edge's endpoints are in a set in $\mathcal{F}_2$, its capacity is 4. It is not possible for there to be an E_2 edge in this cut.

S cannot interact with any other sets in $\mathcal{F}_2$ (by Lemma 15 Property 4 and the fact that $\mathcal{F}_2$ sets are disjoint). Thus, $\delta_E(S)$ has capacity ≤ 4. □

Claim 19. *Sets in $\mathcal{F}_2$ have capacity less than 5 with respect to E.*

Proof. Let D be any set in $\mathcal{F}_2$. As described in Lemma 16, $\delta_{E_1}(D)$ has two edges. By construction, these edges have capacity 2 or 1.

If an edge in $\delta_{E_1}(D)$ has capacity 1, then there is another set in $\mathcal{F}_2$ on the other side of the edge. This set's E_2 edge covers $\delta(S)$, adding 1 to the capacity.

If an edge in $\delta_{E_1}(D)$ has capacity 2, then there is no other set in $\mathcal{F}_2$ on the other side, so there is no E_2 edge on this side.

By construction, no other E_2 edges can be in $\delta(D)$. Thus, $\delta_E(D)$ has capacity ≤ 4. □

Claim 20. *Let $\delta(R)$ be a non-trivial cut with capacity less than 5 with respect to E. Then $R \in \mathcal{F}_1 \cup \mathcal{F}_2$.*

Proof. By Lemma 10, cuts with one edge in E_1 are always induced by sets in $\mathcal{F}_1$. Thus, if $|\delta_{E_1}(R)| = 1$, we are done. We now examine small cuts such that $|\delta_{E_1}(R)| \geq 2$. Immediately, $\delta(R)$ cannot have an edge of capacity 4.

Case 1: $\delta_{E_1}(R)$ includes an edge uv of capacity 3. This means that the second edge $u'v'$ has capacity 1, and that $\delta_{E_2}(R) = \emptyset$. By construction, u and v are in a set $D \in \mathcal{F}_2$.

Remark. *Edges of capacity 1 and 2 are always part of a cycle. Consider some $D' \in \mathcal{F}_2$ crossing sets $S_1, ..., S_k \in \mathcal{F}_1$ such that each set in the sequence is the child of the next. Then there is a path in E_1 from the key node of $S_1 - D'$ to the key node of the parent of $S_k \cup D'$, and these endpoints are also the nodes joined by the E_2 edge corresponding to D'. The E_1 edges with capacity 1 or 2 are the edges at the beginning and end of this path.*

Fact 1. *Let C_1 be a group of edges that form a cycle and C_2 be a group of edges that form a cut. Then $|C_1 \cap C_2|$ is an even number (possibly 0).*

By this fact, uv and $u'v'$ are part of the same cycle. This means that they are part of the path described in the Remark with respect to D.

Let uv be defined such that u is the child of v in the tree and $u'v'$ be defined such that $u' \in D, v' \notin D$. By symmetry of cuts, let $u \in R, v \notin R$. Consider any node below u in D (by "below", we mean with respect to the rooted tree). There is a path of capacity-3 edges from u to that node. To ensure that there isn't a second capacity-3 edge in $\delta(R)$, all

of these nodes must be part of R as well. By opposite justification using v, all other nodes in D are not in R.

Subcase 1: $u'v'$ is below u. This means $u' \in R$, which gives that $v' \notin R$. For any node below v', there is a path of edges. To ensure that there isn't a third edge in $\delta(R)$, all of these nodes are not in R as well. However, since the capacity of $u'v'$ is 1, there exists another $\mathcal{F}_2$ set below u' containing v'. This set's E_2 edge has u' as one of its endpoints. This creates a contradiction because this edge would be in $\delta_{E_2}(R)$.

Subcase 2: $u'v'$ is above v. This means that $u' \notin R, v' \in R$. For any node above v', there is a path of edges. To ensure that there isn't a third edge in $\delta(R)$, all of these nodes are in R as well. However, since the capacity of $u'v'$ is 1, there exists another $\mathcal{F}_2$ set above u' containing v'. This set's E_2 edge has u' as one of its endpoints. This creates a contradiction because this edge would be in $\delta_{E_2}(R)$.

Both subcases have ended in contradiction, so it is impossible for a cut to have $u(\delta(R)) < 5$, $|\delta_{E_1}(R)| \geq 2$, and an edge with capacity 3. This case is visualized in Figure 2.4.

Case 2: $\delta_{E_1}(R)$ does not include an edge of capacity 3. This means that for each set in $\mathcal{F}_2$, R either contains all of the set's nodes, or none of the set's nodes. For each set in $\mathcal{F}_2$, contract all of its nodes to a single pseudonode. Edges that entered the set are now incident to this pseudonode, and edges within the set are no longer present. When we consider a cut-set in this contracted graph, if a pseudonode is part of the set, all of the nodes within it are part of the cut-set in the original graph; and vice-versa.

We now have a graph with the following properties:

- The graph has two types of nodes: **white** ($\mathcal{F}_2$ pseudonodes) and **black** (all other nodes).
- E_1 is a spanning tree. Edge capacities are 4 when both endpoints are black, 1 when both endpoints are white, and 2 when endpoints are different colours.
- White nodes always have degree 2 with respect to E_1. For each white node, there is a unit-capacity edge in E_2 joining its E_1-neighbours. Note that this forms a 3-edge cycle in $E = E_1 \cup E_2$.

Our goal to complete this proof is to show that if a cut with $|\delta_{E_1}(R)| \geq 2$ has total capacity less than 5, then R consists of one white node.

Subcase 1: $\delta_{E_1}(R)$ includes an edge of capacity 2. Let b, w be the endpoints of this edge such that b is black and w is white. By symmetry, construct R such that $w \in R, b \notin R$.

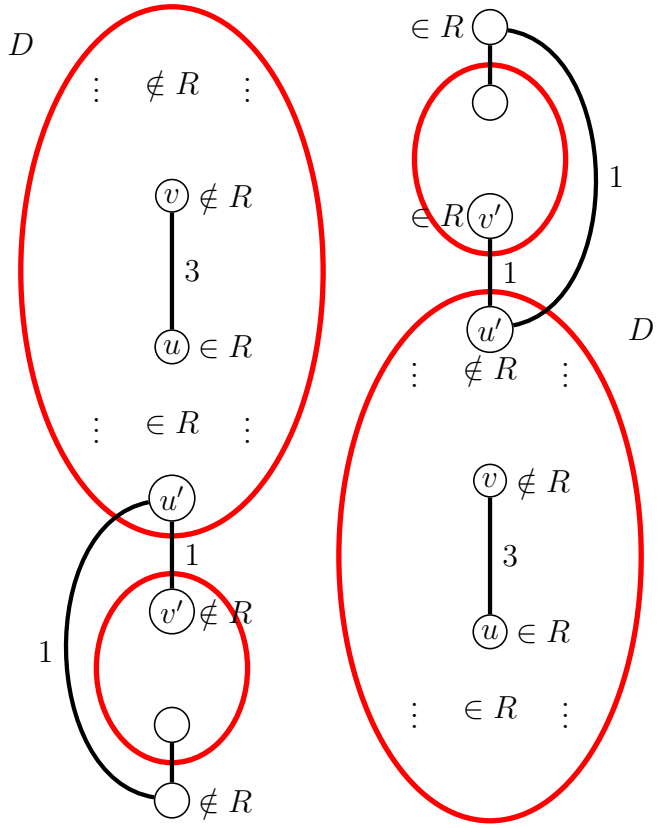


Figure 2.4: Visualization of the subcases of Claim 20, Case 1

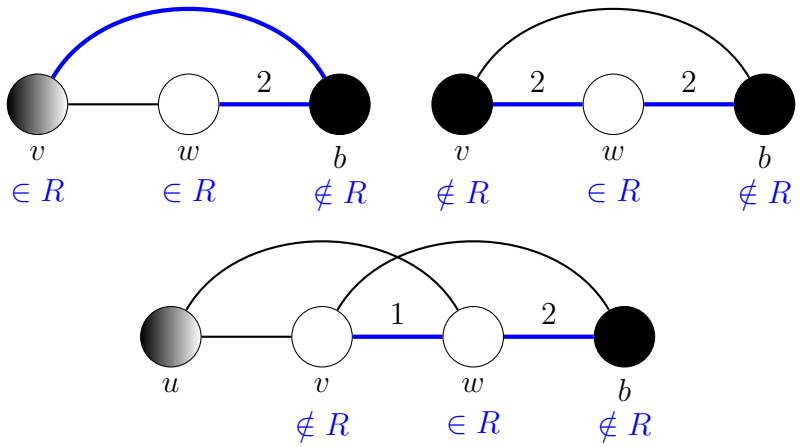


Figure 2.5: Visualization of Claim 20 Case 2, Subcase 1. Nodes that are coloured with a black-white gradient could be either colour. The blue edges are guaranteed to be in $\delta(R)$.

By construction, there exists another node v (could be either colour) such that $vw \in E_1$ and $vb \in E_2$.

By Fact 1, exactly one of vb or vw is also in $\delta(R)$.

1. If $vb \in \delta(R)$, note that $vb \in E_2$, but we assume that $|\delta(R)_{E_1}| \geq 2$. This means that there is at least one other edge in $\delta(R)$. This edge is part of another cycle, so we require one additional edge to enforce an even intersection. Thus, $\delta(R)$ has an edge of capacity 2 and at least three other edges of capacity ≥ 1, contradicting it being a small cut.

2. If $vw \in R$, then note that $v, b \notin R; w \in R$. If v is black, vb has capacity 2 and $u(\delta(R)) \geq 4$ from these edges alone. No other edges can be in $\delta(R)$, but since E_1 forms a tree, all nodes are reachable from either v or b. Thus, all other nodes are not in R and $R = \{w\}$ as desired.

   If v is white, then v has a second neighbour u such that u, v, w form a cycle in w. We require that uw or uv is also in $\delta(R)$, which results in the capacity being at least 4. If $uw \in \delta(R)$, then $u \notin R$, and by the same reasoning as the previous paragraph, no other nodes can be in R. If $uv \in \delta(R)$, then u is white, and forms another cycle that forces us to add one more edge to $\delta(R)$, contradicting the cut's smallness.

Subcase 2: We have considered cuts with edges of capacity 4, 3, and 2; all that remains is cuts where all edges have capacity 1.

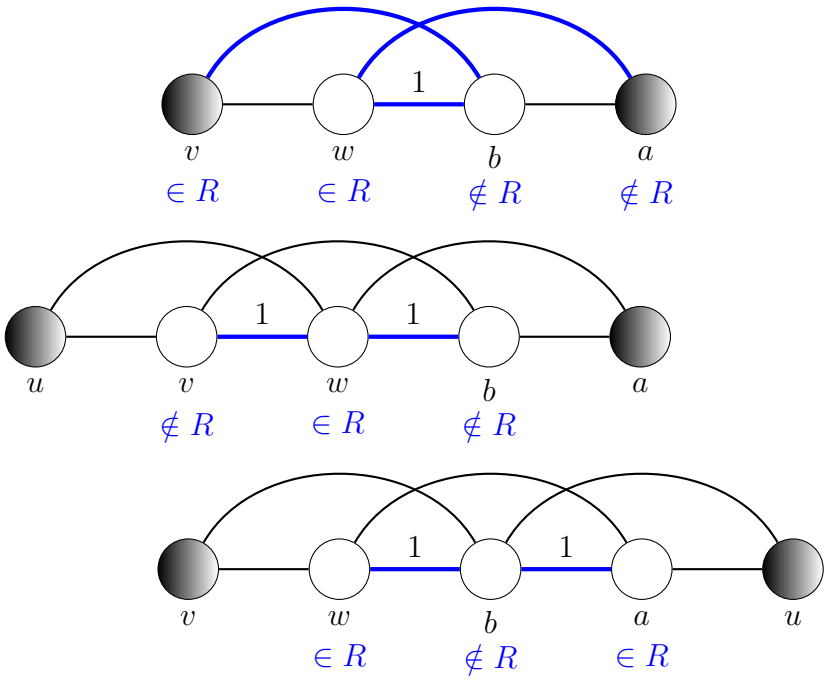


Figure 2.6: Visualization of Claim 20 Case 2, Subcase 2. Nodes that are coloured with a black-white gradient could be either colour. The blue edges are guaranteed to be in $\delta(R)$.

Choose some edge in $\delta_{E_1}(R)$ and let w, b be its endpoints; both are white. Thus, there exists v, a such that $wv, ba \in E_1$, $vb, wa \in E_2$. By symmetry, construct R such that $w \in R, b \notin R$. Note that this part of the graph contains two 3-cycles, v, w, b and w, b, a, both of which contain wb.

By Fact 1, one of wv, vb and one of ba, wa are in $\delta(R)$.

1. If $vb, wa \in \delta(R)$, we can follow the same logic as the Subcase 1's first point to reach a contradiction.

2. Otherwise, at least one of wv or ba is in $\delta(R)$. We follow the same logic as Subcase 2's second point. If $wv \in \delta(R)$, the previous argument follows identically. If $ba \in \delta(R)$, we change the names of v, w, b to a, b, w, respectively, and replace R with $V - R$.

□

2.3 Strongly Pliable Families

2.3.1 Overview

Due to the lack of uncrossability, some results such as Jain's iterative rounding algorithm are not available when analyzing Cover Small Cuts. We explore a property that families of small cuts satisfy, the structure of these set families, and their approximability in cut covering problems.

Note the following about families of small cuts:

Fact 2. *Let $\mathcal{C}$ be the family of small cuts with respect to some set of capacitated edges E, u, λ. Given two $A, B \in \mathcal{C}$ such that A and B cross, then at least one of $A - B$ or $B - A$ is also in $\mathcal{C}$, and at least one of $A \cup B$ or $A \cap B$ is also in $\mathcal{C}$.*

Proof. The edge capacity of cuts satisfies submodularity. Let u define a function such that $u(A)$ is the total edge capacity of the cut $\delta(A)$. This means that

$u(A) + u(B) \geq u(A - B) + u(B - A)$

A cut $\delta(A)$ is small when $u(A) < \lambda$. Assume for contradiction that A and B are small, but $A - B$ and $B - A$ are not small. Then

$u(A) + u(B) < \lambda + \lambda$

$u(A - B) + u(B - A) \geq \lambda + \lambda$

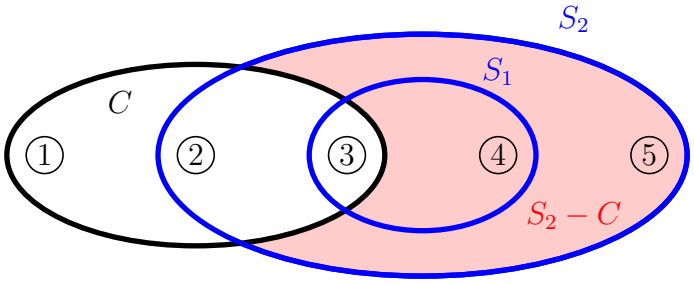
This contradicts the submodularity property. The proof of the second statement follows identically by replacing $A - B, B - A$ with $A \cup B, A \cap B$. □

This is a general property. Recall that this property is called **structural submodularity** in prior work on abstract separation systems [11], [12, Chapter 12],[13], [14],[15],[16].

Definition 21. *A set family $\mathcal{F}$ is **strongly pliable** if for any $A, B \in \mathcal{F}$ that cross, then **both** of the following hold:*

- *At least one of $A \cup B$ or $A \cap B$ is in $\mathcal{F}$.*
- *At least one of $A - B$ or $B - A$ is in $\mathcal{F}$.*

Figure 2.7: Visualization of Property γ for a strongly pliable family



Note that strong pliability immediately implies pliability. If two sets A, B cross, then the property assures that two corner cuts are also part of the family. If A and B do not cross, then either they are disjoint and $A - B = A, B - A = B$, or they are nested ($A \subseteq B$ WLOG) and $A \cap B = A, A \cup B = B$.

We examine some other properties.

Definition 22. ***Property*** γ *[4]: Consider* $C, S_1, S_2 \in \mathcal{F}$ *such that* $S_1 \subset S_2, C$ *is inclusion-wise minimal, and* C *crosses both* S_1, S_2*. Then* $S_2 - (S_1 \cup C)$ *is either empty or in* $\mathcal{F}$*.*

Lemma 21. *Strongly pliable families satisfy property* γ*.*

Proof. Let $\mathcal{F}$ be a strongly pliable family and let $C, S_1, S_2 \in \mathcal{F}$ be as described in the definition of property γ. It is sufficient to suppose that $S_2 - (S_1 \cup C)$ is non-empty, and prove that it is in $\mathcal{F}$.

First, note that since C and S_2 cross, at least one of $C - S_2$ and $S_2 - C$ must be in $\mathcal{F}$. However, C is inclusion-wise minimal, so $C - S_2 \notin \mathcal{F}$, leaving $S_2 - C \in \mathcal{F}$.

Since C and S_1 cross, then $S_1 - C$ and $S_1 \cap C$ are non-empty. Since $S_1 \subseteq S_2, S_1 - C \subseteq S_2 - C$, and $S_1 \cap C \not\subseteq S_2 - C$, then S_1 and $S_2 - C$ cross. By strong pliability, one of $(S_2 - C) - S_1$ or $S_1 - (S_2 - C)$ is in $\mathcal{F}$. However, $S_1 - (S_2 - C) \equiv C \cap S_1 \subset C$, which cannot be in $\mathcal{F}$. Thus, $(S_2 - C) - S_1 \equiv S_2 - (S_1 \cup C) \in \mathcal{F}$ as required. Figure 2.7 visualizes this proof. □

Lemma 22. *Any two inclusion-wise minimal sets in a strongly pliable family are disjoint.*

Proof. Let $\mathcal{F}$ be a strongly pliable family, and $C_1, C_2 \in \mathcal{F}$ be two inclusion-wise minimal sets. They can't be nested within each other, as that violates the inclusion-wise minimal property. Then, suppose C_1 and C_2 cross. By strong pliability, at least one of $C_1 - C_2 \subset C_1$

or $C_2 - C_1 \subset C_1$ must be included in $\mathcal{F}$, but both of these violate the inclusion-wise minimal property. The only case that remains is that C_1 and C_2 are disjoint, as required. □

Definition 23. *A set family $\mathcal{F}$ satisfies the **sparse crossing property** [3] if any set $S \in \mathcal{F}$ crosses at most one inclusion-wise minimal set of $\mathcal{F}$.*

Lemma 23. *Strongly pliable families satisfy the sparse crossing property.*

Proof. Let $\mathcal{F}$ be a strongly pliable family. Assume for contradiction that a set $S \in \mathcal{F}$ crosses two inclusion-wise minimal sets $C_1, C_2 \in \mathcal{F}$.

By strong pliability, at least one of $S \cup C_1, S \cap C_1$ is in $\mathcal{F}$. However, $S \cap C_1 \subset C$, so including it in would violate inclusion-wise minimality. Thus, $S \cup C_1 \in \mathcal{F}$, and by a similar argument, $S \cup C_1 \in \mathcal{F}$.

Since C_1 and C_2 are disjoint by Lemma 22, then $S \cup C_1$ and $S \cup C_2$ cross. Thus, by strong pliability, at least one of $(S \cup C_1) - (S \cup C_2)$ or $(S \cup C_2) - (S \cup C_1)$ is in $\mathcal{F}$. However $(S \cup C_1) - (S \cup C_2) \equiv C_1 - S \subset C_1$ and $(S \cup C_2) - (S \cup C_1) \equiv C_2 - S \subset C_2$, so neither of them can be in $\mathcal{F}$ due to inclusion-wise minimality. We have reached a contradiction, thus S cannot cross two or more inclusion-wise minimal sets, as required. □

2.3.2 Strongly Pliable Non-Cover Small Cuts Example

Section 2.3.1 showed that all families of small cuts are strongly pliable, and that strongly pliable families satisfy several of the same properties previously known for families of small cuts. Are families of small cuts and strongly pliable families equivalent? We confirm that they are **not**.

Theorem 24. *for any integer $k' \geq 2$, it is possible to find a strongly pliable set family that cannot be partitioned into k' uncrossable families, and cannot be realized as a family of small cuts.*

We prove this Theorem throughout this section, beginning with the following construction. Let $k = k' + 1$. Let $\{S_1, ..., S_k\}$ be a group of sets that fully cross each other (that is, for any subfamily $\mathcal{F} \subseteq \{S_1, ..., S_k\}$, there is at least one node in $\bigcap_{S \in \mathcal{S}} S - \bigcup_{S \notin \mathcal{S}} S$). Place exactly one node in each possible "section". Thus, the universe has 2^k nodes and each $S_i, i \in [k]$ has 2^{k-1} nodes. For each $i \in [k]$, let v_i be the vertex in $S_i - (\bigcup_{j \in [k], j \neq i} S_j)$; let u be the vertex in $\bigcap_{i \in [k]} S_i$; and let u' be the vertex in $\overline{\bigcup_{i \in [k]} S_i}$.

Another way to interpret the nodes is in bijection with the family of subsets of $\{1, 2, ..., k\}$ such that a node is in sets $S_{i_1}, ..., S_{i_j}$ and not the other sets if and only if it maps to the subset $\{i_1, ..., i_j\}$. For example, v_i maps to the subset $\{i\}$, u maps to the subset $\{1, 2, ..., k\}$, u' maps to $\emptyset$, and S_i contains the nodes that map to any subset containing i.

We call $S_1, ..., S_k$ the **S-sets** and $v_1, ..., v_k$ the **v-nodes**.

Algorithm 1 constructs a strongly pliable family $\mathcal{F}$ given the defined nodes and S-sets.

Algorithm 1: Strongly Pliable Family with no Cover Small Cuts Realization

---

Initialize: $\mathcal{F} = \mathcal{F}_0 = \{S_1, ..., S_k\}, \ell = 1$.

Begin iteration ℓ, let $\mathcal{F}_\ell = \emptyset$:

1. Examine every pair of sets A, B in $\mathcal{F}$ that cross.
   (a) If $A \cap B \notin \mathcal{F}$, add $A \cap B$ to $\mathcal{F}_\ell$.
   (b) If $A - B, B - A \notin \mathcal{F}$,
      i. If one of $A - B$ or $B - A$ contains v_i and the other contains $v_j, j > i$, add the set difference containing v_j to $\mathcal{F}_\ell$.
      ii. If one of $A - B$ or $B - A$ contains v_i and the other contains no v-nodes, add the set difference containing no v-nodes to $\mathcal{F}_\ell$.
      iii. Otherwise, add one set difference to $\mathcal{F}_\ell$ arbitrarily.
2. Add all sets in $\mathcal{F}_\ell$ to $\mathcal{F}$.
3. If $\mathcal{F}_\ell = \emptyset$, all pairs of crossing sets have the necessary corner cuts in the family. Return $\mathcal{F}$, a strongly pliable family.
4. Otherwise, set $\ell \to \ell + 1$ and proceed to the next iteration.

Figure 2.8 visualizes the "iterations" that make up $\mathcal{F}$.

Example 1. *We present an example for $k = 3$. The names of the nodes are displayed in Figure 2.9.*

$\mathcal{F}_0 = \{S_1 = \{1, 4, 6, 7\}, S_2 = \{2, 5, 6, 7\}, S_3 = \{3, 4, 5, 7\}\}$

$\mathcal{F}_1 = \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 7\}, \{5, 7\}, \{6, 7\}$

$\mathcal{F}_2 = \{3\}, \{4\}, \{5\}, \{7\}, \{6\}, \{3, 4, 7\}$

$\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1 \cup \mathcal{F}_2$ *is strongly pliable.*

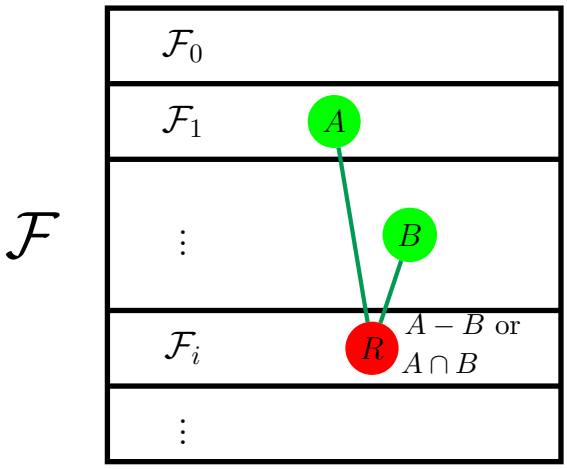


Figure 2.8: A visualization of the partitioning of $\mathcal{F}$, the strongly pliable family returned by Algorithm 1, into iterations. In each iteration, crossing pairs from the previous iterations are examined, and corner cuts are added to $\mathcal{F}$ if they are not already present.

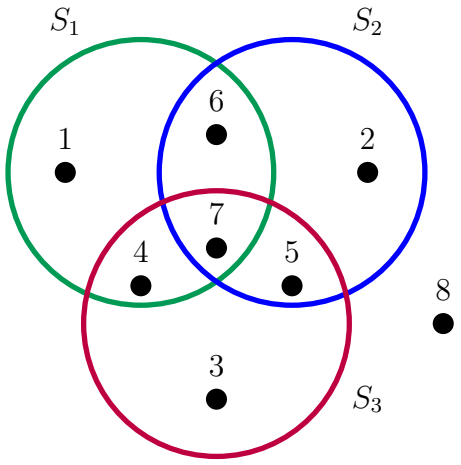


Figure 2.9: Initial setup for Algorithm 1 when $k = 3$.

Lemma 25. *Let $\mathcal{F}$ be the output of Algorithm 1. For each set $R \in \mathcal{F}$, there is at least one $i \in [k]$ such that $R \subseteq S_i$.*

Proof. By induction on the index of the iteration set containing R. As a base case, if $R \in \mathcal{F}_0$, then $R = S_i$ for some i and is a subset of itself. Then, assume all sets added in previous iterations are subsets of at least one S-set. By construction of $\mathcal{F}$ with Algorithm 1, $R \in \mathcal{F}_j$ is either equal to $A - B$ or $A \cap B$ for some $A, B \in \mathcal{F}$ from an iteration earlier than j. By induction, $A \subseteq S_i$ for some $i \in [k]$. Then $A \cap B, A - B \subseteq S_i$. □

Corollary 26. *If a set $R \in \mathcal{F}$ contains v_i, then $R \subseteq S_i$ and contains no other v-nodes.*

Corollary 27. *Algorithm 1 terminates.*

Proof. By the lemma, every set Algorithm 1 adds to $\mathcal{F}$ is a subset of one of the S-sets. With a finite number of nodes, there are a finite number of these subsets, and the family consisting of all subsets of the S-sets is strongly pliable. □

To prove that $\mathcal{F}$ cannot be realized as a family of small cuts, we will identify k sets that are in the family and k sets that are **not** in the family. For any capacitated edge set, we will show that the total summed capacity of edges covering the cuts induced by the sets in $\mathcal{F}$ is at least the total summed capacity of edges crossing the cuts induced by the sets not in $\mathcal{F}$. Thus, it is impossible to create a capacitated edge set such that the former sets are small and the latter sets are not. Before we present the sets that are not in $\mathcal{F}$, some more results will be useful.

Fact 3. *Two sets $A, B \in \mathcal{F}$, such that $v_i \in A, v_j \in B, i \neq j$, cross if $A \cap B \neq \emptyset$.*

Proof. A and B cross if $A \cap B, V - (A \cup B), A - B, B - A$ are all non-empty. We assume that $A \cap B \neq \emptyset$. Since $v_i \in A, v_j \in B$, then $v_i \notin B, v_j \notin A$ by Corollary 26, so $A - B, B - A \neq \emptyset$. Finally, u' is outside all S-sets, so it is outside all sets in $\mathcal{F}$; that is, $u' \in V - (A \cup B)$. □

Claim 28. *Let $R \in \mathcal{F}_\ell$ be a set such that $v_i \in R, R \neq S_i$ for some $i \in [k]$. Then there exist crossing sets $R' \in \bigcup_{h=0}^{\ell} \mathcal{F}_h, R'' \in \bigcup_{h=0}^{\ell-1} \mathcal{F}_h$ such that $R = R' - R''$, and $v_i \in R', v_j \in R''$ with $j < i$.*

Note that, whenever we apply this claim, by Corollary 26, we have $R, R' \subseteq S_i, R'' \subseteq S_j$.

Proof. By induction on the index of the iteration set containing R.

Base Case: We skip $\mathcal{F}_0$ since the only set containing v_i in $\mathcal{F}_0$ is S_i. Let $R \in \mathcal{F}_1$ such that $v_i \in R$. All sets in $\mathcal{F}_1$ are constructed by Algorithm 1 via $S_j - S_\ell, j > \ell$ or $S_j \cap S_\ell$ for some $S_j, S_\ell \in \mathcal{F}_0$. Since each v-node is unique to its respective S-set, there are no v-nodes in any of the intersections. The only S-set that contains v_i is S_i. Thus, $R = S_i - S_j$ for some $j < i$ as required.

Inductive Hypothesis: Let $R \in \bigcup_{h=1}^{\ell} \mathcal{F}_h$ such that $v_i \in R$. Then there exist crossing sets $R', \in \bigcup_{h=0}^{\ell} \mathcal{F}_h, R'' \in \bigcup_{h=0}^{\ell-1} \mathcal{F}_h$ such that $R = R' - R'', v_i \in R', v_j \in R'', j < i$.

Now consider $R \in \mathcal{F}_{\ell+1}$ such that $v_i \in R$. Since $R \in \mathcal{F}$, Algorithm 1 added it to the family due to the crossing of two sets $A, B \in \bigcup_{h=0}^{\ell} \mathcal{F}_h$. If multiple crossing pairs led to the inclusion of R, choose A, B such that $|A| \geq |B|$ and $|A|$ is as large as possible (*).

Case 1: If $R = A - B$, then $v_i \in A - B$. By our construction rules in Step (b)(i), we chose this difference instead of $B - A$ if $v_j \in B, j < i$. Thus $A = R', B = R''$ and we are done. The case in which $R = B - A$ is identical by switching the names of A and B.

Case 2: Now suppose $R = A \cap B$. First note that A and B are not S-sets. This is because $v_i \in R \Rightarrow v_i \in A, B$; the only S-set containing v_i is S_i; and all other sets in $\mathcal{F}$ that contain v_i are subsets of S_i. A set does not cross itself or its subsets, so $A \neq B \neq S_i$.

Since A contains v_i, does not equal S_i, and is in an earlier iteration set than R, by induction, there exist crossing sets $A' \in \bigcup_{h=0}^{\ell} \mathcal{F}_h, A'' \in \bigcup_{h=0}^{\ell-1} \mathcal{F}_h$ such that $A = A' - A'', v_i \in A', v_j \in A'', j < i$.

We can then write $R = (A' - A'') \cap B$, which is equivalent to $R = (A' \cap B) - A''$.

Subcase 1: If A' crosses B, then $A''' = A' \cap B \in \mathcal{F}$ by the Algorithm's construction, and $A' \cap B \in \bigcup_{h=0}^{\ell+1} \mathcal{F}_h$ since $A', B \in \bigcup_{h=0}^{\ell} \mathcal{F}_h$. Thus $R = A''' - A''$ as required, once we confirm that $A''' = A' \cap B$ and A'' cross. $A' \cap B$ and A'' cross by Fact 3 if $A' \cap B \cap A''$ is non-empty. If this intersection was empty, we would have $A' \cap B = R$; this contradicts our assumption (*) because $|A'| > |A|$.

Subcase 2: If A' does not cross B, since $v_i \in A' \cap B$ and $A - B \subset A'$, then $B \subset A'$. We then argue that B and A'' cross and $B - A'' = R$, completing the proof. Consider a node $v \in B$. If $v \in A \cap B$, since $A = A' - A''$, $v \notin A''$, so $v \in B - A''$. If $v \in B - A$, then since $v \in A'$ also, we have $v \in A''$ since otherwise $v \in A' - A'' = A$, a contradiction. Thus $v \notin B - A''$. This gives exactly $B - A'' = A \cap B = R$, and also means that $B \cap A'' \neq \emptyset$, so the two sets cross by Fact 3.

Figure 2.3.2 visualizes these subcases. □

Lemma 29. *Let $\mathcal{F}$ be the output of Algorithm 1. $\mathcal{F}$ satisfies the following:*

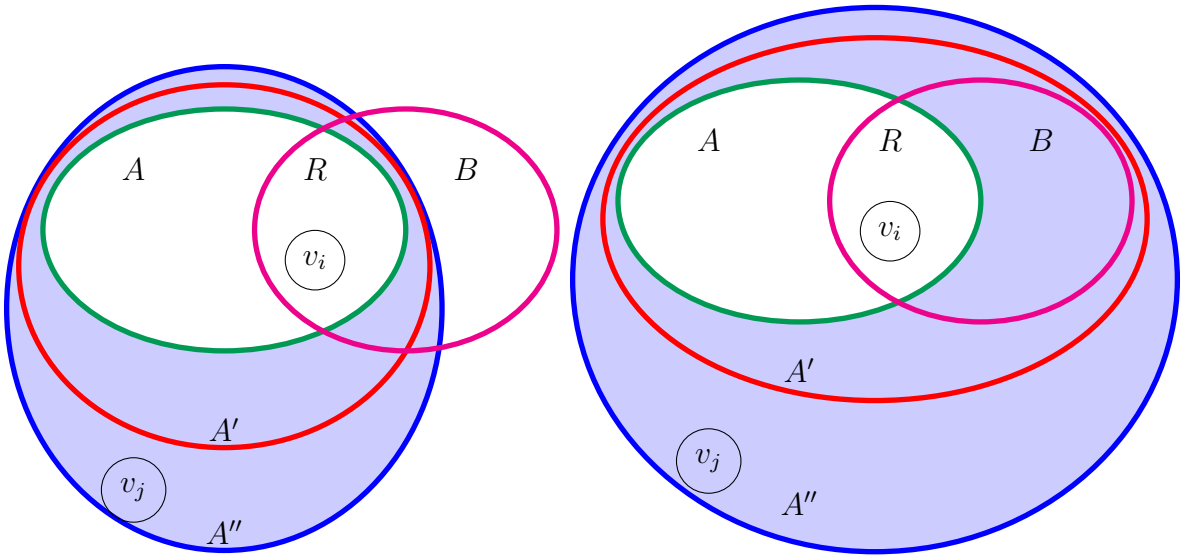


Figure 2.10: Visualization of the subcases of Claim 28, Case 2.

1. *For each* $i \in [k-1]$, $\{v_i\} \notin \mathcal{F}$.

2. $\{v_k, u\} \notin \mathcal{F}$.

Proof. Let R be a set in $\mathcal{F}$ that contains $v_i, i \in [k-1]$. We will prove that $|R| \geq 2^{k-i}$, confirming that $\{v_i\} \notin \mathcal{F}$.

If $R = S_i$, by construction, $|R| = 2^{k-1}$.

If not, suppose $R \in \mathcal{F}_\ell, \ell \geq 1$. By Claim 28, there exist $R_1 \in \mathcal{F}_{\ell_1}, R_2 \in \mathcal{F}_{\ell_2}, \ell_1 \leq \ell, \ell_2 \leq \ell - 1$ such that $v_i \in R_1, v_j \in R_2, j < i$, and $R = R_1 - R_2$.

If $R_1 = S_i$ and $R_2 = S_j$, then $|R| = 2^{k-2}$. This is because the S-sets cross fully, so pairs of them share half of their elements, and $|S_i| = |S_j| = 2^{k-1}$. (Also note that $i \neq 1$ since $j < i$ and all indices are positive.)

Otherwise, we can apply the claim again on R_1 and/or R_2. By the claim, $R_1 = R_3 - R_4$ such that $v_i \in R_3, v_m \in R_4, m < i$; and $R_2 = R_5 - R_6$ such that $v_j \in R_5, v_h \in R_6, h < j$. Then we have $R = (R_3 - R_4) - (R_5 - R_6)$. Also note that $R_3 \in \mathcal{F}_{\ell_3}, R_4 \in \mathcal{F}_{\ell_4}, R_5 \in \mathcal{F}_{\ell_5}, R_6 \in \mathcal{F}_{\ell_6}, \ell_3 \leq \ell_1, \ell_4 \leq \ell_1 - 1, \ell_5 \leq \ell_2, \ell_6 \leq \ell_2 - 1$.

Repeatedly apply the claim on every set in the equation that is not an S-set. Note that when we apply the claim to construct $R^* = R' - R''$, R'' is always from an earlier iteration than R^*, and R' is either from an earlier iteration or the same iteration as R^*.

Additionally, $|R'| > |R^*|$ because $R^* = R' - R'' \Rightarrow R^* \subset R'$. Thus, applying the claim results in sets that are either from earlier iterations, or are from the same iteration but are larger than the initial set. We can apply the claim until all sets in the equation are in $\mathcal{F}_0$; that is, they are S-sets. With enough uses of the claim, we can write R as a series of set differences, nested with brackets, of the S-sets. In particular, the first set in the series is S_i, and all other S-sets have index strictly less than i.

Consider subsets of $\{1, ..., k\}$ containing i and any (possibly empty) combination of elements between $i+1$ and k. There are 2^{k-i} possible subsets. In our graph construction, these subsets map to nodes that are in S_i and possibly in any of $S_{i+1}, ..., S_k$. However, these nodes are **not** in $S_1, ..., S_{i-1}$. Thus, no matter how the series of set differences are arranged, these nodes cannot be removed from S_i by $S_1, ..., S_{i-1}$, so they are present in R. This completes the proof of the first point.

For the second point, suppose for contradiction that $\{v_k, u\} \in \mathcal{F}_\ell$ for some $\ell \geq 1$. By Claim 28, there exist $R_1 \in \mathcal{F}_{\ell_1}, R_2 \in \mathcal{F}_{\ell_2}, \ell_1 \leq \ell, \ell_2 \leq \ell - 1$ such that $\{v_k, u\} = R_1 - R_2, v_k \in R_1 \subseteq S_k, v_j \in R_2 \subseteq S_j, j < k$.

Let w be the node in $S_k \cap S_1 - \bigcup_{\ell=2}^{k-1} S_\ell$; in other words, this node maps to the subset $\{1, k\}$.

Case 1: $w \in R_1$; since $w \notin \{v_k, u\}$, then $w \in R_2$. Note that $R_2 \neq S_j$ since $u \in S_j$ and $u \in R_1 - R_2 \Rightarrow u \notin R_2$. Thus, we can apply the claim again to write $R_2 = R_3 - R_4$ where $v_j, w \in R_3 \subseteq S_j, v_h \in R_4, h < j$. The only S-sets that contain w are S_1 and S_k, but $j < k$ by our choice of R_2, thus $j = 1$. This is a contradiction because $h < j$ and all indices are at least 1.

Case 2: $w \notin R_1$. This means that $R_1 \neq S_k$, so we can apply the claim again to write $R_1 = R_5 - R_6$, and either $w \in R_5, R_6$ or $w \notin R_5$. If the former, we can apply the logic of Case 1 using R_5, R_6 in place of R_1, R_2 to reach a contradiction. Otherwise, we apply the claim on R_5. We repeatedly apply the claim on the "first" set in the difference until we get a set difference such that that the two sets contain w, at which point we derive a contradiction using the logic of Case 1. This will always happen because we can use the claim to construct any set containing v_k as a set difference of two other sets in $\mathcal{F}$, such that the new sets are either from an earlier iteration or have a larger size, unless the set is S_k. Either we find a set containing w along the way, or the first set eventually becomes S_k, which contains w.

□

Lemma 30. *The output of Algorithm 1, $\mathcal{F}$, cannot be realized as a family of small cuts.*

Proof. Assume for contradiction that there exists an edge set E, non-negative capacity assignment u, and threshold λ such that the family of cuts with capacity less than λ is exactly $\mathcal{F}$. We assume that $E = \binom{V}{2}$ (the complete graph of edges), since if an edge is not "present" we can give it capacity 0. For any cut $\delta(S)$, let $u(\delta(S))$ be the total capacity of edges incident to that cut.

We will use the "small cut" sets $S_1, ..., S_k \in \mathcal{F}$ (by construction), and the "large cut" sets $\{v_1\}, ..., \{v_{k-1}\}, \{v_k, u\} \notin \mathcal{F}$ (by Lemma 29). Since the first group of sets form small cuts,

$c' = u(\delta(S_1)) + u(\delta(S_2)) + ... + u(\delta(S_k)) < k * \lambda$

Since the second group of sets form large cuts,

$c'' = u(\delta(\{v_1\})) + u(\delta(\{v_2\}) + ... + u(\delta(\{v_{k-1}\})) + u(\delta(\{v_k, u\})) \geq k * \lambda$

We will show that $c' \geq c''$, which is an immediate contradiction. To do this, we count the number of times an edge can occur in $\delta(\{v_1\}), \delta(\{v_2\}), ..., \delta(\{v_{k-1}\}), \delta(\{v_k, u\})$. If edge e appears α times, it contributes $\alpha * u_e$ to c''. We will then show that e occurs at least as many times in $\delta(S_1), \delta(S_2), ..., \delta(S_k)$, thus also contributing at least $\alpha * u_e$ to c'. Since all capacities are non-negative, we will be done.

Consider an arbitrary edge that appears in $\delta(\{v_1\}), \delta(\{v_2\}), ..., \delta(\{v_{k-1}\}), \delta(\{v_k, u\})$.

Case 1: The edge takes the form $v_i w$, where $i \in [k]$ and $w \in V - \{v_1, v_2, ..., v_k, u\}$. This edge appears once in the large cuts, specifically in $\delta(\{v_i\})$. The edge also appears at least once in the small cuts. If $w \notin S_i$, then $v_i w \in \delta(S_i)$. Otherwise, since the only vertices in exactly one S-set are the v-nodes, w is in some other S_j, which does not contain v_i. Thus $v_i w \in \delta(S_j)$.

Case 2: The edge takes the form $v_i v_j$, where $i, j \in [k], i \neq j$. This edge appears twice in the large cuts, specifically in $\delta(\{v_i\})$ and $\delta(\{v_j\})$. The edge also appears at least twice in the small cuts. Since each v-node is unique to its respective S-set, then $v_j \notin S_i, v_i \notin S_j$, giving $v_i v_j \in \delta(S_i), \delta(S_j)$.

Case 3: The edge takes the form $v_i u$, where $i \in [k-1]$. This edge appears twice in the large cuts, specifically in $\delta(\{v_i\})$ and $\delta(\{v_k, u\})$. This edge also appears at least twice in the small cuts. Note that $v_i \in S_i$ and not in any other S-set, and u is in all S-sets. Thus, $v_i u \in \delta(S_j)$ for any $j \in [k], j \neq i$.

Case 4: The edge takes the form uw, where $w \in V - \{v_1, ..., v_k, u\}$. This edge appears once in the large cuts, specifically in $\delta(\{v_k, u\})$. This edge also appears at least once in the small cuts. Recall that u is the only vertex that is in all S-sets, so there exists j such that $w \notin S_j$. Then $uw \in \delta(S_j)$. □

Lemma 31. *The output of Algorithm 1, $\mathcal{F}$, also satisfies the following: for each pair $S_i, S_j, j < i$; $S_i \cup S_j \notin \mathcal{F}$ and $S_j - S_i \notin \mathcal{F}$.*

Proof. First, by Lemma 25, every set in $\mathcal{F}$ is a subset of one of the S-sets. This immediately proves that $S_i \cup S_j \notin \mathcal{F}$ for any i, j.

Second, assume for contradiction that $S_j - S_i \in \mathcal{F}_\ell$ for some $\ell \geq 1$. By Claim 28, there exist $R_1 \in \mathcal{F}_{\ell_1}, R_2 \in \mathcal{F}_{\ell_2}, \ell_1 \leq \ell, \ell_2 \leq \ell - 1$ such that $S_j - S_i = R_1 - R_2, v_j \in R_1 \subseteq S_j, v_h \in R_2 \subseteq S_h, h < j$.

Let w be the node in $S_j \cap S_i - \bigcup_{\ell \in [k], \ell \neq i,j} S_\ell$; in other words, this node maps to the subset $\{i, j\}$. This node is in S_i and S_j and no other S-sets, and is not in $S_j - S_i$.

Case 1: $w \in R_1$; then $w \in R_2$ also. However, $R_2 \subseteq S_h, h < j < i$, so $w \notin S_h \Rightarrow w \notin R_2$, a contradiction.

Case 2: $w \notin R_1$. This means that $R_1 \neq S_j$, so we can apply the claim again to write $R_1 = R_3 - R_4$, and either $w \in R_3, R_4$ or $w \notin R_3$. If the former, we can apply the logic of Case 1 to reach a contradiction. Otherwise, we apply the claim on R_3. We repeatedly apply the claim on the "first" set in the difference until we get a set difference such that the two sets contain w, at which point we derive a contradiction using the logic of Case 1. This will always happen because we can use the claim to construct any set containing v_j as a set difference of two other sets in $\mathcal{F}$ such that the new sets are either from an earlier iteration or have a larger size, unless the set is S_j. Either we find a set containing w along the way, or the first set eventually becomes S_j, which contains w. □

Corollary 32. *It is impossible to partition $\mathcal{F}$ into $k' = k - 1$ uncrossable families.*

Proof. If such a partition was possible, by the pigeonhole principle, at least two of $S_1, ..., S_k$ would be in the same part of the partition. However, for each of these crossing pairs, only their intersection and one set difference are in the whole set family. This violates the definition of uncrossability. □

Lemma 30 and Corollary 32 together prove Theorem 24.

2.3.3 5-Approximation for Cut Covering Problems on Strongly Pliable Families

Bansal [3] analyzed the approximability of certain types of pliable cut covering problems using the WGMV primal-dual approximation algorithm. He introduced the following definition and result.

Definition 24. *The **crossing density** of a set family $\mathcal{F}_0$ is ρ if for:*

- *any edge set $D \subseteq E$,*
- *any laminar subfamily $\mathcal{L}_0$ of sets in $\mathcal{F}_0$ not covered by D,*
- *the subfamily $\mathcal{C}$ of active sets with respect to the uncovered sets of $\mathcal{F}_0$,*

*we have that the number of **crossing pairs** (S, C) (such that $S \in \mathcal{L}_0$, $C \in \mathcal{C}$, and S and C cross) is at most $\rho|\mathcal{C}|$.*

Theorem 33. *[3, Theorem 1], WGMV's primal-dual algorithm is a $(3+\rho)$-approximation for any pliable family with crossing density ρ.*

Bansal's proof of this Theorem only requires bounding the number of crossing pairs for certain laminar subfamilies, not necessarily all possible laminar subfamilies. Let $\mathcal{F}_0$ be a pliable family and $\mathcal{F}$ be the family of sets not covered by any arbitrary link set D. Let J be any inclusion-minimal link set that covers $\mathcal{F}$ such that each link in J covers at least one active set of $\mathcal{F}$. Let $\mathcal{L}_0 \subseteq \mathcal{F}$ be the laminar witness family associated with the links of J (which exists by [4, Lemma 3.2]), and let $\mathcal{C}$ be the active sets of $\mathcal{F}$.

If the number of crossing pairs between sets of $\mathcal{L}_0$ and $\mathcal{C}$ is at most $\rho|\mathcal{C}|$ for any valid D, J, this is sufficient to prove Theorem 33 for $\mathcal{F}_0$.

We will use this proof framework to improve the known approximability of strongly pliable cut covering problems using WGMV's primal-dual algorithm. Consider an arbitrary valid D, J, and the associated $\mathcal{F}, \mathcal{L}_0, \mathcal{C}$ as defined above, for a **strongly pliable** set family $\mathcal{F}_0$. It is easy to confirm that $\mathcal{F}$ (the violated sets of $\mathcal{F}_0$ with respect to D) is also strongly pliable. Let $\mathcal{L}$ be the subset of $\mathcal{L}_0$ such that the sets in $\mathcal{L}$ cross at least one active set. The sets in $\mathcal{L}_0 - \mathcal{L}$ do not contribute to the number of crossing pairs between $\mathcal{L}_0$ and $\mathcal{C}$. Additionally, since strongly pliable families satisfy the sparse crossing property, each set in $\mathcal{L}$ crosses exactly one active set.

This means that the number of crossing pairs between $\mathcal{L}_0$ and $\mathcal{C}$ is equal to $|\mathcal{L}|$. We will prove the bound $|\mathcal{L}| \leq 2|\mathcal{C}|$ using some intermediate results.

We introduce a property which can be viewed as an extension of Property γ.

Definition 25. *A set family $\mathcal{F}$ satisfies Property γ^* if, for any $A \in \mathcal{F}$ which crosses an active set $C \in \mathcal{F}$ and any subfamily $\mathcal{B} \subseteq \mathcal{F}$ of disjoint subsets of A that all cross C, then $A - C - \bigcup_{B \in \mathcal{B}} B$ is either empty or also in $\mathcal{F}$.*

Proposition 34. *Strongly pliable families satisfy Property γ^*.*

Proof. Let $\mathcal{F}$ be a strongly pliable family, with $A, C, \mathcal{B}$ as defined in the definition of Property γ^*. Suppose $A - C - \bigcup_{B\in\mathcal{B}} B$ is non-empty; it is sufficient to prove that this set is in $\mathcal{F}$.

Let $\mathcal{B}' \subseteq \mathcal{B}$, and we prove by induction on $|\mathcal{B}'|$ that $A - C - \bigcup_{B\in\mathcal{B}'} B \in \mathcal{F}$.

Base Case: $\mathcal{B}' = \emptyset$, $A - C - \bigcup_{B\in\mathcal{B}'} B = A - C$, which is in $\mathcal{F}$ by pliability because $C - A, C \cap A \subset C$, and C is inclusion-wise minimal.

Inductive Hypothesis: Suppose $A_i = A - C - \bigcup_{B\in\mathcal{B}'} B \in \mathcal{F}$, and consider $B_i \in \mathcal{B} - \mathcal{B}'$. A_i and B_i cross:

1. $A_i - B_i \supseteq A - C - \bigcup_{B\in\mathcal{B}} B \neq \emptyset$ by our assumption.
2. $B_i - A_i \equiv B_i \cap C \neq \emptyset$ since B_i crosses C.
3. $A_i \cap B_i \equiv B_i - C \neq \emptyset$ since B_i crosses C.
4. $V - (A_i \cup B) \supseteq C - A \neq \emptyset$ since A crosses C.

By strong pliability, at least one of $A_i - B_i$ or $B_i - A_i$ is in $\mathcal{F}$. However, $B_i - A_i \subseteq C$ which is inclusion-wise minimal. Hence $A_i - B_i = A - C - \bigcup_{B\in\mathcal{B}'\cup\{B_i\}} B$ is in $\mathcal{F}$. Note that strong pliability is necessary over standard pliability here, since $A_i \cap B_i$ and $A_i \cup B_i$ could be part of the family instead if $\mathcal{F}$ was not strongly pliable. □

We define a function $\phi : \mathcal{C} \to \mathcal{L} \cup \{V\}$ such that $\phi(C)$ is the smallest set in $\mathcal{L}$ that properly contains C (or $\phi(C) = V$ if no sets in $\mathcal{L}$ contain C). Each $C \in \mathcal{C}$ has only one set in $\mathcal{L}$ that satisfies $\phi(C)$.

We also define $\psi : \mathcal{L} \to \mathcal{L} \cup \{V\}$ such that $\psi(L)$ is the smallest set in $\mathcal{L}$ that properly contains L (or $\psi(L) = V$ if no sets in $\mathcal{L}$ contain L). That is, $\psi(L)$ is the parent of L in $\mathcal{L}$, and L is a child of $\psi(L)$.

Fact 4. *Let $A \in \mathcal{L}$ be a set that crosses $C_1 \in \mathcal{C}$. Suppose there exists a set $B \in \mathcal{L}$ such that $\psi(B) = A$ and B is disjoint from C_1. Then there exists $C_2 \in \mathcal{C}$ such that $\phi(C_2) = A$.*

Proof. Since $B \in \mathcal{L}$, it crosses some active set, and A can only cross one active set by the sparse crossing property. Thus, there is an active set $C_2 \subset A$ that crosses B. Hence $\phi(C_2) = A$. □

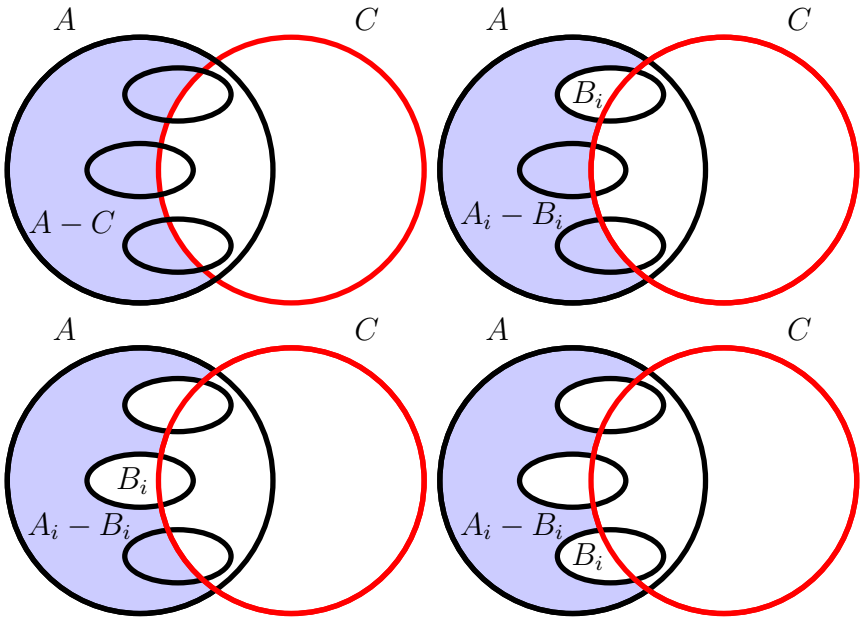


Figure 2.11: Visualization of Property γ^*.

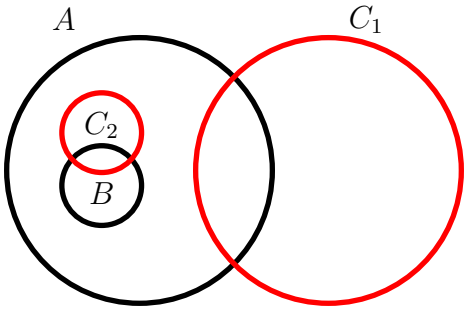


Figure 2.12: Visualization of Fact 4.

In the next lemma, we prove that for every set $A \in \mathcal{L}$, there is an active set $C_2 \in \mathcal{C}$ such that $\phi(C_2) = A$ or $\phi(C_2)$ is a child of A.

Lemma 35. *For every set $A \in \mathcal{L}$, there is a set $C \in \mathcal{C}$ such that $\phi(C) = A$, **or** there is a set $B \in \mathcal{L}$ and set $C \in \mathcal{C}$ such that $\psi(B) = A$ and $\phi(C) = B$.*

Proof. Consider $A \in \mathcal{L}$ which crosses $C_1 \in \mathcal{C}$, and let $\mathcal{B} = \{B \in \mathcal{L}, \psi(B) = A\}$; that is, $\mathcal{B}$ is the set of all children of A in $\mathcal{L}$. We examine multiple cases; cases 1, 2, and 3 each have two subcases, in which subcase (a) handles properties of the set A and subcase (b) handles properties of a set $B \in \mathcal{B}$. Case 4 results in a contradiction.

Case 1(a): A has no children. Since $A \in \mathcal{F}$, A contains an active set $C_2 \Rightarrow \phi(C_2) = A$.

Case 1(b): A has a child $B \in \mathcal{B}$ which has no children. Since $B \in \mathcal{F}$, B contains an active set $C_2 \Rightarrow \phi(C_2) = B$.

Case 2(a): There exists $B \in \mathcal{B}$ that is disjoint from C_1. By Fact 4, there exists $C_2 \in \mathcal{C}$ such that $\phi(C_2) = A$.

Case 2(b): There exists $B \in \mathcal{B}$ which has a child $B' \in \mathcal{L}$ that is disjoint from C_1. By Fact 4 (applied to B and B'), there exists $C_2 \in \mathcal{C}$ such that $\phi(C_2) = B$.

Case 3(a): All sets in $\mathcal{B}$ cross C_1, and $A - C_1 - \bigcup_{B \in \mathcal{L}:\psi(B)=A} B$ is non-empty. By Property γ^* , $A - C_1 - \bigcup_{B \in \mathcal{L}:\psi(B)=A} B$ is in $\mathcal{F}$, so it contains an active set $C_2 \Rightarrow \phi(C_2) = A$.

Case 3(b): There exists $B \in \mathcal{B}$ such that all sets in $\{B' \in \mathcal{L} : \psi(B') = B\}$ cross C_1, and $B - C_1 - \bigcup_{B' \in \mathcal{L}:\psi(B')=B} B'$ is non-empty. By Property γ^* , $B - C_1 - \bigcup_{B' \in \mathcal{L}:\psi(B')=B} B'$ is in $\mathcal{F}$, so it contains an active set $C_2 \Rightarrow \phi(C_2) = B$.

Case 4: The following hold:

- $A - C_1 - \bigcup_{B \in \mathcal{L}:\psi(B)=A} B$ is empty.
- For each $B \in \mathcal{B}$,
  - B crosses C_1.
  - All children of B in $\mathcal{L}$ cross C_1.
  - $B - C_1 - \bigcup_{B' \in \mathcal{L}:\psi(B')=B} B'$ is empty.

Recall that, by construction, $\mathcal{L}_0$ is a witness family for a link set J, such that each link in J covers at least one active set. Consider an arbitrary $B \in \mathcal{B}$. There exists a link $uv \in J$ such that $u \in B - \bigcup_{B' \in \mathcal{L}:\psi(B')=B} B'$ and $v \in A - \bigcup_{B' \in \mathcal{B}-\{B\}} B'$. This holds because B is

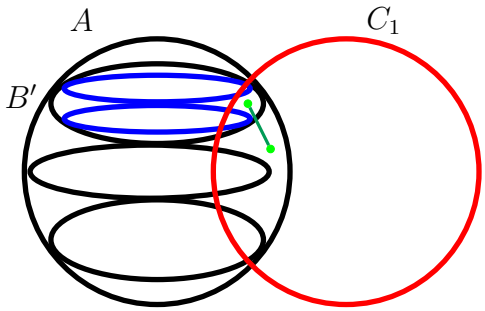


Figure 2.13: Visualization of Lemma 35, Case 4.

the witness set for uv, so $uv \in \delta(B)$, $uv \notin \delta(A)$, uv is not in any of the cuts induced by children of B, and uv is not in any of the cuts induced by other sets in $\mathcal{B}$. We are currently assuming that $B - C_1 - \bigcup_{B' \in \mathcal{L}:\psi(B')=B} B$ and $A - C_1 - \bigcup_{B' \in \mathcal{B}-\{B'\}} B'$ are both empty, hence we have $u \in B \cap C_1$ and $v \in A \cap C_1$. However, $u, v \in C_1$ contradicts the assumption that every link in J covers at least one active set (recall that active sets are disjoint). This case is impossible. □

We have enough information to bound the number of crossings.

Theorem 36. $|\mathcal{L}| \leq 2|\mathcal{C}|$.

Proof. By Lemma 35, each witness set in $\mathcal{L}$ is mapped to a unique active set, or is the unique parent of a witness set mapped to an active set. Thus, each witness set in $\mathcal{L}$ is associated with at least one active set, and each active set is associated with at most two witness sets. □

Combining this result with Bansal's proof of Theorem 33 leads to the following.

Corollary 37. *WGMV's primal-dual approximation algorithm is a 5-approximation for strongly pliable cut covering problems.*

2.3.4 Tightness of Primal-Dual Ratio for Strongly Pliable Cut Covering Problems Using WGMV

In Section 2.3.3, we proved that the WGMV primal-dual algorithm is a 5-approximation for strongly pliable cut covering problems. In this section, we confirm that this analysis is

tight with respect to the primal-dual ratio (recall Definition 6). We construct and analyze a family of strongly pliable cut covering problems with primal-dual ratio approaching 5 using WGMV's primal-dual approximation algorithm.

Theorem 38. *There exists a strongly pliable cut covering problems such that the primal-dual ratio using WGMV's primal-dual approximation algorithm approaches 5.*

We construct the family $\mathcal{F}$ and its universe as follows, given an integer $k \geq 1$. We create k "gadgets" such that gadget i contains five nodes $v_{i,1}, v_{i,2}, ..., v_{i,5}$. Each gadget contains four sets $A_{i,2} = \{v_{i,1}, ..., v_{i,5}\}, A_{i,1} = \{v_{i,1}, ..., v_{i,4}\}, B_{i,2} = \{v_{i,1}, v_{i,2}, v_{i,3}\}, B_{i,1} = \{v_{i,1}\}$. In addition to the gadgets, we add one extra node v and set $C_2 = \{v\}$; and set $C_1 = \bigcup_{i=1}^{k} \{v_{i,4}, v_{i,5}\}$. We denote these sets the "initial sets" of $\mathcal{F}$.

The initial sets and links (described below) for $k = 3$ are shown in Figure 2.14.

Claim 39. *We can add additional sets to extend $\mathcal{F}$ to a strongly pliable family such that the active sets are C_1, C_2, and $B_{i,1}$ for all $i \in [k]$.*

Proof. We create the full family $\mathcal{F}$ with the following sets:

1. The initial sets C_1, C_2, and $A_{i,1}, A_{i,2}, B_{i,1}, B_{i,2}$ $\forall\ i \in [k]$
2. $D_{\mathcal{I}} = \bigcup_{i \in \mathcal{I}} \{v_{i,1}, v_{i,2}, v_{i,3}\}$, for all $\mathcal{I} \subseteq [k], \mathcal{I} \neq \emptyset$
3. $E_{\mathcal{I}} = C_1 \cup \bigcup_{i \in \mathcal{I}} A_{i,2}$, for all $\mathcal{I} \subseteq [k]$.

Note that the inclusion-wise minimal sets are indeed the ones specified in the statement of the claim. We confirm that this family is strongly pliable by examining the crossing pairs of sets in $\mathcal{F}$, and showing that the required corner cuts are present in $\mathcal{F}$. The sets in $\mathcal{F}$ that cross are:

1. $A_{i,j}$ and $D_{\mathcal{I}}$ for $\mathcal{I} \subseteq [k], |\mathcal{I}| \geq 2, i \in \mathcal{I}, j \in [2]$. Then, $A_{i,j} \cap D_{\mathcal{I}} = D_{\{i\}} \in \mathcal{F}$ and $D_{\mathcal{I}} - A_{i,j} = D_{\mathcal{I}-\{i\}} \in \mathcal{F}$.
2. $A_{i,j}$ and C_1 for $i \in [k], j \in [2]$. Then $A_{i,j} \cup C_1 = E_{\{i\}} \in \mathcal{F}$ and $A_{i,j} - C_1 = D_{\{i\}} \in \mathcal{F}$.
3. $A_{i,j}$ and $E_{\mathcal{I}}$ for $\mathcal{I} \subseteq [k], i \in [k] - \mathcal{I}, j \in [2]$. Then, $A_{i,j} \cup E_{\mathcal{I}} = E_{\mathcal{I} \cup \{i\}} \in \mathcal{F}$ and $A_{i,j} - E_{\mathcal{I}} = D_{\{i\}} \in \mathcal{F}$.
4. $D_{\mathcal{I}}$ and $D_{\mathcal{I}'}$ for $\mathcal{I}, \mathcal{I}' \subseteq [k]$ that cross w.r.t. universe $[k]$. Then $D_{\mathcal{I}} \cup D_{\mathcal{I}'} = D_{\mathcal{I} \cup \mathcal{I}'} \in \mathcal{F}$, and $D_{\mathcal{I}} - D_{\mathcal{I}'} = D_{\mathcal{I} - \mathcal{I}'} \in \mathcal{F}$.

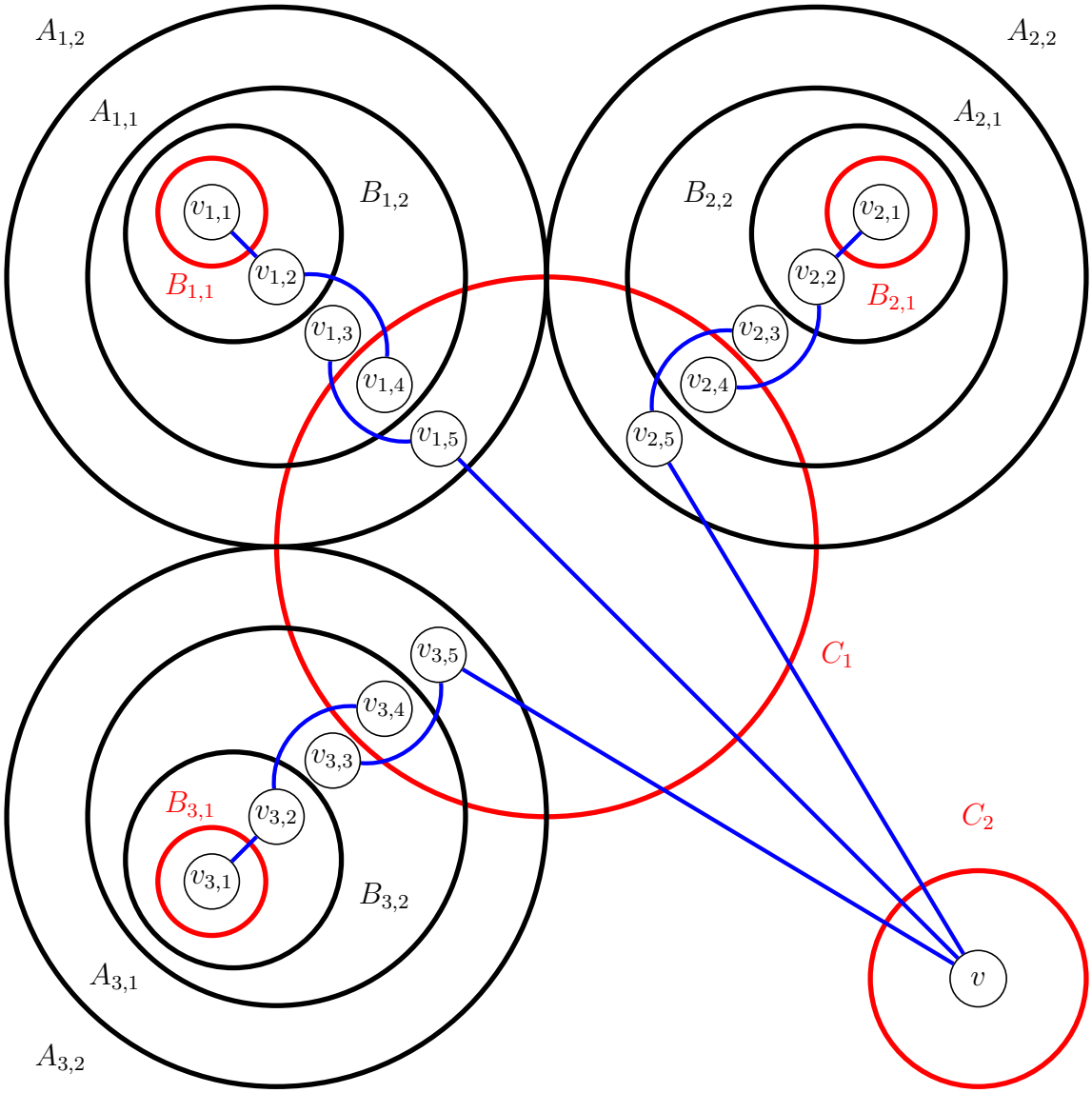


Figure 2.14: Visualization of the initial sets and links for the $k = 3$ instance in the family of examples with primal-dual ratio 5 using the WGMV primal-dual algorithm. The red sets are the active sets.

5. $E_{\mathcal{I}}$ and $E_{\mathcal{I}'}$ for any $\mathcal{I}, \mathcal{I}' \subseteq [k]$ except $\mathcal{I} \subseteq \mathcal{I}'$ or $\mathcal{I}' \subseteq \mathcal{I}$. Then $E_{\mathcal{I}} \cup E_{\mathcal{I}'} = E_{\mathcal{I} \cup \mathcal{I}'} \in \mathcal{F}$, and $E_{\mathcal{I}} - E_{\mathcal{I}'} = D_{\mathcal{I} - \mathcal{I}'} \in \mathcal{F}$.

6. $D_{\mathcal{I}}$ and $E_{\mathcal{I}'}$ for $\mathcal{I}, \mathcal{I}' \subseteq [k]$ that cross w.r.t. universe $[k]$ or $\mathcal{I}' \subset \mathcal{I}$. Then $D_{\mathcal{I}} \cap E_{\mathcal{I}'} = D_{\mathcal{I} \cap \mathcal{I}'} \in \mathcal{F}$, and $D_{\mathcal{I}} - E_{\mathcal{I}'} = D_{\mathcal{I} - \mathcal{I}'} \in \mathcal{F}$.

□

Next, we add links to cover $\mathcal{F}$. For each gadget $i \in [k]$, the following links are added: $(v_{i,1}, v_{i,2}), (v_{i,2}, v_{i,4}), (v_{i,3}, v_{i,5}), (v_{i,5}, v)$. Denote the set of all links by J.

Claim 40. *$\mathcal{F}$ is covered by J.*

Proof. We examine all sets in $\mathcal{F}$.

- $B_{i,1}, i \in [k]$ is uniquely covered by $(v_{i,1}, v_{i,2})$.
- $B_{i,2}, i \in [k]$ is uniquely covered by $(v_{i,2}, v_{i,4})$.
- $A_{i,1}, i \in [k]$ is uniquely covered by $(v_{i,3}, v_{i,5})$.
- $A_{i,2}, i \in [k]$ is uniquely covered by $(v_{i,5}, v)$.
- C_1, C_2, and $E_{\mathcal{I}}, \mathcal{I} \subseteq [k]$ are covered by $(v_{i,5}, v)$ $\forall\, i \in [k]$.
- $D_{\mathcal{I}}, \mathcal{I} \subseteq [k]$ is covered by $(v_{i,3}, v_{i,5})$ $\forall\, i \in \mathcal{I}$.

□

Finally, we analyze the WGMV approximation algorithm with respect to this example. Nutov [24] presented the following variation of WGMV's original result [28].

Lemma 41. *[24, Lemma 5] (also [28], [29, Lemma 7.7]) WGMV's primal-dual approximation algorithm is a ρ-approximation for a cut covering problem if, for any residual family feasibly covered by inclusion-minimal link set J (such that each link of J crosses at least one active set) and active sets $\mathcal{C}$, we have*

$$\sum_{C \in \mathcal{C}} \delta_J(C) \leq \rho |\mathcal{C}|.$$

We can rewrite the sum on the left-hand side by summing over the links instead.

$$\sum_{j \in J} |\{C \in \mathcal{C} : j \in \delta(J)\}| \leq \rho |\mathcal{C}|.$$

The J constructed for this example is inclusion-minimal because every link has a witness set (an A or B set). For each $i \in [k]$, the ith gadget contributes four links. Note that the link $(v_{i,1}, v_{i,2})$ is in $\delta(B_{i,1})$, the links $(v_{i,2}, v_{i,4})$ and $(v_{i,3}, v_{i,5})$ are in $\delta(C_1)$, and the link $(v_{i,5}, v)$ is in both $\delta(C_1)$ and $\delta(C_2)$. Thus, the left-hand sum is equal to $5k$. There are $k+2$ active sets. Hence, we have the inequality

$$\rho \geq \frac{5k}{k+2}$$

which asymptotically approaches 5 as k grows.

Additionally, we can modify this example to obtain primal-dual ratio (asymptotically) 5 when performing the WGMV algorithm, proving Theorem 38. Define the links' costs to be 2 for $(v_{i,5}, v)$ $\forall$ $i \in [k]$, and 1 for all other links.

Suppose we perform the WGMV primal-dual algorithm on this instance. We increase the dual values of the active sets until a link's dual constraint becomes tight. The links that are in two cuts induced by active sets have cost 2, so they will become tight if the dual values are 1. The links that that are in one cut induced by active sets have cost 1, so they will become tight if the dual values are 1. Thus, we increase the dual values of the active sets to 1, every link becomes tight, all links are selected, and none are removed during the reverse-delete stage.

The primal objective value outputted is $1 * (3k) + 2 * k$. The dual objective value outputted is $|\mathcal{C}| = k + 2$. This gives a primal-dual ratio which approaches 5 as k grows.

2.3.5 Subfamilies of Pliable Families

Section 2.3.2 presented an example of a set family that is strongly pliable, but not realizable as a family of small cuts. We examine significant subfamilies of pliable families, and examples that are in a subfamily but **not** in the subfamilies that they contain.

The largest family we examine is all **pliable families**. The primal-dual ratio for the WGMV primal-dual algorithm can be made arbitrarily large for general pliable families (see [4, Appendix A]).

Within pliable families are the subfamily of **γ-pliable families**. By definition, all of these families are pliable, but not all pliable families are γ-pliable. An example of the latter is the "cylinder" example in Appendix A of [4]. Nutov presented a proof and example [24, Section 3] that the WGMV primal-dual approximation algorithm has a tight ratio of 7 for covering γ-pliable families.

Within γ-pliable families are the subfamily of **sparse γ-pliable families**. By definition, all of these families are γ-pliable, but not all γ-pliable families are sparse. An example of the latter is Nutov's tight example for γ-pliable families. Nutov also presented a proof and example [24, Section 4] that the WGMV algorithm has a tight ratio of 6 for covering sparse γ-pliable families.

Within sparse γ-pliable families are the subfamily of **strongly pliable families**. By Section 2.3.1, these families are sparse γ-pliable, but not all sparse γ-pliable families are strongly pliable. An example of the latter is Nutov's tight example for sparse γ-pliable families, which we justify below. In Sections 2.3.3 and 2.3.4, we presented a proof and example that the WGMV algorithm has a tight ratio of 5 for covering strongly pliable families.

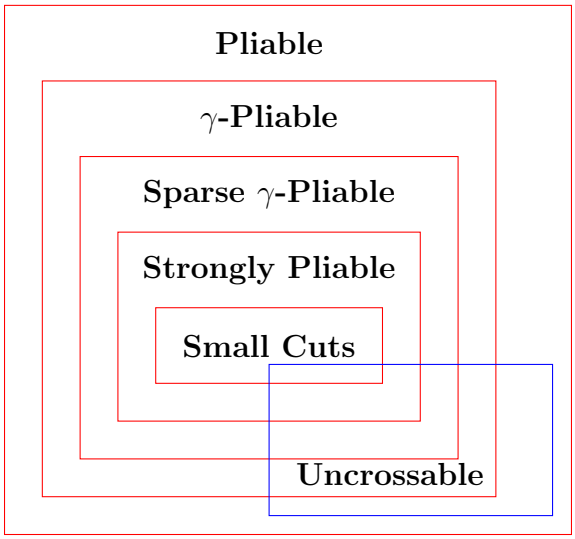
Within strongly pliable families are the subfamily of **small cut families**. By Fact 2, these families are strongly pliable, but not all strongly pliable families are families of small cuts. An example of the latter was in Section 2.3.2.

On the other hand, note that **uncrossable families** are a subfamily of pliable families that intersect the other subfamilies. There are examples of uncrossable families that are γ-pliable, sparse, strongly pliable, and realizable as families of small cuts, and there are examples that are not.

In Nutov's paper [24], he provides an example of a sparse γ-pliable cut covering problem such that the primal-dual ratio using the WGMV approximation algorithm is 6. However, the example is **not** strongly pliable. We explain why with one of his instances, shown in Figure 2.16 (instances of other sizes can be analyzed similarly).

In this example, the orange lines are links and the black circles are witness sets. The pliable set family contains five active sets: four each consisting of one red node, and one consisting of all black nodes (call it B). Note that the blue nodes are not part of any active set.

By themselves, the witness sets and active sets are not a pliable family, because several of the witness sets cross B and their corner cuts are not present. Let's try building the family $\mathcal{F}$. For any witness set S that crosses B, since B is inclusion-wise minimal, we require $B \cap S, B - S \notin \mathcal{F}$. Thus, to ensure pliability, $S - B, S \cup B \in \mathcal{F}$.



Family	WGMV Approx. Ratio	Key Example
Pliable	Unbounded	[4], Appendix A
γ-pliable	7	[24], Section 3
Sparse γ-pliable	6	[24], Section 4
Strongly pliable	5	Section 2.3.2
Small Cuts	≤ 5	
Uncrossable	2	

Figure 2.15: Visualization of the subfamilies of pliable families. The figure is not "to scale".

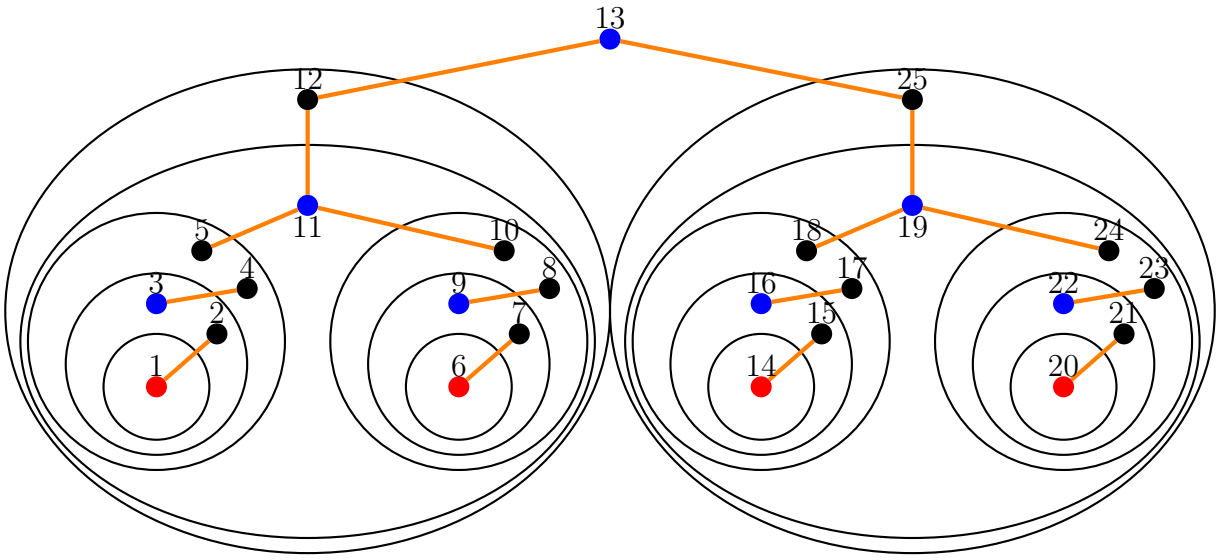


Figure 2.16: One example of Nutov's family of cut covering problems with primal-dual ratio approaching 6. These examples are sparse γ-pliable, but not strongly pliable.

Disproving strong pliability requires three steps. First, the crossing of B and $\{1,2,3,4,5,6,7,8,9,10,11\}$ gives the difference corner cut $\{1,3,6,9,11\}$ (the other difference is a subset of B).

Then, the crossing of $\{1,3,6,9,11\}$ and $\{1,2,3\}$ gives the corner cut $\{6,9,11\}$ (the other difference is a subset of B).

Finally, consider the crossing $S = \{6,9,11\}$ and $T = \{6,7,8,9,10\}$.

$S \cup T = \{6,7,8,9,10,11\}$, which is a valid member of $\mathcal{F}$.

$S \cap T = \{6,9\}$, which is a valid member of $\mathcal{F}$.

$S - T = \{11\}$, which would add an unwanted active set to $\mathcal{F}$.

$T - S = \{7,8,10\}$, which is an invalid subset of B.

Thus, to maintain pliability and the required set of active sets, the intersection and union are added to $\mathcal{F}$, but neither of the set differences are added. Thus, this set family is not strongly pliable.

2.4 Other Cut Covering Problems

2.4.1 Cover Large Cuts

We present the **Cover Large Cuts** problem. The initialization of this problem has the same parameters as Cover Smalls Cuts. However, we aim to cover the cuts with capacity **larger** than the given threshold.

Cover Large Cuts
Input: an instance consists of a graph $G(V,E)$, edge-capacity vector $u \in \mathbb{Q}_{\geq 0}^{|E|}$, threshold value $\lambda \geq 0$, link set $L \in \binom{V}{2}$, and link-cost vector $c \in \mathbb{Q}_{\geq 0}^{|L|}$.
Goal: find a subset $F \subseteq L$ of minimum c-cost such that for each $\delta(S)$ where $S \subset V, S \neq \emptyset$, and $\sum_{e \in \delta_E(S)} u_e > \lambda$; the solution satisfies $|F \cap \delta_L(S)| \geq 1$.

Theorem 42. *The integrality gap for the LP relaxation of Cover Large Cuts is unbounded.*

Proof. We present a family of examples with an integrality gap that increases unboundedly as the size of the example increases. Consider K_n (the complete graph) for even n, and label the vertices $v_1, ..., v_n$. A cut induced by one vertex has $n-1$ edges, a cut induced

by two vertices has $2(n-2)$ edges, a cut induced by three vertices has $3(n-3)$ edges, and so on. The largest cut in K_n is any cut induced by $n/2$ vertices, which has $(\frac{n}{2})^2$ edges. We consider the cut covering problem where the goal is to cover the cuts induced by $n/2$ vertices, which we call **bisection cuts**. This can be realized as a Cover Large Cuts instance by assigning every edge unit capacity and using threshold $\lambda = (\frac{n}{2})^2 - 0.5$. Let the set of links be the same as the set of edges, all with unit cost.

Remark. *This family is not pliable. For any pair of sets that cross, three of their four corner cuts (intersection and both differences) are strictly smaller than at least one of the initial sets; and the union is strictly larger than the initial sets. Here, the only sets in our family are those with size $n/2$.*

One solution for the LP relaxation of this Cover Large Cuts instance is to set $x^*_{(v_1,v_i)} = \frac{2}{n}$ for $i = 2, ..., n$ and $x^*_e = 0$ for all other links e.

Lemma 43. *x^* as described above is a feasible solution to the LP relaxation of this Cover Large Cuts instance.*

Proof. Consider any bisection cut S. By symmetry, suppose that $v_1 \in S$. Then, $\frac{n}{2} - 1$ other vertices are in S, and the remaining $\frac{n}{2}$ vertices are not in S. The non-zero links in $\delta(S)$ are (v_1, v_i) for all $i \in V - S$. Each one is given value $\frac{2}{n}$, so the total value of links crossing the cut is $\frac{n}{2} * \frac{2}{n} = 1$ as required. □

The total objective value of the solution is $\frac{2(n-1)}{n}$.

Lemma 44. *The optimal integer solution to the Cover Large Cuts instance described above requires more than $\frac{n}{4}$ links.*

Proof. Assume for contradiction that all bisection cuts can be covered with $n/4$ links L'. Links join two nodes, so $n/4$ links can be incident to at most $n/2$ nodes. This means that the subgraph induced by L' has at least $n/2$ singleton components. The bisection cut consisting of $n/2$ singleton components on one side and all other nodes on the other side has no links in L'. This contradicts the feasibility of L'. □

By the previous lemmas, the integrality ratio of this family of examples is at least

$\frac{n}{4} * \frac{n}{2(n-1)}$

$= \frac{n^2}{8(n-1)}$

which grows arbitrarily large with n.

□

We solved this Cover Large Cuts instance via direct computation with Maple on $n = 4, 6, 8, 10$. In all of these instances, the exact optimal integer solution required $n/2$ links. We conjecture that this is always the optimum.

Conjecture 1. *Given any graph with an even number of vertices n, the minimum number of links needed to cover every bisection cut is $n/2$.*

2.4.2 Cover Bounded Cuts

The next problem is **Cover Bounded Cuts**. An instance of the problem includes both an upper and lower bound, and the family of cuts to be covered are those with edge capacity within both bounds.

Cover Bounded Cuts
Input: an instance consists of a graph $G(V, E)$, edge-capacity vector $u \in \mathbb{Q}_{\geq 0}^{|E|}$, two threshold values $\lambda_\ell, \lambda_u \geq 0$, link set $L \in \binom{V}{2}$, and link-cost vector $c \in \mathbb{Q}_{\geq 0}^{|L|}$.
Goal: find a subset $F \subseteq L$ of minimum c-cost such that for each $\delta(S)$ where $S \subset V, S \neq \emptyset$, and $\lambda_\ell < \sum_{e \in \delta_E(S)} u_e < \lambda_u$; the solution satisfies $|F \cap \delta_L(S)| \geq 1$.

This problem generalizes both Cover Small Cuts (by choosing an arbitrarily small λ_l) and Cover Large Cuts (by choosing an arbitrarily large λ_u). Thus, because it encapsulates Cover Large Cuts, the integrality gap for its LP relaxation is also unbounded.

Lemma 45. *Cover Bounded Cuts is not necessarily pliable.*

Proof. We can use the Remark in Section 2.4.1 as a counterexample, but we present a separate example for completeness.

Let $G = (V, E)$ with $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{v_1v_2, v_2v_3, v_2v_4, v_3v_4, v_4v_5\}$, $u_e = 1$, $\forall\ e \in E$, $\lambda_l = 1$ and $\lambda_u = 3$. Thus, sets with cut capacity exactly 2 are required to be covered. Consider two such sets $A = \{v_2, v_3, v_4\}, \delta(A) = \{v_1v_2, v_4v_5\}$ and $B = \{v_4, v_5\}, \delta(B) = \{v_2v_4, v_3v_4\}$.

Then, $\delta(A \cap B) = \delta(\{v_4\}) = \{v_2v_4, v_3v_4, v_4v_5\}$

$\delta(A \cup B) = \delta(\{v_2, v_3, v_4, v_5\}) = \{v_1v_2\}$

$\delta(A - B) = \delta(\{v_2, v_3\}) = \{v_1v_2, v_2v_4, v_3v_4\}$

$\delta(B - A) = \delta(\{v_5\}) = \{v_4v_5\}$

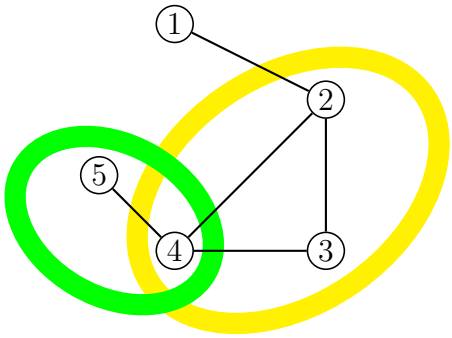


Figure 2.17: Non-pliable Cover Bounded Cuts instance. Edges are depicted and have capacity 1.

All four of these cuts have capacity either 3 or 1, so none of these sets are also in the family of cuts to be covered. This is a violation of the pliable property.

□

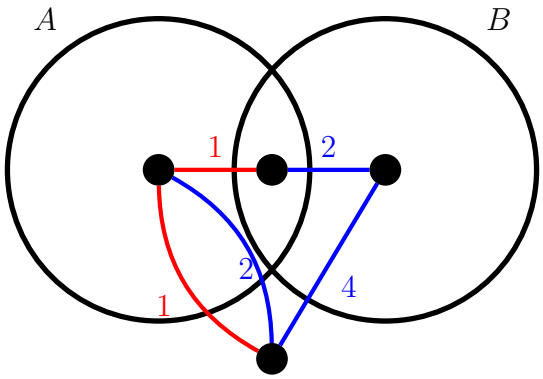
2.4.3 k-Cover Small Cuts

Given an integer $k \geq 1$, we can generalize Cover Small Cuts to **k-Cover Small Cuts**. In this problem, the goal is to cover each small cut with k links. The transition from Cover Small Cuts to k-Cover Small Cuts resembles the transition from the minimum spanning tree problem to the k-edge-connected spanning subgraph problem.

k-Cover Small Cuts
Input: an instance consists of a graph $G(V, E)$, edge-capacity vector $u \in \mathbb{Q}_{\geq 0}^{|E|}$, threshold value $\lambda \geq 0$, link set $L \in \binom{V}{2}$, and link-cost vector $c \in \mathbb{Q}_{\geq 0}^{|L|}$.
Goal: find a subset $F \subseteq L$ of minimum c-cost such that for each $\delta(S)$ where $S \subset V, S \neq \emptyset$, and $\sum_{e \in \delta_E(S)} u_e < \lambda$; the solution satisfies $|F \cap \delta_L(S)| \geq k$.

Using known results on Cover Small Cuts (Corollary 37) and standard results from the area [29], there is an $O(k \log |V|)$-approximation algorithm for this problem. This problem has some useful applications. There is a reduction from a constant-factor approximation algorithm for (integer-capacity) 2-Cover Small Cuts to a constant-factor approximation algorithm for $(2, q)$-FGC, and vice versa [5]. However, no constant-factor approximation algorithm is currently known for either of these problems.

Figure 2.18: Non-pliable Intersection of Small Cuts instance



2.4.4 Intersection of Small Cuts

Consider a graph $G(E_1, E_2)$ with two sets of capacitated edges and two thresholds λ_1 and λ_2. Let $\mathcal{C}_1$ be the family of small cuts with respect to E_1 and λ_1. Let $\mathcal{C}_2$ be the family of small cuts with respect to E_2 and λ_2. The **Intersection of Small Cuts** problem seeks to find a minimum-cost set of links such that each cut in $\mathcal{C}_1 \cap \mathcal{C}_2$ is covered.

Intersection of Small Cuts
Input: an instance consists of a graph $G(V, E_1, E_2)$ with two edge sets, edge-capacity vectors $u_1, \in \mathbb{Q}_{\geq 0}^{|E_1|}, u_2 \in \mathbb{Q}_{\geq 0}^{|E_2|}$ for both edge sets, threshold values $\lambda_1, \lambda_2 \geq 0$, link set $L \in \binom{V}{2}$, and link-cost vector $c \in \mathbb{Q}_{\geq 0}^{|L|}$.
Goal: find a subset $F \subseteq L$ of minimum c-cost such that for each $\delta(S)$ where $S \subset V, S \neq \emptyset$, and $\sum_{e \in \delta_{E_1}(S)} u_{1e} < \lambda_1, \sum_{e \in \delta_{E_2}(S)} u_{2e} < \lambda_2$; the solution satisfies $|F \cap \delta_L(S)| \geq 1$.

Using known results on Cover Small Cuts (Corollary 37) and standard results from the area [29], there is an $O(\log |V|)$-approximation algorithm for this problem.

Lemma 46. *Intersection of Small Cuts is not necessarily pliable.*

Proof. Consider the counterexample in Figure 2.4.4.

Let the blue lines represent E_1 edges with $\lambda_1 = 5$, and let the red lines represent E_2 edges with $\lambda_2 = 2$, with capacities labelled for both colours. Since A and B have capacity 4 from blue edges and 1 from red edges, $A, B \in \mathcal{C}_1 \cap \mathcal{C}_2$. However, $A \cup B \notin \mathcal{C}_1$, $A - B \notin \mathcal{C}_2, B - A \notin \mathcal{C}_1$. Only $A \cap B$ is also small with respect to both edge sets, so the properties of a pliable family do not apply. □

This problem is a useful stepping stone between Cover Small Cuts and k-Cover Small Cuts.

Proposition 47. *Suppose we have an α-approximation algorithm for Intersection of Small Cuts. Then we can formulate an $\alpha + 5$ approximation algorithm for 2-Cover Small Cuts.*

Algorithm 2 is presented below.

Algorithm 2: Approximation algorithm for 2-Cover Small Cuts, given approximation algorithm for Intersection of Small Cuts

---

Given: Let $G(V, E), u, \lambda$ and L, c be an instance of 2-Cover Small Cuts.

1. Use the 5-approximation algorithm for Cover Small Cuts to return a set of links L_1 that 1-cover the cuts in G with capacity less than λ. The cost of 1-covering these cuts is at most the cost of 2-covering these cuts, so this step contributes a factor of 5 to the approximation.

2. Use the assumed α-approximation algorithm for Intersection of Small Cuts to return a set of links L_2 that 1-cover cuts according to the following instance: let $E_1, u_1, \lambda_1 = E, u, \lambda$; that is, cuts that will be covered in this step are small cuts with respect to the original 2-Cover Small Cuts instance. Let $E_2, u_2, \lambda_2 = L_1, \mathbf{1}, 2$; that is, cuts that will be covered in this step have fewer than two links from L_1 crossing them. Let the link set be $L - L_1$. This step will use unselected links to cover cuts that are small with respect to the original problem, but were not 2-covered by the first step. Note that by step 1, all small cuts have at least one link crossing them, so this step, combined with L_1, will 2-cover all small cuts; and incur approximation factor α.

3. Return $L_1 \cup L_2$.

Chapter 3

Flexible Graph Connectivity

3.1 7-Approximation for (1,q)-FGC

Flexible Graph Connectivity ("FGC") problems were first introduced by Adjiashvili et al. [1]. In the (p, q)**-Flexible Graph Connectivity problem**, generalized by Boyd et al. [7], we have a graph $G(V, E)$ such that the edges are partitioned into two sets $E = \mathcal{S} \cup \mathcal{U}$. $\mathcal{S}$ is the set of **safe edges**, representing edges in a network that are consistently reliable. $\mathcal{U}$ is the set of **unsafe edges**, representing edges that could break down. We aim to determine a minimum-cost set of edges that satisfy given connectivity requirements, even when some unsafe edges fail.

> (p, q)**-Flexible Graph Connectivity (**(p, q)**-FGC)**
> **Input:** a graph $G(V, E)$ such that each edge $e \in E$ has cost $c_e \geq 0$; and the edges are partitioned into $E = \mathcal{S} \cup \mathcal{U}$.
> **Goal:** find a minimum c-cost $F \subseteq E$ such that every non-trivial cut $\delta(S), S \subset V$ satisfies either $|F \cap \delta_{\mathcal{S}}(S)| \geq p$ or $|F \cap \delta_E(S)| \geq p + q$.

For the $(1, q)$-FGC problem, the goal is to find a minimum-cost set of edges F such that the subgraph on F is connected, even after at most q unsafe edges are removed. If all non-trivial cuts are covered by at least one safe edge or at least $q + 1$ unsafe edges, they satisfy this requirement.

This section presents a 7-approximation algorithm for $(1, q)$-FGC. For a graph $G = (V, E)$ and $S \subseteq V$, the cut $\delta(S)$ refers to the set of edges that have exactly one end-node in S; $\delta(S)$ is called a non-trivial cut if $\emptyset \neq S \subset V$. (Whenever we use the term "cut $\delta(S)$"

we mean that S is a subset of $V(G)$.) We use the term *small cut* to mean a non-trivial cut $\delta(S)$ with total capacity below a specified threshold λ.

Our starting point is the natural LP relaxation that follows from taking a capacitated network design view of $(1,q)$-FGC where each unsafe edge $e \in \mathcal{U}$ has capacity $u_e = 1$, and each safe edge $e \in \mathcal{S}$ has capacity $u_e = (q+1)$. The natural LP relaxation then seeks to minimize the total cost of edges subject to the constraint that for all $\emptyset \subset S \subset V$, we have $\sum_{e \in \mathcal{S} \cap \delta(S)} (q+1) x_e + \sum_{e \in \mathcal{U} \cap \delta(S)} x_e \geq (q+1)$.

It is easy to see that any feasible solution to this LP with zero/one values is a valid solution to a given instance of $(1,q)$-FGC, and vice versa. However, this LP can have integrality ratio as large as $(q+1)$. Consider a graph with two nodes, joined with one safe edge of cost 1 and q unsafe edges of cost 0. An integer solution must select the safe edge and incur cost 1, but a fractional solution can take all unsafe edges and assign value $\frac{1}{q+1}$ to the safe edge.

To get around this obstruction, we strengthen the LP relaxation of $(1,q)$-FGC using the knapsack-cover inequalities (see [9]) to obtain the following stronger LP. Intuitively, the added knapsack-cover inequalities for a cut $\delta(S)$ ensure that if a safe edge is being used to cover $\delta(S)$ and, moreover, $\delta(S)$ is partly covered by unsafe edges, say, by ℓ of them, then the capacity of the safe edge is reduced to $(q+1) - \ell$.

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e && && \text{(KCLP:}(1,q)\text{-FGC)} \\
\text{s.t.} \quad & \sum_{e \in \mathcal{S} \cap \delta(S)} (q+1) x_e + \sum_{e \in \mathcal{U} \cap \delta(S)} x_e \geq (q+1) && \forall \ \emptyset \subset S \subset V \\
& \sum_{e \in E \cap \delta(S) - A} u_e(A,S) x_e \geq D(A,S) && \forall \ \emptyset \subset S \subset V, A \subseteq E \\
& 0 \leq x_e \leq 1 && \forall \ e \in E,
\end{aligned}$$

where $D(A,S) = \max\{0,\ (q+1) - \sum\{u_e \mid e \in A \cap \delta(S)\}\}$, and $u_e(A,S) = \min\{u_e,\ D(A,S)\}$.

Ibrahimpur & Vegh [20] have presented a polynomial-time separation subroutine for the CND LP relaxation with knapsack-cover inequalities for (p,q)-FGC. Using this, we will design a polynomial-time approximation algorithm for $(1,q)$-FGC.

In this section, let α denote the best approximation ratio known for Cover Small Cuts. We will design an $(\alpha+2)$-approximation algorithm for $(1,q)$-FGC. As of now, we have $(\alpha+2) = 7$, since $\alpha = 5$ (see Section 2.3.3).

Let LP_{opt} denote the optimal value of (KCLP:$(1,q)$-FGC), the LP with the knapsack-cover inequalities. Using binary search for LP_{opt} together with the ellipsoid algorithm and

the polynomial-time subroutine, we can find a vector x^* with cost $c(x^*) = \sum_e c_e x_e$ such that x^* is optimal for (KCLP:$(1,q)$-FGC). Then, we will "round" x^* to an integer solution of cost $\leq (\alpha+2)c(x^*) \leq (\alpha+2)\text{LP}_{\text{opt}}$.

The Rounding Algorithm: Given a value z and a vector x^* of cost $c(x^*) \leq z$ that satisfies (KCLP:$(1,q)$-FGC), Algorithm 3, presented below, rounds it to an integer solution of cost at most $(\alpha+2)z$. We describe the main idea of our rounding scheme.

Recall that our LP relaxation assigns capacity $u_e = 1$ for each unsafe edge $e \in \mathcal{U}$ and capacity $u_e = (q+1)$ to each safe edge $e \in \mathcal{S}$. We call a non-trivial cut $\delta(S)$ a small cut if $|\mathcal{U}_1 \cap \delta(S)| + \sum_{e \in \mathcal{U}_2 \cap \delta(S)} \frac{(\alpha+2)}{2} x^*_e < (q+1)$ where $\mathcal{U}_1 = \{e \in \mathcal{U} : x^*_e \geq \frac{2}{(\alpha+2)}\}$, and $\mathcal{U}_2 = \mathcal{U} - \mathcal{U}_1$. To handle the presence of small cuts, we first construct an instance of Cover Small Cuts with the specified small cuts and with link-set the set of safe edges; then, via Lemma 48, we show that the vector x^* restricted to safe edges and scaled up by a factor of $\frac{(\alpha+2)}{\alpha}$ (i.e., $\frac{(\alpha+2)}{\alpha} x^*_{\mathcal{S}}$) constitutes a feasible solution to the LP relaxation of the Cover Small Cuts instance. We can thus pick a set $\mathcal{S}_1$ of safe edges by applying the α-approximation algorithm to the Cover Small Cuts instance; note that the cost of $\mathcal{S}_1$ is at most α times the cost of $\frac{(\alpha+2)}{\alpha} x^*_{\mathcal{S}}$.

After this step, we contract the connected components formed by the edges in $\mathcal{S}_1$, and get a new instance that does not have any small cuts.

Since there are no small cuts, every non-trivial cut $\delta(S)$ in the contracted graph satisfies the inequality $|\mathcal{U}_1 \cap \delta(S)| + \sum_{e \in \mathcal{U}_2 \cap \delta(S)} \frac{(\alpha+2)}{2} x^*_e \geq (q+1)$. Therefore, we get a feasible solution for the LP relaxation of the f-connectivity problem where $f(S) = q+1$ for every non-trivial set $S \subset V$, by picking all edges in $\mathcal{U}_1$ and scaling up x^* restricted to $\mathcal{U}_2$ by a factor of $(\alpha+2)/2$.

Recall Theorem 1, Jain's iterative rounding 2-approximation [21] for certain f-connectivity problems and cut covering problems.

Definition 26. *A function $f : 2^V \to \mathbb{Z}$ **weakly supermodular** if for any $A, B \subseteq V$, then at least one of the following holds:*

- $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$
- $f(A) + f(B) \leq f(A - B) + f(B - A)$

Since f is weakly supermodular (and integral, and has at least one non-zero value), we can apply Jain's iterative rounding method to solve this f-connectivity problem and obtain a 2-approximate solution, giving us an integral solution whose cost is at most $(\alpha+2)$ times the cost of $x^*_{\mathcal{U}}$ (see Lemma 49). Thus, we get an integral solution whose total cost of safe and unsafe edges is at most $(\alpha+2) * \mathrm{LP}_{\mathrm{opt}}$.

Algorithm 3: $(\alpha+2)$-approximate solution to $(1,q)$-FGC

Require: Graph $G=(V,E)$ where E is partitioned into $\mathcal{S} \cup \mathcal{U}$, with edge costs $\{c_e\}_{e\in E}$. A solution x^* to (KCLP:$(1,q)$-FGC).

1. $\mathcal{U}_1 \leftarrow \{e \in \mathcal{U} : x^*_e \geq \dfrac{2}{(\alpha+2)}\}$, and $\mathcal{U}_2 \leftarrow \mathcal{U} - \mathcal{U}_1$.

2. (a) Apply the approximation algorithm for Cover Small Cuts on the instance with $G' = (V, E' = \mathcal{U})$, where each edge of $\mathcal{U}_1$ is given unit capacity, each edge $e \in \mathcal{U}_2$ is given capacity $\dfrac{(\alpha+2)}{2} x^*_e$, and with link set $\mathcal{S}$. Define the threshold λ (for small cuts) to be $(q+1)$. Thus, we have:

$$|\mathcal{U}_1 \cap \delta(S)| \; + \sum_{e \in \mathcal{U}_2 \cap \delta(S)} \frac{(\alpha+2)}{2} x^*_e < (q+1) \qquad \text{(definition of } \textit{small cuts}\text{)}$$

   (b) Let the output of the call in step (a) be denoted $\mathcal{S}_1$.

3. Construct a graph $G'' = (V, \mathcal{U}_2)$ where each edge $e \in \mathcal{U}_2$ has cost c_e. Define the requirement of a non-trivial cut $\delta(S)$ to be

$$f(S) = \max\{0, \;\; (q+1) - ((q+1)|\delta(S) \cap \mathcal{S}_1| + |\delta(S) \cap \mathcal{U}_1|)\}.$$

   This function f is weakly supermodular, so a 2-approximate solution $\mathcal{U}_3$ for this instance of f-connectivity can be computed using Jain's iterative rounding method [21].

4. Return $\mathcal{S}_1 \cup \mathcal{U}_1 \cup \mathcal{U}_3$.

The next two lemmas formalize the key properties of the solution x^* that are used in the rounding scheme above. We will show that it returns a feasible integral solution of cost at most $(\alpha+2)c(x^*)$.

Lemma 48. *In step 2 of Algorithm 3, a feasible fractional solution to the Cover Small Cuts instance is given by* $\hat{x}_e = \min\{1, \frac{(\alpha+2)}{\alpha} x^*_e\}$ *for* $e \in \mathcal{S}$.

Proof. Consider any non-trivial small cut $\delta(S)$. The cut $\delta(S)$ satisfies the knapsack-cover inequality w.r.t. the set $A = \mathcal{U}_1$:

$$\sum_{e \in \mathcal{S} \cap \delta(S)} u_e(A,S) x_e + \sum_{e \in \mathcal{U}_2 \cap \delta(S)} u_e(A,S) x_e \geq D(A,S),$$

where $D(A,S) = \max\{0,\ (q+1) - \sum\{u_e \mid e \in A \cap \delta(S)\}\} = \max\{0,\ (q+1) - |\mathcal{U}_1 \cap \delta(S)|\}$, and $u_e(A,S) = \min\{u_e,\ D(A,S)\}$.

Moreover, since $\delta(S)$ is a small cut, we have $|\mathcal{U}_1 \cap \delta(S)| + \sum_{e \in \mathcal{U}_2 \cap \delta(S)} \frac{(\alpha+2)}{2} x^*_e < (q+1)$. We rewrite this inequality as $\sum_{e \in \mathcal{U}_2 \cap \delta(S)} \frac{(\alpha+2)}{2} u_e(A,S) x^*_e < D(A,S)$, which is the same as $\sum_{e \in \mathcal{U}_2 \cap \delta(S)} u_e(A,S) x^*_e < \frac{2}{(\alpha+2)} D(A,S)$. Thus, by the knapsack-cover inequality, we have $\sum_{e \in \mathcal{S} \cap \delta(S)} u_e(A,S) x^*_e \geq \frac{\alpha}{(\alpha+2)} D(A,S)$. By definition, $u_e(A,S) \leq D(A,S)$, hence, we have $\sum_{e \in \mathcal{S} \cap \delta(S)} \frac{(\alpha+2)}{\alpha} D(A,S) x^*_e \geq D(A,S)$. Therefore, $\sum_{e \in \mathcal{S} \cap \delta(S)} \hat{x}_e \geq 1$, completing the proof. □

Lemma 49. *In step 3 of Algorithm 3, a feasible fractional solution to the* f*-connectivity problem is given by* $x'_e = \frac{(\alpha+2)}{2} x^*_e$ *for* $e \in \mathcal{U}_2$.

Proof. By way of contradiction, suppose that the claim does not hold. Then for some non-trivial cut $\delta(S)$, we have $(q+1)|\mathcal{S}_1 \cap \delta(S)| + |\mathcal{U}_1 \cap \delta(S)| + \sum_{e \in \mathcal{U}_2 \cap \delta(S)} \frac{(\alpha+2)}{2} x^*_e < (q+1)$. This implies that the cut $\delta(S)$ is a small cut in step 2 of Algorithm 3. Hence, step 2 ensures (via Cover Small Cuts) that $|\mathcal{S}_1 \cap \delta(S)| \geq 1$ and so $(q+1)|\mathcal{S}_1 \cap \delta(S)| \geq (q+1)$. This is a contradiction. □

The output of Algorithm 3 is feasible for $(1,q)$-FGC by Lemmas 48, 49. The cost of the edges in $\mathcal{U}_1$ is $\leq \frac{(\alpha+2)}{2} \sum_{e \in \mathcal{U}_1} c_e x^*_e$ since $x^*_e \geq \frac{2}{(\alpha+2)}$ for each edge e in $\mathcal{U}_1$. Additionally,

the cost of the edges in $\mathcal{S}_1$ is $\leq (\alpha+2)\sum_{e\in\mathcal{S}_1} c_e x^*_e$ by Lemma 48 and our definition of α. Lastly, the cost of the edges in $\mathcal{U}_3$ is at most $\sum_{e\in\mathcal{U}_2}(2)((\alpha+2)/2)c_e x^*_e = (\alpha+2)\sum_{e\in\mathcal{U}_2} c_e x^*_e$, by Lemma 49. Therefore, the cost of the solution returned by Algorithm 3 is at most $(\alpha+2)c(x^*)$.

The outer loop of our algorithm runs a binary search for LP_{opt}, using the polynomial-time separation subroutine. Given a vector x^* (a candidate solution to (KCLP:$(1,q)$-FGC)), the subroutine either finds that x^* violates one of the constraints of (KCLP:$(1,q)$-FGC) or else the algorithm rounds x^* to an integer solution of cost $\leq (\alpha+2)c(x^*)$, where $c(x^*) = \sum_e c_e x^*_e$. The binary search for LP_{opt} starts with the interval $[0, c(E)]$, where $c(E) = \sum_e c_e$. Assume that the instance has a feasible integer solution, let OPT denote the cost of an optimal integer solution, and assume that $0 < \text{OPT} \leq c(E)$.

In an arbitrary iteration, the binary search calls the ellipsoid algorithm with the additional constraint $\sum_e c_e x_e \leq z$, where the current interval is $[\ell_c, h_c]$ and $z = \frac{\ell_c+h_c}{2}$. (The binary search maintains the invariant: $\text{LP}_{\text{opt}} > \ell_c$ and there exists an integer solution of cost $\leq (\alpha+2)h_c$.) The ellipsoid algorithm calls the separation subroutine one or more times, and either (1) reports that the LP (with the additional constraint) is infeasible or else (2) the algorithm finds a vector x^* with $c(x^*) \leq z$ and an integer solution of cost $\leq (\alpha+2)c(x^*)$. The binary search continues as usual, that is, in case (1) it replaces the current interval $[\ell_c, h_c]$ by the upper half-interval $[\frac{\ell_c+h_c}{2}, h_c]$, and in case (2) it replaces the current interval by the lower half-interval $[\ell_c, \frac{\ell_c+h_c}{2}]$. The binary search terminates when the current interval $[\ell_{\mathit{final}}, h_{\mathit{final}}]$ is sufficiently small. The LP with the additional constraint $\sum_e c_e x_e \leq \ell_{\mathit{final}}$ is infeasible, and the Algorithm found an integer solution of cost $\leq (\alpha+2)h_{\mathit{final}}$. Hence, $\text{LP}_{\text{opt}} > \ell_{\mathit{final}}$ and the last integer solution found by the Algorithm has cost $\leq (\alpha+2)\text{LP}_{\text{opt}}$.

Theorem 50. *There is a polynomial-time algorithm that, given an instance of* $(1,q)$-FGC, *computes a vector* x^* *of cost at most* OPT *that satisfies the constraints of* (KCLP:$(1,q)$-FGC), *and rounds it to a feasible integer solution of cost at most* 7 OPT.

3.2 Relation Between FGC and CND

In the previous section, we represented $(1,q)$-FGC as an IP using the format of Cap-k-ECSS. Each safe edge had capacity $(q+1)$, each unsafe edge has capacity 1, and $k = q+1$. Is it possible to use CND to create such an IP for (p,q)-FGC with any value of p, q? We confirm that this is not possible.

Assume for contradiction that we can realize a (p, q)-FGC instance as a Capacitated Network Design instance, with $p, q > 0$. Define functions $f(p, q)$ and $g(p, q)$, such that $f(p, q), g(p, q) > 0$ and let $L = p * f(p, q) = (p + q) * g(p, q)$. We then modify G such that each safe edge has capacity $f(p, q)$, each unsafe edge has capacity $g(p, q)$. Our goal is that this modification results in a CND instance with capacity L that captures (p, q)-FGC.

A cut that does not satisfy the properties of (p, q)-FGC should not satisfy the properties of CND. For any $i = 1, 2, ..., p - 1$, a cut does not satisfy (p, q)-FGC if it has at most $p - i$ safe edges and at most $q + i - 1$ unsafe edges. In the corresponding CND instance, this gives the constraint $(p - i)f(p, q) + (q + i - 1)g(p, q) < L$.

The first requirement tells us $p*f(p, q) = (p+q)*g(p, q) \rightarrow f(p, q) = g(p, q)*(p+q)/p$. We use the second set of constraints to get the following, for all $i = 1, 2, ..., p - 1$:

$$(p - i)f(p, q) + (q + i - 1)g(p, q) < L = p * f(p, q)$$

$$-if(p, q) + (q + i - 1)g(p, q) < 0$$

$$-i * \frac{p + q}{p} * g(p, q) + (q + i - 1)g(p, q) < 0$$

We cancel out $g(p, q)$, which is valid because $g(p, q) > 0$.

$$q + i - 1 < i * \frac{p + q}{p}$$

$$pq + ip - p < i * (p + q)$$

$$pq < iq + p$$

$$p < i + \frac{p}{q}$$

$$q < \frac{iq}{p} + 1$$

Recall that $i \in \{1, ..., p - 1\}$. If $i = 1$, the second-last inequality is $p < 1 + \frac{p}{q}$. If $q \geq p$, then $\frac{p}{q} + 1 \leq 2$, which means $p \leq 1$. Thus, this inequality requires either $q < p$ or $p = 1$.

Also when $i = 1$, the last inequality is $q < \frac{q}{p} + 1$. If $p \geq q$, then $\frac{q}{p} + 1 \leq 2$, which means $q \leq 1$. Thus, this inequality requires either $p < q$ or $q = 1$.

Since $p < q$ and $q < p$ cannot both be true, at least one of p or q is 1, which contradicts the ability to find a conversion for arbitrary p, q.

Thus, (p, q)-FGC can only be exactly modelled as a CND problem if $p = 1$ or $q = 1$.

3.3 2-Approximation for "Soft" (p,q)-FGC

A constant-factor approximation algorithm for (p, q)-FGC has not yet been proven or disproven to exist. Section 3.1 presents an 7-approximation for $(1, q)$-FGC, and several other results provide approximations for FGC with an explicit value for p or q. This includes constant-factor approximation algorithms for $(p, 2)$-FGC [4] and $(p, 3)$-FGC [3], and an $O(q)$-approximation algorithm for $(2, q)$-FGC [10]. The current best approximation for general (p, q)-FGC is an $O(\log(|V|))$-approximation algorithm by Ibrahimpur and Vegh [20].

To explore the approximability of (p, q)-FGC further, we consider a relaxed problem. With reduced restrictions, the problem becomes easier to approximate.

The **"Soft" (p, q)-Flexible Graph Connectivity problem** is defined as follows. We have all of the same initialization of (p, q)-FGC: a graph $G = (V, E)$ such that $E = \mathcal{S} \cup \mathcal{U}$ where $\mathcal{S}$ edges are safe and $\mathcal{U}$ edges are unsafe. **Additionally, a solution is allowed to select an edge multiple times.** An edge that is selected k times contributes k edges of the same type to any cut that it covers, and k times its cost to the objective value.

We present Algorithm 4 and analyze its feasibility and approximation ratio.

Algorithm 4: soft (p, q)-FGC:

---

1. Initialize two "solutions": edge set $F' = \emptyset$ and solution $F = \emptyset$.
2. Replace the cost of each unsafe $e \in \mathcal{U}$ with $\frac{c_e}{p+q}$ and the cost of each safe edge $e \in \mathcal{S}$ with $\frac{c_e}{p}$.
3. Using the updated costs, sort the edges from smallest to largest cost.
4. Iterate through the edges in the order defined. Let e be the currently examined edge.
   (a) If e is a safe edge and $F' \cup \{e\}$ contains no cycles, add e to F', and p copies of e to F.
   (b) If e is an unsafe edge and $F' \cup \{e\}$ contains no cycles, add e to F', and $p + q$ copies of e to F.
5. Return F.

We first note that this algorithm is feasible for (p, q)-FGC. A feasible (p, q)-FGC solution has either p safe edges or $p+q$ total edges in every non-trivial cut. F' is a spanning tree as this algorithm is performing Kruskal's algorithm with scaled edge costs. A spanning tree has at least one edge in every non-trivial cut. When we select a safe edge, we take p copies for F. When we select an unsafe edge, we take $p+q$ copies for F.

Consider the following LP; note that this is a "soft" CND problem where safe edges are represented with x variables and have capacity $p+q$; and unsafe edges are represented with y variables and have capacity q. By Section 3.2, the equivalent IP does not perfectly capture soft (p, q)-FGC, but we will justify that it is a suitable relaxation.

LP (1)

$$\begin{aligned} &\text{minimize} \sum_{e \in S} c_e x_e + \sum_{e \in U} c_e y_e \\ \text{subject to} \quad &(p+q) * x(\delta_S(C)) + p * y(\delta_U(C)) \geq p(p+q), \quad \forall\, C \subset V, C \neq \emptyset \\ &x, y \geq 0 \end{aligned}$$

Claim 51. *All integer solutions to soft (p, q)-FGC are feasible for LP (1).*

Proof. In an integer solution, each non-trivial cut is covered by p safe edges or $p+q$ edges in total. For any given non-trivial cut, p safe edges will satisfy its LP constraint because each safe edge has coefficient $(p+q)$. $p+q$ edges in total will satisfy the constraint because all edges have coefficient at least p. □

Corollary 52. *The optimal value of LP (1) is a lower bound for the optimal integer value of soft (p, q)-FGC.*

Theorem 53. *Algorithm 4 is a 2-approximation for soft (p, q)-FGC.*

Proof. Apply the following transformation on LP (1): $x'_e = (p+q) * x_e, y'_e = p * y_e$. This results in alternate LP:

LP (2)

$$\begin{aligned} &\text{minimize} \sum_{e \in S} \frac{c_e}{p+q} x'_e + \sum_{e \in U} \frac{c_e}{p} y'_e \\ \text{subject to} \quad &x'(\delta_S(C)) + y'(\delta_U(C)) \geq p(p+q), \quad \forall\, C \subset V, C \neq \emptyset \\ &x, y \geq 0 \end{aligned}$$

This is a unit-capacity $p(p+q)$-connectivity problem where the safe edges have a smaller cost by a factor of $\frac{1}{p+q}$ and the unsafe edges have a smaller cost by a factor of $\frac{1}{p}$.

We can perform another transformation on the variables. Let $x''_e = \frac{x'_e}{p(p+q)} = \frac{x_e}{p}, y''_e = \frac{y'_e}{p(p+q)} = \frac{y_e}{p+q}$. This gives another LP:

LP (3)

$$\text{minimize} \sum_{e \in S} p * c_e x''_e + \sum_{e \in U} (p+q) * c_e y''_e$$

$$\text{subject to} \quad x''(\delta_S(C)) + y''(\delta_U(C)) \geq 1, \quad \forall\, C \subset V, C \neq \emptyset$$

$$x, y \geq 0$$

This LP is a minimum spanning tree problem. Note that all three LPs are equivalent up to their variable transformations, so their optimal objective values are the same.

Claim 54. *The cost of F outputted by Algorithm 4 is at most double the optimal cost of LP (3) (and thus the optimal cost of the soft (p,q)-FGC problem).*

Proof. A spanning tree F' with cost k with respect to the $(\frac{c_e}{p+q}, \frac{c_e}{p})$ edge costs has cost $p(p+q)k$ with respect to the $(p * c_e, (p+q) * c_e)$ edge costs. Thus, the cost of the copied tree F with the costs used in LP (1) is the same as the cost of the non-copied tree F' with the costs used in LP (3). Since the optimal values of all three LPs are the same, and the minimum spanning tree LP relaxation has integrality ratio at most 2, the copied tree F has cost at most double the optimal cost, as required. □

3.4 Approximation for "Semi-Hard" (p,q)-FGC

Now that we have a constant-factor approximation for (p,q)-FGC where all edges can be selected multiple times, we re-restrict some (but not all) of the missing constraints. In this problem, we allow ourselves to select multiple copies of some edges. This is an intermediate result in between the previous section's constant-factor approximation for soft (p,q)-FGC, and the $O(\log |V|)$-approximation currently known for "hard" (p,q)-FGC.

The **"Semi-Hard" (p,q)-Flexible Graph Connectivity problem** is defined as follows. We have all of the same initialization as (p,q)-FGC, but **we are allowed to select unsafe edges multiple times in our solution.** Safe edges can be selected at most once.

We use the following LP as a relaxation of this problem. This is is the LP relaxation of the semi-hard Cap-k-ECSS problem, where the capacity of safe edges is $p+q$, the capacity of unsafe edges is p, and the required cut-capacity threshold is $k = p(p+q)$. By Section 3.2, the equivalent IP does not perfectly capture semi-hard (p,q)-FGC; but feasible solutions to

semi-hard (p,q)-FGC also satisfy this LP (see Claim 51). We use x variables to represent the inclusion of safe edges in the solution set, and y variables to represent the inclusion of unsafe edges in the solution set.

$$\min \sum_{e \in \mathcal{S}} c_e x_e + \sum_{e \in \mathcal{U}} c_e y_e$$

$$\text{s.t.} \quad (p+q) \sum_{e \in \mathcal{S} \cap \delta(S)} x_e + p \sum_{e \in \mathcal{U} \cap \delta(S)} y_e \geq p(p+q), \quad \forall\, \emptyset \neq S \subset V$$

$$0 \leq x_e \leq 1, \quad \forall\, e \in \mathcal{S}$$

$$0 \leq y_e \leq p+q, \quad \forall\, e \in \mathcal{U}$$

We also include additional **knapsack cover inequality** constraints for this LP (see [9]). Note that feasible solutions to semi-hard (p,q)-FGC will also satisfy these constraints. For any edge e, let $u_e = p+q$ if $e \in \mathcal{S}$ and $u_e = p$ if $e \in \mathcal{U}$. Then, for any $A \subseteq E$ (which can include multiple copies of unsafe edges) and $\emptyset \neq S \subset V$, we define $D(A,S) = \max(0, p(p+q) - \sum_{e \in \delta(S) \cap A} u_e)$ and $u_e(A,S) = \min(u_e, D(A,S))$. The constraints are:

$$\sum_{e \in (\delta(S) - A) \cap \mathcal{S}} u_e(A,S) x_e + \sum_{e \in \delta(S) \cap \mathcal{U}} u_e(A,S)(y_e - |A \cap e|) \geq D(A,S), \quad \forall\, A \subseteq E, \emptyset \neq S \subset V$$

For this paper, we will assume that we have a polynomial-time separation routine that allows us to find optimal solutions to the LP above with knapsack cover inequalities. We present Algorithm 5, which uses similar methods to the approximation algorithm for Cap-k-ECSS in [6].

Claim 55. *In any iteration i of Algorithm 5, Step 2, the solution $x'_e = \min(1, 2x^*_e)$, $\forall\, e \in \mathcal{S} - F_{\mathcal{S}}$ is a feasible fractional solution for the Cover Small Cuts instance constructed.*

Proof. Let $\delta(S)$ be a non-trivial small cut in iteration i of Step 2(a). Let $A = F_{\mathcal{U}} \cup F_{\mathcal{S}}$ be the edges selected at this point in the Algorithm. Recall the knapsack inequality:

$$\sum_{e \in (\delta(S) - F_{\mathcal{S}}) \cap \mathcal{S}} u_e(A,S) x^*_e + \sum_{e \in \delta(S) \cap \mathcal{U}} u_e(A,S) y'_e \geq D(A,S)$$

Since $\delta(S)$ is a small cut, then

$$\sum_{e \in \delta(S) \cap \mathcal{U}} 2p y'_e + \sum_{e \in \delta(S) \cap F_{\mathcal{U}}} p + \sum_{e \in \delta(S) \cap F_{\mathcal{S}}} (p+q) < p(p+q)$$

Algorithm 5: $O(p^2)$-approximation algorithm for semi-hard (p, q)-FGC

Require: Graph $G = (V, E)$ with partition $E = \mathcal{S} \cup \mathcal{U}$ and costs c_e, $\forall\, e \in E$. A vector (x^*, y^*) which is an optimal solution for the LP relaxation with knapsack inequalities.

1. For each $e \in \mathcal{U}$, initialize $F_{\mathcal{U}}$ to contain $\lfloor y_e^* \rfloor$ copies of edge e. Let $y'_e = y_e^* - \lfloor y_e^* \rfloor$ be the **remainder** of each unsafe edge. Initialize $F_{\mathcal{S}} = \emptyset$.

2. For $i = 1, ..., p$:

   (a) Construct a Cover Small Cuts instance on the graph $G_i(V, \mathcal{U} \cup F_{\mathcal{U}} \cup F_{\mathcal{S}})$. Each edge in $\mathcal{U}$ is given capacity $2py'_e$, each edge in $F_{\mathcal{U}}$ (including multiple copies) is given capacity p, and each edge in $F_{\mathcal{S}}$ is given capacity $p + q$. The small-cut threshold is $p(p + q)$ and the link set is $\mathcal{S} - F_{\mathcal{S}}$.

   (b) Apply the approximation algorithm to the instance of Cover Small Cuts. Add the solution returned to $F_{\mathcal{S}}$.

3. Construct a graph $G' = (V, \mathcal{U})$. Define the requirement of a non-trivial cut $\delta(S)$ to be

$$f(S) = \max(0, p(p + q) - (p + q)|\delta(S) \cap F_{\mathcal{S}}| - p|\delta(S) \cap F_{\mathcal{U}}|)$$

   This function f is weakly supermodular, and we will confirm later that $2py'$ is a feasible solution to the linear relaxation of this problem. Thus, we can use Jain's iterative rounding method to round $2py'$ to a 2-approximate solution F' to this semi-hard f-connectivity problem.

4. Return the union of $F_{\mathcal{S}}$ and p copies of $F' \cup F_{\mathcal{U}}$.

$$\sum_{e \in \delta(S) \cap \mathcal{U}} 2py'_e < D(A,S)$$

$$\sum_{e \in \delta(S) \cap \mathcal{U}} py'_e < \frac{1}{2} D(A,S)$$

Thus, by the knapsack inequalities, $\sum_{e \in (\delta(S) - F_{\mathcal{S}}) \cap \mathcal{S}} u_e(A,S) x^*_e + \geq \frac{1}{2} D(A,S)$

$\sum_{e \in (\delta(S) - F_{\mathcal{S}}) \cap \mathcal{S}} 2x^*_e \geq 1 \Rightarrow \sum_{e \in (\delta(S) - F_{\mathcal{S}}) \cap \mathcal{S}} x'_e + \geq 1$

This means that any small cut is fractionally covered by x'. x' is a feasible solution for the LP relaxation of Cover Small Cuts as desired. □

Claim 56. *At the start of Step 3 of Algorithm 5, all non-trivial cuts $\delta(S)$ satisfy*

$$\sum_{e \in \delta(S) \cap \mathcal{U}} 2py'_e + p|\delta(S) \cap F_{\mathcal{U}}| + (p+q)|\delta(S) \cap F_{\mathcal{S}}| \geq p(p+q)$$

Proof. Suppose not. Then there exists some cut such that

$$\sum_{e \in \delta(S) \cap \mathcal{U}} 2py'_e + p|\delta(S) \cap F_{\mathcal{U}}| + (p+q)|\delta(S) \cap F_{\mathcal{S}}| < p(p+q)$$

The left-hand side of this inequality increases over the course of Step 2. This means that $\delta(S)$ was a small cut for every iteration of Step 2. An iteration of Step 2 adds a safe edge incident to each small cut, and there are p iterations overall. This means that $|\delta(S) \cap F_{\mathcal{S}}| \geq p$, but this leads to a contradiction.

Thus, $2py'$ is a feasible solution to f-connectivity problem in Step 3, and further note that $F' \cup F_{\mathcal{U}} \cup F_{\mathcal{S}}$ is a feasible solution to the capacity-$p(p+q)$-edge connected spanning subgraph problem. □

Lemma 57. *The final output of Algorithm 5 is a feasible solution for semi-hard (p,q)-FGC.*

Proof. As mentioned, $F' \cup F_{\mathcal{U}} \cup F_{\mathcal{S}}$ is a feasible integer solution of the LP relaxation. Consider two cases for an arbitrary non-trivial cut $\delta(S)$.

Case 1: $|F_{\mathcal{S}} \cap \delta(S)| \geq p$. The outputted solution satisfies the requirements of (p,q)-FGC for this cut.

Case 2: $|F_{\mathcal{S}} \cap \delta(S)| = i < p$. Then,

$$(p+q)i + p|\delta(S) \cap (F_{\mathcal{U}} \cup F')| \geq p(p+q)$$

$$|\delta(S) \cap (F_{\mathcal{U}} \cup F')| \geq \frac{p-i}{p}(p+q)$$

Thus, taking p copies of $F_{\mathcal{U}} \cup F'$ satisfies the requirements of (p,q)-FGC for this cut. □

Theorem 58. *Algorithm 5 is an $O(p^2)$ approximation for semi-hard (p,q)-FGC.*

Proof. We examine the cost of the outputted solution in two parts.

Cost of Safe Edges: In Step 2 of the Algorithm, we create Cover Small Cuts instances such that $\min(1, 2x_e^*)$ is a feasible solution. The current best approximation ratio for Cover Small Cuts is 5, so the edges added in one iteration of Step 2 cost at most 10 times more than the cost of the safe edges in (x^*, y^*). Since we perform p iterations, $c(F_{\mathcal{S}}) \leq 10pc(x^*)$.

Cost of Unsafe Edges: In Step 1 of the algorithm, we select unsafe edges corresponding to the floor of each unsafe edge's variable in y^*. These edges contribute the same cost in $y^* - y'$ and $\mathcal{F}_{\mathcal{U}}$. In Step 3 of the Algorithm, we create an f-connectivity problem such that $2py'$ is a feasible solution. Jain's iterative rounding method is a 2-approximation, so the set of edges added in Step 3 cost at most $4p$ times more than the cost of the unsafe edges in (x^*, y^*). Since we then take p additional copies, $c(p(F_{\mathcal{U}} \cup F')) \leq 4p^2c(y^*)$.

The total cost of a solution is the cost of both the safe and unsafe edges. The total cost of the solution returned by the Algorithm is thereby at most $\min(10p, 4p^2)c(x^*, y^*) = O(p^2)c(x^*, y^*)$.

In the case of semi-hard $(2,q)$-FGC, the Algorithm is a $\min(10 * 2, 4 * 2^2) = 20$-approximation. □

Chapter 4

Primal-Dual Computational Results

4.1 WGMV Algorithm Summary

When an optimization problem admits a linear relaxation, it is often useful to examine the **dual** as well. **Primal-dual approximation algorithms** construct a primal solution and a dual solution together.

Williamson, Goemans, Vazirani, and Mihail [28] constructed a 2-approximation algorithm for cut covering problems on uncrossable families. This was an extension of approximation algorithms for the minimum Steiner forest problem by Goemans and Williamson [18] and Agarwal et al.[2]. The WGMV approximation algorithm utilizes the LP relaxation of the cut covering problem and its dual, and introduces a significant **reverse delete** step.

WGMV's algorithm is useful since it can be used for cut covering problems beyond uncrossable families, although the approximation ratio may be worse than 2. For example, Bansal et al. [4] showed that the WGMV algorithm is a 16-approximation for γ-pliable cut covering problems, and Section 2.3.3 showed that the WGMV algorithm is a 5-approximation for strongly pliable cut covering problems (which includes Cover Small Cuts).

We provide a thorough summary of WGMV's primal-dual approximation algorithm for Cover Small Cuts.

Input: Graph $G(V, E)$, with capacities $u_e \in \mathbb{Q}_{\geq 0}$ on edges $e \in E$; a link set L with costs $c_e \in \mathbb{Q}_{\geq 0}$ on links $e \in L$; threshold value $\lambda \geq 0$.

Goal: Choose a minimum-cost set $F \subseteq L$ such that for each non-trivial $S \subset V$ with $\sum_{e \in \delta_E(S)} u_e < \lambda$, at least one link in F covers $\delta(S)$.

Given a non-trivial $S \subset V$, let $u(\delta(S)) = \sum_{e \in \delta_E(S)} u_e$.

Initialization: Let $F = \emptyset$. Create a dual solution y such that $y_S = 0$ for all $S \subseteq V$. Define the **slack** $s(e)$ of a link $e \in L$ as $c_e - \sum_{S: e \in \delta(S)} y_S$. Thus, given the initial dual solution, we have $s(e) = c_e, \ \forall \ e \in L$.

4.1.1 Phase 1: Determine active sets

This subalgorithm to determine active sets is adapted from [23].

Definition 27. *An **active set** (also called a **core**) is a violated set in $\mathcal{F}$ that is inclusion-wise minimal with respect to the subfamily of all violated sets.*

1. Initialize $\mathcal{C} = \{\{v\} : v \in V\}, G' = G$
2. While $|V(G')| \geq 3$,
   (a) Create an ordering of the vertices by choosing a vertex v such that $u(\delta(v)) \leq u(\delta(w)), \ \forall \ w \in V(G')$; removing the vertex and all incident edges from the graph; then repeating until all vertices have been ordered.
   (b) Let u, v be the last two vertices in this order. Update G' to be the graph after contracting u and v into one vertex z.
   (c) Let Z be the set of vertices of V that make up z (if the graph was completely uncontracted). Add Z to $\mathcal{C}$.
3. For each $X \in \mathcal{C}$, compare it to each $Y \in \mathcal{C}$ such that $Y \subset X$. If $\delta(X)$ is not a small cut, remove X from $\mathcal{C}$. If both $\delta(X)$ and $\delta(Y)$ are small cuts, remove X from $\mathcal{C}$.
4. Return $\mathcal{C}$.

4.1.2 Phase 2: Augment dual solution

1. For each link in L, select an e such that $\varepsilon = s(e)/|\{C \in \mathcal{C} : e \in \delta(C)\}|$ is minimized.
2. Add e to F.
3. Update $y_C = y_C + \varepsilon$ for all $C \in \mathcal{C}$.
4. Remove e from L, and add e to E with capacity λ.

4.1.3 Phase 3: Reverse delete

Iterate through the links of F in the opposite order of how they were added to F. When examining link e, if $F - e$ is feasible, update $F \to F - e$.

4.1.4 Algorithm

After initialization, repeat Phases 1 and 2 until Phase 1 returns an empty $\mathcal{C}$. Then perform Phase 3 and return F.

To Confirm Feasibility of F: Let $E' = E \cup F$ with $u'_e = u_e$ for $e \in E$ and $u'_e = \lambda$ for $e \in F$. Use any polynomial-time algorithm that returns the minimum-capacity cut of the graph with respect to E' and u'. Then, F is feasible if and only if the minimum cut capacity is at least λ.

4.2 WGMV Maple Code

To use WGMV's algorithm more efficiently, we wrote a program in the Maple programming language which can implement the algorithm automatically given an inputted graph. Operations that take a long time to perform by hand can be easily run in the background on a computer. In subsequent sections, we use the program to check the primal-dual ratio of various graphs. The following is a full summary of the procedures created to implement the WGMV primal-dual approximation algorithm in Maple. The file itself can be accessed here.

Input: The program requires the standard information about a Cover Small Cuts instance:

- A list of edges E
- A list of capacities u for each edge
- A list of links L
- A list of costs c for each link
- A small-cut threshold λ
- The number of vertices n. Note that the program names the vertices $1, ..., n$.

This input is given at the beginning of the program as global variables. We now list the key procedures.

4.2.1 getOrder: determines minimum-capacity order of vertices

Input: edge list, capacity list (custom edge/capacity lists are given as input. During the active set generation phase of the algorithm, the graph changes due to contractions. Thus, the edge set given here may not be the same as the original graph).

Output: order of vertices.

Steps:

1. This procedure loops n times. In any iteration:
   (a) Loop through all remaining edges. Keep track of the total capacity of each singleton-vertex cut. Add each edge's capacity to the total capacity for its two endpoints.
   (b) Choose a vertex with smallest capacity that has not yet been added to the order; add it to the order.
   (c) Remove all edges incident to that vertex from the list of edges.

4.2.2 contr: contracts an edge-set or link-set

Input: stored-vertices list (before the first contraction, this list is $[\{1\}, \{2\}, ..., \{n\}]$), edge list, capacity list, edge e to contract.

Output: updated stored-vertices list, updated list of edges, updated list of capacities.

Steps:

1. Store e's second endpoint in e's first endpoint. For example, if the initial stored-vertices list is $[\{1\}, \{2\}, \{3\}]$ and we contract $(1, 2)$, the updated list is $[\{1, 2\}, \{\}, \{3\}]$
2. Iterate through the edges to find those incident to e's second endpoint. Update these edges to be incident to e's first endpoint. Remove edge e itself from the edge set.

4.2.3 cutCap: determines cut capacity

Input: cut set (list of vertices), edge list, capacity.

Output: total capacity of inputted cut.

Steps: Iterate through all edges. Add the capacity of an edge to the total if it has exactly one endpoint in the inputted cut set.

4.2.4 activeSets: determines the active sets for any iteration

Input: global variables, current solution.

Output: list of active sets.

Steps:

1. Create a temporary copy of the edge graph, adding the current solution set to the set of edges with capacity λ on each one.
2. Iterate until there are only two vertices in the copied graph:
   (a) Use getOrder (with current copied graph as input) to determine the minimum-capacity order of vertices.
   (b) Add the last two vertices (and all vertices stored within them) to the list of candidate active sets. For example, if the current stored-vertices list is $[\{1,2\},\{\},\{3\},\{4\}]$ and the last two vertices are "1" and 4, then the candidate set is $\{1,2,4\}$.
   (c) Contract the pair of vertices chosen using the contr procedure.
3. Return to examining the original graph (with solution edges added). Iterate through the candidate sets from the previous step.
   (a) If a candidate is not a small cut (checked using cutCap), remove it from the candidate list.
   (b) Iterate through all other inputted sets. If another set is a small cut (checked using cutCap) and a subset of the current candidate, remove it from the candidate list.
4. Return the candidate list.

4.2.5 primalDualIt: runs one iteration of primal-dual algorithm

Input: list of links' slack, current list of dual variables, current list of dual values, current solution, global variables.

Output: next link to add to the solution, updated list of links' slack, updated list of dual variables, updated list of dual values.

Steps:

1. Find the active sets using the activeSets procedure. If there are no active sets, end this procedure early.

2. Iterate through all links.

   (a) Iterate through all active sets. Keep track of the number of active sets that the current link crosses. In order for this link constraint to become tight, we would increase the dual values by the slack on the link divided by the number of active sets covered by the link. (We disregard any links covering zero active sets).

   (b) Keep track of which link has the smallest value described above, called mvpVal in the code.

3. Once the link closest to becoming tight is chosen, iterate through all active sets. Increase the dual of every active set by mvpVal. The dual is stored as two lists, one of sets and one of values, such that the ith set in the first list has dual value equal to the ith number in the second list. All sets not in the list have dual value 0.

4. Iterate through the links to decrease the slack of each link by mvpVal*(number of active sets covered by that link).

4.2.6 primalDualFull: the full algorithm

Input: global variables.

Output: algorithm solution, dual solution.

Steps:

1. Initialize the list of dual variables and values as empty sets, and the list of slack values as the costs of the links.

2. Iterate until primalDualIt terminates early.

   (a) Use procedure primalDualIt to run one iteration of the primal-dual algorithm.

   (b) Add the link returned to the candidate solution set.

3. Reverse-delete: iterate through the links chosen in reverse order.

   (a) Remove the current link from the candidate solution.

(b) Run activeSets given this modified solution. This is equivalent to checking feasibility because if there are no minimum violated sets, then there are no violated sets in general.

(c) If there are no violated sets, replace the solution with the modified version. If there are violated sets, leave the solution as-is.

4.3 WGMV Algorithm Experiments

We ran the primal-dual approximation algorithm for Cover Small Cuts described in the previous section on a variety of graphs. The goal of these experiments was to determine the practical performance of the algorithm. This includes both the runtime of the program, and the accuracy of the approximation (determined via the primal-dual ratio). Graphs were chosen semi-randomly based on interesting properties, and from known noteworthy graphs such as the Petersen graph.

A sample of the graphs tested are displayed in Table 4.1. All graphs were found on the House of Graphs database [8]. Each graph used the same edge set and link set. Tests with "unit" in the Capacity/Costs column means that the capacity of every edge was 1 and the cost of every link was 1. Tests with "random $\in [i]$" in the Capacity/Costs column means that capacities and costs were chosen uniformly at random from $1, 2, ..., i$. For random capacities/costs, the program was ran multiple times with different randomized values. The smallest and largest primal-dual ratios found are recorded.

The program ran well overall. The program can implement the algorithm on graphs with 40 vertices or fewer in no longer than one minute. The examples implemented during these experiments resulted in primal-dual ratios of at most 2. In Section 2.3.4, we found an example with primal-dual ratio (approaching) 5, but this example requires a precise construction. Our experiments suggest that on random/arbitrary graphs, the WGMV primal-dual algorithm achieves a primal-dual ratio of at most 2 fairly consistently.

4.4 Randomized Algorithm for Small Cuts

The WGMV primal-dual approximation algorithm for Cover Small Cuts requires finding only the inclusion-wise minimal violated sets in any given iteration. This can be done in polynomial time [23]. However, if one wants to find *all* small cuts given a capacitated edge set, this is a more complicated problem. This is necessary to determine the **integrality**

Table 4.1: Results of implementing the WGMV primal-dual algorithm on various graphs.

Name	H.o.G. ID	# Nodes	# Links/Edges	Capacities/Costs	λ	Run time	P-D Ratio
Petersen	660	10	15	Unit	6	2s	1.8
Poussin	1273	15	39	Unit	10	3s	1.47
					15		1.87
Babalan 10 Cage	949	70	105	Unit	8	10min	1.74
None	17030	40	65	Unit	8	40s	1.7
					12		1.9
				Random $\in [5]$	20		1.4-1.6
				Random $\in [5]$	25		1.6-1.8
None	27116	40	60	Unit	8	1min	1.8
					12		1.9
				Random $\in [5]$	20		1.5-1.8
					30		1.7-1.9
				Random $\in [10]$	40		1.5-1.7

ratio of Cover Small Cuts problems. The integrality ratio is the ratio between the optimal solution of an IP and the optimal solution of its LP relaxation. To determine this ratio for Cover Small Cuts, we require all small cuts to construct the complete IP and its LP relaxation, then solve both.

The naive method to determine all small cuts is to check the capacity of every non-trivial cut, but this requires an exponential number of checks. One other method is to employ **randomness**. Karger & Stein [22] constructed a randomized algorithm that, with high probability, lists cuts within factor α of the minimum cut size. Given a graph with capacitated edges, to use Karger & Stein's algorithm to find cuts of capacity less then λ, we use a standard polynomial-time algorithm (such as the Stoer-Wagner algorithm [26]) to find the minimum cut size $\lambda(G)$, then let $\alpha = \lambda/\lambda(G)$. We provide a full summary of Karger & Stein's algorithm below.

The main part of the algorithm has two procedures.

Contract_Until (G, k)**:**

1. Choose a random edge in the graph, with randomness weighted based on the capacities of the edges.

2. Update the graph by contracting the chosen edge. Keep parallel edges but remove self-loops.

Table 4.2: Results of random algorithm Test 1

Number of Repetitions	Run Time (s)	Small Cuts Found	Percentage of Total
1	< 1	6	9
5	< 1	29	48
10	1	42	69
20	2	56	92
30	3	61	100

3. Repeat these steps until the resulting graph has k vertices.

Recursive_Contraction (G, n)**:** (where n is the number of vertices in G)

1. If $n \leq \lceil 2\alpha \rceil$, then exhaustively search all non-trivial vertex subsets of G and return cuts with capacity less than λ.
2. Otherwise, do the following **twice**, creating two separate instances of the problem:
   (a) Let $G' = \text{Contract_Until}(G, \lceil n/\sqrt[2\alpha]{2} + 1 \rceil)$.
   (b) Return $\text{Recursive_Contraction}(G', \lceil n/\sqrt[2\alpha]{2} + 1 \rceil)$.

Note that all cuts returned by this algorithm are small cuts, but we rely on probability to return **all** small cuts. Thus, to improve the chances of finding as many cuts as possible, we can run Recursive_Contraction multiple times and return the union of each instance's output.

4.4.1 Accuracy Tests

In the following tests, we implemented the randomized algorithm for small cuts described in the previous section using a Maple program. The program repeats the algorithm multiple times, and then outputs the union of all small cuts found throughout. All graphs were found on the House of Graphs database [8]. The file itself can be accessed here.

Test 1: The graph is the Petersen graph, Graph 660 on House of Graphs. All edges have capacity 1. $\lambda = 6$. This graph has 10 vertices and 15 edges. This instance has 61 small cuts, as outputted by an exhaustive search program.

Table 4.3: Results of random algorithm Test 2

Number of Repetitions	Run Time (s)	Small Cuts Found	Percentage of Total
1	1	7	9
5	2	27	36
10	2	47	62
20	3	64	84
30	3	67	88
40	3	75	99
50	4	75	99
60	5	76	100

Test 2: The graph is the Poussin graph, Graph 1273 on House of Graphs. All edges have capacity 1. $\lambda = 10$. This graph has 15 vertices and 39 edges. This instance has 76 small cuts, as outputted by an exhaustive search program.

Test 3: Same graph as Test 2, with $\lambda = 15$. This instance has 1157 small cuts, as outputted by an exhaustive search program (which took over one hour to run).

Test 4: The graph is the Babalan 10 Cage, Graph 949 on House of Graphs. All edges have capacity 1. $\lambda = 8$. This graph has 70 vertices and 105 edges. This instance is too large to run on our exhaustive search program.

We added an extra procedure to the random algorithm that repeats the algorithm in groups of 100, and stops when a group of 100 does not add any new cuts to the overall list. On Test 4, the program took around 11 minutes to run, ran 4300 iterations, and returned 11138 small cuts.

Then, we had the program stop once **three** 100-iteration sequences didn't return any new small cuts. On Test 4, this version took around 14 minutes, ran 5200 iterations to run, and returned 11154 small cuts.

Table 4.4: Results of random algorithm Test 3

Number of Repetitions	Run Time (s)	Small Cuts Found	Percentage of Total
1	1	78	7
5	2	264	23
10	2	472	41
20	3	707	61
30	3	848	73
40	4	915	79
50	4	984	85
60	4	1070	92
70	4	1067	92
80	4	1106	96
90	5	1118	97
100	5	1133	98
150	6	1148	99
200	7	1155	99
250	7	1155	99
300	9	1157	100
500	14	1157	100

Table 4.5: Results of random algorithm Test 4

Number of Repetitions	Run Time (s)	Small Cuts Found
1	4	51
50	13	1522
100	20	2591
200	34	4166
300	42	5424
400	53	6288
500	65	6935
600	77	7598
1000	120	9200
2000	217	10661

Chapter 5

Conclusion

5.1 Overview

In this thesis, we examined several concepts in the field of network design, specifically Cover Small Cuts, strongly pliable families, Flexible Graph Connectivity, and primal-dual approximation algorithms.

While the Cover Small Cuts problem is relatively new in the literature, its analysis builds on classical results related to uncrossable families and cut covering problems. We explored strongly pliable families (which includes families of small cuts), a subfamily of pliable families, and we showed that they have essential differences from uncrossable families. For example, cut covering problems on uncrossable families can be 2-approximated with Jain's iterative rounding algorithm. On the other hand, we showed that Theorem 2, the key property used to prove the approximation ratio of Jain's algorithm, does not hold for Cover Small Cuts, even when relaxed to any constant bound (see Lemma 3). Cut covering problems on uncrossable families can also be 2-approximated with WGMV's primal-dual approximation algorithm. However, we proved (tightly with respect to primal-dual ratio) that this algorithm is a 5-approximation for strongly pliable cut covering problems.

We explored some other cut covering problems. We showed a Cover Large Cuts/Cover Bounded Cuts instance with integrality ratio $\Omega(n)$, where n is the number of vertices in the graph.

Our main result on the topic of Flexible Graph Connectivity was an 7-approximation algorithm for $(1, q)$-FGC. This relates to the approximability of Cover Small Cuts; if there exists an α-approximation algorithm for Cover Small Cuts, then the algorithm presented in

Section 3.1 achieves approximation ratio $(\alpha+2)$. We explored how relaxing Flexible Graph Connectivity, by allowing the selection of multiple edges, leads to better approximation ratios. We constructed a 2-approximation algorithm for (p,q)-FGC when multiple edges of any kind can be selected, and an $O(p^2)$-approximation algorithm for (p,q)-FGC when multiple unsafe edges can be selected. Recall that the best known approximation ratio for the standard (p,q)-FGC problem, as of the writing of this thesis, is $O(\log|V|)$ [20].

5.2 Open Problems

Generalizing our Result on Set Families Realizable as Families of Small Cuts: in Section 2.2, we presented small cut families that take the form $\mathcal{F}_1 \cup \mathcal{F}_2$, where $\mathcal{F}_1$ is a laminar family and $\mathcal{F}_2$ is a family of disjoint sets. This was useful because the key example in Section 2.1 used families of this form. Is it possible to generalize Theorem 13 to include more set families? Is it possible to give sufficient conditions for a set family to be realizable as a family of small cuts? For example, the following conjecture was explored while working on this thesis.

Conjecture 2. *Let $\mathcal{F}$ be a family with universe V such that $\mathcal{F}$ can be partitioned into $\mathcal{F}_1 \cup \mathcal{F}_2$, such that both subfamilies are laminar; and, for any $S \in \mathcal{F}_1$ and $D \in \mathcal{F}_2$ that cross, then $S \cup D, S - D \in \mathcal{F}_1$. Additionally, a key node of each set in $\mathcal{F}_1 \cup \{V\}$ exists in V. Then $\mathcal{F}$ is realizable as a family of small cuts.*

WGMV Approximation Ratio for Cover Small Cuts Problems: we showed a tight example to prove that the primal-dual ratio for strongly pliable cut covering problems is at most 5 using the WGMV primal-dual approximation algorithm. Can the example in Section 2.3.4 be realized as a family of small cuts? If so, then we have also shown that the WGMV algorithm achieves a tight primal-dual ratio of 5 for the Cover Small Cuts problem. If not, can the approximability of Cover Small Cuts problems be improved further?

Improved Approximability of k-Cover Small Cuts and Intersection of Small Cuts: there exists an $O(k\log|V|)$-approximation algorithm for k-Cover Small Cuts and an $O(\log|V|)$-approximation algorithm for Intersection of Small Cuts. To the best of our knowledge, better approximation algorithms have not been found. Do better approximations exist? These problems do not posses the properties (pliability, etc.) used to prove the approximability of Cover Small Cuts. Thus, analyzing their approximability using WGMV's primal-dual algorithm (or any algorithm) is more difficult. Finding a constant-factor approximation algorithm for these problems would be useful because they have

applications to other problems. For example, Bansal et al. [5] demonstrated constant-factor reductions between approximation algorithms for (integer-capacity) 2-Cover Small Cuts and $(2, q)$-FGC.

Restricting the Soft and Semi-Hard FGC Algorithms: can the approximation algorithms for soft and semi-hard (p, q)-FGC lead to an improved approximation algorithm for (p, q)-FGC? Does a constant-factor approximation algorithm exist for (p, q)-FGC, assuming $p, q \in O(1)$?

References

[1] David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible Graph Connectivity. *Mathematical Programming*, 192:409–441, 2022. doi:10.1007/s10107-021-01664-9.

[2] Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995. arXiv:https://doi.org/10.1137/S0097539792236237, doi:10.1137/S0097539792236237.

[3] Ishan Bansal. A global analysis of the primal-dual method for edge augmentation problems. In *Integer Programming and Combinatorial Optimization - 26th International Conference, IPCO*, volume 15620 of *Lecture Notes in Computer Science*, pages 58–71. Springer, 2025. doi:10.1007/978-3-031-93112-3_5.

[4] Ishan Bansal, Joseph Cheriyan, Logan Grout, and Sharat Ibrahimpur. Improved approximation algorithms by generalizing the primal-dual method beyond uncrossable functions. *Algorithmica*, 86(8):2575–2604, 2024. doi:10.1007/s00453-024-01235-2.

[5] Ishan Bansal, Joseph Cheriyan, Sanjeev Khanna, and Miles Simmons. Improved approximation algorithms for flexible graph connectivity and capacitated network design. 2024. URL: https://arxiv.org/abs/2411.18809, arXiv:2411.18809.

[6] Ishan Bansal, Joseph Cheriyan, Sanjeev Khanna, and Miles Simmons. Improved approximation algorithms for capacitated network design and flexible graph connectivity. In *52nd International Colloquium on Automata, Languages, and Programming, ICALP*, volume 334 of *LIPIcs*, pages 20:1–20:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.ICALP.2025.20, doi:10.4230/LIPICS.ICALP.2025.20.

[7] Sylvia C. Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. Approximation algorithms for flexible graph connectivity. *Mathematical Programming*, 2023. doi:10.1007/s10107-023-01961-5.

[8] Gunnar Brinkmann, Kris Coolsaet, Jan Goedgebeur, and Hadrien Mélot. House of graphs: A database of interesting graphs. *Discrete Applied Mathematics*, 161(1–2):311–314, January 2013. URL: http://dx.doi.org/10.1016/j.dam.2012.07.018, doi:10.1016/j.dam.2012.07.018.

[9] Robert D. Carr, Lisa K. Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, page 106–115, USA, 2000. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=338219.338241.

[10] Chandra Chekuri and Rhea Jain. Approximation algorithms for network design in non-uniform fault models. *CoRR*, abs/2403.15547, 2024. URL: https://doi.org/10.48550/arXiv.2403.15547, arXiv:2403.15547, doi:10.48550/ARXIV.2403.15547.

[11] Reinhard Diestel. Abstract separation systems. *Order*, 35(1):157–170, April 2017. URL: http://dx.doi.org/10.1007/s11083-017-9424-5, doi:10.1007/s11083-017-9424-5.

[12] Reinhard Diestel. *Graph Theory*. Springer Berlin, Heidelberg, 6th edition, 2017. URL: https://doi.org/10.1007/978-3-662-53622-3.

[13] Reinhard Diestel, Philipp Eberenz, and Joshua Erde. Duality theorems for blocks and tangles in graphs. *SIAM Journal on Discrete Mathematics*, 31(3):1514–1528, 2017. doi:10.1137/16M1077763.

[14] Reinhard Diestel, Joshua Erde, and Daniel Weißauer. Structural submodularity and tangles in abstract separation systems. *Journal of Combinatorial Theory, Series A*, 167:155–180, 2019. URL: https://doi.org/10.1016/j.jcta.2019.05.001.

[15] Reinhard Diestel and Sang-il Oum. Tangle-tree duality: In graphs, matroids and beyond. *Combinatorica*, 39(4):879–910, August 2019. doi:10.1007/s00493-019-3798-5.

[16] Christian Elbracht, Jay Lilian Kneip, and Maximilian Teegen. The structure of submodular separation systems, 2021. URL: https://arxiv.org/abs/2103.13162, arXiv:2103.13162.

[17] Michel X. Goemans, Andrew V. Goldberg, Serge A. Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th Symposium on Discrete Algorithms*, pages 223–232, 1994.

[18] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

[19] M Hariprasad. Determinant of binary circulant matrices. *Spec. Matrices*, 7:92–94, 2019. doi:110.1515/spma-2019-0008.

[20] Sharat Ibrahimpur and László A. Végh. An o(log n)-approximation algorithm for (p, q)-flexible graph connectivity via independent rounding. In *Integer Programming and Combinatorial Optimization - 26th International Conference, IPCO 2025, Baltimore, MD, USA, June 11-13, 2025, Proceedings*, volume 15620 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2025. doi:10.1007/978-3-031-93112-3_23.

[21] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. doi:10.1007/s004930170004.

[22] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, 1996. doi:10.1145/234533.234534.

[23] Hiroshi Nagamochi, Kazuhiro Nishimura, and Toshihide Ibaraki. Computing all small cuts in an undirected network. *SIAM Journal on Discrete Mathematics*, 10(3):469–481, 1997. doi:10.1137/S0895480194271323.

[24] Zeev Nutov. Tight analysis of the primal-dual method for edge-covering pliable set families. In *50th International Symposium on Mathematical Foundations of Computer Science, MFCS 2025, August 25-29, 2025, Warsaw, Poland*, volume 345 of *LIPIcs*, pages 82:1–82:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.MFCS.2025.82, doi:10.4230/LIPICS.MFCS.2025.82.

[25] Miles Simmons, Ishan Bansal, and Joseph Cheriyan. A bad example for Jain's iterative rounding theorem for the cover small cuts problem. 2025. URL: https://arxiv.org/abs/2504.13105, arXiv:2504.13105.

[26] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *J. ACM*, 44(4):585–591, July 1997. doi:10.1145/263867.263872.

[27] Wikipedia. Circulant matrix - Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Circulant%20matrix&oldid=1297141144, 2025.

[28] David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15(3):435–454, 1995. doi:10.1007/BF01299747.

[29] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, Cambridge, UK, 2011.

Index

active set, 5

Cap-k-ECSS, 6
corner cuts, 9
Cover Bounded Cuts, 55
Cover Large Cuts, 53
Cover Small Cuts, 3
crossing density, 42
cut, 1
cut covering problem, 2

f-connectivity problem, 2
Flexible Graph Connectivity, 7

Intersection of Small Cuts, 57

k-Cover Small Cuts, 56
knapsack-cover inequalities, 60

laminar, 9

pliable, 7
primal-dual ratio, 5
Property γ, 32
Property γ^*, 42

residual family, 5

Semi-Hard Flexible Graph Connectivity, 68
Soft Flexible Graph Connectivity, 66
sparse crossing property, 33
strongly pliable, 8, 31

uncrossable, 2

weakly supermodular, 3
WGMV, 4, 73
witness set, 9