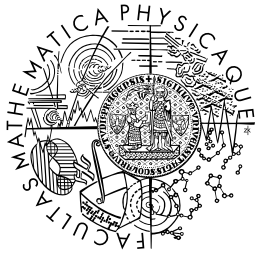


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Tomáš Turek

Connected cuts

Department of Applied Mathematics

Supervisor of the master thesis: doc. Mgr. Petr Kolman, Ph.D.

Study programme: Computer Science - Discrete Models and Algorithms

Prague 2025

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I dedicate this work first and foremost to my family, who has provided unwavering support throughout my entire academic journey and serves as a constant source of inspiration in countless ways.

I also extend my dedication to my supervisor, Petr Kolman, who introduced me to numerous fascinating topics and offered invaluable assistance whenever I needed it.

Finally, I dedicate this work to my friends and peers, whose ability to lift my spirits never fails and whose company has consistently brought joy to my life.

Title: Connected cuts

Author: Tomáš Turek

Department: Department of Applied Mathematics

Supervisor: doc. Mgr. Petr Kolman, Ph.D., Department of Applied Mathematics

Abstract: Cut problems in graph theory are extensively studied and well understood. However, cuts that partition graphs into components that are connected in a prescribed way, have received comparatively less attention. In this thesis, we formally define several connected cut problems, focusing on Minimum k-connected cut and Minimum k-connected cut with source problems. Given a graph $G = (V, E)$, and an interger k, the goal is to partition a vertex set V into two subsets, S and $V \setminus S$, such that the induced subgraph $G[S]$ is connected, the size of S is exactly k, and the number of edges between S and $V \setminus S$ is minimized. In the second problem, we are in addition given a source vertex $s \in V$ that has to be included in S. Our first result is a proof that both Minimum k-connected cut and Minimum k-connected cut with source problems are NP-complete. Additionally, we propose a bicriteria approximation algorithm for the Minimum k-connected cut with source problem. As a result, we also obtain a bicriteria approximation for the general Minimum k-connected cut problem.

Keywords: graph theory, approximation algorithms, cuts, linear programming, heuristics

Název práce: Souvislé řezy

Autor: Tomáš Turek

Katedra: Katedra aplikované matematiky

Vedoucí diplomové práce: doc. Mgr. Petr Kolman, Ph.D., Katedra aplikované matematiky

Abstrakt: Pojem řezu v teorii grafů je intenzivně studovaný. Avšak řezy, které rozdělují graf na komponenty souvislé předem definovaným způsobem, zatím mnoho pozornosti nezískaly. V této práci formálně zavedeme několik problémů tzv. souvislých řezů. Jde především o problémy Minimálního k-souvislého řezu a Minimálního k-souvislého řezu se zdrojem. V prvním případě je cílem rozdělit množinu vrcholů V na dvě podmnožiny S a $V \setminus S$ tak, že indukovaný podgraf $G[S]$ je souvislý, velikost množiny S je k, a počet hran mezi S a $V \setminus S$ je minimální. Ve druhém případě je dán zvláštní zdrojový vrchol, který musí být zahrnut do množiny S. Naším prvním výsledkem je důkaz, že oba problémy Minimálního k-souvislého řezu a Minimálního k-souvislého řezu se zdrojem jsou NP-úplné. Dále jsme vytvořili bikriteriální aproximační algoritmus pro problém Minimálního k-souvislého řezu se zdrojem. V důsledku toho také získáváme bikriteriální aproximaci pro obecný problém Minimálního k-souvislého řezu.

Klíčová slova: teorie grafů, aproximační algoritmy, řezy, lineární programování, heuristiky

Contents

Introduction

In graph theory, the concepts of cuts and connectivity are extensively studied. However, the problem of finding a cut, which partition the graph into components that are connected in a prescribed way, has received comparatively little attention. For some readers, the term connected cut might seem contradictory. One way to understand this problem is to think of disconnecting a given graph in a controlled manner meaning we want to separate the graph without breaking it into too many connected components.

The applications can be found in *divide-and-conquer* algorithms, which partition the input in a controlled way and proceed on the smaller parts recursively. In our case the input would be a graph.

This thesis focuses on such problems. We review existing results and introduce a new problems where the goal is to find a cut that preserves the connectivity of the resulting component(s). In particular, we study the MINIMUM k-CONNECTED CUT problem. For this problem, we prove NP-completeness and propose a integer linear programming formulation for it. Additionally, we demonstrate how a linear relaxation of this program can be used to develop a bicriteria approximation algorithm that achieves a constant-factor approximation of the minimum cut, while allowing some violation in the size of the chosen subset S.

The structure of the thesis is as follows: Chapter 1 reviews the elementary theory and notation, and formally states the connected cut problems. Chapter 2 surveys results from the literature. Chapter 3 contains the proofs of NP-completeness for the MINIMUM k-CONNECTED CUT WITH SOURCE and the MINIMUM k-CONNECTED CUT problems. In Chapter 4, we introduce the integer linear programming formulation and describe the approximation algorithm. Finally, Chapter 5 discusses the implementation details and presents results of several experiments.

To enhance the clarity and readability of certain sections, AI-based language tools, namely Perplexity AI and Claude AI, were employed to assist in rephrasing and smoothing the text. All technical content and original meanings were carefully preserved throughout this process.

1 Preliminaries

In this chapter we introduce the theory and also state all the connected cut problems. We follow a common notation, but for clarity we list the key terms.

1.1 Notation

For the readers which are not well established in graph theory we point them to the well written book by Nešetřil and Matoušek [19] which introduce all the graph theory we work with.

In the text we mention parametrized algorithms and the term *FPT*, which is an abbreviation for *fixed-parameter tractable*, and a graph parameter *treewidth*. To learn more about such topics we advice the reader to check the book by Cygan et al. [5].

We also use *linear* and *integer programming*. Readers not familiar with such topics are advised to look into these books [23, 22, 16, 3].

We will always work with graph denoted as $G = (V, E)$, where V stands for the set of vertices and E for the set of edges. The sizes will be denoted as $n = |V|$ and $m = |E|$. We consider undirected graphs unless stated otherwise, and we assume that the graphs are connected. An undirected edge, with endpoints $u, v \in V$, is denoted as $\{u, v\}$.

For the set $\{1, 2, \ldots, \ell\}$ we use an abbreviation $[\ell]$ and, furthermore, we extend such notation to the set $\{k, k+1, \ldots, \ell\}$, where $k \leq \ell$, which is denoted as $[k, \ell]$.

For real numbers $\ell, r \in \mathbb{R}$, where $\ell \leq r$, we denote the interval $\{x \in \mathbb{R} \mid \ell \leq x \leq r\}$ as $\langle \ell, r \rangle$.

The *partition* of vertices V into $\ell \in \mathbb{N}$ parts is a set $\{V_1, V_2, \ldots, V_\ell\}$ such that $\bigcup_{i=0}^{\ell} V_i = V$ and $\forall i \neq j \in [\ell] : V_i \cap V_j = \emptyset$. The partition itself is denoted as $\mathcal{V}$.

A graph is *connected* if and only if all pairs of vertices have a path between them.

An *induced subgraph* on vertices $S \subseteq V$ is a graph $G' = (S, E')$ where $E' \subseteq E$ and $\{u, v\} \in E' \iff u \in S \wedge v \in S$. We denote such graph as $G[S]$. We say that $S \subseteq V$ is connected if the induced subgraph $G[S]$ is connected.

For a subset $S \subseteq V$, $E(S, V \setminus S)$ is the set of edges having one end point in S and the other in $V \setminus S$. The size of such set will be denoted as $e(S, V \setminus S)$.

A *cut* is a partition of vertices into sets $S \subset V$ and $V \setminus S$. In the text we also say that the set of edges $E(S, V \setminus S)$ is a cut. Easily seen from the definitions that the terms are interchangeable, hence we do not make any significant difference between them. We will assure that the term is always clear from the context.

1.2 Connected cut problems

In this section, we provide precise definitions for several connected cut problems. These definitions extend well-known cut problems by incorporating additional connectivity constraints

Firstly we use the problems were we want to partition the graph only to two partitions.

1.2.1 Single commodity connected cuts

Similarly to single commodity cuts and flows in graph we present cuts, which also have these properties, and on top of that we restrict the solution to be connected.

Definition 1 (Connected cut)**.** *For a connected graph $G = (V, E)$ a connected cut is $S \subset V$, for which $G[S]$ is connected.*

Problem 1 (Minimum connected cut)**.** *Minimum connected cut is to find a connected cut minimizing $e(S, V \setminus S)$.*

This problem is the least restricted. We create another problem which are extended from this one.

Definition 2 (Connected s-cut)**.** *For a connected graph $G = (V, E)$ and given vertex $s \in V$ a connected s-cut is $S \subset V$, for which $G[S]$ is connected and $s \in S$.*

Problem 2 (Minimum connected s-cut)**.** *Minimum connected s-cut is to find a connected s-cut minimizing $e(S, V \setminus S)$.*

See that by solving this problem for all sources $s \in V$ we would also solve the previous Problem 1. One can already know that commonly used cut is defined for two distinct vertices — source $s \in V$ and target $t \in V$. We could also use it in our case and only extend the previous definition by saying that $t \notin S$.

Definition 3 (Connected $s - t$ cut)**.** *For a connected graph $G = (V, E)$ and given two distinct vertices $s, t \in V$ a connected $s - t$ cut is $S \subset V$, for which all following properties hold.*

1. *$G[S]$ is connected.*
2. *$s \in S$ and $t \notin S$.*

Problem 3 (Minimum connected $s - t$ cut)**.** *Minimum connected $s - t$ cut is to find a connected $s - t$ cut minimizing $e(S, V \setminus S)$.*

For all above stated problems we may also require to have $G[V \setminus S]$ connected. Hence all of the previous problem could be extended with such constraint resulting in problems (Minimum) Biconnected cut, (Minimum) Biconnected s-cut and (Minimum) Biconnected $s - t$ cut respectively. Note that Minimum biconnected cut and Minimum biconnected s-cut are same, because if $s \notin S$ we set our new $S' \coloneqq V \setminus S$.

In the next Chapter 2 we discuss, how these problems could be solved by known algorithms.

Let us now turn our attention to more complex problems known as *multi-commodity cuts* and *flows*, which involve partitioning graphs into more than two sets.

1.2.2 Multi-commodity connected cuts

In this section we state connected cut problems which are extended from multi-commodity cuts.

Definition 4 (Multi-way connected cut with sources)**.** *For a connected graph $G = (V, E)$ and pairwise distinct vertices $s_1, s_2, \ldots, s_k \in V$, for $k \in \mathbb{N}$, a multi-way connected cut with sources is a partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ of vertices such that the following holds.*

1. *$\forall i \in [k] : s_i \in V_i$.*
2. *$\forall i \in [k] : G[V_i]$ is connected.*

Problem 4 (Min-sum multi-way connected cut with sources)**.** *A min-sum multi-way connected cut with sources is to find multi-way connected cut minimizing the sum $\sum_{i<j} e(V_i, V_j)$.*

Problem 5 (Min-max multi-way connected cut with sources)**.** *A min-max multi-way connected cut with sources is to find multi-way connected cut minimizing $\max_{i\in[k]} e(V_i, V \setminus V_i)$.*

Typically the problem of minimizing sum (in our case Problem 4) is usually not as difficult as to minimize the maximum (Problem 5).

Definition 5 (Flexible multi-way connected cut)**.** *For a connected graph $G = (V, E)$ and pairwise distinct vertices $s_1, s_2, \ldots, s_k \in V$, for $k \in \mathbb{N}$, the partition $\mathcal{V} = \{V_1, V_2, \ldots, V_\ell\}$, for $1 < \ell \leq k$, is a flexible multi-way connected cut if $\exists\{s_{i_1}, s_{i_2}, \ldots, s_{i_\ell}\} \subseteq \{s_1, s_2, \ldots, s_k\}$ such that $\mathcal{V}$ is a multi-way connected cut with sources $s_{i_1}, s_{i_2}, \ldots, s_{i_\ell}$.*

Problem 6 (Min-sum flexible multi-way connected cut)**.** *A min-sum flexible multi-way connected cut is to find flexible multi-way connected cut minimizing the sum $\sum_{i<j} e(V_i, V_j)$.*

Problem 7 (Min-max flexible multi-way connected cut)**.** *A min-max flexible multi-way connected cut with sources is to find flexible multi-way connected cut minimizing $\max_{i\in[k]} e(V_i, V \setminus V_i)$.*

Now we present another problem which is similar to the previously mentioned one, the only change is that we do not have predefined sources.

Definition 6 (Multi-way connected cut)**.** *For a connected graph $G = (V, E)$ and $k \in \mathbb{N}$ a multi-way connected cut is partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ of vertices such that $\forall i \in [k] : G[V_i]$ is connected and $\forall i \in [k] : V_i \neq \emptyset$.*

Problem 8 (Min-sum multi-way connected cut)**.** *Min-sum multi-way connected cut is to find a multi-way connected cut minimizing $\sum_{i<j} e(V_i, V_j)$.*

Problem 9 (Min-max multi-way connected cut)**.** *Min-max multi-way connected cut is to find a multi-way connected cut minimizing $\max_{i\neq j} e(V_i, V \setminus V_j)$.*

1.2.3 k-connected cut

We can even further increase the number of requirements. In this case to the size of $|S|$.

Definition 7 (k-CONNECTED CUT)**.** *For a connected graph $G = (V, E)$ and $k \in \mathbb{N}$ a k-connected cut is $S \subset V$ such that all following properties hold.*

1. *$G[S]$ is connected.*
2. *$|S| = k$.*

Together with the objective function we obtain the following problem.

Problem 10 (MINIMUM k-CONENCTED CUT)**.** *Minimum k-connected cut is to find k-connected cut minimizing $e(S, V \setminus S)$.*

Again we can define a problem, where we are given a dedicated source vertex.

Definition 8 (k-CONNECTED CUT WITH SOURCE)**.** *For a connected graph $G = (V, E)$, a vertex $s \in V$ and $k \in \mathbb{N}$ a k-connected cut with sources is $S \subset V$ such that all following properties hold.*

1. *$G[S]$ is connected.*
2. *$|S| = k$.*
3. *$s \in S$.*

Problem 11 (MINIMUM k-CONENCTED CUT WITH SOURCE)**.** *Minimum k-connected cut with source is to find k-connected cut with source minimizing $e(S, V \setminus S)$.*

From the algorithmic perspective, we see, that if we solve Problem 11, then by running the algorithm $|V|$ times we get a solution for the original Problem 10.

2 Survey on connected cuts

In this chapter we review which connected cut problems have been studied, and we mention the results already found in this field.

2.1 Minimum connected cut

A well known problem is to find a minimum cut of a graph $G = (V, E)$, which splits vertices into two sets S and $V \setminus S$, where we also have two specified distinct vertices - the source $s \in V$ and the target $t \in V$. The constraints are that $s \in S$ and $t \notin S$ so that we disconnect these two vertices. One of the well studied algorithms which solves this problem is to find a maximum flow from s to t and then find all edges which have flow equal to its capacity. The details can be found in the following book [1] and paper [9].

Note that any optimal solution preserves the connectivity of both parts $G[S]$ and $G[V \setminus S]$. Consider the case where we have disconnected part S into multiple connected components. Because we have a connected graph there must be an edge from every connected component to the rest $V \setminus S$, therefore if we put the connected component not having s to $V \setminus S$ we decrease the size of the cut. For the other case when we would have more connected components in the part $V \setminus S$ we would proceed with similar argument. We would like to emphasize the connectivity property of G, because otherwise these arguments would not hold.

We mention a polyhedral descriptions stated by Garg [12] for the problem of $s - t$ cut, where the vertices of polyhedron correspond to solutions having s-part connected, t-part connected and both parts connected. So that if we build an algorithm using such description, we obtain a solution where the chosen part(s) stay connected.

In this section we mention the problem of general *minimum cut*, that is we do not have neither source nor target vertex and we want to minimize the cut which disconnects the graph. This problem can be solved by Karger's-Stein probabilistic algorithm [14] which uses edge contractions. Here we can clearly see that the contraction operations preserves connectivity for both parts. Alternatively this problem could be solved by running the algorithm for $s - t$ cut problem multiple times, more precisely for each pair $(s, t) \in \binom{V}{2}$.

By using our definitions we can already solve all these problems: Minimum connected cut, Minimum connected s-cut, Minimum connected $s - t$ cut,Minimum biconnected cut, Minimum biconnected s-cut and Minimum biconnected $s - t$ cut.

2.2 Maximum connected cut

In some sense the opposite problem to the first one is to find a subset $S \subset V$ for which $e(S, V \setminus S)$ is maximized, this is called Maximum cut problem. Even without connectivity constraints this problem is known to be NP-complete as shown by Karp [15]. On the other hand there exists an approximation algorithm [13] with the factor 0.87856 using semidefinite programming.

There are two natural expansions for connectivity problems. Either we require to have only $G[S]$ connected or both parts $G[S]$ and $G[V \setminus S]$ connected. These two problems were studied by Duarte et al. [6] and here they called them Maximum Connected Cut and Largest Bond respectively. They have shown that Largest Bond does not admit a constant-factor approximation algorithm unless P = NP. They have shown that both problems are NP-hard even for planar bipartite graphs, whereas the Maximum Cut on such instance has a trivial solution which is given by the two parts. Additionally, they demonstrated that both problems are FPT when parameterized by either the solution size or the treewidth of the input graph.

2.3 ρ-separators and k-edge separator

Consider a graph $G = (V, E)$ with edge capacities and vertex weights and a parameter $0 < \rho < 1$. A *ρ-separator* is defined as a subset of edges whose removal partitions the vertex set into connected components such that the sum of the vertex weights in each component is at most ρ times the weight of the whole graph. Even et al. [7] defined this problem and presented an $\mathcal{O}(\log n)$-approximation algorithm for minimum capacity ρ-separator. The main technique, which was used, is called *spreading metric* and then formulation by an *integer program*. In the same paper, they furthermore generalized the problem into a problem called *simultaneous separators* where the subsets $U_1, U_2, \ldots, U_s$ of vertices are given together with parameter $\rho_1, \rho_2, \ldots, \rho_s$. The goal is to find a minimum capacity cut that partitions the vertex set into connected components such that each connected component contains at most ρ_i of the weight of subset U_i for all i.

A similar problem is called *k-edge separator*, where the goal is to remove minimum number of edges such that each of the connected components has at most k vertices. Lee [18] studied this problem and presented an $\mathcal{O}(\log k)$-approximation algorithm running in time $n^{\mathcal{O}(1)}$.

Both these problems are related to the connected cuts but in a different matter. The goal is to have small connected components, hence the connectivity is not the main goal. In both cases the number of components is not restricted which is the main difference to our defined problems.

2.4 Graph partitioning and clustering

Yet another related problems are graph *partitioning* and *clustering*. In the former problem we partition the vertices into (connected) components while preserving some constraints and optionally optimizing an objective function over such partition, for further information we refer to Buluç et al. [2]. The latter problem is to decompose the given graph so that we end up with clusters (induced subgraphs). Graph clustering is often used in *social networking* [20]. Both these problem classes are well studied and in some cases the connectivity of the components is implicitly satisfied. But we focus on those results which explicitly require connectivity.

Cordero et al. [4] studies the problem where for a given graph $G = (V, E)$, cost function $d : E \to \mathbb{R}^+$ and two integers $k \geq 2, \alpha \geq 2$ we want to find partition

into k parts where every part induces a connected component. Moreover each subset has to have at least α vertices. The objective is to minimize the total sum of edge capacities *within* every subset. They propose a mixed integer program to solve the problem. The program uses a flow to model the partitions, which is a common solving technique. Additionally, they propose a different approach using *spectral clustering* which yields a formulation of another mixed integer program. Their results are about the mixed integer program feasibility, and then they present experiment results, which were done on several graphs, for both proposed algorithms.

We observe the similarity with Problem 8, as the only difference lies in the objective function. Therefore, a promising avenue for investigation would be to modify the objective function in their mixed integer programming formulation to minimize the size of edge cuts. This approach could potentially yield an algorithm that solves Problem 8 and might even lead to the development of an approximation algorithm.

2.5 Expander decomposition problem

Let us have a graph $G = (V, E)$. For $v \in V$ we define $\deg(v)$ as the vertex *degree*, i.e. the number of incident edges. For $S \subseteq V$ the *volume* is defined and denoted as $\mathrm{vol}(S) = \sum_{v \in S} \deg(v)$. Then the *conductance* of a cut $S \subseteq V$ is $\Phi_G(S) = \frac{e(S, V \setminus S)}{\min(\mathrm{vol}(S), \mathrm{vol}(V \setminus S))}$, and the *conductance* of a graph G is $\Phi_G = \min_{S \subset V} \Phi_G(S)$. We say a graph G is a ϕ *expander* if $\Phi_G \geq \phi$, and we call partition $\{V_1, \ldots, V_k\}$ of V a ϕ *expander decomposition* if $\min_i \Phi_{G[Vi]} \geq \phi$. Samurak and Wang [21] described a probabilistic algorithm with parameter $\phi \geq 1/\log^{\mathcal{O}(1)} m$ that constructs a ϕ expander decomposition, where $\sum_i e(S_i, V \setminus S_i) = \mathcal{O}(\phi m \log^3 m)$.

From the definition of conductance, the optimal value ensures all partitions remain highly interconnected. However, the resulting cut size can be quite large — potentially a constant fraction of m. Consequently, this problem-solving technique is not applicable to our connected cut problems. While connectivity emerges naturally from the definition, it is not explicitly required as a constraint.

2.6 Partial Minimum k-connected cut

Recall that in Minimum k-connected cut problem we have three following requirements.

1. $G[S]$ is connected.
2. $|S| = k$.
3. The size $e(S, V \setminus S)$ is minimized.

Let us consider what happens when we omit one of the three requirements. We can demonstrate that for each case, there exists a known algorithm that either solve or approximate the resulting problem.

First, when requiring only conditions 1 and 3, we can apply the results from Garg [12]. Additionally, we have examined several alternative approaches in section 2.1.

Second, when requiring only conditions 2 and 3, we can employ the approximation algorithm for solving minimal bisection proposed by Feige and Krauthgamer [8]. Furthermore, they presented an extension in section 5.4, where for a given $k \leq 1/2$, they developed an algorithm for computing the minimum bisection with $|S| = k$, achieving an $\mathcal{O}(\log^2 n)$ approximation ratio on the cut size.

Finally, when selecting only conditions 1 and 2, we can utilize standard graph traversal algorithms such as *breadth-first* or *depth-first search*, as the cut size minimization is not required.

3 NP-completeness of Minimum k-conencted cut (with source)

An important question is whether the MINIMUM k-CONNECTED CUT problem is NP-complete. In such case we cannot expect any polynomial time algorithm which solves it, unless P = NP. In this chapter we show the reduction from MINIMUM BISECTION problem for both MINIMUM k-CONNECTED CUT and MINIMUM k-CONNECTED CUT WITH SOURCE problems. First, we recall the definition of MINIMUM BISECTION problem.

Problem 12 (MINIMUM BISECTION)**.** *Given graph $G = (V, E)$ the goal is to find a partition $\mathcal{V} = \{V_1, V_2\}$ where $|V_1| = |V_2| = \frac{1}{2}|V|$ and $e(V_1, V_2)$ is minimized.*

To see that the Problem 12 is NP-complete we refer to the sources [10, 11, 17].

3.1 Minimum k-connected cut with source

First, we deal the MINIMUM k-CONNECTED CUT WITH SOURCE problem which has a straightforward reduction from MINIMUM BISECTION.

Lemma 1. *The MINIMUM k-CONNECTED CUT WITH SOURCE problem is* NP*-complete.*

Proof. Lets have a graph $G = (V, E)$ for the MINIMUM BISECTION problem. Assume $s \notin V$. We create a graph $G' = (V \cup \{s\}, E \cup \{\{u, s\} \mid u \in V\})$ and set $k = \frac{n}{2} + 1$. Now we use the solver for the minimum k-connected cut with source s.

Since s has to be included in the solution we look for $\frac{n}{2}$ vertices from V which will be also added into the solution S. The number of edges adjacent to s always contributes to the cut size with the value $\frac{n}{2}$. Therefore we minimize the cut only within the original graph G. Since s is connected to all vertices we always have solution which induces a connected subgraph. Thus the solution to MINIMUM BISECTION problem is $S \setminus \{s\}$.

See that this reduction runs in linear time with respect to $|V|$, hence in polynomial time. □

3.2 Minimum k-connected cut

Proving the NP-completeness of the MINIMUM k-CONNECTED CUT problem requires more careful consideration. Simply applying the same reduction as for the MINIMUM k-CONNECTED CUT WITH SOURCE problem is insufficient, as it doesn't guarantee that the source s would be included in our solution. Below, we present a proof that establish the NP-completeness of this problem

Theorem 2. *The MINIMUM k-CONNECTED CUT problem is* NP*-complete.*

First proof. On input we have a graph $G = (V, E)$ for a minimum bisection problem. Assume n is even. We add $H \coloneqq K_{n^2}$ complete graph and arbitrarily

choose n vertices from H into a set B. Now create a perfect matching with V and B arbitrarily. Set $k = n^2 + \frac{n}{2}$.

Now we run MINIMUM k-CONNECTED CUT on such instance. Observe that in every possible solution we choose ℓ vertices from H, where $\ell \in I \coloneqq [n^2 - \frac{n}{2}, n^2]$.

If we choose the whole subgraph H then we obtain minimum bisection. If not then the cut solely within H will be of size $\ell(n^2 - \ell)$, which is always greater than $e(S, V \setminus S)$ for any $S \subset V$. □

Second proof. Lets have $G = (V, E)$ on the input for minimum bisection problem, assume that $V = \{v_1, v_2, \ldots, v_n\}$. Lets create an auxiliary graph $H = (W, F)$, where

$$W = s \cup \bigcup_{i=1}^{n} \{v_j^i | \forall j \in [n+1]\}$$

and

$$F = \bigcup_{i=1}^{n} \{\{v_j^i, v_{j'}^i\} | \forall j \neq j' \in [n+2]\}\} \cup \{\{v_{n+1}^i, s\} | i \in [n]\} \cup E'$$

Here $\{v_j^i, v_i^j\} \in E' \iff \{v_i, v_j\} \in E$. The description of our created graph is that we added new source vertex s and every original vertex $v_i \in V$ is replaced by a clique of size $n+1$. The edges between the original vertices are simulated by choosing one vertex from each clique as its representative for such connection. Lastly we also add edges from the source.

We take the graph H, set $k = 1 + \frac{n}{2}(n+1)$ and solve the minimum k-connected cut. If the solution will take exactly $\frac{n}{2}$ cliques and s we have optimal value for minimal bisection by choosing vertices represented by chosen cliques. Observe that the whole procedure can be easily done in polynomial time.

Observe that the decision on choosing one vertex from a clique may result into at most one edge going in between cliques. But there may be more edges inside such clique.

Now suppose we have different solution, which is optimal. Firstly we relax the problem and allow the solution S to be of size in $[k-n, k+n]$ and that $G[S]$ does not have to be connected.

Now we can augment the solution so that all cliques are fully contained in S. Both relaxed properties allow us to always either remove the vertices or add them. For each clique we decrease the cut by $l_i(n+1-l_i)$ and increase by at most l_i or $n+1-l_i$ and since the total increase is at most $l_i^2 - nl_i$ or $l_i^2 - (n+2)l_i + n + 1$ respectively. We only care about $l_i \in [1, n]$ and $n \geq 1$. In these ranges the function is ≤ 0.

Hence we have a solution which has all cliques fully contained, which also results in the fact that we have $\frac{n}{2}$ cliques. Otherwise we would violate our range for S. Therefore we may add s if it was not already in the solution, this does not increase the cut.

Now observe that we have $1 + \frac{n}{2}(n+1)$ vertices in S which has only better optimum. Since $s \in S$ and all cliques are fully contained, i.e. v_{n+1}^i for all such cliques is also included then the subgraph $G[S]$ is also connected. Meaning we have better solution for minimum k-connected cut.

Note that all steps were not increasing. But in case of adding cliques we may order these additions and removals so that at least one step will be strictly better. □

4 Bicriteria approximation algorithm for k-connected cut

In this chapter we present a bicriteria approximation algorithm which uses a new notion of *absorptive flow* from which we create an integer linear program, abbreviated by ILP. We approximate the size of cut with a constant factor, whilst the size of S is within bounds $1 \leq |S| \leq \mathcal{O}(k)$.

Besides these result we also show the lower bound for integrality gap between the integer linear program and its linear relaxation.

4.1 Absorptive flow

We define a new notion of *absorptive flow*. First, we state the definition in a common sense.

For a graph $G = (V, E)$ and a source vertex $s \in V$ we find a flow which flows throughout the graph and every time it goes through a vertex some portion of the flow may get absorbed. The absorption marks our desired vertices.

Later we define a boundary induced by such flow, and state the integer linear program.

Definition 9 (Absorptive flow)**.** *For a graph $G = (V, E)$ and a vertex $s \in V$, also called the source, and for $k \in \mathbb{N}$, such that $|V| \geq k$, we define absorptive flow as functions (f_V, f_E), where $f_V : V \to \mathbb{R}$ and $f_E : E \to \mathbb{R}$ satisfying the following properties.*

1. $\sum_{v \in V} f_V(v) = k$
2. $f_V(s) = 1$
3. $\forall v \in V : 0 \leq f_V(v) \leq 1$
4. $\sum_{v \in V, \{s,v\} \in E} f_E(\{s, v\}) = k - 1$
5. $\forall e \in E : 0 \leq f_E(e)$
6. $\forall v \in V \setminus \{s\} : \sum_{u \in V, \{u,v\} \in E} f_E(\{u, v\}) = \sum_{u \in V, \{v,u\} \in E} f_E(\{v, u\}) + f_V(v)$

Property 1 ensures the absorption of the entire flow. Property 2 establishes that the source s always serves as an absorber. Properties 3 and 5 define bounds on the functions. Property 4 indicates that flow originates from the source. Finally, Property 6 represents Kirchhoff's law, which states that flow continues uninterrupted *unless* a portion is absorbed.

We call function f_E a *flow* and f_V an *absorption*. Reader can view the flow in a dynamic approach. That is we have source s and multiple targets which are dynamically chosen by their absorptions.

It is evident that this formulation defines a linear program. However, we still need to address the connectivity requirement. To accomplish this, we will augment the existing flow with additional inequalities that enforce connectivity. Let us now define a boundary.

Definition 10 (Boundary of the absorptive flow)**.** *Given an absorptive flow* (f_V, f_E) *on a graph* $G = (V, E)$ *we set* $S = \{v \in V \mid f_V(v) > 0\}$*. The boundary is* $E(S, V \setminus S)$*, where its size is* $e(S, V \setminus S)$*.*

Now we propose our integer linear program which solves the MINIMUM k-CONNECTED CUT WITH SOURCE problem.

4.1.1 Integer linear program

In this section we establish the integer linear program, which uses absorptive flow and its boundary. We ensure connectivity of the subgraph $G[S]$, thus the term boundary can be interchanged with the term cut.

Variables

First, we declare the variables for edges and vertices.

$$f_v = \begin{cases} 1 & \text{if the vertex } v \text{ absorbs.} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

$f_{uv}, f_{vu} \geq 0$ the amount of flow on edge $\{u, v\}$ going in the respective direction. (4.2)

$$x_e = \begin{cases} 1 & \text{if } e \in E(S, V \setminus S). \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

See that these variables arise only from the definition of the problem. We call variables (4.1) as an *absorption*, (4.2) as *flow* and (4.3) as a *cut*.

The integer linear program

In (4.4) we propose the whole ILP formulation, which we denote as $\mathcal{M}$, and describe the constraints.

$$\min \sum_{e \in E} x_e \tag{4.4a}$$

$$x_{\{u,v\}} \geq f_u - f_v \qquad \forall \{u, v\} \in E \tag{4.4b}$$

$$x_{\{u,v\}} \geq f_v - f_u \qquad \forall \{u, v\} \in E \tag{4.4c}$$

$$\sum_{v \in V, \{s,v\} \in E} f_{sv} = k - 1 \tag{4.4d}$$

$$f_s = 1 \tag{4.4e}$$

$$\sum_{u \in V, \{u,v\} \in E} f_{uv} = \sum_{u \in V, \{v,u\} \in E} f_{vu} + f_v \qquad \forall v \in V \setminus \{s\} \tag{4.4f}$$

$$\sum_{u \in V} f_u = k \tag{4.4g}$$

$$(k-1) \cdot f_v \geq \sum_{u \in V, \{uv\} \in E} f_{uv} \qquad \forall v \in V \setminus \{s\} \tag{4.4h}$$

$$f_v \in \{0, 1\} \qquad \forall v \in V \tag{4.4i}$$

$$f_{uv}, f_{vu} \in \mathbb{Z}^+ \qquad \forall \{u, v\} \in E \tag{4.4j}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{4.4k}$$

The objective function (4.4a) stems from our goal to minimize the cut size. Inequalities (4.4b) and (4.4c) represent the constraint $x_{\{u,v\}} \geq |f_u - f_v|$, which captures the difference in absorbance. To ensure a proper flow, equality (4.4d) combined with (4.4e) establishes that k resources are distributed from the source s. The modified Kirchhoff's law appears in (4.4f), indicating that any flow entering a vertex either continues outward or is partially absorbed by that vertex. Inequality (4.4h) enforces connectivity by requiring that any vertex receiving non-zero flow must absorb a portion of it. Finally, inequalities (4.4i), (4.4j) and (4.4k) simply define the boundaries for all variables.

4.1.2 Consider the distance from source

We modify our integer linear program $\mathcal{M}$ by incorporating the distance from s. We'll use the notation $d(u, v)$ to represent the shortest path between vertices u and v. This is a well-established graph property that can be computed in polynomial time (e.g., using Dijkstra's algorithm). In our modification, we adjust the absorption requirement for each vertex by changing the fraction from $\frac{1}{k-1}$ to $\frac{1}{k-d(s,u)}$ for every vertex u. This approach combines two different path lengths: one based on standard distance and another based on flow. The resulting inequality is as follows.

$$(k - d(s, v)) \cdot f_v \geq \sum_{u \in V, \{uv\} \in E} f_{uv} \qquad \forall v \in V \setminus \{s\} \tag{4.5a}$$

We replace the inequality (4.4h) with more strict version (4.5a) and denote it as $\mathcal{M}_d$. This modification enforces greater absorption and implicitly excludes vertices that are farther away (i.e., at a distance greater than k) from consideration in the solution. As a result, this approach may expedite the solver's search for an optimal solution.

4.1.3 Properties

Let us now observe several properties of this integer linear program $\mathcal{M}_d$.

Observation 1. *Every vertex in the flow absorb.*

Proof. See that due to the constraint (4.5) whenever a flow goes inside the vertex we must set the vertex to absorb some portion of the flow. Exactly 1 in the case of integer program. □

Observation 2. *A set $S = \{v \in V \mid f_v = 1\}$ defines a connected induced subgraph $G[S]$.*

Proof. Since $f_s = 1$ we know that s is inside the S. For contradiction assume there is a vertex $v \in S$ which is not connected to the source s by a path, and is the closest one to s in terms of $d(s, v)$. Because $f_v = 1$, there must be a flow going to the vertex v from the constraint (4.4f). Since this flow starts at s and every vertex in the flow absorbs, from Observation 1, we must have a path from s to v. Hence no such v exist. □

Lemma 3. *The optimal solution (x^*, f^*) of $\mathcal{M}_d$ correspond to a* Minimum k-connected cut *and vice versa.*

Proof. We show the following.

1. For every feasible (x, f) of $\mathcal{M}_d$ there exists k-connected cut with the same value. – Suppose we have a feasible solution of the integer linear program. Then we define the cut as $S = \{v \in V \mid f_v = 1\}$. From the previous observations we know that the induced subgraph $G[S]$ is connected. And also from the constraint (4.4g) we have that $|S| = k$.

2. For every k-connected cut there exists a feasible solution (x, f) of our $\mathcal{M}_d$ with the same value. – Suppose we have S as a k-connected cut. Let us build a shortest path tree starting from s where we consider only the connected induced subgraph $G[S]$. For every $v \in S$ set $f_v = 1$ and $\forall u \notin S$ set $f_u = 0$. Set cut x_e to be 1 if and only if exactly one end is inside S. Finally the values f_{uv} will be defined by the shortest path tree. That is from s we send the flow with the size of the sub-tree and recursively call on that. See that it satisfy all constraints. Both (4.4b) and (4.4c) are satisfied purely from the definitions. The sum of outgoing flow from s is equal to the sum of all sub-trees which is indeed $k - 1$. There is k vertices with value $f_v = 1$ and finally $f_s = 1$. From the tree structure we can easily observed that the Kirchhoff's law (4.4f) is satisfied. And also the connected constraint (4.4h) holds.

Therefore the optimality follows. □

Lemma 4 (Integrality gap)**.** *The integrality gap between the integer linear program $\mathcal{M}_d$ and its linear relaxation is at least k.*

Proof. Suppose we have $k \in \mathbb{N}$ and a clique graph with $\mathbb{N} \ni n \geq k$ vertices. The optimal value of $\mathcal{M}_d$ will choose arbitrarily k vertices, because we cannot get better solution by exchanging one vertex for another. Hence we have $n-k$ vertices in the rest. The size of the cut is therefore $k(n-k)$. Conversely in the linear relaxation we can send from s the same value to all vertices, therefore only $n-1$ edges will have non-zero cut-value. For $n-1$ vertices we split $k-1$ equally, hence we have a cut of size $(n-1) \cdot \left(1 - \frac{k-1}{n-1}\right) = n - 1 - k + 1 = n - k$. The integrality gap is in this case k. □

4.2 Approximation of the linear relaxation

In this section we use our integer linear program $\mathcal{M}_d$ and create a linear relaxation. Let us denote d_x as a pseudo-metric on $V \times V \to \mathbb{R}_0^+$. For two vertices $u, v \in V$ we have $d_x(u, v)$ defined as the shortest path between these two vertices, where the length of each edge e is set to be the cut variable x_e. We denote our LP optimum as $\mathrm{OPT}_{\mathrm{LP}}$.

Definition 11. *For a given graph $G = (V, E)$ and a pseudo-metric $d : V \times V \to \mathbb{R}_0^+$, a ball around vertex $u \in V$ with radius $0 \leq r \leq 1$ is a set $\{v \in V \mid d(u, v) \leq r\}$, denoted as $\mathcal{B}_d(u, r)$.*

Lemma 5. *For all vertices $v \in V$ the inequality $f_v \geq 1 - d_x(s, v)$ holds.*

Proof. Let us prove this by an induction on the distance d_x. First, see that for s the inequality $f_s \geq 1 - 0 = 1$ holds due to the constraint (4.4e). Then for all neighbors of s we have $f_v \geq 1 - d_x(s, v) \geq 1 - x_{sv}$, and from the inequalities (4.4b) and (4.4c) we have that $f_v \geq 1 - f_s + f_v = 1 - 1 + f_v = f_v$. Thus it holds.

A vertex $v \in V$ is said to be *bad* if $f_v < 1 - d_x(s, v)$. For contradiction suppose there is a bad vertex v, and take the one closest to the source with respect to the pseudo-metric d_x. Hence for all neighbors which are closer it must hold from induction. Take the one, which has shortest path from source summed with the value of their common edge, denote this vertex u. Therefore $f_u \geq 1 - d_x(s, u)$. Now $1 - d_x(s, v) = 1 - (d_x(s, u) + x_{uv}) \leq 1 - d_x(s, u) - (f_u - f_v) = 1 - d_x(s, u) - f_u + f_v \leq f_v$. Which is a contradiction with the fact that v is bad. □

The probabilistic algorithm is in a following matter. Uniformly at random choose $0 \leq r \leq 1$ and set $S = \mathcal{B}_{d_x}(s, r)$. With such approach we get optimal expected values.

Lemma 6. *The expected values of the cut and the size of $\mathcal{B}_{d_x}(s, r)$ are upper bounded by the optimum values.*

Proof. First, see the probability $\mathsf{P}[e \in E(\mathcal{B}_{d_x}(s, r), V \setminus \mathcal{B}_{d_x}(s, r))] \leq x_e$. This can be seen by a simple observation. The edge e is in cut if r is inside the interval $\langle \alpha, \alpha + x_e \rangle$ for some $\alpha \in \mathbb{R}_0^+$. But the probability is at most as in the case of r being in the interval $\langle 0, x_e \rangle$. Therefore the expected value can be computed.

$$\begin{aligned} \mathsf{E}[e(\mathcal{B}_{d_x}(s, r), V \setminus \mathcal{B}_{d_x}(s, r))] &= \sum_{e \in E} \mathsf{P}[e \in E(\mathcal{B}_{d_x}(s, r), V \setminus \mathcal{B}_{d_x}(s, r))] \\ &\leq \sum_{e \in E} x_e = \mathrm{OPT}_{\mathrm{LP}} \end{aligned} \tag{4.6}$$

Now we compute the expected size of the set $\mathcal{B}_{d_x}(s, r)$.

$$\mathsf{P}[v \in \mathcal{B}_{d_x}(s, r)] = \mathsf{P}[d_x(s, v) \leq r] \leq \mathsf{P}[1 - f_v \leq r] = f_v \tag{4.7}$$

Observe that having r larger than $1 - f_v$ is same as the value of f_v itself, hence the last equality holds. The previous inequality follows from the Lemma 5. The expected value can be computed in the following way.

$$\mathsf{E}[|\mathcal{B}_{d_x}(s, r)|] = \sum_{v \in V} \mathsf{P}[v \in \mathcal{B}_{d_x}(s, r)] \leq \sum_{v \in V} f_v = k \tag{4.8}$$

□

Let us now show a bounds on the size of the ball $\mathcal{B}_{d_x}(s, r)$.

Lemma 7. *For any $0 \leq r < 1$ these bounds $1 \leq |\mathcal{B}_{d_x}(s, r)| \leq 1 + \frac{k-1}{1-r}$ holds.*

Proof. The lower bound can be easily seen only from the inequality (4.4e).

The upper bound uses Lemma 5, which results in the inequality $f_v \geq 1 - r$ for all $v \in \mathcal{B}_{d_x}(s, r)$. The total sum of f_v values is equal to k and since $f_s = 1$ we get that $\sum_{v \in V \setminus \{s\}} f_v = k - 1$.

$$
\begin{aligned}
(1-r)\,|\mathcal{B}_{d_x}(s,r)\setminus\{s\}| &= \sum_{v\in\mathcal{B}_{d_x}(s,r)\setminus\{s\}} 1-r \\
&\leq \sum_{v\in\mathcal{B}_{d_x}(s,r)\setminus\{s\}} f_v \\
&\leq \sum_{v\in V\setminus\{s\}} f_v \\
&= k-1
\end{aligned}
\tag{4.9}
$$

From which we obtain the desired inequality. □

Here we present a constant factor approximation of the optimal value. We use the basic property of linear relaxation, where its optimal value can be only lower than the optimum of the integer linear program.

Lemma 8. *Given a constant $0 < c < 1$, let $\mathcal{I}_c = \{r \in \langle 0,1\rangle \mid e(\mathcal{B}_{d_x}(s,r), V \setminus \mathcal{B}_{d_x}(s,r)) > \frac{1}{c}\mathrm{OPT_{LP}}\}$. Then $\mathcal{I}_c$ corresponds to a set of subintervals of $\langle 0,1\rangle$ and the sum of their length is at most c.*

Proof. We partition the interval $\langle 0,1\rangle$ into maximal inclusion-wise sub-intervals $I_1, I_2, \ldots, I_\ell$, such that for all r within a given sub-interval I_i, the number of edges $e(\mathcal{B}_{d_x}(s,r), V \setminus \mathcal{B}_{d_x}(s,r))$ remains constant. For each $i \in [\ell]$, we denote this constant number of edges $e(\mathcal{B}_{d_x}(s,r), V \setminus \mathcal{B}_{d_x}(s,r))$ as δ_i.

Set $\mathcal{I} = \{i \in [\ell] \mid \delta_i \geq \frac{1}{c} \cdot \mathrm{OPT_{LP}}\}$. Since $\sum_{i=1}^{\ell} \delta_i\,|I_i| = \mathrm{OPT_{LP}}$ holds from the definition, we have that $\sum_{i\in\mathcal{I}} |I_i| \leq c$. □

We now propose a bicriteria approximation algorithm for the Minimum k-connected cut with source problem, and consequently also for Minimum k-connected cut problem. The technique we use is known as a *Ball growing technique* [12].

Theorem 9. *For a given graph $G = (V,E)$, $k \in \mathbb{N}$ and $0 < c < 1$ there exists a polynomial time algorithm, which produces a connected cut S, such that $1 \leq |S| \leq \mathcal{O}(k)$ and $e(S, V \setminus S) \leq \frac{1}{c}\mathrm{OPT_{LP}}$.*

Proof. From both Lemmata 7 and 8 it follows that there exists $0 \leq r < 1$ such that $1 \leq |\mathcal{B}_{d_x}(s,r)| \leq \mathcal{O}(k)$ and $e(\mathcal{B}_{d_x}(s,r), V \setminus \mathcal{B}_{d_x}(s,r)) \leq \frac{1}{c}\mathrm{OPT_{LP}}$. As the function $f(r) \coloneqq e(\mathcal{B}_{d_x}(s,r), V \setminus \mathcal{B}_{d_x}(s,r))$ is piece-wise constant with at most $|V|$ pieces, then such radius can be found in polynomial time. □

5 Computational experiments

In this last chapter we briefly describe our implementation of the algorithm for Minimum k-connected cut with source which was mentioned in Chapter 4.

We implemented the algorithm in Rust with the use of Gurobi optimizer. In order to run the program both has to be installed. Additionally, for creating images of graphs, we use Graphviz's DOT language[1]. We also generate reports into a markdown format[2] and then it can be transformed to any other type of document with the usage of pandoc.

The source code is given as an attachment together with the auto-generated reports.

5.1 Documentation

The program's architecture is organized across several source files:

- `main.rs` – Contains the program's entry point and main function.
- `lp.rs` – Houses all linear programming components, including `Instance` and `Solution` classes, as well as functionality for creating linear programs from the input graph and solving them.
- `apx.rs` – Implements functions related to the approximation algorithm.
- `generator.rs` – Provides functions for generating the various test graphs used in experiments.
- `images.rs` – Contains all procedures responsible for graph visualization and image generation.
- `reporter.rs` – Includes functions that generate markdown files summarizing experimental results.

As implementation details are not the primary focus of this thesis, we will not delve deeper into the code structure. Additionally, we present several graphs that were examined during testing. The accompanying script allows for reproduction of all experiments.

5.2 Considered graphs

We examine several graph types in our experiments:

- *clique* – A complete graph with n vertices, where every vertex is connected to all others, as illustrated in Figure 5.1a.

[1] Structured text for drawing graphs. For more information see graphviz.org/doc/info/lang.html.

[2] Structured text format. For more details visit www.markdownguide.org/.

- *star* – A graph structure featuring one central node connected to all other $n-1$ vertices, while the non-central vertices have no connections between them, as shown in Figure 5.1b.
- *Petersen graph* – A well-known graph frequently used as a counterexample, depicted in Figure 5.1c.
- *comet* – A modified star graph where one edge has been subdivided into a path, as demonstrated in Figure 5.1d.
- *necklace* – A graph formed by taking a clique and connecting two of its vertices with an additional path containing new vertices, as shown in Figure 5.1e.
- *tree* – A tree with predefined number of ancestors and specified depth, illustrated in Figure 5.1f.
- *complete bipartite* – A graph consisting of two disjoint vertex sets V_1 and V_2, where an edge $\{u, v\}$ exists if and only if $u \in V_1 \wedge v \in V_2$, as presented in Figure 5.1g.

5.3 Experiment results

Last, we present the results of our experiments. For more details we refer to look into the attached reports.

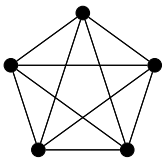
In Table 5.1, we present the graph names alongside their properties. The results from both integer linear program and its linear relaxation are displayed as $\mathrm{OPT}_{\mathrm{MIP}}$ and $\mathrm{OPT}_{\mathrm{LP}}$ respectively. The integer linear program value represents the optimal solution. Notably, several instances achieved the optimal value in their linear relaxation, specifically *star*, *star-alt*, *tree*, and *tree-alt* instances. Although all these examples are trees, it's important to clarify that tree structures don't universally yield optimal values in linear relaxation. This is demonstrated by the *comet* and *comet-alt* instances, which are both trees that differ only in their designated source vertex.

We ran our approximation algorithm for three constants $\frac{1}{10}$, $\frac{1}{2}$ and $\frac{9}{10}$ to observe key properties. For each constant we record the size of the cut and the size of the solution $S \subseteq V$. This values are also present in the Table 5.1.

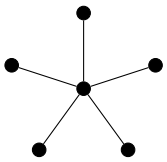
First, Table 5.1 demonstrates that for each constant c and instance, the value $e(S, V \setminus S)$ does not exceed $\frac{1}{c}\mathrm{OPT}_{\mathrm{LP}}$. Specifically for the instances *comet*, *comet-alt*, *clique*, *necklace-alt*, and *k-2-20-large*, we observe that when $c = \frac{9}{10}$, the solution set S undergoes a change. This modification is necessary because otherwise, the value would exceed the $\frac{10}{9}\mathrm{OPT}_{\mathrm{LP}}$ threshold.

The reader may observe that for graphs, where $\mathrm{OPT}_{\mathrm{LP}} = \mathrm{OPT}_{\mathrm{MIP}}$ we obtain the same result for any c, hence our approximation algorithm preserves the optimum.

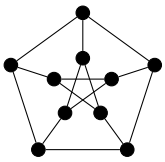
Observe that all *clique*s, *k-2-20*s and *Petersen* have trivial solutions, which is either $S = V$ or $S = \{s\}$. In the other instances the size of solution $|S|$ is close to the value k, specifically for both constants $\frac{1}{10}$ and $\frac{1}{2}$, which arises from



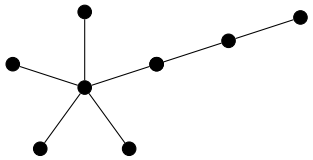
(a) Complete graph for $n = 5$.



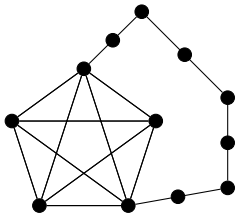
(b) Star graph for $n = 6$.



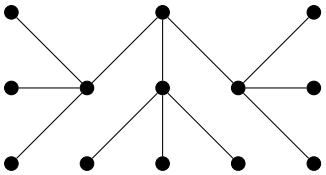
(c) Petersen graph.



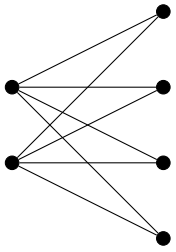
(d) Comet graph for star of size 5 and path of length 3.



(e) Necklace graph for clique of size 5 and path of length 7.



(f) Regular tree graph for 3 ancestors and depth 3.



(g) Complete bipartite for $|V_1| = 2$ and $|V_2| = 4$.

Figure 5.1 Considered type of graphs in experiments.

more possible choices of radii. Last, for constant $\frac{9}{10}$ we also get trivial solution for *comet*, *comet-alt* and *neckalce-alt*.

We want to emphasize that having $S = V$ is not a proper cut. Let us explain why this can occur. In such instances we may see that n is relatively close to k, or it is a dense graph, i.e. having a large number of edges. Therefore all vertices are close to the source vertex s with respect to the pseudo-metric d_x. Hence this solution satisfies our properties. Moreover in several cases, mainly for constant $c = \frac{9}{10}$, this is the only feasible solution satisfying the bound $\frac{1}{c}\mathrm{OPT}_{\mathrm{LP}}$. Hence it makes sense to consider such solutions.

Therefore we conclude that for constant $c = \frac{1}{2}$ where the value is at most $2 \cdot \mathrm{OPT}_{\mathrm{LP}}$ we get solutions relatively close to k unless the graphs are largely intact, i.e. having large number of edges. This holds for $c = \frac{1}{10}$ as well, but here we have guaranteed only the bound $10 \cdot \mathrm{OPT}_{\mathrm{LP}}$.

Graph name	k	$\|V\|$	$\mathrm{OPT}_{\mathrm{MIP}}$	$\mathrm{OPT}_{\mathrm{LP}}$	$c = 1/10$		$c = 1/2$		$c = 9/10$	
					$e(S, V \setminus S)$	$\|S\|$	$e(S, V \setminus S)$	$\|S\|$	$e(S, V \setminus S)$	$\|S\|$
comet	12	21	2	1.636	2	10	2	10	0	21
comet-alt	12	21	2	1.8	2	11	2	11	0	21
comet-path	6	21	4	4	4	6	4	6	4	6
clique	3	10	42	14	18	1	18	1	0	10
clique-alt	6	10	48	8	0	10	0	10	0	10
star	4	21	34	34	34	4	34	4	34	4
star-alt	8	21	26	26	26	8	26	8	26	8
tree	15	81	6	6	6	15	6	15	6	15
tree-alt	3	216	22	22	22	3	22	3	22	3
Petersen	6	10	12	2.667	0	10	0	10	0	10
necklace	10	20	22	5.6	4	11	4	11	4	11
necklace-alt	10	20	22	2.222	4	6	4	6	0	20
k-2-20	8	22	40	26.667	40	1	40	1	0	22
k-2-20-alt	3	22	40	36.19	40	1	40	1	40	1
k-2-20-large	14	22	32	15.238	0	22	0	22	0	22

Table 5.1 Experiment results.

Conclusion

In this thesis, we dealt with connected cut problems that partition graph in a more constrained way than classical cuts. We mainly focused on MINIMUM k-CONNECTED CUT and MINIMUM k-CONNECTED CUT WITH SOURCE problems, which we defined. First, we proved that both these problems are NP-complete by reducing them from the MINIMUM BISECTION problem.

Second, we introduced a new concept called *absorptive flow* and developed an integer linear programming formulation that solves the MINIMUM k-CONNECTED CUT WITH SOURCE problem, and consequently, the MINIMUM k-CONNECTED CUT as well. We demonstrated that the integrality gap of this integer linear program and its relaxation is at least k, and identified several key properties that enable the design of an approximation algorithm.

For the approximation algorithm, we used the pseudo-metric defined by the cut variables x_e and considered all vertices within a ball $\mathcal{B}_{d_x}(s, r)$, centered at the source s with radius $0 \leq r < 1$. We showed that the expected values obtained are bounded by the optimal value. Additionally, we presented an algorithm which, for any constant $0 < c < 1$, produces a solution within a constant factor $\frac{1}{c}$ of the optimum, while allowing the size constraint to be within the bounds $1 \leq |S| \leq \mathcal{O}(k)$.

Finally, we implemented our algorithm and conducted several experiments on special graph classes. Our observations indicate that complete graphs do not yield good solutions, whereas trees tend to produce better results, which are closer to the optimal value.

We highlight some open questions that remain. One is whether the ball $\mathcal{B}_{d_x}(s, 1)$ always contains all vertices, or if there exists an instance $G = (V, E)$ with a vertex $v \in V$ such that the distance $d_x(s, v) > 1$. We conjecture that $\mathcal{B}_{d_x}(s, 1) = V$, but this remains unproven. Another observation is that all graphs we found with absorptive flow induce a tree structure, raising the question of whether this is always the case.

Bibliography

1. Ahuja, Ravindra; Magnanti, Thomas; Orlin, James. Network Flows. 1993. Available also from: https://www.researchgate.net/publication/38009578_Network_Flows.

2. Buluç, Aydın; Meyerhenke, Henning; Safro, Ilya; Sanders, Peter; Schulz, Christian. Recent Advances in Graph Partitioning. In: Kliemann, Lasse; Sanders, Peter (eds.). *Algorithm Engineering: Selected Results and Surveys*. Cham: Springer International Publishing, 2016, pp. 117–158. ISBN 978-3-319-49487-6. Available from DOI: 10.1007/978-3-319-49487-6_4.

3. Cook, William J.; Cunningham, William H.; Pulleyblank, William R.; Schrijver, Alexander. *Combinatorial Optimization*. Wiley, 1997. ISBN 9781118033142. Available from DOI: 10.1002/9781118033142.

4. Cordero, Mishelle; Miniguano-Trujillo, Andrés; Recalde, Diego; Torres, Ramiro; Vaca, Polo. Optimizing Connected Components Graph Partitioning With Minimum Size Constraints Using Integer Programming and Spectral Clustering Techniques. *Networks*. 2025, vol. 85, no. 3, pp. 245–260. Available from DOI: 10.1002/NET.22257.

5. Cygan, Marek; Fomin, Fedor V.; Kowalik, Łukasz; Lokshtanov, Daniel; Marx, Dániel; Pilipczuk, Marcin; Pilipczuk, Michał; Saurabh, Saket. *Parameterized Algorithms*. Springer International Publishing, 2015. ISBN 9783319212753. Available from DOI: 10.1007/978-3-319-21275-3.

6. Duarte, Gabriel L.; Eto, Hiroshi; Hanaka, Tesshu; Kobayashi, Yasuaki; Kobayashi, Yusuke; Lokshtanov, Daniel; Pedrosa, Lehilton L. C.; Schouery, Rafael C. S.; Souza, Uéverton S. Computing the Largest Bond and the Maximum Connected Cut of a Graph. *Algorithmica*. 2021, vol. 83, no. 5, pp. 1421–1458. ISSN 1432-0541. Available from DOI: 10.1007/s00453-020-00789-1.

7. Even, Guy; Naor, Joseph (Seffi); Rao, Satish; Schieber, Baruch. Fast Approximate Graph Partitioning Algorithms. *SIAM Journal on Computing*. 1999, vol. 28, no. 6, pp. 2187–2214. ISSN 1095-7111. Available from DOI: 10.1137/s0097539796308217.

8. Feige, Uriel; Krauthgamer, Robert. A Polylogarithmic Approximation of the Minimum Bisection. *SIAM Journal on Computing*. 2002, vol. 31, no. 4, pp. 1090–1118. Available from DOI: 10.1137/S0097539701387660.

9. Ford, Lester R.; Fulkerson, Delbert Ray. *Flows in networks*. Ed. by Fulkerson, Delbert Ray. Princeton: Princeton Univ. Press, 2010. Princeton landmarks in mathematics. ISBN 0691146675. Originally published: 1962.

10. Garey, M.R.; Johnson, D.S.; Stockmeyer, L. Some simplified NP-complete graph problems. *Theoretical Computer Science*. 1976, vol. 1, no. 3, pp. 237–267. ISSN 0304-3975. Available from DOI: https://doi.org/10.1016/0304-3975(76)90059-1.

11. Garey, Michael R.; Johnson, David S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

12. Garg, Naveen; Vazirani, Vijay. Multicommodity Flows and Approximation Algorithms. 1996, pp. 10–17. Available also from: https://www.researchgate.net/publication/2513735_Multicommodity_Flows_and_Approximation_Algorithms.

13. Goemans, Michel X.; Williamson, David P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*. 1995, vol. 42, no. 6, pp. 1115–1145. ISSN 0004-5411. Available from DOI: 10.1145/227683.227684.

14. Karger, David R.; Stein, Clifford. A new approach to the minimum cut problem. *J. ACM*. 1996, vol. 43, no. 4, pp. 601–640. ISSN 0004-5411. Available from DOI: 10.1145/234533.234534.

15. Karp, Richard M. Reducibility among Combinatorial Problems. In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Miller, Raymond E.; Thatcher, James W.; Bohlinger, Jean D. Boston, MA: Springer US, 1972, pp. 85–103. ISBN 978-1-4684-2001-2. Available from DOI: 10.1007/978-1-4684-2001-2_9.

16. Korte, Bernhard; Vygen, Jens. *Combinatorial Optimization: Theory and Algorithms*. Springer Berlin Heidelberg, 2018. ISBN 9783662560396. ISSN 2197-6783. Available from DOI: 10.1007/978-3-662-56039-6.

17. Krauthgamer, Robert. Minimum Bisection. In: Kao, Ming-Yang (ed.). *Encyclopedia of Algorithms*. New York, NY: Springer New York, 2016, pp. 1294–1297. ISBN 978-1-4939-2864-4. Available from DOI: 10.1007/978-1-4939-2864-4_231.

18. Lee, Euiwoong. Partitioning a Graph into Small Pieces with Applications to Path Transversal. In: Klein, Philip N. (ed.). *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*. SIAM, 2017, pp. 1546–1558. Available from DOI: 10.1137/1.9781611974782.101.

19. Matoušek, Jiří; Nešetřil, Jaroslav. *Invitation to Discrete Mathematics (2. ed.)* Oxford University Press, 2009. ISBN 978-0-19-857042-4.

20. Mishra, Nina; Schreiber, Robert; Stanton, Isabelle; Tarjan, Robert E. Clustering Social Networks. In: Bonato, Anthony; Chung, Fan R. K. (eds.). *Algorithms and Models for the Web-Graph*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 56–67. ISBN 978-3-540-77004-6.

21. Saranurak, Thatchaphol; Wang, Di. Expander decomposition and pruning: faster, stronger, and simpler. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. San Diego, California: SIAM, 2019, pp. 2616–2635. SODA '19.

22. Schrijver, Alexander. *Combinatorial optimization: Polyhedra and efficiency*. Berlin: Springer, 2003. Algorithms and combinatorics, no. 24. ISBN 9783540443896.

23. Schrijver, Alexander. *Theory of linear and integer programming*. Wiley, 1999. Wiley-Interscience series in discrete mathematics and optimization. ISBN 978-0-471-98232-6.

List of Figures

List of Tables

A Attachments

A.1 `attachment.zip`

In the attached file `attachment.zip` we have two directories.

A.1.1 `solver`

In the directory `solver` we find the source code files corresponding to the implementation described in Chapter 5. If all required tools are installed, the reader can simply run `cargo run` or use the provided script `automate.sh` to generate images and reports with a single command.

A.1.2 `reports`

The second directory, `reports`, contains the automatically generated PDF reports produced by our implementation. These reports include images of graphs, flows, cuts, and approximation results. Specifically, for the constants 0.1, 0.5, and 0.9, the files are named `report-`c`.pdf`, where c represents the respective constant value.