IMPOSING CONNECTIVITY IN NETWORK DESIGN PROBLEMS

By

HAMIDREZA VALIDI

Bachelor of Science in Industrial Engineering
Ferdowsi University of Mashhad
Mashhad, Iran
2012

Master of Science in Industrial Engineering
Sharif University of Technology
Tehran, Iran
2014

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
June, 2020

IMPOSING CONNECTIVITY IN NETWORK DESIGN PROBLEMS

Dissertation Approved:

Dr. Austin Buchanan

Dissertation Advisor

Dr. Baski Balasundaram

Dr. Sunderesh Heragu

Dr. Jay Schweig

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor, Dr. Austin Buchanan, for all the support he provided to me. He treated me like a professional researcher and I really enjoyed working with him on interesting optimization problems. He follows high-level standards in (i) doing research, (ii) writing papers and (iii) coding optimization models; so, I learned a lot in these three areas from him. Dr. Buchanan has been always approachable, and I always feel comfortable and excited to talk on optimization problems with him. He has always encouraged me to follow my passions in optimization by providing all the needed resources. In a nutshell, he is surely a close and professional friend, and I never forget his favors.

Also, I would like to thank my committee members - Dr. Baski Balasundaram, Dr. Sunderesh Heragu, and Dr. Jay Schweig - for their support and invaluable comments on my dissertation. I really enjoyed taking optimization courses with Dr. Baski. I am also excited to work with FLAT project team, leading by Dr. Baski, on a real-world industrial problem. I truly appreciate Dr. Heragu, the head of IEM, for giving wonderful teaching opportunities to me in Fall 2018 and Fall 2019. It was surely my pleasure to teach IEM 3103 to undergraduate students.

I should also thank my summer mentors - Dr. Moon Duchin (Tufts University) and Dr. Justin Solomon (MIT) - at Voting Rights Data Institute (VRDI) for giving a great research opportunity to me. The internship experience at VRDI opened my eyes to new dimensions of the districting problem. I enjoyed working with my knowledgeable teammates and friends at

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

VRDI: Amy, Chris, Daryl, Gabe, Jasmine, Lorenzo, Nestor, Parker, and Tara. I am grateful to Eugene Lykhovyd for his outstanding contribution in Chapter III. Also, David Applegate did a great favor to me and solved a complicated districting puzzle, just in one day, at MIP workshop 2019.

Last but not least, I would like to thank my family for their support during last four years. Although I have not seen them during my Ph.D. study, my mom has been always online on Thursday nights to talk on her paintings and my works.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: HAMIDREZA VALIDI

Date of Degree: JUNE, 2020

Title of Study: IMPOSING CONNECTIVITY IN NETWORK DESIGN PROBLEMS

Major Field: INDUSTRIAL ENGINEERING AND MANAGEMENT

Abstract: Imposing connectivity arises in multiple real-world problems – telecommunication network design, social network analysis, reserve network design, and redistricting to name a few. In network optimization, connectivity constraints are usually imposed in three spaces: (i) vertex space, (ii) edge space, and (iii) vertex-and-edge space. In this dissertation, we focus on imposing connectivity in the vertex and edge spaces. We study connectivity constraints in telecommunication and redistricting networks (both in the vertex space), revisit the spanning tree polytope in planar graphs (in the edge space), and conduct a polyhedral study of k connected components (in the vertex space).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION[1]

Discrete network optimization contains a wide range of interesting combinatorial problems such as the traveling salesman problem, the maximum matching problem, the minimum spanning tree problem, the shortest path problem, and others. Although they usually have easy definitions, many of them (e.g., the traveling salesman problem) are considered as hard problems (i.e., no one has found a polynomial-time algorithm for solving them yet). However, there are practical (but not polynomial-time) algorithms for solving the hard ones to optimality on real-life instances. These methods include but are not limited to branch-and-bound (Markowitz and Manne, 1957; Eastman, 1958; Land and Doig, 1960), branch-and-price (Davidon, 1991; Broyden, 1970; Fletcher, 1970), cutting-plane method (Jünger et al., 1995), and Lagrangian relaxation (Held and Karp, 1971). Thanks to some of these methods, large-scale instances of many optimization problems are solvable nowadays (Cook, 2019).

In some of these optimization problems, we observe that any feasible solution induces a connected subgraph (i.e., there is a path between any two vertices of the induced subgraph). For example, consider a feasible solution of the minimum spanning tree problem. This solution corresponds to an induced subgraph that is (i) connected, (ii) contains no cycle, and (iii) spans all vertices of the graph. In Figure 1.1, the solid edges represent a spanning tree. Because any feasible solution of such problems (e.g., the minimum spanning tree problem and the traveling salesman problem) induces a connected subgraph, their corresponding optimization models must contain some constraints to impose connectivity.

[1]Some parts of this chapter are based on work with Austin Buchanan (Validi and Buchanan, 2019c)

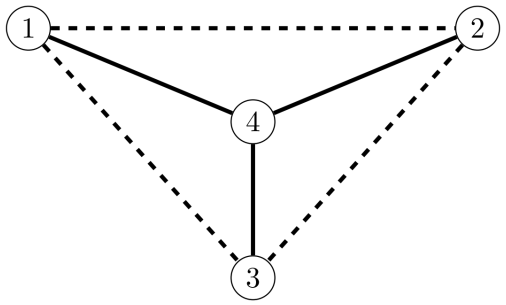


Figure 1.1: solid edges represent a spanning tree on the complete graph K_4.

1.1 Connectivity Constraints

Connectivity constraints appear in many network optimization problems such as the minimum spanning tree problem, the minimum Steiner tree problem, the traveling salesman problem, the minimum connected dominating set problem, the political districting problem, and others. For some problems (e.g., the districting problem), connectivity constraints are considered to be challenging to impose well (Ricca and Simeone, 2008; Ricca et al., 2013; Goderbauer and Winandy, 2018; Swamy et al., 2019b).

In this dissertation, we study connectivity constraints in network optimization problems, and conduct associated polyhedral studies. For the sake of organization, we provide a categorization of them. Depending on which decisions need to be made, connectivity constraints are usually imposed in three spaces: the (i) vertex space, (ii) edge space, and (iii) vertex-and-edge space. Before discussing these spaces, we provide some basic notations of this chapter. Let $G = (V, E)$ be a simple graph with vertex set V and edge set E. Also, let $n := |V|$ and $m := |E|$. The subgraph induced by a vertex subset $V' \subseteq V$ and an edge set $E' \subseteq E$ are denoted by $G[V']$ and $G[E']$, respectively.

1.1.1 Vertex Space

Consider the county-level map of Oklahoma in Figure 1.2. One can build a corresponding graph (dual graph) for it as follows: (i) put a vertex in each county, and (ii) draw an edge between any two adjacent vertices (counties). Because Oklahoma has 77 counties and 195 internal common borders between counties, its corresponding dual graph has 77 vertices and 195 edges. Also, if following these instructions for drawing the dual graph, then no pair of edges will cross each other. This means that the dual graph is planar. We will provide more details on planar graphs in Chapter IV.

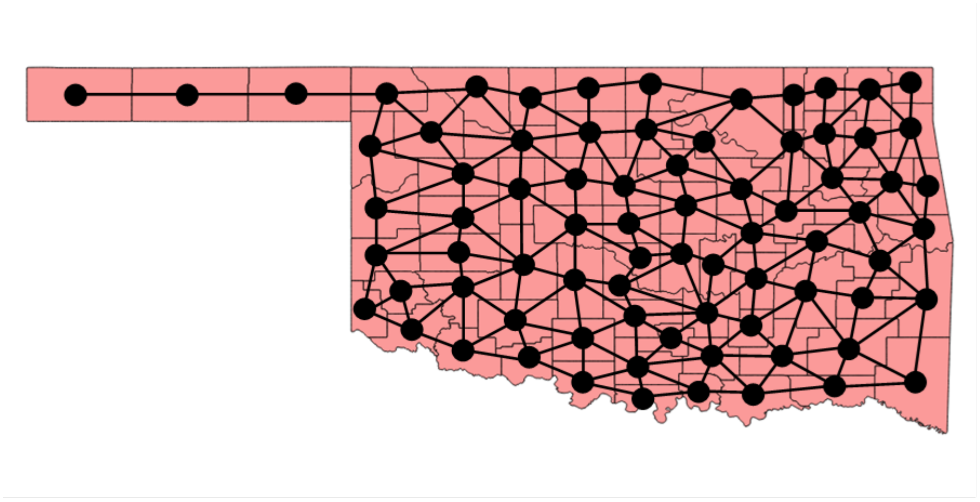


Figure 1.2: the dual graph of Oklahoma at the county level.

Now suppose you are asked to partition the state of Oklahoma into five non-empty connected subsets of counties. How do you partition it? Note that a subset of counties is connected if you are able to travel from any county of the subset to another county (in the same subset) without leaving the subset. Figure 1.3 gives a plan with connected partitions for Oklahoma at the county level. Connectivity is considered as a “hard constraint”, but not the only one, in the districting process. We will provide more details on the districting process and its criteria in Chapter III.

In some network optimization problems (e.g., partitioning the vertices of a graph into connected subgraphs), the edges are immaterial, and the real decision is to determine which *vertices* should be chosen so that the resulting induced subgraph is connected. So, one can impose connectivity constraints by employing only binary decision variables for choosing

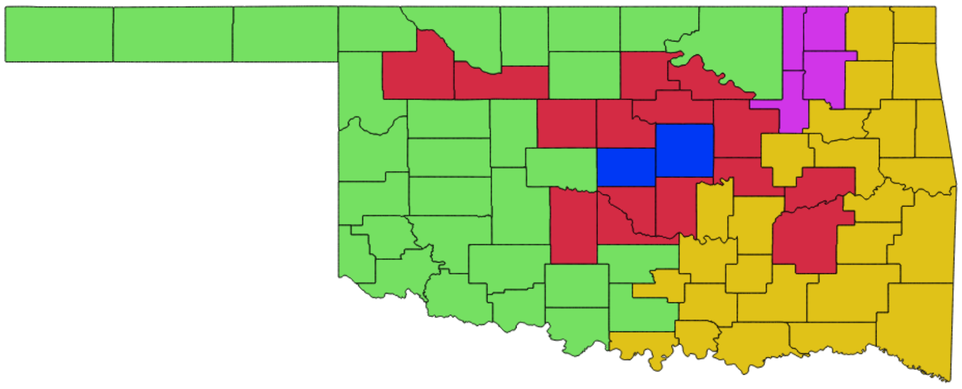


Figure 1.3: a contiguous districting plan of Oklahoma at the county level.

vertices. In other words, any feasible solution in a vertex-centric connectivity problem belongs to the vertex-induced connected subgraph polytope $P_V(G)$.

$$P_V(G) := \text{conv.hull}\left\{x^{V'} \;\middle|\; G[V'] \text{ is connected}\right\}.$$

Here, $x^{V'} \in \{0,1\}^n$ represents the characteristic vector of $V' \subseteq V$. Connectivity constraints in the vertex space arise in the following problems:

- designing "virtual backbones" for wireless sensor networks (Validi and Buchanan, 2019b; Buchanan et al., 2015; Du and Wan, 2013);
- choosing contiguous pieces of land to create a wildlife preservation (Dilkina and Gomes, 2010; Conrad et al., 2012; Carvajal et al., 2013);
- selecting contiguous census tracts to form a political district (Validi et al., 2020; Swamy et al., 2019b; Garfinkel and Nemhauser, 1970; Mehrotra et al., 1998);
- searching for clusters in social (Wasserman and Faust, 1994; Moody and White, 2003; Veremyev and Boginski, 2012) and biological networks (Dittrich et al., 2008; Bailly-

Bechet et al., 2011; Backes et al., 2012);

- trying to distinguish an object from the rest of an image (Vijayanarasimhan and Grauman, 2011; Chen and Grauman, 2012);
- designing aisle space in a high density storage system (Gue, 2006).

In Chapter II of this dissertation, we consider how to impose connectivity for the *minimum latency-constrained connected dominating set* problem in the vertex space. We employ a cut-based formulation to impose connectivity in this space. Although the cut-based formulation has exponentially many connectivity constraints, we can solve the LP relaxation of the problem in polynomial time thanks to the classical "optimization = separation" result of Grötschel et al. (1993). In Chapter III, we propose two formulations for the districting problem: (i) a new cut-based formulation in the vertex space; and (ii) an extended flow-based formulation for an existing cut-based formulation (in the vertex space). In Chapter V, we conduct a polyhedral study of k connected components in the vertex space.

1.1.2 Edge Space

In some combinatorial optimization problems (e.g., see Section 1.1 and Figure 1.1 for the minimum spanning tree problem), the decision is to determine which edges should be selected in a connected subgraph. So, one can impose connectivity constraints by employing only binary decision variables for selecting edges. In other words, any feasible solution in an edge-centric connectivity problem belongs to the edge-induced connected subgraph polytope $P_E(G)$.

$$P_E(G) := \text{conv.hull}\left\{ y^{E'} \;\middle|\; G[E'] \text{ is connected} \right\}.$$

Here, $y^{E'} \in \{0,1\}^m$ represents the characteristic vector of $E' \subseteq E$. We can see connectivity constraints of this type in the following problems:

- designing Steiner and spanning trees (Hwang et al., 1995; Goemans and Myung, 1993; Polzin and Daneshmand, 2001; Voß, 2006; Validi and Buchanan, 2019a; Williams, 2002a);
- designing survivable networks (Grötschel et al., 1995; Kerivin and Mahjoub, 2005; Grötschel et al., 1992b,c).

In Chapter IV, we note that how ignoring a rule (root rule) renders the extended formulation (in the edge space) of Williams (2002a) for finding spanning trees in planar graphs incorrect.

1.1.3 Vertex-and-Edge Space

Transmitting quality signals in a telecommunication network requires an efficient network design that can minimize the cost of installing regenerators on vertices as well as establishing new edges. The *network design problem with regenerators (NDPR)* is defined on an undirected graph $G = (V, E)$ with commodity set K, where K denotes the set of commodities or communication pairs. Edges are categorized into two sets: (1) "free" edges denoted by E_f, and (2) potential edges, to possibly be added, represented by E_p. In other words, $E = E_f \cup E_p$.

Since signals can be traversed only through a path of length at most d_{max}, a regenerator can be placed on an intermediate vertex v with cost c_v to reamplify, reshape, and retime them. Moreover, a potential edge e also can be added to the network with length of d_e at cost of c_e to facilitate communications between commodities. The optimization version of NDPR is defined as follows.

Problem: NDPR.

Input: A graph $G = (V, E_p \cup E_f)$, commodity set K, regenerator costs c_v and edge costs c_e (which can be zero for free edges).

Output: A minimum cost regenerator subset $V^* \subseteq V$ of vertices and edge subset $E_p^* \subseteq E_p$ of edges that is feasible for NDPR.

A small NDPR instance and an optimal solution are provided in Figure 1.4. In this instance, $d_{max} = 4$ and $K = \{\{1,3\},\{1,4\}\}$. The installation cost of each regenerator is shown as a label outside of each vertex. The potential edges are shown by dashed lines while the free ones are represented by solid lines. Moreover, the cost of establishing each potential edge is shown in parentheses after its length. Since free edges have been established before, they are only labeled by their lengths.

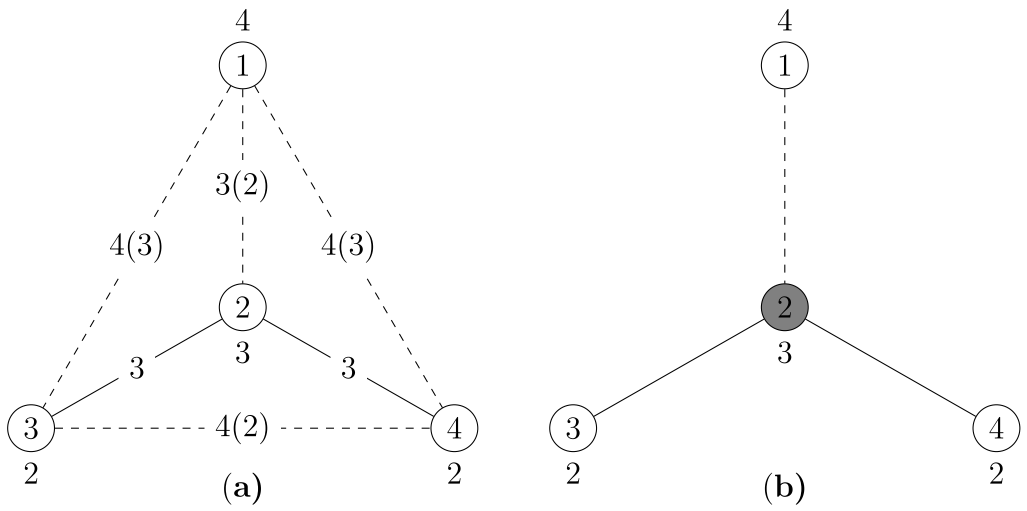


Figure 1.4: An instance of NDPR (**a**); an optimal solution (**b**).

There are some network optimization problems (e.g., NDPR) in which we make decision on both vertices and edges. So, one can impose connectivity constraints by employing binary decision variables for selecting vertices and edges. In other words, any feasible solution in a vertex-and-edge-centric connectivity problem belongs to the connected subgraph polytope $P(G)$.

$$P(G) := \text{conv.hull}\left\{(x^{V'}, y^{E'}) \;\middle|\; \text{subgraph } (V', E') \subseteq G \text{ is connected}\right\}.$$

Here, $x^{V'} \in \{0,1\}^n$ and $y^{E'} \in \{0,1\}^m$ represent the characteristic vectors of $V' \subseteq V$ and $E' \subseteq E$, respectively. These types of constraints appear in the following connectivity problems:

- prize-collecting Steiner tree problem (Chopra and Rao, 1994; Ljubić et al., 2006; Leitner et al., 2018; Rehfeldt et al., 2019);
- network design problems with relays (Leitner et al., 2019; Cabral et al., 2007, 2008).

1.2 Summary of Contributions

In Chapter II, we consider imposing connectivity in wireless networks. Two nodes of a wireless network may not be able to communicate with each other directly perhaps due to obstacles or insufficient signal strength. This necessitates the use of intermediate nodes to relay information. Often, one designates a (preferably small) subset of them to relay these messages (i.e., to serve as a virtual backbone for the wireless network) which can be seen as a connected dominating set (CDS) of the associated graph. Ideally, these communication paths should be short, leading to the notion of a latency-constrained CDS. We point out several shortcomings of a previously studied formalization of a latency-constrained connected dominating set and propose an alternative one. We introduce an integer programming formulation for the problem that has a variable for each node and imposes the latency constraints via an exponential number of cut-like inequalities. Two nice properties of this formulation are that: (1) it applies when distances are hop-based and also when they are weighted; and (2) it easily generalizes to ensure fault tolerance. We provide a branch-and-cut implementation of this formulation and compare it with a new polynomial-size formulation. Computational experiments demonstrate the superiority of the cut-like formulation. We also study related questions from computational complexity such as approximation hardness and answer an open problem regarding the fault diameter of graphs.

In Chapter III, we study the political districting problem from a combinatorial optimization point-of-view. Beginning in the 1960s, techniques from operations research began to be used to generate political districting maps. A classical example is the integer programming model of Hess et al. (*Operations Research* 13(6):998–1006, 1965). Due to the model's compactness-seeking objective, it tends to generate contiguous or nearly-contiguous districts, although none of the model's constraints explicitly impose contiguity. Consequently, Hess et al. had to manually adjust their solutions to make them contiguous. Since then, there have been several attempts to adjust the Hess model and other models so that contiguity is explicitly ensured. We review two existing models for imposing contiguity, propose two new ones, and analytically compare them in terms of their strength and size. We conduct an extensive set of numerical experiments to evaluate their performance. While many believe that contiguity constraints are particularly difficult to deal with, we find that the problem does not become harder when contiguity is imposed. In fact, a branch-and-cut implementation of a cut-based model generates, for the first time, optimally compact districting plans for 21 different US states at the census tract level (under the compactness objective proposed by Hess et al.). To encourage future research in this area, and for purposes of transparency, we make our test instances, source code, and log files publicly available.

In Chapter IV, we consider the minimum spanning tree problem in planar graphs. In the paper "A linear-size zero-one programming model for the minimum spanning tree problem in planar graphs" (Networks 39(1):53–60, 2002), Williams introduced an extended formulation for the spanning tree polytope of a planar graph. This formulation is remarkably small (using only $O(n)$ variables and constraints) and remarkably strong (defining an integral polytope). We point out that Williams' formulation, as originally stated, is incorrect. Specifically, we construct a binary feasible solution to Williams' formulation that does not represent a spanning tree. Fortunately, there is a simple fix, which is to restrict the choice of the root vertices in the primal and dual spanning trees, whereas Williams explicitly allowed them to be chosen arbitrarily. The same flaw and fix apply to a subsequent formulation of Williams

("A Zero-One Programming Model for Contiguous Land Acquisition." Geographical Analysis 34(4): 330-349, 2002).

In Chapter V, we conduct a polyhedral study on connectivity constraints for imposing at most k connected subgraphs. We generalize results of Wang et al. (Math Prog, 166(1-2):241–271, 2017) that were obtained for the $k = 1$ case of our problem. Two classes of inequalities are discussed: (i) separator inequalities, and (ii) indegree inequalities. We identify when these classes are facet-defining, and when they, along with trivial 0-1 bounds, provide perfect formulations. While fractional separation problem is solvable in polynomial time for indegree inequalities, we show that it is NP-hard for sparator inequalities when $k \geq 2$. Also, we provide two extended formulations that are at least as strong as both classes of the inequalities.

In Chapter VI, we conclude the dissertation and provide potential future research work.

CHAPTER II

THE OPTIMAL DESIGN OF LOW-LATENCY VIRTUAL BACKBONES[1]

Two nodes of a wireless network may not be able to communicate with each other *directly* perhaps due to obstacles or insufficient signal strength. This necessitates the use of intermediate nodes to relay information. Often, one designates a small subset of them to relay messages (i.e., to serve as a virtual backbone for the wireless network) which amounts to a connected dominating set of the associated graph, defined below.

Definition 1 (CDS). *A subset $D \subseteq V$ of vertices is a connected dominating set (CDS) for an undirected graph $G = (V, E)$ if:*

1. *D is* dominating*, i.e., every vertex from $V \setminus D$ neighbors a vertex of D; and*

2. *D is* connected*, i.e., the subgraph $G[D]$ induced by D is connected.*

If the graph G is not complete[2], a CDS can equivalently be defined as a subset $D \subseteq V$ of vertices such that, for every vertex pair $\{a, b\} \in \binom{V}{2}$, there exists a path connecting a and b whose interior vertices belong to D.

A CDS ensures that the nodes of the network can communicate with each other. It provides little guarantee on *how long* it will take for a message to be received once it has been sent. This has led some researchers to impose additional constraints on the CDS $D \subseteq V$, namely that the subgraph induced by the dominating set D has diameter at most s, i.e., is a

[1]Reprinted with permission from "The optimal design of low-latency virtual backbones" by H. Validi and A. Buchanan. To appear at *INFORMS Journal on Computing*.

[2]The complete graph is the only exception. In this case, no virtual backbone is needed and yet Definition 1 would disallow the empty set. For this reason, the complete graph is treated "with generous disregard" in the virtual backbone literature.

dominating s-club (Li et al., 2007; Zhang et al., 2008; Buchanan et al., 2014). This ensures that messages will be received in $s+2$ hops: one hop to reach the CDS, at most s hops within the CDS, and one hop to reach the destination.

Definition 2 (Dominating s-club)**.** *A subset $D \subseteq V$ of vertices is a dominating s-club for an undirected graph $G = (V, E)$ if:*

1. *D is* dominating*, i.e., every vertex from $V \setminus D$ neighbors a vertex of D; and*

2. *D is an s-*club*, i.e., the subgraph $G[D]$ induced by D has diameter at most s.*

We argue that this formalization of the problem is less than ideal. First, and most importantly, a dominating s-club does not quite capture the intent of the hop constraints, as we will illustrate. Suppose that we want a CDS that facilitates 4-hop communication in the graph in Figure 2.1. This can be ensured by the dominating 2-club given in Figure 2.1(a). Indeed, a message sent from node 5 to node 8 through this virtual backbone must follow the path 5-4-3-6-8, which takes four hops.

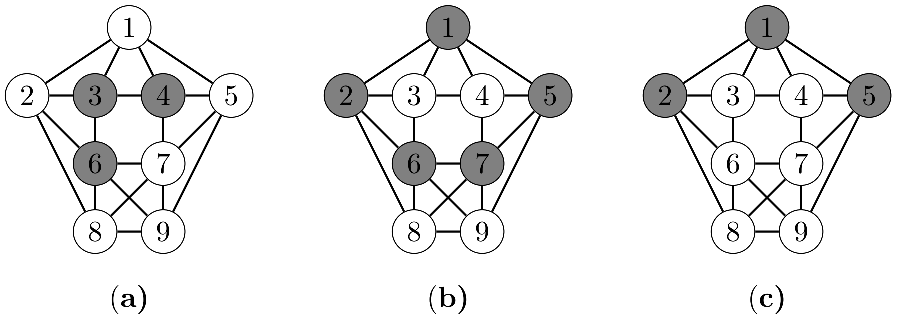


Figure 2.1: (**a**) dominating 2-club; (**b**) latency-2 CDS; (**c**) latency-3 CDS.

If 3-hop communication were required, one might search for a dominating 1-club, but none exist in this graph. This may lead us to believe that a CDS that facilitates 3-hop communication does not exist, but this belief would be false. Indeed, in Figure 2.1(b) we provide a CDS which needs at most *two* hops to transmit information, so we could call it a latency-2 CDS. Further, Figure 2.1(c) gives a latency-3 CDS which is also a minimum CDS!

Note that a message can be passed *directly* from node 8 to node 9 in the wireless network since they are adjacent; it does *not* have to be relayed through the CDS nodes.

Another limitation of previous works is that they make the simplifying assumption that distances are measured by the number of hops, see Li et al. (2007); Zhang et al. (2008); Buchanan et al. (2014) and Chapter 7 of Du and Wan (2013). However, this may ultimately provide a poor approximation to the actual end-to-end delay when the delays at the nodes differ. For example, a particular node may play a central role in the CDS, needing to relay a large number of messages. This may cause messages to have to wait to be transmitted, and these *queueing delays* may be more realistically captured for our purposes via *node-weighted* delays as opposed to hop-based delays. For more information about this and other delays in wireless networks, consult Xie and Haenggi (2009) and Zhong et al. (2017).

With these shortcomings in mind, we propose a new formalization of a low-latency virtual backbone, which we call a latency-s CDS. For purposes of generality, it is defined in terms of a *directed* and—without loss of generality—*edge-weighted* graph. By allowing for directed edges, we can model non-uniform transmission ranges. For example, consider the case where a node i has a large transmission range and is far away from a node j that has a small transmission range. In this scenario, the edge (i, j) should exist, but not the edge (j, i). Note that, as we make the transition to directed graphs, we are no longer referring to a "CDS" in the sense of Definition 1, but rather in terms of a *strongly connected dominating set* in the sense of Li et al. (2009). This is defined as a subset D of vertices such that: (i) D induces a strongly connected subgraph; and (ii) every vertex from $V \setminus D$ has both an in-neighbor and an out-neighbor in D.

Definition 3 (latency-s CDS)**.** *A vertex subset $D \subseteq V$ is a latency-s CDS for a directed graph $G = (V, E)$ under edge weights $w : E \to \mathbb{R}_+$ if, for every vertex pair $(a, b) \in V \times V$, there is a path from a to b of length at most s whose interior vertices belong to D.*

Observe that the graph G in Definition 3 is edge-weighted but not vertex-weighted. This is without loss of generality, as the following will illustrate. Consider the 3-vertex, undirected path graph 1-2-3 representing a wireless network. Sending a message from node 1 to node 3 would incur delays at nodes 1 and 2 (since the delay is based on the transmitting node), as well as delays on edges $\{1,2\}$ and $\{2,3\}$, for an end-to-end delay that might be denoted $d_1 + d_2 + d_{\{1,2\}} + d_{\{2,3\}}$. Instead, we can replace each undirected edge $\{i,j\}$ by its directed counterparts (i,j) and (j,i) and let the delay of each directed edge $d_{(i,j)}$ be the delay of its undirected counterpart $d_{\{i,j\}}$ plus the delay of its tail node d_i. In this way, it is sufficient to consider a directed graph with only edge weights.

Definition 3 overcomes the aforementioned issues with the previous formalization based on dominating s-clubs. As an added bonus, it has superior computational properties. Indeed, checking whether a graph admits a latency-s CDS is as simple as checking whether the graph's diameter is at most s. In contrast, the problem of checking whether there exists *any* dominating s-club is NP-complete; specifically, this is true under hop-based distances for the two most restrictive (but nontrivial) cases where $s = \text{diam}(G) - 2$ (Schaudt, 2013) and $s = \text{diam}(G) - 1$ (Buchanan et al., 2014), where diam denotes the graph's diameter.

The associated optimization problem is as follows.

Problem: The minimum latency-s CDS problem.
Input: A directed graph $G = (V, E)$, a weight $w_e \geq 0$ for each edge $e \in E$, and a number s.
Output: (if any exist) A smallest subset $D \subseteq V$ of vertices that is a latency-s CDS.

In this chapter, we propose an integer programming (IP) formulation for this problem that uses an exponential number of cut-like inequalities. As we will see, a relatively simple implementation of it significantly outperforms a polynomial-size formulation that we introduce. This second formulation has $O(sn^2)$ variables and $O(snm)$ constraints and applies when the distances are hop-based, where n and m denote the number of vertices and edges, respectively.

In contrast, the cut-like formulation applies when there are weighted delays.

Previous Work

The minimum CDS problem is a well-studied NP-hard problem (Garey and Johnson, 1979) in which the task is to find a CDS of minimum cardinality. For example, Figure 2.2(a) provides a CDS and Figure 2.2(b) provides a *minimum* CDS. The reader is encouraged to consult the book by Du and Wan (2013) for motivating applications, approximation algorithms, and hardness results. There are a number of IP formulations and implementations for the minimum CDS problem and for the equivalent maximum-leaf spanning tree problem (Lucena et al., 2010; Simonetti et al., 2011; Morgan and Grout, 2008; Fan and Watson, 2012; Fujie, 2004; Gendron et al., 2014; Buchanan et al., 2015). See also the literature on the regenerator location problem (Chen et al., 2010, 2015; Li and Aneja, 2017). To our knowledge, the state-of-the-art IP formulation and implementation are due to Fujie (2004) and Buchanan et al. (2015), respectively, although several of the previously mentioned approaches work well. As far as we know, the only previous work to propose an IP formulation for a latency-constrained variant of the CDS problem is by Buchanan et al. (2014); however, it is for dominating s-clubs.

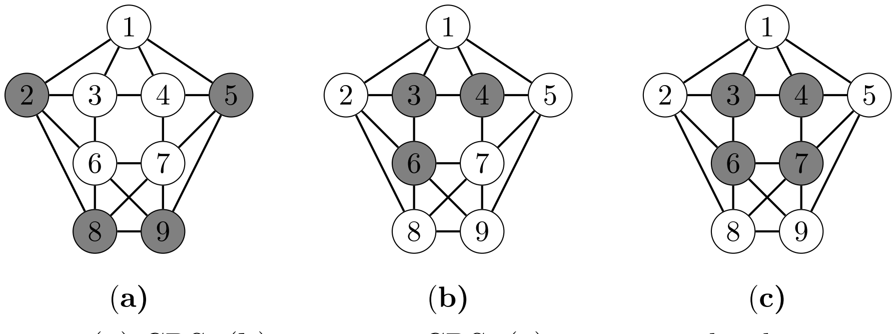


Figure 2.2: (**a**) CDS; (**b**) *minimum* CDS; (**c**) 2-connected 2-dominating set.

Assuming the input graph is not complete, the minimum CDS problem can be formulated as an IP as follows, where x_i is a binary variable representing the decision to include vertex

i in the CDS. A vertex cut is a subset $C \subset V$ of vertices such that $G - C := G[V \setminus C]$ is disconnected and nontrivial (i.e., has at least two nodes).

$$\min \sum_{i \in V} x_i \tag{2.1}$$

$$\sum_{i \in C} x_i \geq 1, \ \forall \text{ vertex cut } C \subset V \tag{2.2}$$

$$x_i \in \{0, 1\}, \ \forall i \in V. \tag{2.3}$$

This particularly elegant formulation is essentially due to Fujie (2004), and its linear programming relaxation can be solved in polynomial time despite having exponentially many constraints, as the separation problem for the vertex cut constraints (2.2) can be solved in polynomial time. The implementation of Buchanan et al. (2015) used this formulation to solve 42 of 47 standard test instances each in under 10 seconds (and never taking longer than 500 seconds), whereas no earlier approach solved 42 instances each in a 1-hour time limit. In their implementation, Buchanan et al. (2015) add violated vertex cut inequalities on-the-fly, cutting off infeasible *integer* points. Our proposed formulation in this chapter, which generalizes Fujie's formulation, is implemented in the same manner.

One drawback of a CDS is that it can be vulnerable to node or arc failures. For example, consider the minimum CDS from Figure 2.2(b). If node 3 fails, this renders the virtual backbone inoperative as it no longer can relay information (say, from node 5 to node 8).

This motivates the notion of a fault-tolerant CDS—one that remains a CDS when fewer than k nodes fail. This has been called a k-connected k-dominating set (k-k-CDS) as it can equivalently be defined as a subset $S \subseteq V$ of vertices such that $G[S]$ is k-vertex-connected and every vertex of $V \setminus S$ has k neighbors in S. Figure 2.2(c) gives a 2-2-CDS, which remains a CDS if one vertex fails. The associated optimization problem, the minimum k-k-CDS problem, admits the following formulation (Buchanan et al., 2015; Ahn and Park, 2015).

$$\min \sum_{i \in V} x_i \tag{2.4}$$

$$\sum_{i \in C} x_i \geq k, \ \forall \text{ vertex cut } C \subset V \tag{2.5}$$

$$x_i \in \{0,1\}, \ \forall i \in V. \tag{2.6}$$

The formulations that we propose in this chapter generalize this k-k-CDS formulation as well as Fujie's CDS formulation.

Notation and Terminology

From now on, unless stated otherwise, $G = (V, E)$ will be a directed graph, with vertex set V and edge set $E \subset V \times V$, that has no loops and no parallel edges. By "no parallel edges", we mean that there is at most one directed edge from a vertex i to a vertex j, and so we can refer to it by the notation (i, j). Here, i is called the tail and j is the head. Frequently, we *bidirect* an undirected edge, which we define to be the operation in which an undirected edge $\{i, j\}$ is replaced by its directed counterparts (i, j) and (j, i). When the edges of G are reciprocated, i.e., if $(i, j) \in E$ implies $(j, i) \in E$, then we say that G is bidirected—not to be confused with the bidirected graphs of Edmonds and Johnson (1970), see also Schrijver (2003).

For each edge $e \in E$ of G, there is an associated nonnegative weight w_e representing the delay. In the *hop-based* case, each weight is one. The distance from vertex a to vertex b in graph G, denoted $\text{dist}_G(a, b)$, is the length of a shortest path from a to b in G, edge-weighted by w. Convention states that if there is no path from a to b in G, then $\text{dist}_G(a, b) = \infty$. The diameter of G, denoted $\text{diam}(G)$, is the maximum of these pair-wise distances, i.e., $\text{diam}(G) := \max\{\text{dist}_G(a, b) \mid a, b \in V\}$.

The out-neighborhood and in-neighborhood of a vertex $v \in V$ in G are denoted $N_G^+(v) := \{w \in V \mid (v,w) \in E\}$ and $N^-(v) := \{u \in V \mid (u,v) \in E\}$, respectively. For a vertex subset S, $\delta_G^+(S)$ denotes the subset of edges whose tail belongs to S and whose head does not. Similarly, $\delta_G^-(S)$ denotes the subset of edges whose head belongs to S but whose tail does not. For a singleton $S = \{v\}$, let $\delta_G^+(v) := \delta_G^+(\{v\})$ and $\delta_G^-(v) := \delta_G^-(\{v\})$. When the graph G in question is clear, the subscripts G in $N_G^+(\cdot)$, $N_G^-(\cdot)$, $\delta_G^+(\cdot)$, and $\delta_G^-(\cdot)$ are omitted. The subset of edges having both endpoints in $S \subseteq V$ is denoted $E(S) := \{(i,j) \in E \mid i,j \in S\}$.

Our Contributions

In Section 2.1, we examine the complexity of latency-s CDS's. Specifically, we answer questions like: How quickly can one verify that a given subset of vertices is a latency-s CDS? And, how hard is the minimum latency-s CDS problem?

In Section 2.2, we propose IP formulations for the minimum latency-s CDS problem. The first formulation, which we call CUT, has n binary variables and an exponential number of cut-like constraints. We then generalize this formulation so that it models the fault-tolerant variant in which one seeks a latency-s CDS that maintains feasibility after a small number of vertex failures. Then, we give a second IP formulation, which we call POLY, that has $O(sn^2)$ variables and $O(snm)$ constraints. It serves as a baseline for computational comparisons.

In Section 2.3, we examine the complexity of the separation problem associated with formulation CUT. Specifically, we show that, under hop-based distances, it is polynomial-time solvable for $s \in \{2,3,4\}$ and NP-hard when $s \geq 5$. En route to proving this, we answer an open question of Xu et al. (2005), by showing that it is indeed NP-hard to compute a graph's fault diameter.

In Section 2.4, we perform computational experiments. Our results demonstrate that a branch-and-cut implementation of formulation CUT significantly outperforms the polynomial-size formulation POLY. Notably, CUT makes easy work of a real-life instance with 300 nodes, while formulation POLY struggles to solve instances with 50 nodes in an hour.

2.1 The Complexity of Latency-s CDS

In this section, we examine the complexity of latency-s CDS's. First, we pinpoint the complexity of verifying feasible solutions, showing essentially that a quadratic running time is unavoidable under a plausible complexity assumption. Then, we establish the inapproximability of the minimum latency-s CDS problem.

2.1.1 The Complexity of Verifying Feasible Solutions

To verify that a given subset $D \subseteq V$ of vertices is a latency-s CDS, we can compute, for each vertex $v \in V$, the shortest paths from v to all other nodes $t \in V \setminus \{v\}$ and check that these paths are short enough. However, we are not interested in just any paths from v to t; these paths must not cross vertices from $V \setminus D$. In our proposed approach, we solve an instance of the single source shortest path problem (SSSP) in the subgraph $\overrightarrow{G}_v^D = (V, \overrightarrow{E}_v^D)$, which has edge set

$$\overrightarrow{E}_v^D := E(D) \cup \delta^+(D) \cup \left(\delta^+(v) \cap \delta^-(D)\right). \tag{2.7}$$

Here, we preserve the edges $E(D)$ that have both endpoints in D, those edges $\delta^+(D)$ that point out of D, and those edges $\delta^+(v) \cap \delta^-(D)$ whose tail is v and whose head is in D. This set $\overrightarrow{E}_v^D$ includes all edges that might be used in a suitable path from v to another node. Of course, we need not create the graph $\overrightarrow{G}_v^D$ in the implementation, as nearly any shortest path algorithm can be reconfigured to work implicitly on $\overrightarrow{G}_v^D$ when given G, D, and v.

IsLatencyConstrainedCDS(G, D, s**):**

```
1. for each $v \in V$ do
```

(a) compute shortest paths from v in $\overrightarrow{G}_v^D$;

(b) if $\text{dist}_{\overrightarrow{G}_v^D}(v,t) > s$ for some $t \in V \setminus \{v\}$, then return "no";

2. return "yes".

Proposition 1. *The algorithm* IsLatencyConstrainedCDS *correctly determines whether a given subset $D \subseteq V$ of vertices is a latency-s CDS for a directed graph $G = (V, E)$:*

- *in $O(mn + n^2 \log\log n)$ time and linear space under nonnegative edge weights;*
- *in $O(mn)$ time and linear space in the hop-based case.*

Here, we are using the algorithm of Thorup (2004) to compute SSSP in time $O(m + n \log\log n)$ in the nonnegative weights case, and BFS to solve SSSP in the hop-based case.

Given that this or some other verification procedure will be called repeatedly in our implementation, it is important that it runs as quickly as possible. For example, we would like to know: is there a different verification procedure that, say, runs in linear time $O(m+n)$? Unfortunately, under a complexity assumption called the strong exponential time hypothesis (SETH) of Impagliazzo et al. (2001) and Impagliazzo and Paturi (2001), this is not possible. SETH is an unproven complexity assumption that is stronger than P≠NP. While it is unproven and some doubt that it is true, it is nevertheless a benchmark for gauging how surprising or noteworthy a faster algorithm would be. The reader is referred to the survey of Lokshtanov et al. (2013) for more information about SETH. To prove our results, we use the following theorem of Roditty and Vassilevska Williams (2013).

Theorem 1. *If SETH holds, then for every $\varepsilon > 0$ there is no algorithm for verifying that a simple, connected graph $G = (V, E)$ has* $\text{diam}(G) = 2$ *that runs in time $O(m^{2-\varepsilon})$.*

Proposition 2. *If SETH holds, then for every $\varepsilon > 0$ there exists no algorithm for verifying that a subset D of vertices is a latency-s CDS that runs in time $O(m^{2-\varepsilon})$, even in the simplest nontrivial case of hop-based distances and $s = 2$.*

Proof. The proof follows by reduction from the problem in Theorem 1. Namely, bidirect the edges of the graph to get $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$ and let $s = 2$, $D = V$, and $w_e = 1$ for each $e \in \overleftrightarrow{E}$. It can be observed that D is a latency-s CDS for $\overleftrightarrow{G}$ if and only if G has $\text{diam}(G) = 2$, and this reduction runs in linear time, so the proposition follows. □

A similar negative result shows a difficulty that would be encountered when applying local search to the minimum latency-s CDS problem. In the following proposition, we refer to the local search move in which one is given a known-to-be-feasible solution $D \subseteq V$ along with a vertex $v \in D$, and the task is to determine whether one can move to $D \setminus \{v\}$ and maintain feasibility.

Proposition 3. *If SETH holds, then for every $\varepsilon > 0$ there exists no algorithm for the local search move defined above that runs in time $O(m^{2-\varepsilon})$, even in the simplest nontrivial case of hop-based distances and $s = 2$.*

Proof. The proof follows by reduction from the problem in Theorem 1, where we assume, without loss, that G is not complete and thus $\text{diam}(G) \geq 2$. Add a new node v to the graph $G = (V, E)$ and connect it to all other nodes. Call this new graph $G' = (V', E')$, where $V' = V \cup \{v\}$ and $E' = E \cup \{\{u, v\} \mid u \in V\}$. Bidirect all edges of G' to get $\overleftrightarrow{G}' = (V', \overleftrightarrow{E}')$. Let $s = 2$ and $D = V'$ and $w_e = 1$ for each $e \in \overleftrightarrow{E}'$. Since v is an in-neighbor and an out-neighbor of all other nodes in $\overleftrightarrow{G}'$ and since $v \in D$, it is clear that D is a latency-s CDS for $\overleftrightarrow{G}'$. It can be observed that $D \setminus \{v\}$ is a latency-s CDS for $\overleftrightarrow{G}'$ if and only if G has $\text{diam}(G) = 2$, and this reduction runs in linear time, so the proposition follows. □

2.1.2 The Inapproximability of the Minimum Latency-s CDS Problem

We provide a hardness result for approximating the size of a minimum latency-s CDS. It is based on the hardness result of Dinur and Steurer (2014) which states that approximating the minimum hitting set problem to within a factor of $(1 - \varepsilon) \ln h$ is NP-hard for every $\varepsilon > 0$, where h refers to the number of subsets to hit, cf. Raz and Safra (1997); Alon et al. (2006); Moshkovitz (2012). Also, see similar hardness results based on the stronger assumption that NP does not have quasipolynomial-time algorithms (Lund and Yannakakis, 1994; Feige, 1998). Note that $|U| = O(h^c)$ for some constant c in Dinur and Steurer's result.

Problem: The minimum hitting set problem.

Input: a family $F_1, \ldots, F_h \subseteq U$ of subsets of U.

Output: A minimum cardinality subset $D \subseteq U$ such that $|D \cap F_i| \geq 1$ for every $i = 1, \ldots, h$.

Theorem 2. *There is a polynomial-time algorithm, when given an instance $((F_1, \ldots, F_h), U)$ of the minimum hitting set problem, that creates an instance $(G = (V, E), w, s)$ of the minimum latency-s CDS problem that satisfies:*

- $|V| = 4 + h + |U|$ *and* $s = 2 = \operatorname{diam}(G)$ *and* $w_e = 1$ *for each* $e \in E$*;*
- *there exists a k-hitting set if and only if there exists a $(k + 2)$-vertex latency-s CDS.*

Proof. Let $V = \{r, a, b, c\} \cup T \cup U$, where $T = \{t_1, \ldots, t_h\}$. Thus, $|V| = 4 + h + |U|$. Construct E by bidirecting the following edges. Connect r to every vertex of $U \cup \{a\}$. Connect a to every vertex of U. Connect b to every vertex of $T \cup U$. Make $\{a, b, c\}$ a triangle. Finally, for each F_i in the hitting set instance, connect t_i to every vertex $v \in F_i \subseteq U$.

($\Longrightarrow$) Suppose that $D \subseteq U$ is a hitting set of size k. It can be verified that $D \cup \{a, b\}$ is a latency-2 CDS for G, i.e., that for every ordered pair of nodes (i, j) with $i \neq j$ and $(i, j) \notin E$, there is a node $v \in D \cup \{a, b\}$ such that (i, v) and (v, j) are edges in E.

($\Longleftarrow$) Now, suppose that $D \subseteq V$ is a latency-2 CDS of size $k + 2$. We argue that $D \cap U$ is a hitting set of size at most k. Observe that there is no edge (c, r) and so to ensure 2-hop communication from c to r, D must contain a vertex from $N^+(c) \cap N^-(r)$, and $N^+(c) \cap N^-(r) = \{a\}$ so $a \in D$. Similarly, (c, t_1) is not an edge and $N^+(c) \cap N^-(t_1) = \{b\}$ so $b \in D$. This shows that $|D \cap U| \leq |D| - 2 = k$. Now we show that $D \cap U$ is a hitting set.

Recall that, for each $i = 1, \dots, h$, the edge (r, t_i) does not exist. So, since D is a latency-2 CDS, at least one vertex from $N^+(r) \cap N^-(t_i) = F_i \subseteq U$ must belong to D. Thus, $D \cap U$ is a hitting set of size at most k. □

Corollary 1 (Inapproximability). *There is a constant $\alpha > 0$ such that it is NP-hard to approximate the minimum latency-2 CDS problem to within a factor of $\alpha \ln n$, where n refers to the number of vertices, even under bidirected edges and hop-based distances.*

Proof. This follows by Theorem 2 and the inapproximability of hitting set (Raz and Safra, 1997; Alon et al., 2006; Moshkovitz, 2012; Dinur and Steurer, 2014). □

2.2 Integer Programming Formulations

In what follows, we propose two IP formulations for the minimum latency-s CDS problem: CUT and POLY.

2.2.1 Formulation CUT

Here we propose the formulation called CUT. It has n binary variables and an exponential number of constraints—one for each (minimal) length-s vertex cut.

Definition 4 (length-s vertex cut). *A subset $C \subseteq V$ of vertices is a length-s vertex cut of a directed, edge-weighted graph $G = (V, E)$ if* $\text{diam}(G - C) > s$.

The correctness of formulation CUT is a consequence of the following characterization.

Proposition 4 (Characterization of latency-s CDS). *A subset $D \subseteq V$ of vertices is a latency-s CDS for G if and only if $|D \cap C| \geq 1$ for every length-s vertex cut $C \subset V$.*

Proof. ($\implies$) Assume that $D \subseteq V$ is a latency-s CDS and suppose, for sake of contradiction, that $C \subset V$ is a length-s vertex cut with $|D \cap C| = 0$. By definition of length-s vertex cut, $\text{diam}(G - C) > s$, i.e., there exist vertices $a, b \in V \setminus C$ such that $\text{dist}_{G-C}(a, b) > s$. By assumption that D is a latency-s CDS, there is an a-b path of length at most s whose interior vertices belong solely to D, i.e., $\text{dist}_{G[D\cup\{a,b\}]}(a,b) \leq s$. Since $D \cup \{a,b\} \subseteq V \setminus C$ we have $\text{dist}_{G[V\setminus C]}(a,b) \leq \text{dist}_{G[D\cup\{a,b\}]}(a,b)$ which results in the following contradiction:

$$s < \text{dist}_{G-C}(a,b) \triangleq \text{dist}_{G[V\setminus C]}(a,b) \leq \text{dist}_{G[D\cup\{a,b\}]}(a,b) \leq s.$$

($\Longleftarrow$) By the contrapositive. Suppose that $D \subseteq V$ is not a latency-s CDS, i.e., there exist vertices $a, b \in V$ such that there is no a-b path of length at most s whose interior vertices belong to D, i.e., $\text{dist}_{G[D\cup\{a,b\}]}(a,b) > s$. This implies that $\text{diam}(G[D\cup\{a,b\}]) > s$, and so $C := V \setminus (D \cup \{a,b\})$ is a length-s vertex cut. Moreover, $|D \cap C| = 0$, as desired. $\square$

Proposition 4 immediately implies the correctness of the formulation CUT:

$$\min \sum_{i\in V} x_i \tag{2.8}$$

$$\sum_{i\in C} x_i \geq 1, \ \forall \text{ length-}s \text{ vertex cut } C \subset V \tag{2.9}$$

$$x_i \in \{0,1\}, \ \forall i \in V. \tag{2.10}$$

In general, there can be exponentially many of the constraints (2.9), even if we restrict ourselves to *inclusion-minimal* length-s vertex cuts. This formulation generalizes the CDS formulation based on vertex cuts that is essentially due to Fujie (2004). We address the separation complexity for constraints (2.9) in Section 2.3.

Not every valid inequality of the form $\sum_{i\in C} x_i \geq 1$ is a length-s vertex cut inequality. For example, $\sum_{i\in V\setminus\{v\}} x_i \geq 1$ is valid when $G = (V,E)$ is the bidirected 4-cycle, but $V \setminus \{v\}$ is not a length-s vertex cut. However, the following shows that the length-s vertex cut inequalities are the only *meaningful* valid inequalities of this type.

Lemma 1. *Let $C \subset V$. The inequality $|S \cap C| \geq 1$ holds for every latency-s CDS $S \subseteq V$ if* *and only if C is a* <u>*superset*</u> *of some length-s vertex cut C'.*

Proof. The 'if' direction follows easily by Proposition 4, so suppose that $|S \cap C| \geq 1$ holds for every latency-s CDS $S \subseteq V$. Let $D = V \setminus C$. By our assumption, D cannot be a latency-s CDS, i.e., there exist vertices $a, b \in V$ such that $\text{dist}_{G[D \cup \{a,b\}]}(a, b) > s$. So, $s < \text{diam}(G[D \cup \{a, b\}]) = \text{diam}(G - C')$, where $C' = V \setminus (D \cup \{a, b\})$. Thus, C' is a length-s vertex cut for G, and $C \supseteq C'$, as desired. □

Given that the minimum latency-s CDS problem admits the formulation CUT (and by Lemma 1), there are immediate polyhedral consequences (cf. Sassano (1989)), so we provide the following proposition without proof.

Proposition 5 (Basic polyhedral analysis)**.** *The convex hull of (characteristic vectors of) latency-s CDS's is full-dimensional if and only if every length-s vertex cut has size at least two. Further, if it is full-dimensional, then*

1. *for each $v \in V$,*
   (a) *$x_v \leq 1$ induces a facet;*
   (b) *$x_v \geq 0$ induces a facet if and only if v does not belong to a length-s vertex cut of size two.*
2. *for $C \subset V$, the inequality $\sum_{i \in C} x_i \geq 1$ induces a facet if and only if*
   (a) *C is a minimal length-s vertex cut, and*
   (b) *for each $v \in V \setminus C$ there exists $c \in C$ such that $(V \setminus C) \cup \{c\} \setminus \{v\}$ is a latency-s CDS.*

2.2.2 Generalizing the Formulation CUT for Fault-Tolerance

Here, we consider the *robust* or *fault-tolerant* variant of a latency-s CDS. That is, we are interested in a vertex subset that remains a latency-s CDS when few vertices fail.

Definition 5. *A subset $D \subseteq V$ of vertices is an* <u>*r-robust latency-s CDS*</u> *for graph G if, for every $F \subseteq D$ with $|F| < r$, the vertex subset $D \setminus F$ is a latency-s CDS for G.*

A consequence of Proposition 4 is the following characterization.

Corollary 2 (Characterization of r-robust latency-s CDS)**.** *A subset $D \subseteq V$ of vertices is an r-robust latency-s CDS if and only if $|D \cap C| \geq r$ for every length-s vertex cut $C \subset V$.*

Corollary 2 immediately implies the correctness of the following formulation for the minimum r-robust latency-s CDS problem:

$$\min \sum_{i \in V} x_i \tag{2.11}$$

$$\sum_{i \in C} x_i \geq r, \ \forall \text{ length-}s \text{ vertex cut } C \subset V \tag{2.12}$$

$$x_i \in \{0,1\}, \ \forall i \in V. \tag{2.13}$$

This formulation generalizes previously existing formulations for the minimum k-k-CDS problem (Ahn and Park, 2015; Buchanan et al., 2015).

Figure 2.1(b) gives a feasible solution to this problem when $(s, r) = (3, 2)$ (using hop-based distances and treating the undirected edges as bidirected edges). That is, the gray vertices remain a latency-3 CDS when one of them fails. This is also an optimal solution for $(s, r) = (2, 1)$. However, there is no solution for $(s, r) = (2, 2)$, as evidenced by the length-2 vertex cut $C = \{5\}$. Indeed, this implies that the inequality $x_5 \geq 2$ is valid, but of course no binary vector x can satisfy this constraint.

We remark that our formalization of a fault-tolerant low-latency virtual backbone might *not* provide for r vertex-disjoint paths of length at most s (of CDS vertices) between every pair of vertices. An example is given in Figure 2.3, treating the undirected edges as bidirected edges. This should not be surprising given that there is, in general, no "Menger's theorem" for length-bounded paths, cf. Lovász et al. (1978).

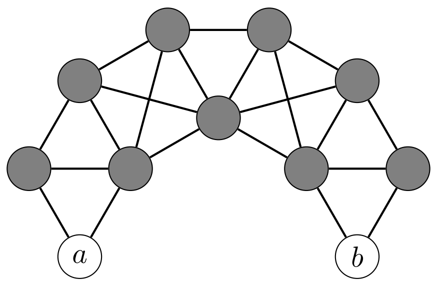


Figure 2.3: A CDS that maintains 5-hop communication paths when any one vertex fails, but that does <u>not</u> have a pair of vertex-disjoint length-5 a-b paths. Observe that this is a unit disk graph.

2.2.3 Formulation POLY

Here we propose the formulation called POLY. It is introduced primarily for comparison purposes and is inspired by a formulation for s-clubs given by Veremyev and Boginski (2012). It applies to hop-based case.

As before, the binary variable x_i represents the decision to include vertex i in the latency-s CDS. The binary variable y^t_{ij} equals one if and only if there exists a directed path in G from i to j of length exactly t whose interior vertices belong to the chosen CDS. This variable is only defined when $t \geq 2$ and should not be confused with y_{ij} raised to the t-th power. To formulate our problem, we should write constraints that impose the following condition:

$$y^t_{ik} = 1 \iff \big(\text{there exists } j \in N^-(k) \text{ such that } y^{t-1}_{ij} = 1 \text{ and } x_j = 1\big).$$

In words, there is a path (across CDS nodes) from i to k of length t if and only if (i) there is a path (across CDS nodes) of length $t-1$ from i to some in-neighbor j of node k, and (ii) node j belongs to the CDS. When $t \geq 3$, this equivalence can be formulated as follows.

$$\begin{aligned} &(\Longleftarrow) && y^{t-1}_{ij} + x_j \leq y^t_{ik} + 1 && \forall j \in N^-(k) \\ &(\Longrightarrow) && y^t_{ik} \leq \sum_{j \in N^-(k)} y^{t-1}_{ij} x_j. && \end{aligned}$$

The second implication is enforced via a constraint that has products of binary variables. For linearization purposes, introduce (binary) variables z_{ij}^{t-1} to replace the terms $y_{ij}^{t-1}x_j$. To impose that $z_{ij}^{t-1} = y_{ij}^{t-1}x_j$, use the usual linear constraints:

$$\begin{aligned}
&z_{ij}^{t-1} \leq y_{ij}^{t-1}\\
&z_{ij}^{t-1} \leq x_j\\
&y_{ij}^{t-1} + x_j \leq z_{ij}^{t-1} + 1.
\end{aligned}$$

These ideas lead to the following formulation, where the special case $t = 2$ is handled via constraints (2.15) and (2.16). Let $T_{\geq 3} := \{3, \dots, s\}$ and $N^{-}[j] := N^{-}(j) \cup \{j\}$.

$$\min \sum_{i \in V} x_i \tag{2.14}$$

$$x_j \leq y_{ik}^2 \qquad j \in N^+(i) \cap N^-(k),\ i \in V \setminus \{k\},\ k \in V \tag{2.15}$$

$$y_{ik}^2 \leq \sum_{j \in N^+(i) \cap N^-(k)} x_j \qquad i \in V \setminus \{k\},\ k \in V \tag{2.16}$$

$$y_{ij}^{t-1} + x_j \leq y_{ik}^t + 1 \qquad i \in V \setminus \{j,k\},\ (j,k) \in E,\ t \in T_{\geq 3} \tag{2.17}$$

$$y_{ik}^t \leq \sum_{j \in N^-(k)} z_{ij}^{t-1} \qquad i \in V \setminus \{k\},\ k \in V,\ t \in T_{\geq 3} \tag{2.18}$$

$$z_{ij}^{t-1} \leq y_{ij}^{t-1} \qquad i \in V \setminus \{j\},\ j \in V,\ t \in T_{\geq 3} \tag{2.19}$$

$$z_{ij}^{t-1} \leq x_j \qquad i \in V \setminus \{j\},\ j \in V,\ t \in T_{\geq 3} \tag{2.20}$$

$$y_{ij}^{t-1} + x_j \leq z_{ij}^{t-1} + 1 \qquad i \in V \setminus \{j\},\ j \in V,\ t \in T_{\geq 3} \tag{2.21}$$

$$\sum_{t=2}^{s} y_{ij}^t \geq 1 \qquad i \in V \setminus N^-[j],\ j \in V \tag{2.22}$$

$$x_i \in \{0,1\} \qquad i \in V \tag{2.23}$$

$$y_{ij}^t \in \{0,1\} \qquad i \in V \setminus \{j\},\ j \in V,\ t \in \{2, \dots, s\} \tag{2.24}$$

$$z_{ij}^t \in \{0,1\} \qquad i \in V \setminus \{j\},\ j \in V,\ t \in \{2, \dots, s-1\}. \tag{2.25}$$

The constraints (2.22) ensure that there is a path of length at most s (across CDS vertices) from i to j when $(i,j) \notin E$. So, by the ideas presented above, it is straightforward to prove the following.

Theorem 3. *Under hop-based distances, the above is a correct formulation for the minimum latency-s CDS problem and has $\Theta(sn^2)$ variables, $\Theta(snm)$ constraints, and $\Theta(snm)$ nonzeros.*

Since modern MIP solvers use sparse matrix representation, this formulation's size in computer memory can be approximated by the number $\Theta(snm)$ of nonzeros. This is much less than the quantity obtained by multiplying the number of variables by the number of constraints.

Not all of these variables and constraints may be necessary. For example, if G is bidirected, we can assume that $y^t_{ij} = y^t_{ji}$. If desired, the user can impose these constraints $y^t_{ij} = y^t_{ji}$ when implementing the formulation, and the MIP solver will perform the appropriate substitutions in its presolve phase.

Based on our computational experiments, it is possible that formulation POLY is weaker than CUT, although we could not find a proof. Figure 2.4 shows that POLY cannot be stronger than CUT. We were unable to prove/disprove that CUT is stronger than POLY.

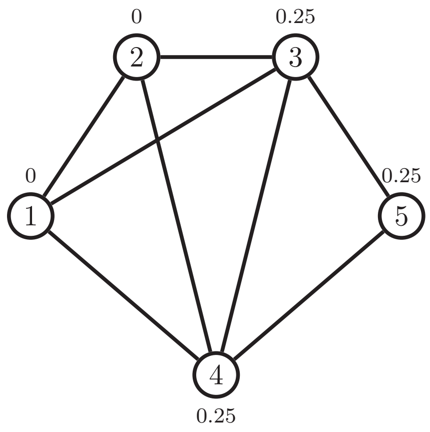


Figure 2.4: A graph with nodes numbered $\{1, 2, 3, 4, 5\}$ and values for x^* given by the nodes. When the edges shown are treated as bidirected and unit-weighted, this vector x^* is infeasible for formulation CUT (for every s), since the length-s vertex cut inequality for $C = \{3, 4\}$ is violated. However, there exist values y^* and z^* for which (x^*, y^*, z^*) satisfies formulation POLY when $s = 3$. Indeed, these values for x^* are *optimal* for POLY's LP relaxation. The solve logs, obtained via Gurobi, are below.

```
Optimize a model with 230 rows, 80 columns and 590 nonzeros

Coefficient statistics:

  Matrix range     [1e+00, 1e+00]

  Objective range  [1e+00, 1e+00]

  Bounds range     [0e+00, 0e+00]

  RHS range        [1e+00, 1e+00]
```

```
Presolve removed 131 rows and 45 columns

Presolve time: 0.00s

Presolved: 99 rows, 35 columns, 241 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time

       0    0.0000000e+00   2.000000e+00   0.000000e+00      0s

      20    5.0000000e-01   0.000000e+00   0.000000e+00      0s

Solved in 20 iterations and 0.00 seconds

Optimal objective  5.000000000e-01


X_1 = 0

X_2 = 0

X_3 = 0.25

X_4 = 0.25

X_5 = 0

Y_{1,2}^2 = 0.25

Y_{1,3}^2 = 0.25

Y_{1,4}^2 = 0.25

Y_{1,5}^2 = 0.5

Y_{2,1}^2 = 0.25

Y_{2,3}^2 = 0.25

Y_{2,4}^2 = 0.25

Y_{2,5}^2 = 0.5

Y_{3,1}^2 = 0.25

Y_{3,2}^2 = 0.25

Y_{3,4}^2 = 0

Y_{3,5}^2 = 0.25
```

```
Y_{4,1}^2 = 0.25

Y_{4,2}^2 = 0.25

Y_{4,3}^2 = 0

Y_{4,5}^2 = 0.25

Y_{5,1}^2 = 0.5

Y_{5,2}^2 = 0.5

Y_{5,3}^2 = 0.25

Y_{5,4}^2 = 0.25

Y_{1,2}^3 = 0

Y_{1,3}^3 = 0

Y_{1,4}^3 = 0

Y_{1,5}^3 = 0.5

Y_{2,1}^3 = 0

Y_{2,3}^3 = 0

Y_{2,4}^3 = 0

Y_{2,5}^3 = 0.5

Y_{3,1}^3 = 0

Y_{3,2}^3 = 0

Y_{3,4}^3 = 0

Y_{3,5}^3 = 0

Y_{4,1}^3 = 0

Y_{4,2}^3 = 0

Y_{4,3}^3 = 0

Y_{4,5}^3 = 0

Y_{5,1}^3 = 0.5

Y_{5,2}^3 = 0.5
```

```
Y_{5,3}^3 = 0

Y_{5,4}^3 = 0

Z_{1,2}^2 = 0

Z_{1,3}^2 = 0

Z_{1,4}^2 = 0

Z_{1,5}^2 = 0

Z_{2,1}^2 = 0

Z_{2,3}^2 = 0

Z_{2,4}^2 = 0

Z_{2,5}^2 = 0

Z_{3,1}^2 = 0.25

Z_{3,2}^2 = 0.25

Z_{3,4}^2 = 0

Z_{3,5}^2 = 0.25

Z_{4,1}^2 = 0.25

Z_{4,2}^2 = 0.25

Z_{4,3}^2 = 0

Z_{4,5}^2 = 0.25

Z_{5,1}^2 = 0

Z_{5,2}^2 = 0

Z_{5,3}^2 = 0

Z_{5,4}^2 = 0
```

2.3 The Complexity of the Formulations

In this section, we determine the separation complexity for the constraints defining formulation CUT and its fault-tolerant generalization. On the way, we answer an open question of Xu et al. (2005) regarding the complexity of computing a graph's fault diameter.

2.3.1 Computing the Fault Diameter of Graphs

As a helpful first step to determining the separation complexity, we show that a related problem, which we call Diameter Interdiction by Node Deletion, is NP-complete.

Problem: Diameter Interdiction by Node Deletion.
Input: a simple graph $G = (V, E)$ and integers q and L.
Question: Is there a subset $C \subset V$ of q vertices such that $\text{diam}(G - C) > L$?

This problem is defined for an undirected and unweighted graph G, and the diameter that is referred to is hop-based.

Theorem 4. *For each $L \geq 5$,* Diameter Interdiction by Node Deletion *is NP-complete.*

To prove this theorem, we craft reductions from Length-Bounded a-b Node Cut, which is known to be NP-complete and hard to approximate (Baier et al., 2010). Notice that this problem has specified end nodes a and b, while Diameter Interdiction by Node Deletion does not. We provide two reductions, given in Lemma 2 and Lemma 3, that together prove Theorem 4.

Problem: Length-Bounded a-b Node Cut.
Input: A simple graph $G' = (V', E')$, nonadjacent $a, b \in V'$, and integers q' and L'.
Question: Is there a subset $C' \subseteq V' \setminus \{a, b\}$ of q' vertices such that $\text{dist}_{G'-C'}(a, b) > L'$?

Lemma 2. *For each $\underline{\text{odd}}$ $L \geq 5$,* Diameter Interdiction by Node Deletion *is NP-complete.*

Proof. Membership in NP is obvious. For the reduction, consider an instance of Length-Bounded a-b Node Cut defined by graph $G' = (V', E')$, vertices $a, b \in V'$, and integers

q' and odd $L' \geq 5$. Let $q = q'$ and $L = L'$. Now we construct $G = (V, E)$. The idea is to connect every pair of vertices from G' (besides a and b) by carefully adding many short paths so that the only possible way to cheaply disrupt the diameter of G is to cut all short paths from a to b. Construct V as follows.

$$\begin{aligned}
V &:= V' \cup T \cup A \cup B \cup W \\
T &:= \{t_i \mid 1 \leq i \leq q+1\} \\
A &:= \left\{ a_i^j \;\middle|\; 1 \leq i \leq q+1,\ 1 \leq j \leq \frac{L-1}{2} \right\} \\
B &:= \left\{ b_i^j \;\middle|\; 1 \leq i \leq q+1,\ 1 \leq j \leq \frac{L-1}{2} \right\} \\
W &:= \left\{ v_i^j \;\middle|\; 1 \leq i \leq q+1,\ 1 \leq j \leq \frac{L-3}{2},\ v \in V' \setminus \{a, b\} \right\}.
\end{aligned}$$

Notice that $|V| = O(qL|V'|)$ and $q \leq |V'|$ and $L \leq |V'|$, so the reduction will be polynomial.

Construct the edge set E of G as follows. First, connect the vertices of V' so that $G[V'] = G'$. Then make T a clique in G. Similarly, make each $A^j := \{a_i^j \mid 1 \leq i \leq q+1\}$ a clique. Do the same for each $B^j := \{b_i^j \mid 1 \leq i \leq q+1\}$ and for each $W^j(v) := \{v_i^j \mid 1 \leq i \leq q+1\}$. Connect a to every vertex of A^1; and every vertex of A^1 to every vertex of A^2; and so on. Connect b to every vertex of B^1; and every vertex of B^1 to every vertex of B^2; and so on. Then for every $v \in V' \setminus \{a, b\}$, connect v to every vertex of $W^1(v)$; and every vertex of $W^1(v)$ to every vertex of $W^2(v)$; and so on. Finally, letting $p = \frac{L-1}{2}$, connect every vertex of T to every vertex of $A^p \cup B^p \cup \left(\cup_{v \in V' \setminus \{a,b\}} W^{p-1}(v)\right)$. Creating E obviously can be done in polynomial time. See Figure 2.5 for an illustration.

Observe that there exist at least $q+1$ (internally) node-disjoint paths of length at most L between every pair of vertices of G (the interior vertices of which belong to $V \setminus V'$), except possibly for the pair $\{a, b\}$. Moreover, (simple) a-b paths of length at most L in G can only cross vertices of V'. Thus, it can be argued that the instance (G', a, b, q', L')

of LENGTH-BOUNDED a-b NODE CUT is a "yes" if and only if the instance (G, q, L) of DIAMETER INTERDICTION BY NODE DELETION is a "yes." $\square$

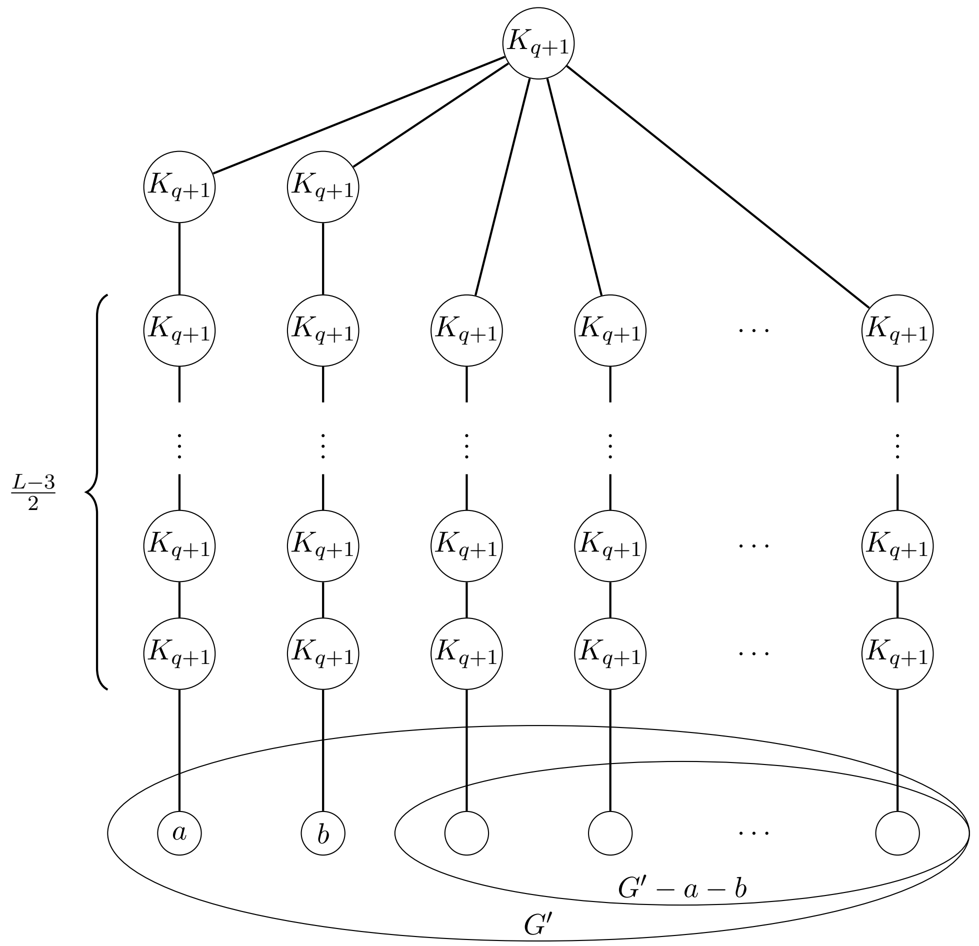


Figure 2.5: Illustration of the reduction for odd $L \geq 5$. Here, K_n is a complete graph on n nodes.

Lemma 3. *For each* <u>*even*</u> $L \geq 5$*,* DIAMETER INTERDICTION BY NODE DELETION *is NP-complete.*

Proof. Membership in NP is obvious. For the reduction, consider an instance of LENGTH-BOUNDED a-b NODE CUT defined by graph $G' = (V', E')$, vertices $a, b \in V'$, and integers q' and even $L' \geq 5$. Let $q = q'$ and $L = L'$. Now we construct $G = (V, E)$. The main idea behind the reduction is the same as before, but the construction is slightly different. Construct V as follows.

$$\begin{aligned} V &:= V' \cup T \cup T' \cup W \\ T &:= \{t_i \mid 1 \leq i \leq q+1\} \\ T' &:= \{t'_i \mid 1 \leq i \leq q+1\} \\ W &:= \left\{ v_i^j \;\middle|\; 1 \leq i \leq q+1,\ 1 \leq j \leq \frac{L}{2} - 1,\ v \in V' \right\}. \end{aligned}$$

Notice that $|V| = O(qL|V'|)$ and $q \leq |V'|$ and $L \leq |V'|$, so the reduction will be polynomial.

Construct the edge set E of G as follows. First, connect the vertices of V' so that $G[V'] = G'$. Then make $T \cup T'$ a clique in G. Similarly, make each $W^j(v) := \{v_i^j \mid 1 \leq i \leq q+1\}$ a clique. For every $v \in V'$, connect v to every vertex of $W^1(v)$; and every vertex of $W^1(v)$ to every vertex of $W^2(v)$; and so on. Let $p = \frac{L}{2} - 1$. Connect every vertex of T to every vertex of $\cup_{v \in V' \setminus \{b\}} W^p(v)$. Similarly, connect every vertex of T' to every vertex of $\cup_{v \in V' \setminus \{a\}} W^p(v)$. Creating E obviously can be done in polynomial time. See Figure 2.6 for an illustration.

Observe that there exist at least $q+1$ (internally) node-disjoint paths of length at most L between every pair of vertices of G (the interior vertices of which belong to $V \setminus V'$), except possibly for the pair $\{a, b\}$. Moreover, (simple) a-b paths of length at most L in G can only cross vertices of V'. Thus, it can be argued that the instance (G', a, b, q', L') of Length-Bounded a-b Node Cut is a "yes" if and only if the instance (G, q, L) of Diameter Interdiction by Node Deletion is a "yes." $\square$

Researchers have studied related notions of the *fault diameter* of a graph (Krishnamoorthy and Krishnamurthy, 1987; Xu, 2001). For example, Xu (2001) defines the f-fault diameter of

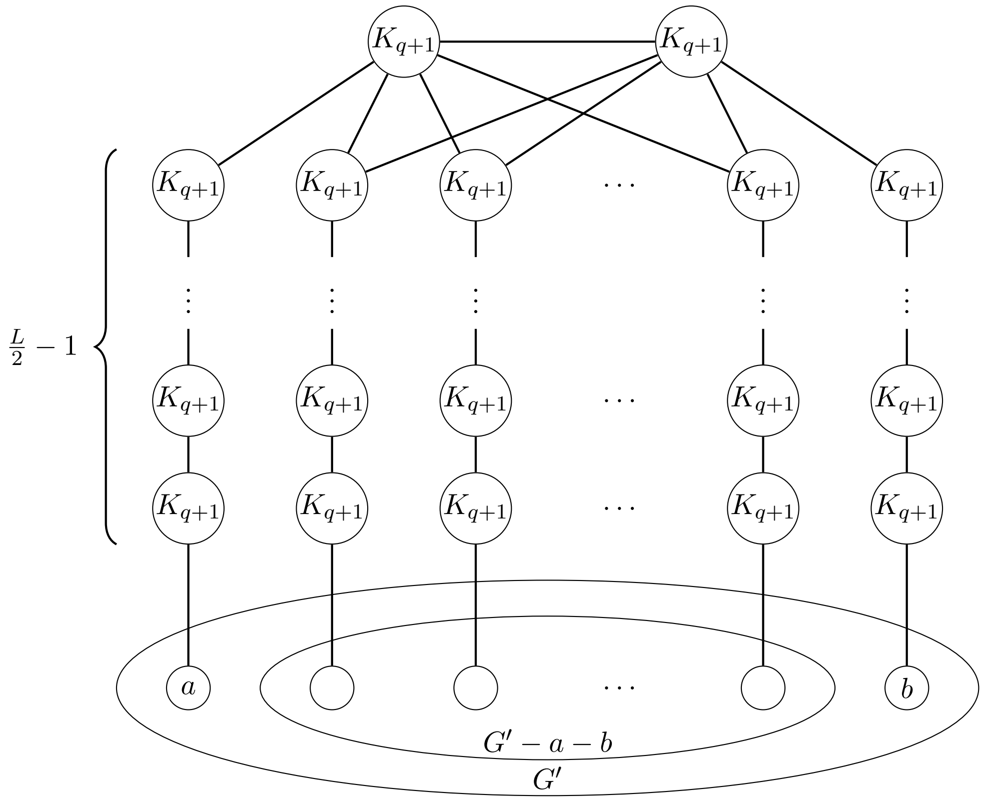


Figure 2.6: Illustration of the reduction for even $L \geq 5$.

graph $G = (V, E)$ to be

$$D_f(G) := \max\{\text{diam}(G - F) \mid F \subseteq V,\ |F| < f\},$$

and states that computing this value "is a quite difficult problem," but no justification is given[3]. Later, Xu et al. (2005) listed its NP-hardness as an open problem. Theorem 4 implies that computing $D_f(G)$ is indeed NP-hard when f is part of the input, say, by letting $f = q+1$ and returning "yes" if $D_f(G) > L$.

Corollary 3. *Computing the f-fault diameter is NP-hard when f is part of the input.*

2.3.2 The Separation Problem for CUT

Formulation CUT has an exponential number of constraints (2.9), as does its fault-tolerant generalization (2.12), making it a nontrivial question as to how they should be used. A helpful observation, however, is that by the polynomial equivalence of optimization and separation (Grötschel et al., 1993), their LP relaxations can be solved in polynomial time if and only if their separation problems (defined below) can be solved in polynomial time.

Problem: Separation Problem for Formulation CUT.
Input: a directed and edge-weighted graph $G = (V, E)$, a weight $x^*_v \in [0, 1]$ for each $v \in V$, an integer $r \geq 1$, a number s.
Output: (if any exist) a length-s vertex cut $C \subseteq V$ with $\sum_{i \in C} x^*_i < r$.

For purposes of generality, we define this separation problem for the fault-tolerant gener-

[3] Schoone et al. (1987) show a related result for increasing the diameter by *edge deletions*, but it is not clear how to modify their result for our purposes. Their definition of the problem (strangely) only allows edge deletions that maintain connectivity of the graph. This allows them to perform a reduction from Hamiltonian Path by seeking subsets of $m - (n-1)$ edges whose removal increases the diameter to $n - 1$. We feel that this is an unsatisfying hardness reduction since a minimum cut likely has fewer edges, and its removal would make the diameter infinite. In contrast, our diameter parameter can be a small constant, and we allow for arbitrary vertex deletions.

alization, which has right-hand-side r. We provide both positive and negative results.

Theorem 5. *Under hop-based distances, the separation problem is:*

1. *polynomial-time solvable for $s \in \{2, 3, 4\}$, for every $r \geq 1$;*
2. *(in its decision version) NP-complete for every $s \geq 5$, even when $r = 1$.*

Proof. First, we prove that item 2 holds. Membership in NP is clear, so we only show hardness. The reduction is from an instance of Diameter Interdiction by Node Deletion given by (G, q, L), which is NP-complete for each $L \geq 5$ by Theorem 4. Bidirect $G = (V, E)$ yielding directed graph $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$. Let $r = 1$, $s = L$, and $x_i^* = \frac{1}{q+1}$ for every $i \in V$. We argue that (G, q, L) is a "yes" instance of Diameter Interdiction by Node Deletion if and only if $(\overleftrightarrow{G}, x^*, r, s)$ admits a violated length-s vertex cut inequality (2.12). Suppose there is a violated length-s vertex cut inequality (2.12) for some $C \subseteq V$. Then, $\frac{|C|}{q+1} = \sum_{i \in C} x_i^* < 1$, i.e., $|C| \leq q$, and the instance of Diameter Interdiction by Node Deletion is a "yes." Now, if there is a length-s vertex cut $C' \subseteq V$ with $|C'| \leq q$ for $\overleftrightarrow{G}$, then $\sum_{i \in C'} x_i^* = \frac{|C'|}{q+1} \leq \frac{q}{q+1} < 1$ and so x^* violates the length-s vertex cut inequality $\sum_{i \in C'} x_i \geq 1$.

Now, we discuss *why* item 1 holds. In the cases $s \in \{2, 3, 4\}$, we can find a *most-violated* length-s vertex cut inequality (2.12) by computing, for each $(a, b) \in (V \times V) \setminus E$, a minimum-weight length-s a, b-vertex cut and comparing its weight to r. The cases $s \in \{2, 3\}$ are fairly straightforward, e.g., for $s = 2$ the solution is $N^+(a) \cap N^-(b)$. The case $s = 4$ was (essentially) shown by Lovász et al. (1978) to be polynomial-time solvable by reducing it to a particular instance of the min-cut problem, cf. Theorem 4.3.1 of Xu (2001). Since this min-cut instance can be constructed in linear time (and it is actually a subgraph of the input graph), this minimum-weight length-s a, b-vertex cut subproblem can be solved in time $O(mn)$ by Orlin (2013). Solving these subproblems for every missing edge (a, b) gives a total time of $O(mn^3)$, which is polynomial. □

By standard arguments, the flow-based separation routines referenced in the proof of Theorem 5 imply polynomial-size extended formulations for the LP relaxation of CUT when $s \in \{3, 4\}$, see Martin (1991). However, these formulations would have roughly mn^2 variables, making them too large to be practical. Hence, we do not discuss them further.

2.3.3 Verification and Integer Separation for the Fault-Tolerant Variant

The problem of verifying whether a given subset $D \subseteq V$ of vertices is an r-robust latency-s CDS is nontrivial. In a brute force approach, enumerate all subsets $F \subseteq D$ of $r-1$ vertices and verify that $D \setminus F$ is indeed a latency-s CDS. By algorithm IsLatencyConstrainedCDS, this takes time $\binom{|D|}{r-1}O(n^3) = O(n^{r+2})$, which is polynomial for any constant r. A natural question is whether this test can be performed in polynomial time when r is part of the problem input. Unfortunately, the likely answer is "no," as this is coNP-complete.

Corollary 4. *When r is part of the input, the problem of verifying whether $D \subseteq V$ is an r-robust latency-s CDS is coNP-complete for each fixed $s \geq 5$. This holds even for bidirected edges and hop-based distances.*

Proof. Membership in coNP follows because a length-s vertex cut $C \subseteq V$ with $|C| < r$ is a suitable witness when it is a "no" instance. For the reduction, consider an instance of Diameter Interdiction by Node Deletion defined by a simple graph $G = (V, E)$ and integers q and L. Bidirect its edges and let $s = L$, $r = q + 1$, and $D = V$. It can be observed that the instance of Diameter Interdiction by Node Deletion is a "yes" instance if and only if D is *not* an r-robust latency-s CDS of this bidirected graph. □

Remark 1. *As a consequence of Corollary 4, the separation problem for the constraints* (2.12)*, with r being part of the input, is hard* even when x^* is integer.

2.4 Computational Experiments

In this section, we provide results from our computational experiments. First, we demonstrate the importance of (quickly) strengthening the length-s vertex cut inequalities. Second, we provide computational results demonstrating the importance of providing an initial heuristic solution to the MIP solver. Third, we compare our full implementation of CUT with the polynomial-size formulation POLY. Our tests demonstrate the superiority of CUT over POLY. Finally, we experiment with formulation CUT for:

1. $s \in \{\text{diam}(G), \text{diam}(G)+1, \text{diam}(G)+2, n-1\}$;
2. the fault-tolerant case with $r = 2$;
3. a class of instances representing node-weighted, transmitter-based delays.

All of our experiments are conducted on a Dell Precision Tower 7000 Series (7810) machine running Windows 10 enterprise, x64, with Intel® Xeon® Processor E52630 v4 (10 cores, 2.2GHz, 3.1GHz Turbo, 2133MHz, 25MB, 85W) – that is 20 logical processors – and 32 GB memory. The IP formulations were implemented in Microsoft Visual Studio 2015 in C++ for Gurobi version 7.0.2. We use default settings with the exception that we force Gurobi to use the concurrent method (which uses primal simplex, dual simplex, and barrier on different threads) for solving the root LP relaxation for POLY, as this formulation is highly degenerate and typically barrier is fastest. We impose a time limit of 3600 seconds on each instance and use the same test instances that have been used in the previous literature on the minimum CDS problem by Lucena et al. (2010); Simonetti et al. (2011); Fan and Watson (2012); Gendron et al. (2014); Buchanan et al. (2015); Li and Aneja (2017). This testbed includes both real-life and synthetically generated instances, all of which are undirected. In our experiments, we bidirect their edges.

2.4.1 The Importance of Strengthening the Inequalities

Since formulation CUT can have exponentially many constraints when $s \geq 3$, we initialize it with only some of the constraints. Others are added as needed via Gurobi's lazy constraint callback features. Specifically, we start with the vertex cuts given by $N^+(i),\ i \in V$ (or inclusion-minimal subsets thereof that are also length-s vertex cuts). Then, within the branch-and-bound tree, violated length-s vertex cut inequalities are added on-the-fly.

Since the separation problem for the length-s vertex cut inequalities is NP-hard, we only separate *integer* points that the solver encounters.

Each of these possible solutions $D \subseteq V$ will satisfy the initial constraints given to the solver, but D may not actually be feasible for the latency-s CDS problem. In this case, $C := V \setminus D$ is a length-s vertex cut for the graph, and the inequality $\sum_{i \in C} x_i \geq 1$ would be valid for our problem and would cut off the binary point representing D. However, this inequality is likely very weak, so we strengthen the inequality, i.e., find a minimal subset of C that is also a length-s vertex cut. This is done when initializing the formulation with the vertex cuts $N^+(i),\ i \in V$ and also when adding inequalities on-the-fly.

The following algorithm can be used to strengthen length-s vertex cut inequalities, i.e., to find *inclusion-minimal* length-s vertex cuts. For our purposes, it is best described in terms of a "bad" vertex subset $B \subseteq V$, i.e., one that is not a latency-s CDS for G. (If given a length-s vertex cut, initialize the algorithm using its complement.) The algorithm takes B as input and returns an inclusion-minimal length-s vertex cut $C \subseteq V \setminus B$ for G.

```
MinimalizeBasic($G$, $B$, $s$):
  1. $Q \leftarrow V \setminus B$;
  2. for $q \in Q$ do
       • if $B \cup \{q\}$ is not a latency-$s$ CDS for $G$ then update $B \leftarrow B \cup \{q\}$;
  3. return $C := V \setminus B$.
```

Proposition 6. *Algorithm* MinimalizeBasic *finds a minimal length-s vertex cut in time* $O(n^4)$.

Proof. The runtime is dominated by the $|Q|$ different calls to IsLatencyConstrainedCDS, which take time $O(|Q|n^3) = O(n^4)$. Since latency-s CDS's are closed under taking supersets (as a consequence of Proposition 3 of Chapter II), it can be argued that, in step 3, $B \cup \{v\}$ is a latency-s CDS for every $v \in V \setminus B$. Further, B is not. Thus, $|D \cap C| \geq 1$ for every latency-s CDS $D \subseteq V$, and no proper subset of C satisfies this property. So, by Lemma 1 of Chapter II, C is a minimal length-s vertex cut. □

The algorithm MinimalizeBasic for finding a minimal length-s vertex cut runs in time $O(n^4)$. Here, we provide an improved $O(n^3)$ implementation. It is based on maintaining the collection F of "far" pairs of vertices, i.e., pairs of vertices that cannot communicate quickly enough through $B \subseteq V$.

First we provide pseudocode for the subroutine EnumerateFarPairs, which finds all far pairs. It is similar to IsLatencyConstrainedCDS described earlier but requires more space, since it requests the set F (and not just a query as to whether F is empty). Recall the subgraph of G denoted $\overrightarrow{G}_v^B$ whose edge set $\overrightarrow{E}_v^B$ was defined in equation 7 of Chapter II. This is used to compute shortest paths from v to all other nodes, using nodes from B as intermediaries.

```
EnumerateFarPairs($G$, $B$, $s$):
  1. $F \leftarrow \emptyset$;
  2. for each $v \in V$ do
       • compute shortest paths from $v$ in digraph $\overrightarrow{G}_v^B$;
       • for $t \in V \setminus \{v\}$ do
           – if $\text{dist}_{\overrightarrow{G}_v^B}(v,t) > s$ then $F \leftarrow F \cup \{(v,t)\}$;
```

3. return F.

Now we give an improved implementation of MINIMALIZEBASIC. In its pseudocode, we need notation for another subgraph of G, denoted $\overleftarrow{G}_v^B = (V, \overleftarrow{E}_v^B)$, that is used when computing the shortest paths from all other nodes to v, using only nodes from B as intermediaries. Its edge set $\overleftarrow{E}_v^B$ contains all edges that might be used in one of these paths.

$$\overleftarrow{E}_v^B := E(B) \cup \delta^-(B) \cup \left(\delta^+(B) \cap \delta^-(v)\right). \tag{2.26}$$

Minimalize(G, B, s):

```
1. $Q \leftarrow V \setminus B$ and $F \leftarrow$ENUMERATEFARPAIRS$(G, B, s)$;
2. for $v \in Q$ do
     • compute shortest paths from $v$ in graph $\overrightarrow{G}_v^B$;
     • compute shortest paths to $v$ in graph $\overleftarrow{G}_v^B$;
     • $R \leftarrow \{(a,b) \in F \mid \mathrm{dist}_{\overleftarrow{G}_v^B}(a,v) + \mathrm{dist}_{\overrightarrow{G}_v^B}(v,b) \leq s\}$;
     • if $F \neq R$ then update $B \leftarrow B \cup \{v\}$ and $F \leftarrow F \setminus R$;
3. return $C := V \setminus B$.
```

Here, $R \subseteq F$ represents the subset of far pairs that would no longer be far if paths could cross node v. If the sets R and F are the same, then this means that $B \cup \{v\}$ a latency-s CDS (which we do not want), otherwise the algorithm adds v to B, thus moving to the larger infeasible set $B \cup \{v\}$—whose complement is a smaller length-s vertex cut.

Theorem 6. *Algorithm* MINIMALIZE *finds a minimal length-s vertex cut in time $O(n^3)$ and space $O(n^2)$, or, more precisely, in:*

- *$O(nm + n^2 \log\log n + |Q||F_0|)$ time and $O(m + |F_0|)$ space under nonnegative weights;*

- *$O(nm + |Q||F_0|)$ time and $O(m + |F_0|)$ space in the hop-based case.*

Here, F_0 denotes the original set of far pairs, i.e., the set F from step 1.

In Table 2.1, we compare the performance of formulation CUT with three different settings: (1) without minimalizing the length-s vertex cuts, (2) applying algorithm MinimalizeBasic, and (3) applying algorithm Minimalize. In these tests, we set $s = \text{diam}(G)$ since this is the smallest feasible value of s. Moreover, we exclude the cases where $s = \text{diam}(G) = 2$, as there is nothing to minimalize. Note that these tests use the BestInHeuristic that is described in Section 5. CUT only solves 4 instances if we do not minimalize the length-s vertex cuts, in which case the formulation is practically useless. Using MinimalizeBasic, we can solve 20 of the 30 instances, but the callback time is large in many cases. For example, see the instance v200_d20 in which 37 minutes is spent in the callback. If Minimalize is used instead, the callback time reduces to 1 minute. Moreover, the time saved by Minimalize allows the solver to improve the lower bound in 5 of the 10 unsolved cases.

We also experimented with separating fractional points, particularly when $s = 3$ as this case of the separation problem is polynomial-time solvable. However, the fastest separation procedure that we are aware of takes time $O(mn^3)$ and was ultimately unhelpful—in all nine of the different implementations that we tried.

When experimenting with fractional separation, we felt it was important to try different implementations, as the number of cuts and the cut violation can greatly impact the performance. With this in mind, we tried nine different implementations.

The generic pseudocode that we use in the callback is as follows, where x^* is the branch-and-bound node's LP solution, $\epsilon \in \{0.01, 0.1, 0.5\}$ is the cut violation threshold, and $I \in \{1, 2, 3\}$ controls how many cuts are added per callback.

- for every vertex $a \in \{1, 2, \ldots, n\}$ do
    - for every vertex $b \in \{1, 2, \ldots, n\}$ do

Table 2.1: An evaluation of the usefulness of MINIMALIZE as compared to no minimalizing and to MINIMALIZEBASIC. For all graphs G, we set $s = \text{diam}(G)$ and exclude instances where $s = 2$. We report the optimal objective (or the best lower/upper bounds $[L, U]$ after one hour) under the columns labeled obj. We also give the time spent in the callback (call), and the total solve time (total), where a dash indicates > 3600 seconds.

		No minimalizing			MINIMALIZEBASIC			MINIMALIZE		
graph	s	obj	call	total	obj	call	total	obj	call	total
v30_d10	8	15	10.29	280.24	15	0.00	0.01	15	0.00	0.02
v30_d20	5	8	0.01	0.05	8	0.02	0.06	8	0.00	0.03
v30_d30	3	[7,8]	229.27	-	8	0.11	0.21	8	0.02	0.10
v50_d5	14	[26,32]	49.88	-	32	0.02	0.07	32	0.02	0.07
v50_d10	5	[13,20]	31.27	-	18	0.44	0.58	18	0.08	0.18
v50_d20	3	[8,14]	51.24	-	14	0.16	0.25	14	0.03	0.11
v50_d30	3	[6,8]	45.67	-	8	1.69	2.70	8	0.16	1.12
v70_d5	8	[24,36]	14.41	-	32	1.13	1.36	32	0.32	0.53
v70_d10	4	[14,31]	19.93	-	29	3.14	5.77	29	0.44	2.98
v70_d20	3	[8,18]	19.60	-	17	47.27	335.22	17	3.40	305.96
v70_d30	3	[6,8]	66.95	-	7	9.68	12.93	7	0.50	3.33
v100_d5	5	[24,57]	9.71	-	56	16.38	33.14	56	2.10	17.91
v100_d10	4	[14,31]	10.82	-	[22,26]	322.14	-	[22,26]	18.79	-
v100_d20	3	[8,20]	15.26	-	[14,20]	507.29	-	[14,20]	24.16	-
v120_d5	6	[25,40]	8.80	-	31	464.72	1511.87	31	25.86	1087.17
v120_d10	3	[13,68]	7.92	-	63	27.31	35.60	63	2.21	10.31
v120_d20	3	[8,21]	13.89	-	[10,21]	973.37	-	[10,21]	42.57	-
v120_d30	3	[6,12]	19.98	-	[7,12]	737.91	-	[7,12]	21.65	-
v150_d5	5	[25,54]	7.19	-	[29,54]	1088.47	-	[30,54]	121.73	-
v150_d10	3	[13,65]	7.35	-	[41,63]	717.64	-	[43,61]	39.95	-
v150_d20	3	[8,22]	12.70	-	[9,22]	1915.12	-	[9,22]	58.24	-
v200_d5	4	[25,92]	6.10	-	[48,92]	1851.87	-	[49,92]	169.57	-
v200_d10	3	[13,64]	6.26	-	[23,64]	1734.04	-	[26,64]	132.19	-
v200_d20	3	[7,22]	7.58	-	[7,22]	2226.01	-	[8,22]	58.71	-
IEEE-14	5	5	0.00	0.00	5	0.00	0.00	5	0.00	0.01
IEEE-30	6	14	0.33	3.98	14	0.00	0.01	14	0.00	0.01
IEEE-57	12	[24,35]	23.24	-	35	0.02	0.05	35	0.01	0.04
RTS-96	13	[27,40]	13.53	-	37	0.09	0.17	37	0.07	0.14
IEEE-118	14	[39,48]	15.22	-	48	0.01	0.17	48	0.00	0.15
IEEE-300	24	[104,139]	16.49	-	135	14.35	18.00	135	8.65	11.98

```
      * compute a minimum-weight length-3 $a$,$b$-separator $C \subseteq V \setminus \{a, b\}$ in graph
        $G = (V, E)$, where each vertex $i$ has weight $x^*_i$;
      * if the cut violation $1 - x^*(C)$ is more than $\epsilon$,
          · add the cut $x(C) \geq 1$;
          · if $I = 1$, then return;
          · if $I = 2$, then break the for-loop over $b$ (i.e., go to the next $a$);
```

As can be observed in the pseudocode, setting $I = 1$ adds ≤ 1 cut per callback, setting $I = 2$ adds $\leq n$ cuts per callback, and setting $I = 3$ adds $\leq n^2$ cuts per callback.

There are several ways to speed up separation in practice. In all of our implementations, we use the following speedups:

1. We can skip the cases $a = b$.
2. Our test instances are bidirected, so it suffices to consider $b \in \{a + 1, a + 2, \ldots, n\}$.
3. If $x^*(N^+(a) \cap N^-(b)) + \epsilon \geq 1$, then no sufficiently violated cut will exist, and one can skip to the next vertex b.

We tried each of the three settings $I \in \{1, 2, 3\}$ across three different values of $\epsilon \in \{0.01, 0.1, 0.5\}$, resulting in a total of nine implementations. The results are provided in Tables 2.2, 2.3, 2.4. In each table, we report the lower and upper bounds on the objective value after a one-hour time limit. For reference, we also report the bounds obtained using our integer separation routine that we have been using all along.

Among the nine fractional separation implementations, there is no clear winner. For the graph v150_d10, the best bounds are achieved with the following parameter settings.

$$(I, \epsilon) \in \{(1, 0.01), (3, 0.01), (3, 0.1)\}$$

While, for the graph v200_d20, the best bounds are achieved with a different parameter setting $(I, \epsilon) = (1, 0.1)$. However, if we include our original implementation into the mix, it is the clear winner. This is despite a good-faith effort towards using fractional separation, including a polynomial-time separation routine for $s = 3$ and helpful speedups. We imagine that the results would be even worse if we had to perform exact separation for $s \geq 5$, where separation is NP-hard. This confirms our suspicions that we had obtained when studying another semi-related problem (Salemi and Buchanan, 2020).

One explanation for the poor performance of $s = 3$ fractional separation is the time to solve the separation problem. It requires the solution of up to $\Theta(n^2)$ different minimum cut problems, making the runtime something like mn^3. In contrast, our integer separation routine requires $O(n^3)$ time. Note that for the $s = 3$ instance v200_d20, $n^3 = 8,000,000$, while $mn^3 = 31,840,000,000$. Any time spent on the time mn^3 procedure in the callback is time not spent on other things (like branching). Apparently, this tradeoff is not worth it.

As a final remark, see that several of our instances remain unsolved even when $s = 2$. Recall that we add all length-2 vertex cut inequalities upfront in the $s = 2$ case since there are only $O(n^2)$ of them, in which case there is no need to perform fractional separation. Thus, even when fractional separation takes "zero time", the instances cannot be solved. This lends credence to the idea that there is little room for fractional separation to help; these instances may just be challenging. We note that one of our $s = 2$ instances was submitted to MIPLIB 2017 and has been included in their Collection Set. At the time of writing, it is still considered an "open" instance, meaning that no MIP solver has been able to solve it (even with the latest software releases). The instance (and its "open" status) can be found on the MIPLIB website: `http://miplib.zib.de/instance_details_v150d30-2hopcds.html`

Table 2.2: Lower and upper bounds after one hour when cut violation threshold is $\epsilon = 0.01$

		LAZY	$I = 1$	$I = 2$	$I = 3$
graph	s		≤ 1	$\leq n$	All Pairs
v100_d20	3	[14,20]	[11,20]	[10,20]	[11,20]
v120_d20	3	[10,21]	[8,21]	[7,21]	[8,21]
v120_d30	3	[7,12]	[4,12]	[4,12]	[4,12]
v150_d10	3	[43,61]	[38,65]	[35,65]	[38,65]
v150_d20	3	[9,22]	[7,22]	[7,22]	[7,22]
v200_d10	3	[26,64]	[27,64]	[24,64]	[27,64]
v200_d20	3	[8,22]	[6,22]	[6,22]	[6,22]

Table 2.3: Lower and upper bounds after one hour when cut violation threshold is $\epsilon = 0.1$

		LAZY	$I = 1$	$I = 2$	$I = 3$
graph	s		≤ 1	$\leq n$	All Pairs
v100_d20	3	[14,20]	[10,20]	[10,20]	[11,20]
v120_d20	3	[10,21]	[9,21]	[7,21]	[8,21]
v120_d30	3	[7,12]	[6,12]	[4,12]	[4,12]
v150_d10	3	[43,61]	[28,65]	[37,65]	[38,65]
v150_d20	3	[9,22]	[8,22]	[7,22]	[7,22]
v200_d10	3	[26,64]	[19,64]	[24,64]	[26,64]
v200_d20	3	[8,22]	[7,22]	[6,22]	[6,22]

Table 2.4: Lower and upper bounds after one hour when cut violation threshold is $\epsilon = 0.5$

		LAZY	$I = 1$	$I = 2$	$I = 3$
graph	s		≤ 1	$\leq n$	All Pairs
v100_d20	3	[14,20]	[11,20]	[9,20]	[10,20]
v120_d20	3	[10,21]	[8,21]	[7,21]	[7,21]
v120_d30	3	[7,12]	[4,12]	[4,12]	[4,12]
v150_d10	3	[43,61]	[28,65]	[37,65]	[36,65]
v150_d20	3	[9,22]	[7,22]	[6,22]	[6,22]
v200_d10	3	[26,64]	[20,64]	[23,64]	[24,64]
v200_d20	3	[8,22]	[6,22]	[6,22]	[6,22]

2.4.2 The Importance of Providing a Heuristic Solution to the Solver

We provide a simple "best-in" heuristic for the minimum latency-s CDS problem. In the pseudocode, the vertex subset D represents a partial solution; the set F represents the set of "far pairs," i.e., those pairs of vertices that inhibit D from being a feasible solution; and the score of a vertex is how many far pairs it would "close" or eliminate. Note that the scores can increase during the heuristic, and they can all be zero while F is nonempty.

BestInHeuristic(G, s):

```
1. compute diam($G$);
2. if diam($G$) $> s$, return "infeasible";
3. initialize $D \leftarrow \emptyset$ and $F \leftarrow \{(i,j) \in V \times V \mid (i,j) \notin E,\ i \neq j\}$;
4. while $F \neq \emptyset$ do
    (a) for $v \in V \setminus D$ do
        • compute shortest paths from $v$ in graph $\overrightarrow{G}_v^D$;
        • compute shortest paths to $v$ in graph $\overleftarrow{G}_v^D$;
        • compute score($v$) := $\left|\left\{(a,b) \in F \;\middle|\; \text{dist}_{\overleftarrow{G}_v^D}(a,v) + \text{dist}_{\overrightarrow{G}_v^D}(v,b) \leq s\right\}\right|$;
    (b) let $v^* \in V \setminus D$ be a vertex of maximum score;
    (c) $R \leftarrow \left\{(a,b) \in F \;\middle|\; \text{dist}_{\overleftarrow{G}_{v^*}^D}(a,v^*) + \text{dist}_{\overrightarrow{G}_{v^*}^D}(v^*,b) \leq s\right\}$;
    (d) update $D \leftarrow D \cup \{v^*\}$ and $F \leftarrow F \setminus R$;
5. return $D$.
```

This heuristic returns a feasible solution (when the instance is feasible), and its runtime is $O(n^4)$, or, more precisely, $O(n^3 + |D|mn + |D||F_0|n)$, where D is the heuristic solution at the end, and F_0 is the initial set of far pairs from step 3. Given that it takes time $O(n^3)$ to verify feasibility, this heuristic is not too costly. The output of this heuristic is not necessarily inclusion-minimal, so we run a post-processing procedure to make it so.

We also experimented with a "worst-out" heuristic that starts with V as an initial solution and greedily deletes vertices v, in order of decreasing indegree $|N^-(v)|$ plus outdegree $|N^+(v)|$, as long as this is still feasible. This heuristic, when applied to the minimum CDS problem, is "provably best" in the sense of Kahruman-Anderoglu et al. (2016) meaning that, unless P=NP, no polynomial-time algorithm always finds a better solution than this heuristic (when a better solution exists). Similar heuristics work well in practice for the minimum CDS problem (Butenko et al., 2004). However, this particular worst-out heuristic performed poorly compared to the best-in heuristic in initial experiments, so it is excluded.

In Table 2.5, we evaluate the usefulness of the proposed best-in heuristic (while employing Minimalize to strengthen the inequalities). In most cases, the impact of the heuristic is marginal (positively or negatively). Of the instances that were solved to optimality, its effect is most pronounced on v70_d20 and v120_d5 where it slowed down the solver by 173 seconds and sped up the solver by 409 seconds, respectively. It may seem strange that providing an MIP start could worsen the performance, but this is likely a natural consequence of solver variability, and we did not attempt to tame or take advantage of this behavior. Note, however, that Gurobi was unable to find a feasible solution on four instances if left unaided. Further, on the instances v150_d20 and v200_d20 the best-in heuristic found solutions (in under a second) that were 12 vertices and 13 vertices better, respectively, than what Gurobi found in an hour. Frankly, Gurobi's performance on some of the $s = 2$ instances surprised us. For example, on the instance v120_d50, CUT has only 120 binary variables and 3570 covering constraints, and yet it did not solve within an hour.

2.4.3 Comparison with Formulation POLY

In Table 2.6, we compare the performance of CUT with that of POLY. In these tests, we set $s = \text{diam}(G)$, provide an MIP start using BestInHeuristic, and exclude instances with $s = 2$. The reason for excluding the $s = 2$ comparisons is that CUT and POLY are equally

Table 2.5: An evaluation of the usefulness of BestInHeuristic as compared to running Gurobi without an MIP start. We report the objective of the heuristic solution (hobj) and the time spent in the heuristic (htime). For the meanings of other reported quantities, refer to Table 2.1.

		No heuristic		with BestInHeuristic			
graph	s	obj	total	hobj	obj	htime	total
v30_d10	8	15	0.01	15	15	0.00	0.02
v30_d20	5	8	0.09	8	8	0.00	0.03
v30_d30	3	8	0.16	8	8	0.00	0.10
v30_d50	2	7	0.02	7	7	0.00	0.01
v30_d70	2	3	0.02	4	3	0.00	0.04
v50_d5	14	32	0.05	32	32	0.01	0.07
v50_d10	5	18	0.18	20	18	0.01	0.18
v50_d20	3	14	0.14	14	14	0.00	0.11
v50_d30	3	8	7.23	8	8	0.00	1.12
v50_d50	2	9	0.18	11	9	0.00	0.17
v50_d70	2	4	0.89	4	4	0.00	0.79
v70_d5	8	32	0.44	36	32	0.01	0.53
v70_d10	4	29	1.34	31	29	0.01	2.98
v70_d20	3	17	132.69	18	17	0.01	305.96
v70_d30	3	7	1.95	8	7	0.00	3.33
v70_d50	2	10	0.76	10	10	0.00	0.56
v70_d70	2	5	2.04	5	5	0.00	1.57
v100_d5	5	56	17.02	57	56	0.04	17.91
v100_d10	4	[22,26]	-	31	[22,26]	0.02	-
v100_d20	3	[14,18]	-	20	[14,20]	0.01	-
v100_d30	2	39	5.35	42	39	0.03	5.00
v100_d50	2	12	59.15	13	12	0.01	61.27
v100_d70	2	5	8.91	5	5	0.00	8.17
v120_d5	6	31	1496.49	40	31	0.05	1087.17
v120_d10	3	63	10.33	68	63	0.06	10.31
v120_d20	3	[10,26]	-	21	[10,21]	0.02	-
v120_d30	3	[7,12]	-	12	[7,12]	0.01	-
v120_d50	2	[11,12]	-	13	[11,12]	0.01	-
v120_d70	2	5	32.86	6	5	0.00	29.67
v150_d5	5	[30,∞]	-	54	[30,54]	0.10	-
v150_d10	3	[39,∞]	-	65	[43,61]	0.10	-
v150_d20	3	[9,34]	-	22	[9,22]	0.04	-
v150_d30	2	[35,42]	-	45	[35,41]	0.06	-
v150_d50	2	[9,13]	-	13	[9,13]	0.02	-
v150_d70	2	6	560.68	7	6	0.01	569.10
v200_d5	4	[46,∞]	-	92	[49,92]	0.34	-
v200_d10	3	[23,∞]	-	64	[26,64]	0.20	-
v200_d20	3	[7,35]	-	22	[8,22]	0.07	-
v200_d30	2	[27,43]	-	45	[27,44]	0.12	-
v200_d50	2	[8,15]	-	16	[8,15]	0.04	-
v200_d70	2	[4,6]	-	7	[4,7]	0.01	-
IEEE-14	5	5	0.00	5	5	0.00	0.01
IEEE-30	6	14	0.01	14	14	0.00	0.01
IEEE-57	12	35	0.02	35	35	0.01	0.04
RTS-96	13	37	0.15	40	37	0.02	0.14
IEEE-118	14	48	0.07	48	48	0.06	0.15
IEEE-300	24	135	8.91	139	135	2.31	11.98

strong when $s = 2$, and so CUT will obviously perform better due to its smaller size. Since our instances are bidirected, we fix $y^t_{ij} = y^t_{ji}$ as discussed in Section 2.2.3.

The results demonstrate the superiority of CUT. It solves the 11 instances solved by POLY—and 9 others. Ten instances are left unsolved by both approaches; CUT provides better bounds on all of them. The formulation CUT also *quickly* solves some instances that POLY left unsolved after an hour. For example, CUT solved v50_d5, v70_d5, v70_d10, and v70_d30 each in under five seconds, while POLY solved none of them in the time limit.

2.4.4 The Cost of Low Latency

Imposing that a dominating set be *connected* is not too costly. Indeed, the domination number $\gamma(G)$ and the connected domination number $\gamma_c(G)$ are a constant factor apart. Specifically, they satisfy $\gamma(G) \leq \gamma_c(G) \leq 3\gamma(G) - 2$, see Haynes et al. (1998). In contrast, we show that the cost of low latency can be very large—even when decreasing the latency parameter s by one. We denote by $\gamma^{lat}_s(G)$ the size of a minimum latency-s CDS in G.

Proposition 7 (Potentially large cost of low latency)**.** *For every latency parameter $s \geq 2$, there is an infinite class of graphs G for which $\gamma^{lat}_{s+1}(G) \leq s$, but $\gamma^{lat}_s(G) \geq \Omega(n)$. This holds even when edges are bidirected and distances are hop-based.*

Proof. One such class of graphs are obtained by taking the Cartesian products $K_q \Box P_s$ of a complete graph K_q and a path graph P_s and then bidirecting the edges. These graphs have sq vertices and diameter s when $q \geq 2$. These graphs $K_q \Box P_s$ can be defined as having vertex set $V^1 \cup \cdots \cup V^s$, where each $V^i = \{v^i_1, \ldots, v^i_q\}$. For the edges, let each V^i be a clique, and connect each vertex v^i_j to its counterpart v^{i+1}_j from the next V^{i+1}. Bidirect all edges.

See that $\gamma^{lat}_{s+1}(K_q \Box P_s) \leq s$, since the s vertices v^i_1 form a feasible solution. Now we show that $\gamma^{lat}_s(K_q \Box P_s) \geq \Omega(n)$ in two cases. When $s = 2$, the q vertex subsets $\{v^1_i, v^2_{i+1}\}$ for $i = 1, \ldots, q-1$ and $\{v^1_q, v^2_1\}$ form length-2 cuts and are disjoint. Thus, $\gamma^{lat}_2(K_q \Box P_2) \geq q = n/2$.

Table 2.6: A comparison of the performance of formulation CUT with that of POLY. For all graphs G, we set $s = \text{diam}(G)$ and exclude instances where $s = 2$. We report the optimal objective (or the best lower/upper bounds $[L, U]$ after one hour) under the columns labeled obj. We also give the total solve time (total), where a dash indicates > 3600 seconds.

			POLY		CUT	
graph	s	hobj	obj	total	obj	total
v30_d10	8	15	15	2.71	15	0.02
v30_d20	5	8	8	111.14	8	0.03
v30_d30	3	8	8	12.59	8	0.10
v50_d5	14	32	[31,32]	-	32	0.07
v50_d10	5	20	18	2711.41	18	0.18
v50_d20	3	14	14	6.31	14	0.11
v50_d30	3	8	8	398.41	8	1.12
v70_d5	8	36	[26,36]	-	32	0.53
v70_d10	4	31	[28,29]	-	29	2.98
v70_d20	3	18	[11,18]	-	17	305.96
v70_d30	3	8	[6,7]	-	7	3.33
v100_d5	5	57	[42,57]	-	56	17.91
v100_d10	4	31	[13,31]	-	[22,26]	-
v100_d20	3	20	[9,20]	-	[14,20]	-
v120_d5	6	40	[16,40]	-	31	1087.17
v120_d10	3	68	63	1522.53	63	10.31
v120_d20	3	21	[7,21]	-	[10,21]	-
v120_d30	3	12	[4,12]	-	[7,12]	-
v150_d5	5	54	[19,54]	-	[30,54]	-
v150_d10	3	65	[35,65]	-	[43,61]	-
v150_d20	3	22	[6,22]	-	[9,22]	-
v200_d5	4	92	[43,92]	-	[49,92]	-
v200_d10	3	64	[24,64]	-	[26,64]	-
v200_d20	3	22	[6,22]	-	[8,22]	-
IEEE-14	5	5	5	0.08	5	0.01
IEEE-30	6	14	14	0.29	14	0.01
IEEE-57	12	35	35	57.65	35	0.04
RTS-96	13	40	[35,39]	-	37	0.14
IEEE-118	14	48	48	130.70	48	0.15
IEEE-300	24	139	[6,139]	-	135	11.98

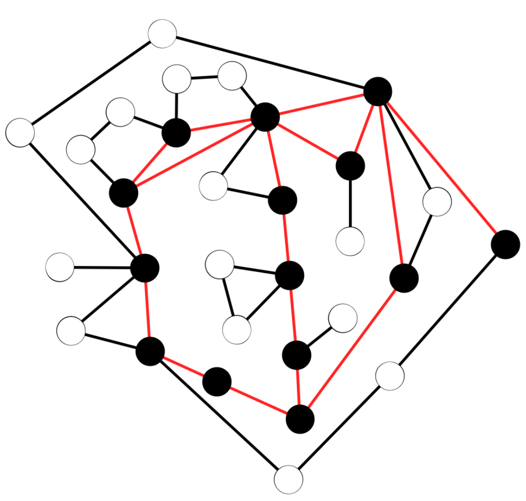
When $s \geq 3$, each vertex v_j^i with $2 \leq i \leq s-1$ is a length-s vertex cut on its own (by the resulting distance between nodes v_j^1 and v_j^s), so $\gamma_s^{lat}(K_q \square P_s) \geq (s-2)q \geq n/3$. $\square$

We observe the cost of low latency "in practice" through the computational results given in Table 2.7. We report the solution sizes and runtimes for different values of the latency parameter $s \in \{\text{diam}, \text{diam}+1, \text{diam}+2, n-1\}$ under hop-based distances. Thus, we have the strictest case of $s = \text{diam}$ and the most relaxed value of $s = n-1$, which corresponds to the minimum CDS problem when edges are bidirected. The solve times tend to improve as s increases, and we are able to solve all instances when $s = n-1$. However, this is not universally the case, e.g., for graph v100_d5. Some of the lower bounds can immediately be improved based on the table. For example, we can claim a lower bound of 10 for the instance v150_d20 when $s = \text{diam}$, since 10 is optimal for the less restrictive case $s = \text{diam}+1$.

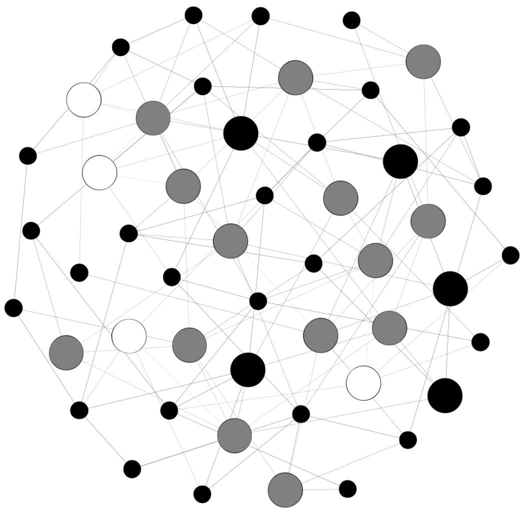
The runtimes for the case $s = n-1$ closely resemble those given by Buchanan et al. (2015) for the minimum CDS problem. This is unsurprising given that the approach taken here is very similar. However, the instance IEEE-300 takes longer here (514.34 vs. 52.88 seconds). This can be attributed to the 492.86 seconds spent in our slower callback routines.

In some applications, achieving low latency is desirable but should not be viewed as a "hard" constraint. In this case, the tradeoff between CDS size and the latency guarantee should be considered. For example, the results for graphs v30_d10 and IEEE-14 show that low latency comes for free; there is a minimum CDS that also satisfies the most restrictive (but feasible) latency value $s = \text{diam}$. On the other hand, for the graphs v100_d30 and v150_d30, the optimal objective triples when tightening the latency parameter from $s = 3$ to $s = 2$ and may not be justified.

We also give the optimal objectives for the dominating $(s-2)$-club problem. As observed in the introduction, the formalization based on dominating $(s-2)$-clubs does not quite capture the intent of the latency constraints, and here we see that it usually gives the impression that no suitable low-latency CDS exists, and yet there exists a latency-s CDS. This occurs for 37 of the 47 graphs when $s = \text{diam}(G)$. An example is given in Figure 2.7(a). Also, Figure 2.7(b) shows an instance where both problems are feasible but have different optimal solutions.



(a) IEEE-30



(b) v50_d10

Figure 2.7: Side (a) shows a minimum latency-6 CDS for IEEE-30; there is no dominating 4-club. Side (b) shows a minimum latency-5 CDS D and a minimum dominating 3-club D' for the graph v50_d10. Nodes in $D \cup D'$ are larger; nodes in $D \setminus D'$ are white; nodes in $D' \setminus D$ are black; nodes in $D \cap D'$ are gray.

2.4.5 Results for the Fault-Tolerant Variant

In Table 2.8, we provide results for the fault-tolerant variant, where $r = 2$, and s is set to the smallest feasible value[4] $\max\{\text{diam}(G - v) \mid v \in V\}$. The reason for including results for $r = 2$ (and not for larger values) is that network topologies that continue to function after the failure of a single entity are often considered sufficient for practical purposes (Monma and Shallcross, 1989; Grötschel et al., 1992a).

The implementation needed to be modified to accommodate $r = 2$. For one, we needed to alter the heuristic. Our approach is to first find a feasible solution $D \subseteq V$ for $r = 1$ using the BestInHeuristic from before. Then we consider some $v \in D$ and check if $D \setminus \{v\}$ is a

[4]If the graph has a cut vertex, we exclude it from consideration since there is no feasible (finite) value for s.

Table 2.7: Results for different values of the latency parameter s. The case where $s = n - 1$ is identical to the minimum strongly connected dominating set problem. We also report the objective of the dominating $(s - 2)$-club problem (club). Here, ∞ denotes infeasibility, and blank cells indicate that $s - 2 \leq 0$.

		$s =$ diam			$s =$ diam $+1$			$s =$ diam $+2$			$s = n - 1$	
graph	diam	club	obj	total	club	obj	total	club	obj	total	obj	total
v30_d10	8	∞	15	0.02	15	15	0.02	15	15	0.02	15	0.04
v30_d20	5	8	8	0.03	7	7	0.01	7	7	0.02	7	0.01
v30_d30	3	∞	8	0.10	5	5	0.05	4	4	0.01	4	0.01
v30_d50	2		7	0.01	3	3	0.01	3	3	0.01	3	0.01
v30_d70	2		3	0.04	2	2	0.01	2	2	0.01	2	0.01
v50_d5	14	32	32	0.07	32	32	0.13	31	31	0.19	31	0.36
v50_d10	5	19	18	0.18	14	14	0.18	13	13	0.11	12	0.23
v50_d20	3	∞	14	0.11	7	7	0.27	7	7	0.17	7	0.17
v50_d30	3	∞	8	1.12	5	5	0.10	5	5	0.12	5	0.12
v50_d50	2		9	0.17	3	3	0.10	3	3	0.02	3	0.02
v50_d70	2		4	0.79	2	2	0.03	2	2	0.02	2	0.02
v70_d5	8	32	32	0.53	29	29	0.74	28	28	1.38	27	0.76
v70_d10	4	∞	29	2.98	17	16	8.87	13	13	0.47	13	0.10
v70_d20	3	∞	17	305.96	8	8	0.21	7	7	0.19	7	0.14
v70_d30	3	∞	7	3.33	5	5	0.17	5	5	0.16	5	0.16
v70_d50	2		10	0.56	3	3	0.05	3	3	0.05	3	0.05
v70_d70	2		5	1.57	2	2	0.10	2	2	0.06	2	0.06
v100_d5	5	∞	56	17.91	40	[34,36]	-	29	29	2018.43	24	0.56
v100_d10	4	∞	[22,26]	-	15	15	4.24	14	14	0.41	13	0.25
v100_d20	3	∞	[14,20]	-	9	9	2.35	8	8	0.53	8	0.51
v100_d30	2		39	5.00	∞	[7,12]	-	6	6	0.94	6	0.98
v100_d50	2		12	61.27	4	4	0.87	4	4	0.89	4	0.93
v100_d70	2		5	8.17	3	3	1.20	3	3	1.18	3	1.20
v120_d5	6	31	31	1087.17	28	28	97.26	26	26	4.36	25	0.62
v120_d10	3	∞	63	10.31	∞	[17,31]	-	15	15	202.51	13	0.69
v120_d20	3	∞	[10,21]	-	9	9	5.67	8	8	3.46	8	1.54
v120_d30	3	∞	[7,12]	-	6	6	1.10	6	6	1.20	6	1.10
v120_d50	2		[11,12]	-	4	4	4.78	4	4	3.71	4	3.63
v120_d70	2		5	29.67	3	3	2.06	3	3	2.16	3	2.08
v150_d5	5	∞	[30,54]	-	[28,33]	[26,40]	-	[26,28]	[26,33]	-	26	2.10
v150_d10	3	∞	[43,61]	-	∞	[14,28]	-	16	[15,18]	-	14	4.94
v150_d20	3	∞	[9,22]	-	10	10	379.34	9	9	7.13	9	6.88
v150_d30	2		[35,41]	-	∞	[6,11]	-	6	6	7.65	6	3.96
v150_d50	2		[9,13]	-	4	4	2.38	4	4	2.49	4	2.77
v150_d70	2		6	569.09	3	3	3.29	3	3	3.35	3	3.41
v200_d5	4	∞	[49,92]	-	[31,52]	[26,50]	-	[25,46]	[26,35]	-	27	10.00
v200_d10	3	∞	[26,64]	-	∞	[14,29]	-	[14,19]	[14,21]	-	16	301.83
v200_d20	3	∞	[8,22]	-	10	[9,11]	-	9	9	183.40	9	184.27
v200_d30	2		[27,44]	-	∞	[6,12]	-	7	7	223.04	7	205.41
v200_d50	2		[8,15]	-	4	4	145.90	4	4	7.44	4	7.84
v200_d70	2		[4,7]	-	3	3	6.43	3	3	6.29	3	6.59
IEEE-14	5	5	5	0.01	5	5	0.01	5	5	0.01	5	0.01
IEEE-30	6	∞	14	0.01	13	13	0.01	11	11	0.01	11	0.01
IEEE-57	12	35	35	0.04	31	31	0.08	31	31	0.19	31	1.88
RTS-96	13	37	37	0.14	35	35	0.46	34	34	0.38	32	1.90
IEEE-118	14	48	48	0.15	46	46	0.25	45	45	0.25	43	1.18
IEEE-300	24	135	135	11.98	131	131	35.08	130	130	66.30	129	514.34

latency-s CDS. If not, then we augment it by adding high *score* vertices from $V \setminus D$ in the same way as in BESTINHEURISTIC until the failure of $v \in D$ will not cause problems. We do this for each vertex $v \in D$ from the initial solution (for $r = 1$) until our heuristic solution is feasible for $r = 2$.

We also had to modify the callback routines. When the solver encounters a vertex subset D that satisfies the initial constraints, we check if it is a latency-s CDS (for $r = 1$). If not then $V \setminus D$ is a length-s vertex cut, and we strengthen it using MINIMALIZE as before. Supposing D is a latency-s CDS, then we check if it is a solution for $r = 2$, i.e., if $D \setminus \{v\}$ is a latency-s CDS for each $v \in D$. If not, then $(V \setminus D) \cup \{v\}$ is a length-s vertex cut, and we use MINIMALIZE on it.

2.4.6 Results when Delays are Node-Weighted and Transmitter-Based

In this section, we provide computational results for instances in which delays are node-weighted and transmitter-based. The intent is to model wireless sensor networks in which delays *depend on the transmitting node.* In our experiments, we make the simplifying assumption that the delay at node i is a given constant w_i. This is the time for node i to pass a message to any neighboring node. Following the transformation given in the introduction, this means that the edges pointing away from node i should have weight w_i. However, if one were to look at our implementation, they would see that our weights are stored by node. We prefer this representation since it is more space efficient.

To ensure that the node-based delays w_i used in our experiments are somewhat reasonable and reproducible, we define them as follows, where $\text{dist}(i, j)$ is hop-based.

$$w_i := \left\lfloor \frac{1000(n-1)}{\sum_{j \in V} \text{dist}(i, j)} \right\rfloor .$$

The reasoning for defining w_i in this way is as follows. So-called *central* nodes in the network

Table 2.8: Results for the fault-tolerant variant, where $r = 2$ and $s = \max\{\text{diam}(G - v) \mid v \in V\}$. Only biconnected graphs are considered. The number of branch-and-bound nodes is "BB node."

graph	diam	s	BB node	htime	hobj	obj	total
v30_d30	3	3	462	0.00	12	12	0.09
v30_d50	2	2	0	0.01	15	14	0.02
v30_d70	2	2	0	0.00	6	5	0.02
v50_d10	5	5	239	0.12	34	32	0.33
v50_d20	3	4	432	0.03	15	13	0.25
v50_d30	3	3	10625	0.01	13	12	2.28
v50_d50	2	2	9	0.02	18	15	0.19
v50_d70	2	2	0	0.00	7	6	0.33
v70_d5	8	10	201	0.27	53	51	0.61
v70_d10	4	5	7519	0.13	33	27	4.44
v70_d20	3	3	384357	0.12	31	26	106.28
v70_d30	3	3	29771	0.02	14	12	7.30
v70_d50	2	2	174	0.03	18	16	0.56
v70_d70	2	2	163	0.01	9	7	1.77
v100_d5	5	6	134933	0.71	63	56	102.40
v100_d10	4	4	943671	0.29	48	[38,41]	-
v100_d20	3	3	1260796	0.15	32	[24,27]	-
v100_d30	2	3	366610	0.07	18	[11,16]	-
v100_d50	2	2	21597	0.07	19	18	36.17
v100_d70	2	2	4146	0.02	10	8	14.25
v120_d5	6	7	134501	1.22	59	50	164.92
v120_d10	3	4	570958	0.56	42	[28,39]	-
v120_d20	3	3	357951	0.28	32	[18,30]	-
v120_d30	3	3	265467	0.08	19	[10,17]	-
v120_d50	2	2	1109463	0.13	22	[16,18]	-
v120_d70	2	2	6075	0.02	10	8	37.63
v150_d5	5	5	388468	2.44	80	[53,80]	-
v150_d10	3	4	283107	0.49	45	[25,45]	-
v150_d20	3	3	273843	0.51	36	[16,34]	-
v150_d30	2	3	244025	0.20	19	[9,19]	-
v150_d50	2	2	222265	0.18	22	[15,18]	-
v150_d70	2	2	628745	0.04	11	[8,9]	-
v200_d5	4	4	644105	12.81	128	[104,118]	-
v200_d10	3	3	426782	5.56	95	[55,95]	-
v200_d20	3	3	224602	0.79	32	[13,32]	-
v200_d30	2	2	77784	6.57	72	[49,63]	-
v200_d50	2	2	33061	0.64	25	[13,22]	-
v200_d70	2	2	119071	0.07	12	[7,9]	-

will be used more frequently to transmit information, resulting in longer queueing delays. The quantity $(n-1)/\sum_{j \in V} \operatorname{dist}(i,j)$ is the definition of (normalized) closeness centrality and the first definition for w_i that we tried. However, it is fractional, and the inexactness of later floating point calculations caused problems. To avoid exact rational arithmetic, we wanted to round closeness centrality to an integer, but this would give either a zero or a one. Multiplying by a large integer (1000 in our case) allowed for more diverse delays.

Table 2.9 provides our results with these delays where $s = \operatorname{diam}(G)$ is set to be as restrictive as possible while maintaining feasibility. They indicate a strength of our approach—that using weighted distances has little impact on the performance.

2.5 Conclusion

In this chapter, we introduce a latency-constrained variant of the minimum CDS problem motivated by applications in wireless sensor networks in which one seeks a virtual backbone that provides for small end-to-end delays. We propose integer programming formulations based on length-bounded vertex cuts. These formulations generalize the best-performing existing formulations for the minimum CDS problem and also generalize the best-performing formulations for the fault-tolerant variant—the minimum k-connected k-dominating set problem. A branch-and-cut implementation of formulation CUT makes easy work of synthetic instances having fewer than 100 nodes and real-life instances with up to 300 nodes, significantly outperforming formulation POLY. Our proposed formulations are in the same vein as the recent "thin" approaches for other optimization problems (Fischetti et al., 2017a, 2016). In ongoing and future research, we study the potential of using similar thin formulations based on length-bounded vertex cuts for other distance-constrained problems in networks, e.g., Salemi and Buchanan (2020). We also focused on exact approaches. Only because our MIP solver Gurobi had problems finding feasible solutions in an hour did we employ a simple construction heuristic. There is certainly room for improvement on this front, although the

Table 2.9: Results for the weighted-distance variant. See Section 2.4.6 for weighting information.

graph	s	BB node	hobj	obj	total
v30_d10	2281	0	21	21	0.02
v30_d20	2317	31	12	11	0.04
v30_d30	1751	117	19	18	0.07
v30_d50	1422	48	10	9	0.06
v30_d70	1585	0	8	7	0.04
v50_d5	2549	21	37	37	0.11
v50_d10	1961	377	29	29	0.29
v50_d20	1614	158	33	32	0.21
v50_d30	1743	6317	25	22	1.17
v50_d50	1400	523	14	14	0.27
v50_d70	1606	285	8	6	0.21
v70_d5	2055	919	50	48	0.71
v70_d10	1701	42	52	50	0.51
v70_d20	1624	10527	43	40	2.37
v70_d30	1716	3556	35	30	1.47
v70_d50	1404	5631	17	15	3.04
v70_d70	1581	328	8	8	0.47
v100_d5	1701	10376	79	76	3.37
v100_d10	1785	1820846	55	[47,51]	-
v100_d20	1678	290399	34	[21,33]	-
v100_d30	1211	13773	51	47	6.46
v100_d50	1378	244673	19	17	61.96
v100_d70	1571	1250	9	8	1.82
v120_d5	1948	542530	59	53	774.43
v120_d10	1471	97436	76	70	31.07
v120_d20	1659	2327430	47	[37,43]	-
v120_d30	1695	4844453	47	40	1101.31
v120_d50	1389	6911583	18	[14,16]	-
v120_d70	1559	475	10	9	3.21
v150_d5	1757	4407299	89	87	2378.46
v150_d10	1469	1686790	102	[81,92]	-
v150_d20	1663	337800	50	[30,47]	-
v150_d30	1209	4540949	58	[43,50]	-
v150_d50	1371	4427905	24	[18,21]	-
v150_d70	1559	317	12	10	4.38
v200_d5	1594	374407	137	[80,135]	-
v200_d10	1503	419384	122	[83,109]	-
v200_d20	1640	2184837	106	[84,96]	-
v200_d30	1201	1162600	67	[45,63]	-
v200_d50	1361	4196920	26	[20,23]	-
v200_d70	1557	8868	12	11	46.41
IEEE-14	2154	0	8	8	0.01
IEEE-30	2121	0	16	16	0.02
IEEE-57	2306	0	41	41	0.11
RTS-96	2241	285	42	41	0.51
IEEE-118	2556	0	48	48	0.84
IEEE-300	2646	6558	141	137	66.90

negative results given in Section 2.1.1 should not be ignored.

CHAPTER III

IMPOSING CONTIGUITY CONSTRAINTS IN POLITICAL DISTRICTING MODELS[1]

In the USA, congressional redistricting occurs every 10 years, soon after the census has been taken and the number of representatives for each state has been determined based on their populations (a process called reapportionment). A redistricting plan, which specifies how the district lines will be drawn within a US state, must satisfy certain constraints. Two typical constraints are that: (i) each district must be contiguous, and (ii) each district must have the "same" population. There are other properties that redistricting plans must satisfy (e.g., the Voting Rights Act prohibits racial gerrymandering) but they are often not as clear-cut as contiguity and population-equality or may vary by state (e.g., some states require the preservation of political subdivisions like counties).

Many redistricting plans will satisfy the contiguity and population-equality constraints, allowing redistricters to optimize a particular objective or to satisfy a set of additional constraints. This opens the door for state legislatures (who often control redistricting) to leverage the process to their benefit, resulting in partisan or incumbent gerrymanders. Indeed, gerrymandering has become so contentious lately, that it led to several cases before the Supreme Court of the United States in the last few years. These cases included: *Lamone v. Benisek* regarding Democratic gerrymandering in Maryland, *Rucho v. Common Cause* regarding Republican gerrymandering in North Carolina, and *Abbott v. Perez* regarding racial gerrymandering in Texas. Preliminary evidence of a gerrymander often includes

[1]This chapter is based on work with Austin Buchanan and Eugene Lykhovyd (Validi et al., 2020).

disproportionate electoral outcomes (e.g., in 2016, Republicans won 77% of North Carolina's congressional seats despite winning only 53% of the votes) or unusually shaped, non-compact districts (e.g., Maryland's 3rd congressional district[2], depicted in Figure 3.1, was drawn by the state's Democrats and has been compared to a "broken-winged pterodactyl, lying prostrate across the center of the state" by a federal judge (Linskey, 2012)). However, neither a disproportionate outcome nor an unusually shaped district is a tell-tale sign of gerrymandering (Duchin et al., 2019). In June 2019, the Supreme Court decided that partisan gerrymandering falls outside the purview of the federal courts, unleashing a new era of gerrymandering.

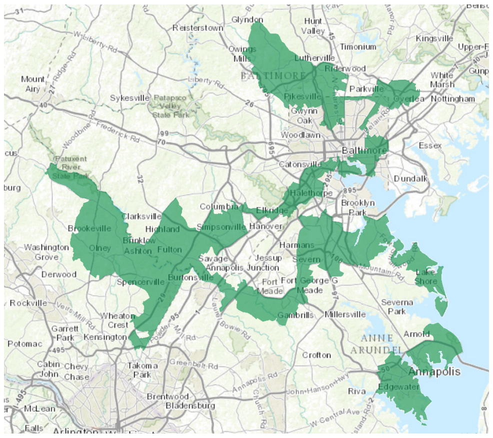


Figure 3.1: Maryland's 3rd congressional district following the 2010 census.

In an effort to de-politicize the redistricting process, some have suggested that "redistricting should be a bureaucratic, boring process where you get the census data, you turn the crank, and you get new maps for the next decade" (Boehm, 2018). Others have commented unfavorably on automated redistricting because: (1) reasonable people can disagree about what properties the "best" redistricting plan should satisfy, and (2) even if there were universal agreement on the desiderata, the resulting problem would almost certainly be NP-hard (Altman and McDonald, 2010). Nevertheless, the show must go on. Some redistricting plan must be

[2]Operations researchers might be interested to know that the INFORMS offices sit just outside of this district.

chosen, and computers will inevitably be used in its creation, whether to provide starting points for discussion, to refine preliminary plans, or to understand the limits of what is possible. In any case, imposing contiguity will be important; 23 states require contiguity by law, and the others almost always practice it anyway (Altman, 1998; Duchin et al., 2019).

Researchers have struggled to handle the contiguity constraints inherent in redistricting problems, despite a long history. Hess et al. (1965) proposed perhaps the first optimization model for redistricting. It sought an optimally compact redistricting plan, where compactness was measured in terms of a moment-of-inertia objective, subject to constraints on population equality. Due to the model's compactness-seeking objective, it tends to generate contiguous or nearly-contiguous districts, although none of the model's constraints explicitly impose contiguity. Consequently, Hess et al. manually adjusted their solutions to make them contiguous.

Since then, there have been several attempts to adjust the Hess model (or others similar to it) so that contiguity is explicitly ensured (Zoltners and Sinha, 1983; Drexl and Haase, 1999; Caro et al., 2004; Shirabe, 2009; Duque et al., 2011; Oehrlein and Haunert, 2017; Kim and Xiao, 2017). Nevertheless, researchers have remained pessimistic.

- "[Contiguity] constraints make [districting] much more difficult than other partitioning problems in combinatorial optimization, such as coloring or frequency assignment." (Ricca and Simeone, 2008)
- "[Contiguity] is particularly difficult to deal with and, sometimes, it is even discarded from [political districting] models and considered only a posteriori." (Ricca et al., 2013)
- "Ensuring contiguity efficiently seems to be an issue in exact methods [for political districting]." (Goderbauer and Winandy, 2018)
- "For exact methods, contiguity enforcement has been a major challenge." (Swamy et al., 2019b)

With this in mind, this chapter studies how to best impose contiguity in the context of the Hess model. We consider the following models:

1. SHIR, a flow-based model credited to Shirabe (2005, 2009) and detailed by Oehrlein and Haunert (2017);
2. CUT, a cut-based model proposed by Oehrlein and Haunert (2017) that draws from Carvajal et al. (2013) and others;
3. MCF, a new flow-based model that we show is equivalent in strength to CUT and stronger than SHIR at the cost of having more variables;
4. LCUT, a new cut-based model that is shown to be stronger than the other models.

We also examine the separation problems associated with the CUT and LCUT models; the former is shown to be solvable in time $O(n^2 \log^3 n)$, whereas the procedure used by Oehrlein and Haunert (2017) takes time $O(n^4)$. (Here, n is the number of vertices.)

To evaluate the performance of the four models, we conduct a thorough set of computational experiments, testing the four models on redistricting instances for every US state at the county and census tract levels. In a nod to Hess et al. (1965), we use the original moment-of-inertia objective function in our experiments. We find that the new LCUT model is the best-performing formulation on the county-level instances where the problem has more of a combinatorial flavor. On the tract-level instances, which have significantly more granularity, CUT and LCUT perform nearly the same and solve 21 instances to optimality.

The largest instance that CUT solves is for Indiana. This instance has $1,511$ census tracts and uses $(1,511)^2 = 2,283,121$ binary variables. To our knowledge, this is significantly larger than any districting instance ever solved in the literature by an exact method—with or without contiguity constraints. For example, Mehrotra et al. (1998) use branch-and-price for South Carolina, essentially at the county level, having approximately 50 vertices and 6 districts. Several of their steps were not automated, including the splitting and joining of several counties pre-solve, and manual adjustments post-solve for population-equality, and were ultimately unable to guarantee optimality. A more recent paper by Swamy et al. (2019b) heuristically reduces the sizes of instances by a series of graph contractions until $n \leq 200$ ($n^2 \leq 40,000$) at which point they are able to deploy their exact method. They motivate this graph contraction procedure by noting that Wisconsin has $1,409$ tracts and by pointing out the enormous size of the resulting MIP. This instance can also be solved with our techniques if Hess et al.'s original objective function is used.

To encourage future research in this area, and for purposes of transparency, we make our test instances, C++ source code, and redistricting plans (maps and block equivalency files) available at `https://github.com/zhelih/districting`. The source code is released under a GNU General Public License, which gives users the ability to run, study, share, and modify it.

3.1 Background and Literature Review

In this section, we give a brief overview of redistricting, particularly congressional redistricting for the US, and approaches for constructing redistricting plans. The literature on these topics is vast, and we can only cover the highlights. Interested readers are encouraged to refer to Di Cortona et al. (1999), Murphy et al. (2013), Ricca et al. (2013), and Goderbauer and Winandy (2018) for perspectives on redistricting from operations researchers, as well as Grofman (1985), Arrington (2010), and Bullock III (2010) for perspectives from social scientists.

3.1.1 Redistricting principles and laws in the US

A redistricting plan must satisfy certain state and federal laws. These laws are often crafted to ensure that *traditional redistricting principles* are followed (e.g., population-equality, contiguity, compactness, preservation of political subdivisions and communities of interest) or that the districting process does not disadvantage a particular group (e.g., a racial minority or members of a political party). Below we mention some examples of redistricting laws in the US. Our intent is to provide some context for the stylized redistricting problem that we will consider in this chapter, while also recognizing that the districting plans from our computational experiments will not consider all of these redistricting principles or laws (which also vary by state).

Population balance. Federal laws in the US require that congressional districts within a state all have the "same" population. This *one-person, one-vote* principle was formally interpreted by the Supreme Court to be a consequence of Article I, Section 2 of the US Constitution in the 1964 case *Reynolds v. Sims.* In the decades following *Reynolds*, the courts established tighter and tighter restrictions on how much population deviation is allowed. There is currently no threshold for population deviation beyond which a districting plan will necessarily be deemed legal (i.e., a "safe harbor"), and a population deviation of just *19 people* (0.0029%!) was ruled unconstitutional in 2002 by a federal district court in Pennsylvania (Hebert et al., 2010). Nevertheless, larger population deviations of up to 1% have been allowed if there is a compelling justification, such as the desire to satisfy a traditional redistricting principle. For example, West Virginia kept all of its counties intact at the price of a 0.79% population deviation (NCSL, 2019).

Race. Federal law also dictates what role race should (or should not) play in redistricting. For example, Section 2 of the Voting Rights Act (VRA) prohibits racial gerrymandering, disallowing any practice or procedure that inhibits a protected minority group from electing candidates of their choice. In the 1986 case *Thornburg v. Gingles*, the Supreme Court established when states must create "majority-minority" or minority-opportunity districts with the three-pronged *Gingles* test, the first prong of which requires that the minority group be sufficiently numerous and geographically compact. There are also *constitutional* limits on racial gerrymandering. For example, in the 1993 case *Shaw v. Reno*, the Supreme Court established that the Equal Protection Clause of the 14th Amendment prohibits states from the excessive or unjustified use of race when redistricting, especially if race predominates the map-making process to the exclusion of traditional redistricting principles (Hebert et al., 2010).

State laws (e.g., contiguity). States also enact laws regarding congressional redistricting. For example, contiguity is not federally required, so 23 states have imposed this requirement themselves; the other states almost always enact contiguous districts anyway (Duchin et al., 2019). States such as Iowa have additional laws regarding compactness, the preservation of political subdivisions, and the non-use of partisan data (NCSL, 2019).

3.1.2 Algorithms and models for redistricting

Any practical variant of redistricting is NP-hard (Altman, 1997), leading many researchers to propose their own heuristics (Ricca et al., 2013). A non-exhaustive list of examples include greedy construction heuristics (Vickrey, 1961; Kim, 2019), local search heuristics (King et al., 2012, 2015, 2018), metaheuristics like simulated annealing and tabu search (Bozkaya et al., 2003; Ricca and Simeone, 2008; Altman et al., 2011; Guo and Jin, 2011; Liu et al., 2016; Olson, 2019; Gutiérrez-Andrade et al., 2019), and generalizations of Voronoi diagrams (Miller, 2007; Svec et al., 2007; Ricca et al., 2008; Cohen-Addad et al., 2018; Levin and Friedler, 2019).

Recently, several researchers propose Markov chain Monte Carlo (MCMC) methods for generating large collections of redistricting plans, where the aim is understand the *distribution* of redistricting plans, which can provide a baseline with which to compare proposed or implemented plans (Fifield et al., 2015; Cho and Liu, 2018; Adler and Wang, 2019; DeFord et al., 2019). If a proposed redistricting plan is an outlier (say, with respect to seat share distribution), this might suggest an intent to gerrymander. MCMC sampling methods for redistricting are quite similar to local search in that they move from one feasible solution to a neighboring feasible solution.

The literature also contains many exact methods for redistricting, including numerous IP formulations (Ricca et al., 2013). Perhaps the two most notable are what we will call the *Hess* model and the *set partition* model.

The Hess model, detailed in Section 3.1.4, is a constrained k-median model. Its essence can be found in numerous papers on redistricting in the OR literature (Hess et al., 1965; Hojati, 1996; Gentry et al., 2015). The most popular technique for imposing contiguity in the context of the Hess model is a flow-based formulation credited to Shirabe (2005, 2009) and fully fleshed out by Oehrlein and Haunert (2017). This formulation, which is detailed in Section 3.2.1, is easy to implement and has been used in one form or another in several papers (Duque et al., 2011; Gopalan et al., 2013; Oehrlein and Haunert, 2017; Kong et al., 2019; Swamy et al., 2019b). Another notable approach for imposing contiguity uses graph cuts. Specifically, Oehrlein and Haunert (2017) propose to use a, b-separator inequalities for redistricting (detailed in Section 3.2.3); others have proposed related inequalities that are weaker (Drexl and Haase, 1999) or not valid (Zoltners and Sinha, 1983).

In the set partition model, there is a binary variable for each possible district, and the task is to select k of them such that every part of the state is covered exactly once. The approaches of Garfinkel and Nemhauser (1970) and Mehrotra et al. (1998) are based on formulations of this type. It should be noted that, in general, the set of possible districts grows exponentially, meaning that formulations of this type are solved either using a select subset of district variables or the variables are introduced on-the-fly via column generation. Contiguity is handled during pricing.

3.1.3 Notation and problem definition

When trying to impose the contiguity constraints involved in political districting, it is helpful to use the so-called *contiguity graph* (Ricca et al., 2013), also known as the adjacency graph or dual graph. In this graph $G = (V, E)$, each vertex $v \in V$ represents a contiguous parcel of land (e.g., a county or census tract), and there is an edge $\{u, v\} \in E$ connecting vertices u and v when the corresponding land parcels share a border of nonzero length (e.g., it is not enough to meet at a point[3]). By this construction, G will be simple and planar, and thus sparse. Indeed, its number of edges $m := |E|$ is linear with respect to the number of vertices $n := |V|$; Euler's polyhedral formula implies that $m \leq 3n - 6$ when $n \geq 3$. The (open) neighborhood of vertex i is denoted by $N(i) := \{j \in V \mid \{i, j\} \in E\}$, and the closed neighborhood is denoted by $N[i] := N(i) \cup \{i\}$. Other necessary data includes:

- the number k of districts to be created;
- the population p_v of each land parcel $v \in V$;
- the minimum and maximum population (L and U) allowed in a district.

Another piece of data that might be used to construct a "compact" districting plan is the distance d_{ij} between (the centers of) land parcels i and j. Indeed, these distances appear in the compactness-seeking objective function in the model of Hess et al. (1965).

For the purposes of this chapter, the districting problem is defined as follows. If any exist, find a partition of the vertex set V into *districts* such that:

1. each vertex belongs to exactly one district;
2. there are k districts;
3. each district V' satisfies the population bounds, i.e., $L \leq \sum_{i \in V'} p_i \leq U$;

[3]Authors sometimes use the notions of *queen contiguity* and *rook contiguity.* In the former, land parcels that meet at a point are considered to be adjacent, while in the latter (which we use) they are not.

4. each district V' is contiguous, i.e., $G[V']$ is connected.

We call these four constraints the *bright-line rules*, and any districting plan that satisfies them is said to be *feasible*. If there are multiple feasible solutions, the task is to find one that is most compact with respect to the penalties w, which are discussed next.

3.1.4 The Hess model

Hess et al. (1965) introduced *the* classical integer program for political districting, which uses the following n^2 binary variables.

$$x_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is assigned to (the district centered at) vertex } j \\ 0 & \text{otherwise.} \end{cases}$$

The Hess formulation is as follows, where w_{ij} is a penalty charged for assigning i to j.

$$\min \sum_{i\in V}\sum_{j\in V} w_{ij}x_{ij} \tag{3.1a}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{3.1b}$$

$$\sum_{j\in V} x_{jj} = k \tag{3.1c}$$

$$Lx_{jj} \leq \sum_{i\in V} p_i x_{ij} \leq Ux_{jj} \qquad \forall j \in V \tag{3.1d}$$

$$x_{ij} \leq x_{jj} \qquad \forall i,j \in V \tag{3.1e}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V. \tag{3.1f}$$

Constraints (3.1b) ensure that each vertex is assigned to a district. Constraint (3.1c) ensures that k districts are chosen. Constraints (3.1d) ensure that the population of each district lies between L and U. Constraints (3.1e) were not originally included by Hess et al. (1965)

but are usually added for strength (Ricca et al., 2013). In our code, we also introduce a new variable to replace the expression $\sum_{i \in V} p_i x_{ij}$ in constraints (3.1d), thus reducing the formulation's size in computer memory by 20%.

Hess et al. (1965) considered a *moment-of-inertia* objective, defining the penalties w_{ij} as follows.

$$\text{(penalty for moment-of-inertia objective)} \quad w_{ij} := p_i d_{ij}^2.$$

Meanwhile, the traditional objective used in facility location would capture the total number of miles traveled if the state's inhabitants were to drive to their district centers with the penalties $w_{ij} := p_i d_{ij}$, where d is set using road distances (Daskin and Tucker, 2018). Duchin (2018) has humorously described this measure of compactness as an "appealing one, particularly if you imagine replacing distance with travel time" because you can think of it as answering the question: "How long does it take you to go yell at your representative?" Another objective that has been considered by Hojati (1996) and Gopalan et al. (2013) uses squared Euclidean distances $w_{ij} := d_{ij}^2$. Others such as Swamy et al. (2019a) and Mehrotra et al. (1998) use the simpler penalty $w_{ij} := d_{ij}$. In fact, Mehrotra et al. (1998) take d_{ij} as the hop-based distance between i and j in the contiguity graph. In a nod to Hess et al. (1965), we use the original moment-of-inertia objective in our experiments; however, our analysis and techniques apply regardless of which penalties are used. For more information about the many measures of compactness in the literature, we refer the reader to the compactness critiques of Young (1988) and the classification by Niemi et al. (1990).

The Hess model is the foundation of our proposed formulations. The feasible region of its LP relaxation, which we denote by $\mathcal{P}_{\text{HESS}}$, is defined as follows.

$$\mathcal{P}_{\text{HESS}} := \left\{ x \in \mathbb{R}_+^{n \times n} \;\middle|\; x \text{ satisfies constraints (3.1b), (3.1c), (3.1d), (3.1e)} \right\}.$$

Note that the bounds $x_{ij} \leq 1$ are implied by nonnegativity and the assignment constraints (3.1b).

3.2 Contiguity Models

Here, we consider four different models for imposing contiguity in the context of the Hess model: SHIR, CUT, MCF, and LCUT. Two of them have appeared in the previous literature (SHIR and CUT), while two others are new (MCF and LCUT). The flow-based models SHIR and MCF refer to the "bidirected" version of the contiguity graph which we denote by $D = (V, A)$. This directed graph D is obtained from $G = (V, E)$ by replacing each undirected edge $\{u, v\} \in E$ by its directed counterparts (u, v) and (v, u). Thus, $|A| = 2|E|$. The set of edges pointing away from vertex i is denoted by $\delta^+(i)$, and the set of edges pointing towards vertex j is denoted by $\delta^-(j)$.

3.2.1 SHIR

Oehrlein and Haunert (2017) propose a flow-based formulation, adapted from Shirabe (2009). We provide (essentially) the same formulation below and call it SHIR. This formulation uses the following flow variables.

$f^v_{ij} =$ the amount of flow, originating at district center v, that is sent across edge (i, j).

The SHIR formulation is as follows, where $f^j(S)$ for $S \subseteq A$ is shorthand for $\sum_{(u,v)\in S} f^j_{uv}$.

$$
\begin{aligned}
&x \in \mathcal{P}_{\text{HESS}} && && \text{(3.2a)}\\
&f^j(\delta^-(i)) - f^j(\delta^+(i)) = x_{ij} && \forall i \in V \setminus \{j\},\ \forall j \in V && \text{(3.2b)}\\
&f^j(\delta^-(i)) \le (n-1)x_{ij} && \forall i \in V \setminus \{j\},\ \forall j \in V && \text{(3.2c)}\\
&f^j(\delta^-(j)) = 0 && \forall j \in V && \text{(3.2d)}\\
&f^v_{ij} \ge 0 && \forall (i,j) \in A,\ \forall v \in V && \text{(3.2e)}\\
&x_{ij} \in \{0,1\} && \forall i,j \in V. && \text{(3.2f)}
\end{aligned}
$$

Now, we define the polytope $\mathcal{P}_{\text{SHIR}}$ as follows.

$$
\mathcal{P}_{\text{SHIR}} := \left\{ (x,f) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{2mn} \;\middle|\; (x,f) \text{ satisfies constraints (3.2a)} - \text{(3.2e)} \right\}.
$$

Remark 2. *The following equations are implied in* $\mathcal{P}_{\text{SHIR}}$.

$$
f^j(\delta^+(j)) = \sum_{i \in V\setminus\{j\}} x_{ij} \qquad \forall j \in V. \tag{3.3}
$$

Proof. Consider a point $(\hat{x}, \hat{f})$ that belongs to $\mathcal{P}_{\text{SHIR}}$. Then, for every vertex $j \in V$,

$$
\begin{aligned}
\hat{f}^j(\delta^+(j)) &= \hat{f}^j(\delta^+(j)) - \hat{f}^j(\delta^-(j)) && \text{(3.4a)}\\
&= \hat{f}^j(\delta^+(j)) - \hat{f}^j(\delta^-(j)) - \sum_{i\in V}\left(\hat{f}^j(\delta^+(i)) - \hat{f}^j(\delta^-(i))\right) && \text{(3.4b)}\\
&= -\sum_{i \in V\setminus\{j\}}\left(\hat{f}^j(\delta^+(i)) - \hat{f}^j(\delta^-(i))\right) && \text{(3.4c)}\\
&= \sum_{i \in V\setminus\{j\}}\left(\hat{f}^j(\delta^-(i)) - \hat{f}^j(\delta^+(i))\right) = \sum_{i\in V\setminus\{j\}} \hat{x}_{ij}. && \text{(3.4d)}
\end{aligned}
$$

Here, equation (3.4a) holds by constraints (3.2d), equation (3.4b) holds because the flow variables in the summation cancel each other, and equation (3.4d) holds by constraints (3.2b).

□

3.2.2 MCF

The SHIR formulation uses "big-M" constraints (3.2c), which can result in a weak linear programming relaxation. With this in mind, we propose a different flow-based formulation which avoids the big-M constraints via a different variable definition.

$$f_{ij}^{ab} = \begin{cases} 1 & \text{if edge } (i,j) \in A \text{ is on the path to vertex } a \text{ from its district's center } b \\ 0 & \text{otherwise.} \end{cases}$$

The resulting MCF formulation, which can be viewed as a disaggregation of SHIR, is as follows, where $f^{ab}(S)$ for $S \subseteq A$ is shorthand for $\sum_{(u,v)\in S} f_{uv}^{ab}$.

$$x \in \mathcal{P}_{\text{HESS}} \tag{3.5a}$$

$$f^{ab}(\delta^+(b)) - f^{ab}(\delta^-(b)) = x_{ab} \qquad \forall a \in V \setminus \{b\},\ \forall b \in V \tag{3.5b}$$

$$f^{ab}(\delta^+(i)) - f^{ab}(\delta^-(i)) = 0 \qquad \forall i \in V \setminus \{a,b\},\ \forall a \in V \setminus \{b\},\ \forall b \in V \tag{3.5c}$$

$$f^{ab}(\delta^-(b)) = 0 \qquad \forall a \in V \setminus \{b\},\ \forall b \in V \tag{3.5d}$$

$$f^{ab}(\delta^-(j)) \leq x_{jb} \qquad \forall j \in V \setminus \{b\},\ \forall a \in V \setminus \{b\},\ \forall b \in V \tag{3.5e}$$

$$f_{ij}^{ab} \geq 0 \qquad \forall (i,j) \in A,\ \forall a \in V \setminus \{b\},\ \forall b \in V \tag{3.5f}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V. \tag{3.5g}$$

To our knowledge, this formulation is new in the context of districting. Now, we define the polytope $\mathcal{P}_{\text{MCF}}$ as follows.

$$\mathcal{P}_{\text{MCF}} := \left\{(x,f) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{2mn(n-1)} \;\middle|\; (x,f) \text{ satisfies constraints (3.5a)} - \text{(3.5f)}\right\}.$$

Note that the bounds $f_{ij}^{ab} \leq 1$ are implied in $\mathcal{P}_{\text{MCF}}$ by nonnegativity and constraints (3.5e).

Remark 3. *The following equations are implied in* $\mathcal{P}_{\text{MCF}}$*.*

$$f^{ab}(\delta^+(a)) - f^{ab}(\delta^-(a)) = -x_{ab} \qquad \forall a \in V \setminus \{b\},\ \forall b \in V \tag{3.6a}$$

$$f^{ab}(\delta^+(a)) = 0 \qquad \forall a \in V \setminus \{b\},\ \forall b \in V. \tag{3.6b}$$

Proof. Consider a point $(\hat{x}, \hat{f})$ that belongs to $\mathcal{P}_{\text{MCF}}$. For distinct vertices $a, b \in V$,

$$\hat{f}^{ab}(\delta^+(a)) - \hat{f}^{ab}(\delta^-(a)) = \hat{f}^{ab}(\delta^+(a)) - \hat{f}^{ab}(\delta^-(a)) - \sum_{j \in V} \left(\hat{f}^{ab}(\delta^+(j)) - \hat{f}^{ab}(\delta^-(j))\right) \tag{3.7a}$$

$$= - \sum_{j \in V \setminus \{a\}} \left(\hat{f}^{ab}(\delta^+(j)) - \hat{f}^{ab}(\delta^-(j))\right) \tag{3.7b}$$

$$= - \left(\hat{f}^{ab}(\delta^+(b)) - \hat{f}^{ab}(\delta^-(b))\right) = -\hat{x}_{ab}. \tag{3.7c}$$

Here, equation (3.7a) holds because the flow variables in the summation cancel each other. The first equation in line (3.7c) holds by constraints (3.5c) and the second holds by constraints (3.5b). So, the point $(\hat{x}, \hat{f})$ satisfies the equations (3.6a). Further,

$$0 \leq \hat{f}^{ab}(\delta^+(a)) = -\hat{x}_{ab} + \hat{f}^{ab}(\delta^-(a)) \leq -\hat{x}_{ab} + \hat{x}_{ab} = 0.$$

Here, the first equation holds by the implied equations (3.6a), and the last inequality holds by constraints (3.5e). Thus, the equations (3.6b) are implied in $\mathcal{P}_{\text{MCF}}$. $\square$

3.2.3 CUT

The CUT formulation, which was first used for districting by Oehrlein and Haunert (2017), is based on the concept of a, b-separators, see also Carvajal et al. (2013); Buchanan et al. (2015); Ahn and Park (2015); Fischetti et al. (2017b); Wang et al. (2017). An example of an

a, b-separator is given in Figure 3.2.

Definition 6 (a, b-separator)**.** *A subset $C \subseteq V \setminus \{a, b\}$ of vertices is called an a, b-separator for $G = (V, E)$ if there is no path from a to b in $G - C$.*

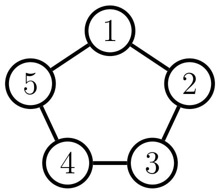


Figure 3.2: Here, $C = \{2, 5\}$ is a $1, 3$-separator, and the inequality $x_{13} \leq x_{23} + x_{53}$ is valid when districts are required to be connected.

The resulting a, b-separator inequalities (3.8b) are written for every ordered pair (a, b) of nonadjacent vertices and every a, b-separator C, which we denote by the shorthand $\forall (a, b, C)$. As is usual, it is sufficient to consider *minimal* a, b-separators, where minimality is taken by inclusion.

$$x \in \mathcal{P}_{\text{HESS}} \tag{3.8a}$$

$$x_{ab} \leq \sum_{c \in C} x_{cb} \qquad \forall (a, b, C) \tag{3.8b}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{3.8c}$$

Now, we define the polytope $\mathcal{P}_{\text{CUT}}$ as follows.

$$\mathcal{P}_{\text{CUT}} := \left\{ x \in \mathbb{R}^{n \times n} \;\middle|\; x \text{ satisfies constraints (3.8a)} - \text{(3.8b)} \right\}.$$

3.2.4 LCUT

The a, b-separator inequalities (3.8b) use the fact that the vertices assigned to a vertex b are required to induce a connected subgraph. By exploiting additional information (e.g., that the population bound U must also be satisfied), we can write stronger inequalities.

We formalize this using the concept of a length-U a,b-separator. To do this, we need to refer to vertex-weighted distances, where the weight of a vertex i is its population p_i. The distance $\text{dist}_{G,p}(a,b)$ from a to b is the length $\sum_{v \in V(P)} p_v$ of a shortest vertex-weighted path P from a to b. An example is given in Figure 3.3.

Definition 7 (length-U a,b-separator)**.** *A subset $C \subseteq V \setminus \{a,b\}$ of vertices is called a length-U a,b-separator in $G=(V,E)$, with respect to vertex weights p, if* $\text{dist}_{G-C,p}(a,b) > U$.

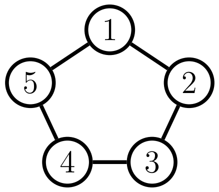


Figure 3.3: When $p = \mathbf{1}$, $C = \{2\}$ is a length-3 1,3-separator, and the inequality $x_{13} \leq x_{23}$ is valid when districts are required to have population at most 3 and be connected.

With this definition, we can write inequalities having the exact same form as the a,b-separator inequalities (3.8b), except that C will now be a length-U a,b-separator. We write these inequalities for every ordered pair (a,b) of *distinct* (possibly adjacent) vertices and every (minimal) length-U a,b-separator C, which we again denote by the shorthand $\forall(a,b,C)$.

$$x \in \mathcal{P}_{\text{HESS}} \tag{3.9a}$$

$$x_{ab} \leq \sum_{c \in C} x_{cb} \qquad \forall(a,b,C) \tag{3.9b}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V. \tag{3.9c}$$

Similar length-bounded cut models have been proposed recently for other problems (Salemi and Buchanan, 2020; Validi and Buchanan, 2019b; Arslan et al., 2019, 2020).

Remark 4. *If* $\text{dist}_{G,p}(a,b) > U$*, then $C = \emptyset$ is a length-U a,b-separator and $x_{ab} \leq 0$ is the associated length-U a,b-separator inequality.*

Now, we define polytope $\mathcal{P}_{\text{LCUT}}$ as follows.

$$\mathcal{P}_{\text{LCUT}} := \left\{ x \in \mathbb{R}^{n \times n} \;\middle|\; x \text{ satisfies constraints (3.9a)} - \text{(3.9b)} \right\}.$$

3.3 Analysis of the Formulations

We now compare the strength of the four formulations, prove their correctness, and analyze the separation problems associated with the exponentially-sized models CUT and LCUT.

3.3.1 Formulation strength

The main result of this subsection is the following theorem. A specific result is that the newly proposed LCUT formulation is the strongest formulation in this chapter.

Theorem 7. *For every instance of districting,*

$$\mathcal{P}_{\text{LCUT}} \subseteq \mathcal{P}_{\text{CUT}} = \operatorname{proj}_x \mathcal{P}_{\text{MCF}} \subseteq \operatorname{proj}_x \mathcal{P}_{\text{SHIR}},$$

and there exist instances for which the inclusions are strict.

Proof. This follows by Lemmata 4, 5, and 6, which are proven below. □

Lemma 4. *For every instance of districting, $\mathcal{P}_{\text{LCUT}} \subseteq \mathcal{P}_{\text{CUT}}$, and this inclusion can be strict.*

Proof. Since a, b-separator inequalities are length-U a, b-separator inequalities for every U, the inclusion $\mathcal{P}_{\text{LCUT}} \subseteq \mathcal{P}_{\text{CUT}}$ holds. Figure 3.4 gives an example where $\mathcal{P}_{\text{LCUT}} \neq \mathcal{P}_{\text{CUT}}$. So, the inclusion can be strict. □

Lemma 5. *For every instance of districting, $\mathcal{P}_{\text{CUT}} = \operatorname{proj}_x \mathcal{P}_{\text{MCF}}$.*

Proof. ($\supseteq$) Suppose that $(\hat{x}, \hat{f})$ belongs to $\mathcal{P}_{\text{MCF}}$. To show that $\hat{x}$ belongs to $\mathcal{P}_{\text{CUT}}$, it suffices to show that $\hat{x}$ satisfies constraints (3.8b) for an arbitrary pair of nonadjacent vertices $a, b \in V$

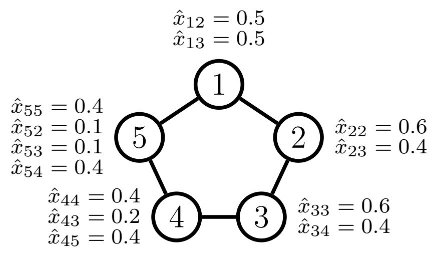


Figure 3.4: An example showing $\mathcal{P}_{\text{LCUT}} \neq \mathcal{P}_{\text{CUT}}$. Here, $L = k = 2$, $U = 3$ and $p = \mathbf{1}$. While the point $\hat{x}$ belongs to $\mathcal{P}_{\text{CUT}}$, it does not belong to $\mathcal{P}_{\text{LCUT}}$ because it violates the length-3 $1,3$-separator inequality (3.9b) for $a = 1$, $b = 3$, and $C = \{2\}$.

and a,b-separator $C \subseteq V \setminus \{a, b\}$. Let B the set of vertices reachable from b in the graph $G - C$. Then,

$$\begin{aligned}
\hat{x}_{ab} &= \hat{f}^{ab}(\delta^+(b)) - \hat{f}^{ab}(\delta^-(b)) && \text{(3.10a)}\\
&= \hat{f}^{ab}(\delta^+(B)) - \hat{f}^{ab}(\delta^-(B)) && \text{(3.10b)}\\
&\leq \hat{f}^{ab}(\delta^+(B)) && \text{(3.10c)}\\
&\leq \hat{f}^{ab}(\delta^-(C)) && \text{(3.10d)}\\
&\leq \sum_{j \in C} \hat{f}^{ab}(\delta^-(j)) && \text{(3.10e)}\\
&\leq \sum_{j \in C} \hat{x}_{jb}. && \text{(3.10f)}
\end{aligned}$$

Here, equation (3.10a) holds by constraints (3.5b), equation (3.10b) holds by constraints (3.5c), and inequalities (3.10c)–(3.10e) hold by nonnegativity of $\hat{f}$. Finally, inequality (3.10f) holds by constraints (3.5e).

($\subseteq$) Suppose that $\hat{x}$ belongs to $\mathcal{P}_{\text{CUT}}$. For every pair of distinct vertices $a, b \in V$, we define $\hat{f}^{ab}$ as follows. If vertices a and b are adjacent, then define $\hat{f}^{ab}_{ba} := \hat{x}_{ab}$ and $\hat{f}^{ab}_{ij} := 0$ for all other arcs $(i, j) \in A \setminus \{(b, a)\}$. If a and b are nonadjacent, consider the following maximum

b-a flow problem in which the fixed $\hat{x}_{jb}$ values are used as vertex capacities.

$$\max f^{ab}(\delta^+(b)) \tag{3.11a}$$

$$f^{ab}(\delta^+(i)) - f^{ab}(\delta^-(i)) = 0 \qquad \forall i \in V \setminus \{a, b\} \tag{3.11b}$$

$$f^{ab}(\delta^-(b)) = 0 \tag{3.11c}$$

$$f^{ab}(\delta^-(j)) \leq \hat{x}_{jb} \qquad \forall j \in V \setminus \{b\} \tag{3.11d}$$

$$f^{ab}_{ij} \geq 0 \qquad \forall (i, j) \in A. \tag{3.11e}$$

This problem is feasible (by the zero flow), and its objective is at most one because the sink a has capacity $\hat{x}_{ab}$ by (3.11d). Let $\hat{f}^{ab}_{ij}$ for $(i, j) \in A$ be an optimal solution to this flow problem.

Now we are to show that $(\hat{x}, \hat{f})$ satisfies constraints (3.5b)–(3.5f). They are easily satisfied when the constraint quantifiers a and b are adjacent, by the simple definition of $\hat{f}^{ab}$ in this case. When the constraint quantifiers a and b are nonadjacent, constraints (3.5c)–(3.5f) hold by the constraints defining the flow problem. So it remains to show that $(\hat{x}, \hat{f})$ satisfies constraints (3.5b) for nonadjacent vertices a and b. By classical results of Ford Jr and Fulkerson (1962) (see section 1.11 on vertex capacities), the flow value $\hat{f}^{ab}(\delta^+(b))$ is equal to the weight $\sum_{j \in C} \hat{x}_{jb}$ of a minimum-weight b, a-separator C, where vertex j has weight $\hat{x}_{jb}$. Then,

$$\hat{x}_{ab} \leq \sum_{j \in C} \hat{x}_{jb} = \hat{f}^{ab}(\delta^+(b)) = \hat{f}^{ab}(\delta^-(a)) \leq \hat{x}_{ab}.$$

Here, the first inequality holds by assumption that $\hat{x}$ satisfies constraints (3.8b), and the last inequality holds by the capacity of vertex a in the flow problem (3.11d). Then, because $\hat{f}^{ab}(\delta^-(b)) = 0$ by the flow problem (3.11c), the constraint $\hat{f}^{ab}(\delta^+(b)) - \hat{f}^{ab}(\delta^-(b)) = \hat{x}_{ab}$ holds. □

Lemma 6. *For every instance of districting,* $\text{proj}_x\, \mathcal{P}_{\text{MCF}} \subseteq \text{proj}_x\, \mathcal{P}_{\text{SHIR}}$*, and this can be* *strict.*

Proof. To prove $\operatorname{proj}_x \mathcal{P}_{\text{MCF}} \subseteq \operatorname{proj}_x \mathcal{P}_{\text{SHIR}}$, consider a point $(\bar{x}, \hat{f})$ that belongs to $\mathcal{P}_{\text{MCF}}$. We construct $\bar{f}$ such that $(\bar{x}, \bar{f})$ belongs to $\mathcal{P}_{\text{SHIR}}$. For every vertex $b \in V$ and every edge $(i, j) \in A$, let

$$\bar{f}^b_{ij} := \sum_{a \in V \setminus \{b\}} \hat{f}^{ab}_{ij}.$$

To show that $(\bar{x}, \bar{f})$ satisfies constraints (3.2b), consider distinct vertices $i, b \in V$. Then,

$$\bar{f}^b(\delta^-(i)) - \bar{f}^b(\delta^+(i)) = \sum_{a \in V \setminus \{b\}} \hat{f}^{ab}(\delta^-(i)) - \sum_{a \in V \setminus \{b\}} \hat{f}^{ab}(\delta^+(i)) \tag{3.12a}$$

$$= \sum_{a \in V \setminus \{b,i\}} \left(\hat{f}^{ab}(\delta^-(i)) - \hat{f}^{ab}(\delta^+(i)) \right) + \hat{f}^{ib}(\delta^-(i)) - \hat{f}^{ib}(\delta^+(i)) \tag{3.12b}$$

$$= 0 + \hat{f}^{ib}(\delta^-(i)) - \hat{f}^{ib}(\delta^+(i)) = \bar{x}_{ib}. \tag{3.12c}$$

Here, equation (3.12a) holds by the definition of $\bar{f}$, and equations (3.12c) holds by constraints (3.5c) and (3.6a).

To show that $(\bar{x}, \bar{f})$ satisfies constraints (3.2c), consider distinct vertices $i, b \in V$. Then,

$$\bar{f}^b(\delta^-(i)) = \sum_{a \in V \setminus \{b\}} \hat{f}^{ab}(\delta^-(i)) \le \sum_{a \in V \setminus \{b\}} \bar{x}_{ib} = (n-1)\bar{x}_{ib}.$$

Here, the first equation holds by the definition of $\bar{f}$, and the inequality holds by constraints (3.5e).

To show that $(\bar{x}, \bar{f})$ satisfies constraints (3.2d), consider a vertex $j \in V$. Then,

$$\bar{f}^j(\delta^-(j)) = \sum_{i \in V \setminus \{j\}} \hat{f}^{ij}(\delta^-(j)) = 0.$$

Here, the first equation holds by the definition of $\bar{f}$, and the second holds by constraints (3.5d).

Figure 3.5 gives an example where $\mathcal{P}_{\text{CUT}} \neq \text{proj}_x \mathcal{P}_{\text{SHIR}}$. Since $\mathcal{P}_{\text{CUT}} = \text{proj}_x \mathcal{P}_{\text{MCF}}$ by Lemma 5, this shows that the inclusion $\text{proj}_x \mathcal{P}_{\text{MCF}} \subseteq \text{proj}_x \mathcal{P}_{\text{SHIR}}$ can be strict. $\square$

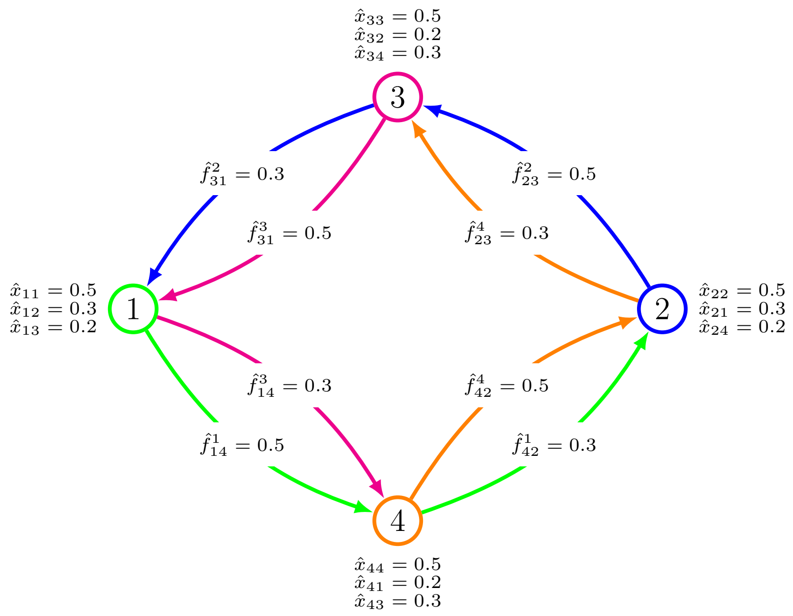


Figure 3.5: An example showing $\text{proj}_x \mathcal{P}_{\text{SHIR}} \neq \mathcal{P}_{\text{CUT}}$. Here, $L = U = k = 2$ and $p = \mathbf{1}$. While the point $(\hat{x}, \hat{f})$ belongs to $\mathcal{P}_{\text{SHIR}}$, $\hat{x}$ does not belong to $\mathcal{P}_{\text{CUT}}$ because it violates the a,b-separator inequality (3.8b) for $a = 1$, $b = 2$, and $C = \{3, 4\}$.

3.3.2 Formulation correctness

Here we consider the *correctness* of the formulations. By this, we mean that they allow precisely those plans that satisfy the four bright-line rules. To our knowledge, no previous work has explicitly proven the correctness of the SHIR and CUT formulations. While their correctness is not surprising, we feel that this step is critical to safeguard against seemingly innocuous formulations, see Validi and Buchanan (2019a).

Definition 8. *A given* $\hat{x} \in \{0,1\}^{n \times n}$ *is consistent if* $\hat{x}_{ij} \leq \hat{x}_{jj}$ *for all* $i, j \in V$.

In the context of the x variables, a districting plan is represented as follows. If a given

$\hat{x} \in \{0,1\}^{n\times n}$ is consistent, the associated set of district "roots" is given by

$$R(\hat{x}) := \{b \in V \mid \hat{x}_{bb} = 1\},$$

and the district rooted at $b \in R(\hat{x})$ is given by

$$V_b(\hat{x}) := \{a \in V \mid \hat{x}_{ab} = 1\}.$$

Supposing that $\hat{x}$ is consistent, the four bright-line rules can be expressed mathematically as follows.

1. each vertex $i \in V$ belongs to exactly one of the sets $V_r(\hat{x})$, $r \in R(\hat{x})$;
2. $|R(\hat{x})| = k$;
3. for every $r \in R(\hat{x})$, $L \leq \sum_{i \in V_r(\hat{x})} p_i \leq U$;
4. for every $r \in R(\hat{x})$, $V_r(\hat{x})$ induces a connected subgraph in G.

We restate the correctness theorem for completeness.

Theorem. *Formulations SHIR, MCF, CUT, and LCUT are correct.*

Proof. We are to show that, for each formulation F, $\hat{x} \in \{0,1\}^{n\times n}$ is consistent and satisfies the four bright-line rules if and only if $\hat{x} \in \operatorname{proj}_x F$. By Theorem 7, it suffices to show the two claims:

1. if $\hat{x} \in \{0,1\}^{n\times n}$ is consistent and satisfies the four bright-line rules, then $\hat{x} \in \mathcal{P}_{\text{LCUT}}$; and
2. if $(\hat{x}, \hat{f}) \in \mathcal{P}_{\text{SHIR}}$ and $\hat{x} \in \{0,1\}^{n\times n}$, then $\hat{x}$ is consistent and satisfies the four bright-line rules.

To prove the first claim, suppose that $\hat{x}$ satisfies the four bright-line rules and is consistent. By the first three rules, $\hat{x}$ satisfies the constraints (3.9a) from the Hess model. So, all that remains is to show that $\hat{x}$ satisfies the length-U a,b-separator inequalities (3.9b). So, consider vertices a and b and a length-U a,b-separator $C \subseteq V \setminus \{a,b\}$. If $\hat{x}_{ab} = 0$, then the constraint (3.9b) is trivially satisfied, so suppose $\hat{x}_{ab} = 1$. This implies that $a \in V_b(\hat{x})$. By the fourth rule, $V_b(\hat{x})$ induces a connected subgraph in G, implying that there exists a path P from a to b in $G[V_b(\hat{x})]$. Moreover, this path P has length at most U, because $V(P)$ is a subset of $V_b(\hat{x})$ and because $p(V_b(\hat{x})) \leq U$ by the third rule. By the definition of a length-U a,b-separator, there is at least one vertex $c \in V_b(\hat{x})$ that belongs to both C and $V(P)$, and so constraint (3.9b) is satisfied as

$$\hat{x}_{ab} = 1 = \hat{x}_{cb} \leq \sum_{j \in C} \hat{x}_{jb}.$$

To prove the second claim, suppose that $(\hat{x}, \hat{f}) \in \mathcal{P}_{\text{SHIR}}$ and $\hat{x} \in \{0,1\}^{n \times n}$. This implies $\hat{x}$ is consistent. By the correctness of the Hess formulation, the districting plan $V_b(\hat{x})$, $b \in R(\hat{x})$, satisfies the first three bright-line rules. So, it suffices to show that the fourth holds, i.e., that each vertex subset $V_b(\hat{x})$ induces a connected subgraph $G_b := G[V_b(\hat{x})]$ in G. For contradiction purposes, suppose that some $V_b(\hat{x})$ induces at least two connected components in G, and let S be the vertex set of a component of G_b that does not contain vertex b. Let $N(S)$ be the neighborhood of S, i.e.,

$$N(S) := \{ j \in V \setminus S \mid \exists\, v \in S \ni \{j,v\} \in E \}.$$

Then,

$$1 \leq |S| = \sum_{j \in S} \hat{x}_{jb} = \sum_{j \in S} \left(\hat{f}^b(\delta^-(j)) - \hat{f}^b(\delta^+(j)) \right) \tag{3.13a}$$

$$= \sum_{j \in S} \left(\sum_{i \in N(j)} (\hat{f}^b_{ij} - \hat{f}^b_{ji}) \right) \tag{3.13b}$$

$$= \sum_{j \in S} \sum_{i \in N(j) \cap S} \left(\hat{f}^b_{ij} - \hat{f}^b_{ji} \right) + \sum_{j \in S} \sum_{i \in N(j) \cap (V \setminus S)} \left(\hat{f}^b_{ij} - \hat{f}^b_{ji} \right) \tag{3.13c}$$

$$= 0 + \left(\sum_{j \in S} \sum_{i \in N(j) \cap (V \setminus S)} \hat{f}^b_{ij} - \sum_{j \in S} \sum_{i \in N(j) \cap (V \setminus S)} \hat{f}^b_{ji} \right) \tag{3.13d}$$

$$= 0 + \left(\sum_{j \in S} \sum_{i \in N(j) \cap (V \setminus S)} \hat{f}^b_{ij} - 0 \right) \tag{3.13e}$$

$$\leq 0 + \left(\sum_{i \in N(S)} \hat{f}^b(\delta^+(i)) - 0 \right) \tag{3.13f}$$

$$= \sum_{i \in N(S)} \hat{f}^b(\delta^-(i)) - \sum_{i \in N(S)} \hat{x}_{ib} \tag{3.13g}$$

$$= \sum_{i \in N(S)} \hat{f}^b(\delta^-(i)) - 0 \leq 0. \tag{3.13h}$$

Equation (3.13a) holds by constraints (3.2b). Equation (3.13d) holds because, in the left sum, each flow variable $\hat{f}^b_{ij}$ with $i, j \in S$ appears once with a positive coefficient and once with a negative coefficient, so they cancel each other. Equation (3.13e) holds by constraints (3.2c) and (3.2e), because $\hat{x}_{ib} = 0$ for every vertex $i \in N(S)$. Inequality (3.13f) holds by constraints (3.2e). Equation (3.13g) holds by constraints (3.2b). The equation in line (3.13h) holds because $\hat{x}_{ib} = 0$ for every vertex $i \in N(S)$. The inequality in line (3.13h) then holds by constraints (3.2c). This results in the contradiction $1 \leq 0$. □

3.3.3 The separation problems

Since the models CUT and LCUT have exponentially many constraints, it is important to study the associated separation problems so that inequalities can be added to the model on-the-fly as needed, instead of all up front which would render the models useless. This problem asks: given x^*, is there an inequality from the model that x^* violates? We show that this can be solved in time $O(n^2 \log^3 n)$ for CUT, which is significantly faster than what was previously published: $O(n^4)$ by Oehrlein and Haunert (2017). We then show that the separation problem for LCUT is NP-hard. However, we show that both separation problems can be solved in time $O(n^2)$ when x^* is integer.

Fractional separation for CUT

Oehrlein and Haunert propose to solve the separation problem for inequalities (3.8b) as follows, when given a (fractional) point $\hat{x}$. For every pair (a, b) of nonadjacent vertices, solve a minimum-weight a, b-separator problem in graph G, where vertex i has weight $\hat{x}_{ib}$. If this weight is less than $\hat{x}_{ab}$, then an inequality (3.8b) is violated. The minimum-weight a, b-separator problem is solved in the usual way, by a node-splitting transformation to a minimum cut problem. This procedure runs in time $O(mn) = O(n^2)$ for a particular (a, b) pair, and in time $O(n^4)$ overall.

Proposition 8 (Oehrlein and Haunert (2017)). *The separation problem for constraints* (3.8b) *can be solved in time* $O(n^4)$.

We note that, by exploiting the planarity of D, the separation problem can be solved significantly faster. Namely, when given a fractional point $\hat{x}$, do the following for each $b \in V$. Start with graph D and let each vertex v have capacity $\hat{x}_{vb}$. Apply the linear-time reduction of Kaplan and Nussbaum (2011) to convert vertex capacities to edge capacities (while preserving planarity). Then, apply the algorithm of Lacki et al. (2012) to find the max

b, v-flow *value* (for all v) in time $O(n \log^3 n)$. This will tell us the value of a minimum-weight v, b-separator for all vertices v. Compare these flow values to $\hat{x}_{vb}$ to see if any inequalities (3.8b) are violated. Pick the most-violated one of them, say $v = a$, if any exist. Then apply the algorithm of Kaplan and Nussbaum (2011) to compute a max b, a-flow, and from it find the associated separator $C \subseteq V \setminus \{a, b\}$.

Lemma 7. *The separation problem for constraints* (3.8b) *for fixed* $b \in V$ *can be solved in time* $O(n \log^3 n)$.

Applying Lemma 7 for each vertex $b \in V$ gives the following proposition. We find this result quite striking given that there are n^2 variables.

Proposition 9. *The separation problem for constraints* (3.8b) *can be solved in time* $O(n^2 \log^3 n)$.

Fractional separation for LCUT

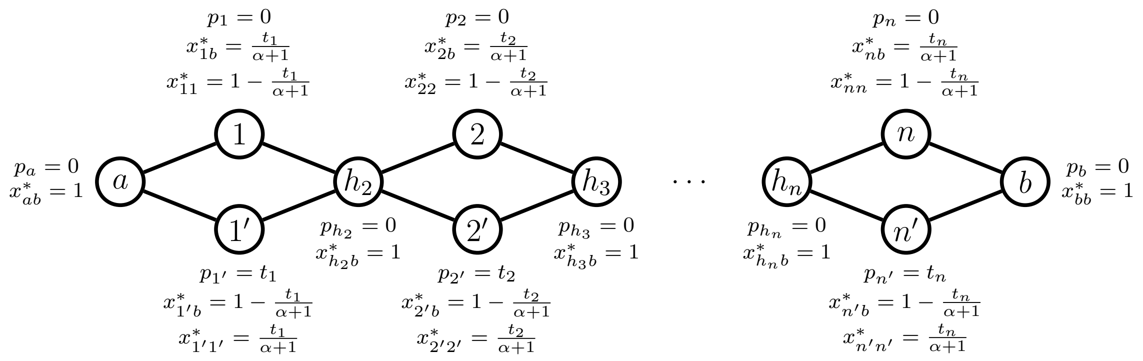
We show that the separation problem for LCUT is NP-hard. Hardness persists for outerplanar graphs, which are planar graphs in which all vertices touch the outer face. To prove this, we require the following lemma. It refers to the PARTITION problem, in which positive integers $t_1, t_2, \ldots, t_n$ are given as input and the task is to determine whether there exists $T \subseteq [n]$ with $\sum_{i \in T} t_i = \sum_{i \in [n] \setminus T} t_i$.

Lemma 8. PARTITION *remains NP-hard when two of the integers in the input equal* $(3/8) \sum_{i=1}^{n} t_i$.

Proof. The reduction is from an arbitrary instance of PARTITION with positive integers $s_1, s_2, \ldots, s_q$. Let $\sigma = \frac{1}{2} \sum_{i=1}^{q} s_i$ be the associated target value, and let $n := q + 2$. We construct an equivalent instance of partition $t_1, t_2, \ldots, t_n$, where $t_i := s_i$ for every $i \in [q]$, and the last two integers t_{q+1} and t_{q+2} are each set to 3σ. The sum $\sum_{i=1}^{n} t_i$ equals 8σ, meaning that the new target value is 4σ. This implies that t_{q+1} and t_{q+2} cannot be on the same side of the partition, in which case the new instance of PARTITION is equivalent to the original one. Since the reduction runs in polynomial time and since $t_{q+1} = t_{q+2} = 3\sigma = (3/8)\sum_{i=1}^{n} t_i$, this proves the lemma. $\square$

Theorem 8. *The separation problem for LCUT is NP-hard, even when the graph is outerplanar and the given point x^* is known to belong to $\mathcal{P}_{\text{CUT}}$.*

Proof. We show that it is NP-complete to determine whether a given point x^* lies outside of $\mathcal{P}_{\text{LCUT}}$. This problem belongs to NP because a violated inequality from $\mathcal{P}_{\text{LCUT}}$ is a suitable witness. The hardness reduction is from a PARTITION instance from the special class described in Lemma 8. Without loss, suppose that $t_1 = t_2 = \frac{3}{4}\alpha$, where $\alpha := \frac{1}{2}\sum_{i=1}^{n} t_i$ is the target value, and that $\alpha \geq 17$ (otherwise it is solvable in polynomial time by dynamic programming). Figure 3.6 details the construction of the graph $G = (V, E)$, the population vector p, and the nonzeros of the point x^*. Also, set $L := 0$, $U := \alpha - 1$, and $k := n + 1$. The graph G has $3n + 1$ vertices and is outerplanar.


$p_1 = 0$
$x^*_{1b} = \frac{t_1}{\alpha+1}$
$x^*_{11} = 1 - \frac{t_1}{\alpha+1}$
$p_2 = 0$
$x^*_{2b} = \frac{t_2}{\alpha+1}$
$x^*_{22} = 1 - \frac{t_2}{\alpha+1}$
$p_n = 0$
$x^*_{nb} = \frac{t_n}{\alpha+1}$
$x^*_{nn} = 1 - \frac{t_n}{\alpha+1}$
$p_a = 0$
$x^*_{ab} = 1$
a
1
$1'$
h_2
2
$2'$
h_3
$\cdots$
h_n
n
n'
b
$p_b = 0$
$x^*_{bb} = 1$
$p_{h_2} = 0$
$x^*_{h_2 b} = 1$
$p_{h_3} = 0$
$x^*_{h_3 b} = 1$
$p_{h_n} = 0$
$x^*_{h_n b} = 1$
$p_{1'} = t_1$
$x^*_{1'b} = 1 - \frac{t_1}{\alpha+1}$
$x^*_{1'1'} = \frac{t_1}{\alpha+1}$
$p_{2'} = t_2$
$x^*_{2'b} = 1 - \frac{t_2}{\alpha+1}$
$x^*_{2'2'} = \frac{t_2}{\alpha+1}$
$p_{n'} = t_n$
$x^*_{n'b} = 1 - \frac{t_n}{\alpha+1}$
$x^*_{n'n'} = \frac{t_n}{\alpha+1}$
Figure 3.6: An illustration of the graph $G = (V, E)$, the population vector p, and the nonzeros of x^*.

First, we show that x^* belongs to $\mathcal{P}_{\text{CUT}}$. For every vertex $i \in V$, $\sum_{j \in V} x^*_{ij} = 1$, so the assignment constraints (3.1b) are satisfied. Furthermore, $\sum_{j \in V} x^*_{jj} = n + 1 = k$, so constraint (3.1c) is satisfied. The population constraints (3.1d) are clearly satisfied for a, the h_i vertices, and the top-most vertices, so consider a vertex of the type v' on the bottom, and see that

$$Lx^*_{v'v'} = 0\left(\frac{t_v}{\alpha+1}\right) \leq \sum_{i\in V} p_i x^*_{iv'} = t_v\left(\frac{t_v}{\alpha+1}\right) \leq (\alpha-1)\left(\frac{t_v}{\alpha+1}\right) = Ux^*_{v'v'},$$

where the last inequality holds by $t_v \leq \frac{3}{4}\alpha$ and $\alpha \geq 4$. For vertex b, we have

$$\begin{aligned} Lx^*_{bb} = 0(1) \leq \sum_{i\in V} p_i x^*_{ib} &= \sum_{i=1}^{n} p_{i'} x^*_{i'b} \\ &= \sum_{i=1}^{n} t_i\left(1 - \frac{t_i}{\alpha+1}\right) \\ &= \sum_{i=1}^{n} t_i - \left(\frac{1}{\alpha+1}\right)\sum_{i=1}^{n} t_i^2 \\ &< \sum_{i=1}^{n} t_i - \left(\frac{1}{\alpha+1}\right)\left(t_1^2+t_2^2\right) \\ &= 2\alpha - \left(\frac{1}{\alpha+1}\right)\left(\frac{9}{8}\alpha^2\right) \\ &\leq (\alpha-1)(1) = Ux^*_{bb}. \end{aligned}$$

The last inequality holds because $\alpha \geq 17$. It is clear that x^* satisfies the coupling constraints (3.1e). The i,j-separator inequalities (3.8b) obviously hold for most variables x^*_{ij}, because most of them are zero or have $i = j$. The only nontrivial case is when $j = b$. For this case, observe that a unit flow can be sent from a to b that respects the "capacities" x^*_{vb}, by sending a flow of x^*_{vb} across arc (h_v, v) and a flow of $x^*_{v'b}$ across arc (h_v, v'). A similar flow proves the claim for $i \neq a$. So, x^* satisfies all inequalities (3.8b), and thus satisfies all constraints defining $\mathcal{P}_{\text{CUT}}$.

Now, to show NP-hardness, we must demonstrate that $(t_1, t_2, \dots, t_n)$ is a "yes" instance of PARTITION if and only if there is an inequality defining $\mathcal{P}_{\text{LCUT}}$ that x^* violates.

($\implies$) Suppose that $(t_1, t_2, \dots, t_n)$ is a "yes" instance of PARTITION, meaning that there is a subset $C \subset [n]$ such that $\sum_{i\in C} t_i = \sum_{i\in[n]\setminus C} t_i$. We argue that C is a length-U

a, b-separator in G and that x^* violates the associated inequality. First, see that if a vertex $i \in C$ is removed from G, then any a, b-path must cross its copy i' which will contribute $p_{i'}$ towards the length of the path. Thus, if C is removed, then every a, b-path will have length at least $\sum_{i \in C} p_{i'} = \sum_{i \in C} t_i = \alpha = U + 1$. That is, C is a length-U a, b-separator. Then, x^* lies outside of $\mathcal{P}_{\text{LCUT}}$ because

$$x^*_{ab} = 1 > \frac{\alpha}{\alpha + 1} = \frac{1}{\alpha + 1} \sum_{i \in C} t_i = \sum_{i \in C} \frac{t_i}{\alpha + 1} = \sum_{i \in C} x^*_{ib}.$$

($\Longleftarrow$) For the other direction, suppose that there is an inequality defining $\mathcal{P}_{\text{LCUT}}$ that x^* violates. As we have shown, x^* belongs to $\mathcal{P}_{\text{CUT}}$ (and thus $\mathcal{P}_{\text{HESS}}$), so any violated inequality must take the form $x_{ij} \leq \sum_{c \in C} x_{cj}$ where $i \neq j$. Observe that C must be a length-U i, j-separator and not just an i, j-separator (by $x^* \in \mathcal{P}_{\text{CUT}}$), so there exists an i, j-path in $G - C$, but its length is more than U. Without loss of generality, we suppose that C is a *minimal* length-U i, j-separator.

Claim 1. *$j = b$.*

Proof. The claim follows because if $j \neq b$, then $x^*_{ij} = 0 \leq \sum_{c \in C} x^*_{cj}$. ■

Claim 2. *C contains no vertices from the middle row $\{a, h_2, h_3, \dots, h_n, b\}$.*

Proof. If C contains a vertex v from the middle row, then $x^*_{ib} \leq 1 = x^*_{vb} \leq \sum_{c \in C} x^*_{cb}$, and x^* satisfies the length-U i, b-separator inequality for C. ■

Claim 3. *C contains no vertices from the bottom row $\{1', 2', \dots, n'\}$.*

Proof. For contradiction purposes, suppose that C contains a vertex v' from the bottom row. Let $C' := C \setminus \{v'\}$ and consider a shortest i, b-path P' in $G - C'$. This path P' cannot cross

v', for if it did then replacing v' by its upper counterpart v gives a *shorter* path P of length

$$\text{length}(P) = \text{length}(P') + p_v - p_{v'} = \text{length}(P') + 0 - t_v < \text{length}(P').$$

Thus, P' also belongs to $G - C$, implying that $\text{dist}_{G-C',p}(i,b) = \text{dist}_{G-C,p}(i,b)$. Recalling that $\text{dist}_{G-C,p}(i,b) > U$ as C is a length-U i,b-separator, we see $\text{dist}_{G-C',p}(i,b) = \text{dist}_{G-C,p}(i,b) > U$. Consequently, C' is also a length-U i,b-separator, contradicting the minimality of C. ■

Claim 4. *The inequalities* $\alpha \leq p_i + \sum_{c \in C} t_c$ *and* $(\alpha+1)\sum_{c \in C} x^*_{cb} \geq (\alpha - p_i)$ *hold.*

Proof. By Claims 2 and 3, $C \subseteq [n]$. By minimality of C, i must be somewhere to the left of the C vertices in Figure 3.6. So, every i,b-path must cross i and the lower counterparts C' of C, and thus has length at least $p_i + p(C')$. Meanwhile, a path P of the same length can be constructed by moving from left to right, opting for $v \in [n]$ when it is available ($v \notin C$) and taking v' otherwise. Thus, P is a shortest i,b-path in $G - C$, and

$$\alpha = U + 1 \leq \text{dist}_{G-C,p}(i,b) = p_i + p(C') = p_i + \sum_{c \in C} t_c = p_i + (\alpha+1)\sum_{c \in C} x^*_{cb}.$$

■

Claim 5. *Vertex i cannot belong to the top row* $\{1, 2, \ldots, n\}$.

Proof. If $i \in \{1, 2, \ldots, n\}$, we arrive at the following contradiction.

$$(3/4)\alpha \geq t_i = (\alpha+1)x^*_{ib} > (\alpha+1)\sum_{c \in C} x^*_{cb} \geq \alpha > (3/4)\alpha.$$

The first inequality holds by special class of PARTITION instances that we reduce from. The second inequality holds by the assumption. The third inequality holds by Claim 4 and $p_i = 0$. ■

Claim 6. $(t_1, t_2, \ldots, t_n)$ *is a "yes" instance of* PARTITION.

Proof. By Claim 5, either $i \in \{a, h_2, ..., h_n\}$ or $i \in \{1', 2', \ldots, n'\}$. In the former case,

$$1 = x^*_{ib} > \sum_{c \in C} x^*_{cb} = \sum_{c \in C} \frac{t_c}{\alpha + 1}, \tag{3.14}$$

implying that $\alpha + 1 > \sum_{c \in C} t_c$ and so

$$\alpha + 1 > \sum_{c \in C} t_c = \sum_{c \in C} t_c + p_i \geq \alpha, \tag{3.15}$$

where the last inequality holds by Claim 4. Because all terms of (3.15) are integers, it follows that $\sum_{c \in C} t_c = \alpha$. Thus, C is a solution to the partition instance. In the latter case, i belongs to the bottom row $\{1', 2', \ldots, n'\}$. Let $j \in \{1, 2, \ldots, n\}$ be its upper counterpart. Then,

$$1 - \frac{t_j}{\alpha + 1} = x^*_{ib} > \sum_{c \in C} x^*_{cb} = \sum_{c \in C} \frac{t_c}{\alpha + 1}. \tag{3.16}$$

implying that $\alpha + 1 - t_j > \sum_{c \in C} t_c$ and so

$$\alpha + 1 > \sum_{c \in C} t_c + t_j = \sum_{c \in C} t_c + p_i \geq \alpha, \tag{3.17}$$

where the last inequality holds by Claim 4. Because all terms of (3.17) are integers, it follows that $\sum_{c \in C} t_c + t_j = \alpha$. Thus, $C \cup \{j\}$ is a solution to the partition instance. ■

□

Integer separation for CUT and LCUT

Here we give a separation procedure for CUT and LCUT that applies when x^* is integer. The motivation is threefold. First, the fractional separation algorithm for CUT (Proposition 9) relies on complex algorithms for planar graphs that, to our knowledge, have never been implemented. Meanwhile, the straightforward procedure for CUT taken by Oehrlein and

Haunert (2017) requires the solution of roughly n^2 minimum cut problems; this price is often too expensive to justify (Fischetti et al., 2017b; Validi and Buchanan, 2019b; Salemi and Buchanan, 2020). Finally, the separation problem for LCUT is NP-hard. This motivates the following procedure which builds upon one proposed by Fischetti et al. (2017b).

Algorithm 1 IntegerSeparation(G, p, U, x^*)

```
1: for $b \in V$ do
2:     if $x^*_{bb} = 1$ then
3:         let $V_b := \{i \in V \mid x^*_{ib} = 1\}$
4:         for every component $G'$ of $G[V_b]$ that does not contain $b$ do
5:             let $a$ be an arbitrary vertex of $G'$ (e.g., one with the largest population)
6:             let $C$ be the minimal $a, b$-separator obtained by Fischetti et al. (2017b)
7:             if model = LCUT then
8:                 for $c \in C$ do
9:                     if $\text{dist}_{G-(C\setminus\{c\}),p}(a, b) > U$ then
10:                        $C \leftarrow C \setminus \{c\}$
11:            add cut $x_{ab} \leq \sum_{c\in C} x_{cb}$ to the model
```

When G is planar, Algorithm 1 runs in time $O(n^2)$ and returns a collection of violated (minimal) separator inequalities. This time is quite modest given that the solution x^* has n^2 entries. When the solution is already contiguous, the separation procedure runs in time $O(kn)$. Another nice property is that the cuts added on line 11 will have a combined $O(n)$ nonzeros.

Key to our arguments is the observation that each vertex i will belong to at most $\deg_G(i)$ many of the sets C from line 6. This follows because each minimal a, b-separator obtained from the algorithm of Fischetti et al. (2017b) is a subset of the shore of $V(G')$, and i can belong to at most $\deg_G(i)$ such shores (once for each of its neighbors). This implies that the total number of nonzeros coming from the round of cuts on line 11 is at most $\sum_{i\in V} \deg_G(i) + n = 2m + n = O(n)$.

To argue for a runtime of $O(n^2)$ when Algorithm 1 is applied to CUT, see that each vertex i can take at most one turn as the vertex a from line 5 and the fact that the algorithm of Fischetti et al. (2017b) takes time $O(m)$, and $m = O(n)$ since G is planar.

To argue for a runtime of $O(n^2)$ when Algorithm 1 is applied to LCUT, it now suffices to argue that the total time spent on lines 7 through 10 is at most $O(n^2)$. To see this, observe that the distance found in line 9 can be computed in time $O(n)$ since G is planar (Henzinger et al., 1997), also recalling that the number of distance computations will be at most $2m = O(n)$ because each vertex i will make at most $\deg_G(i)$ appearances in the sets C. Note that the inequalities added for LCUT will be as strong or stronger than the inequalities added for CUT. For ease of implementation, we instead use Dijkstra's shortest path algorithm, meaning that our code will take time $O(n^2 \log n)$ instead of time $O(n^2)$ when applied to LCUT.

3.4 Variable Fixing and a Heuristic

In initial experiments, the MIP solver had difficulties handling most tract-level instances, even when contiguity was not imposed. The large number of variables (n^2) and the numerical instability coming from the wide range of the objective coefficients ($w_{ij} := p_i d_{ij}^2$) lead to the following issues.

1. The root LP took a long time to solve with simplex; typically barrier was significantly faster.
2. If the root LP was solved with barrier, the crossover step took a very long time. The primal and dual push phases were not too costly, but the final simplex cleanup was slow—often taking ten times longer than barrier.
3. Even when an optimal basis for the root LP could be found, the MIP solver took a long time to find a good MIP-feasible solution.

To mitigate these issues, we sought to reduce the model's size by safely fixing some of the variables. By *safe*, we mean that doing so preserves an optimal solution. Intuitively, opportunities for variable fixing should be quite common. For example, if a vertex j is near the border of the state, it is likely not an optimal district center; instead, we expect the district centers to be in the state's interior. If we can rigorously argue that this is the case, we can fix the n variables x_{ij}, $i \in V$, to zero. Similarly, if vertices i and j are far from each other (say, at opposite corners of a state), then we expect that we can fix $x_{ij} = 0$. It is also possible to safely fix some variables to one, although this occurred so infrequently that we chose not to pursue it. This is perhaps not surprising; very few variables could safely be fixed to one in the experiments of Beasley (1993) for the k-median problem when k was small.

To safely fix variables to zero, we use Lagrangian arguments. We run a heuristic to find an upper bound UB on the optimal objective value. Then, we construct a Lagrangian relaxation model that provides a lower bound LB_{ij} on the objective value when a variable x_{ij} is tentatively fixed to one. The bounds LB_{ij} can be computed almost for free while solving the Lagrangian. Now, if $LB_{ij} > UB$, then x_{ij} cannot equal one in an optimal solution, meaning that we can safely fix $x_{ij} = 0$. The Lagrangian is solved using an implementation of Shor's r-algorithm that was developed by one of the authors (Lykhovyd, 2019), see also Shor (1985) and Kappel and Kuntsevich (2000). We found that the Lagrangian terminated more quickly and with a larger objective if the Lagrangian multipliers were initialized using optimal dual variables from the LP relaxation, which we obtained with barrier (no crossover). These initial multipliers can be found in the `ralg_warm` directory.

Figure 3.7 illustrates the power of this variable fixing procedure for Oklahoma. The heuristic runs for 4 seconds and the Lagrangian runs for 3 seconds (when given optimal LP dual variables). The total number of x_{ij} variables reduces from 1,094,116 to 12,425, a savings of 99%.

3.4.1 Heuristic

After some testing, we settled on the following heuristic. It draws upon the districting experience of Hess et al. (1965) and the k-median experience of Resende and Werneck (2004).

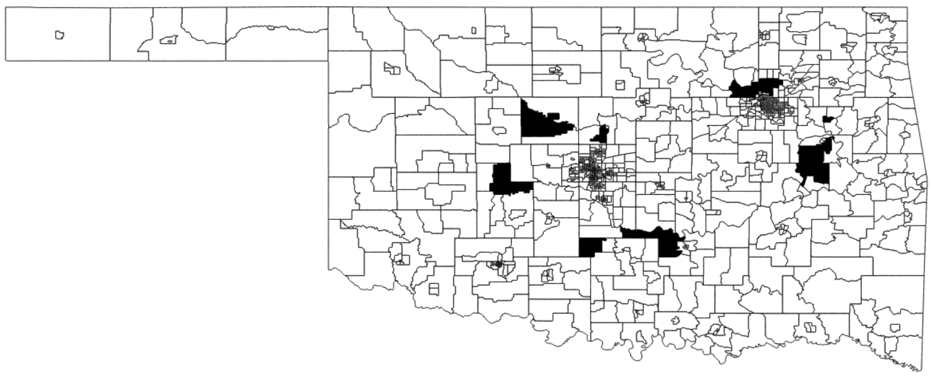


Figure 3.7: When contiguity is imposed, 96% of the center variables x_{jj} are safely fixed to zero for Oklahoma at the tract level; the non-fixed tracts are filled in black.

Pseudocode for `RandomizedHeuristic`() and `LocalSearch`() are given in Algorithms 2 and 3, respectively.

1. call `RandomizedHeuristic` for 10 iterations to find an initial set S of centers;
2. call `LocalSearch`(S) to improve the set of centers, using a first-improvement strategy;
3. if contiguity is required, solve a particular instance of the SHIR model (detailed below).

The descent steps in the inner loop of `RandomizedHeuristic` originate with Hess et al. (1965), see their Figure 1. They propose to find a heuristic solution by solving a restricted problem. That is, a set S of k centers is fixed (i.e., fix $x_{jj} = 1$ for $j \in S$) and the task is to assign the other vertices to them. At the time, Hess et al. heuristically solved this restricted problem by transportation techniques, but we use a MIP solver and denote the objective of this restricted problem by obj(S). The solution to this restricted problem partitions the vertices into k districts. The best center of each district is calculated, and the restricted problem is resolved taking this new set of centers as S. Repeat until convergence. This heuristic works surprisingly well even when the initial set of centers is chosen uniformly at random (as observed by Resende and Werneck). Following Resende and Werneck, we run this for 10 iterations and return the best set of centers that was found.

Algorithm 2 `RandomizedHeuristic`($maxit$)

```
1:  $S^* \leftarrow \emptyset$
2:  for $i = 1, 2, \ldots, maxit$ do
3:      pick a set $S$ of $k$ centers uniformly at random from $V$
4:      repeat
5:          find a solution $x^*$ to the Hess model (3.1) restricted to centers $\{v_1, v_2, \ldots, v_k\} \leftarrow S$
6:          for $j = 1, 2, \ldots, k$ do
7:              let $V_j$ be the vertices assigned to $v_j$ in $x^*$
8:              let $v_j^*$ be the best center of $V_j$, i.e., a vertex $b \in V_j$ that minimizes $\sum_{i \in V_j} w_{ib}$
9:          $S \leftarrow \{v_1^*, v_2^*, \ldots, v_k^*\}$
10:     until convergence
11:     if obj($S$) < obj($S^*$) then
12:         $S^* \leftarrow S$
13: return $S^*$
```

However, the subsequent local search procedure used by Resende and Werneck would be too burdensome for us. In the traditional k-median problem, the problem of assigning customers to facilities is trivial once the facilities have been opened: assign each customer to its nearest open facility. This allows one to quickly evaluate whether it is beneficial to swap a center with a non-center. The same cannot be said for districting due to the population bounds and contiguity constraints. This makes the problem of assigning vertices to centers, i.e., the function evaluation obj(S), an NP-hard problem. Consequently, we find just one local minimum with `LocalSearch`, starting the search from the best set of centers found by `RandomizedHeuristic`. To further speed up local search, we only consider swapping a center $s \in S$ with one of its neighbors $s' \in V \setminus S$. This reduces the number of function evaluations from $|S|(n - |S|)$ to roughly $\sum_{i \in S} \deg_G(i)$ in each call of `LocalSearch`. For California at the tract level, this reduction is from 425,000 to roughly 300.

Lastly, if required, we solve a final MIP to find a contiguous solution. Typically, the solution identified in local search is nearly contiguous and few changes are needed to achieve contiguity. To exploit this observation, we take the current set of district centers $S \subseteq V$ and

Algorithm 3 `LocalSearch`(S)

```
1: for $s \in S$ do
2:     for $s' \in N_G(s) \cap (V \setminus S)$ do
3:         $S' \leftarrow (S \setminus \{s\}) \cup \{s'\}$
4:         if $\text{obj}(S') < \text{obj}(S)$ then
5:             return LocalSearch$(S')$
6: return $S$
```

fix the associated variables x_{jj} to one in the SHIR model. For most states, this is sufficient for the MIP solver to identify a good contiguous solution within a few seconds. However, we observed that the MIP solver still struggled to solve this restricted problem on some of the largest instances. In response, we take the following approach. For each center $j \in S$, do the following:

1. identify the vertices V_j that are assigned to j in the local search solution;
2. identify the component of $G[V_j]$ that contains j and let J be its vertex set;
3. for each vertex $i \in J$, provide $x_{ij} = 1$ as part of a warm start solution; and
4. if working with a tract-level instance: for each vertex i in the interior[4] of J, fix $x_{ij} = 1$.

In practice, we found that these last two steps guide the MIP solver to a contiguous solution without sacrificing much in terms of solution quality. This tweak allowed us to find (contiguous) feasible solutions for states like Illinois, New York, and Texas at the tract level.

3.4.2 Lagrangian-based variable fixing for Hess model

Lagrangian techniques are quite common and useful for variants of the k-median problem (Beasley, 1993) and have also been used for districting purposes (Hojati, 1996). Here, we propose to use Lagrangian reduced costs to safely fix many of the variables x_{ij} to zero *before* building the Hess model, thus reducing it to a more manageable size. A similar approach was taken by Briant and Naddef (2004) for the diversity management problem.

[4]We define the *interior* of J to be the vertices in J whose neighbors in G also belong to J.

The Lagrangian relaxation model is obtained from the Hess model as follows. We relax the assignment constraints (3.1b), population lower bounds (3.1d), and population upper bounds (3.1d) and penalize their violation in the objective function with (vector) multipliers α, λ, and υ, respectively. Following Beasley (1993) and Hojati (1996), we scale the population constraints by dividing them by L and U, respectively. The optimal objective of the following Langragian is denoted by $\mathcal{L}(\alpha, \lambda, \upsilon)$.

$$\min \sum_{i \in V} \sum_{j \in V} w_{ij} x_{ij} + \sum_{i \in V} \alpha_i \left(1 - \sum_{j \in V} x_{ij}\right) + \sum_{j \in V} |\lambda_j| \left(x_{jj} - \sum_{i \in V} \frac{p_i}{L} x_{ij}\right)$$

$$+ \sum_{j \in V} |\upsilon_j| \left(\sum_{i \in V} \frac{p_i}{U} x_{ij} - x_{jj}\right) \tag{3.18a}$$

$$\sum_{j \in V} x_{jj} = k \tag{3.18b}$$

$$x_{ij} \leq x_{jj} \qquad \forall i, j \in V \tag{3.18c}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{3.18d}$$

There are two main differences with previous works (besides the definition of w). First, we have population lower and upper bounds (L and U), while Beasley (1993) had no analogue of L and Hojati (1996) had an equality constraint (i.e., $L = U$). Second, we take the absolute values of λ_j and υ_j in the Lagrangian's objective function, while Beasley (1993) required υ_j to be nonnegative; absolute values do not cause problems for Shor's r-algorithm (Shor, 1985).

To simplify the Lagrangian's objective function, we can collect like terms. For this, define

$$\hat{w}_{ij} = \begin{cases} w_{ij} - \alpha_i - |\lambda_j| \left(\dfrac{p_i}{L} \right) + |\upsilon_j| \left(\dfrac{p_i}{U} \right) & \text{if } i \neq j \\ \\ w_{ij} - \alpha_i - |\lambda_j| \left(\dfrac{p_i}{L} \right) + |\upsilon_j| \left(\dfrac{p_i}{U} \right) + |\lambda_j| - |\upsilon_j| & \text{if } i = j \end{cases}$$

for every $i \in V$ and $j \in V$. The Lagrangian's objective function now reduces to

$$\min \sum_{i \in V} \alpha_i + \sum_{i \in V} \sum_{j \in V} \hat{w}_{ij} x_{ij}.$$

To find an optimal solution x^* to the Lagrangian, consider the case where vertex $j \in V$ is selected as a center. In this case, it would be optimal to assign vertex i to center j if and only if $\hat{w}_{ij} \leq 0$. Thus, if $j \in V$ were selected as a center, its contribution to the objective would be

$$W_j := \hat{w}_{jj} + \sum_{i \in V \setminus \{j\}} \min \{0, \hat{w}_{ij}\},$$

otherwise the contribution would be zero.

With these observations, the Lagrangian reduces to the following n-variable problem.

$$\min \sum_{i \in V} \alpha_i + \sum_{j \in V} W_j x_{jj} \tag{3.19a}$$

$$\sum_{j \in V} x_{jj} = k \tag{3.19b}$$

$$x_{jj} \in \{0, 1\} \qquad \forall j \in V. \tag{3.19c}$$

This can be solved by identifying the k different vertices j with the smallest W_j values and setting their variables x_{jj} to one; set all others to zero. The objective value of this solution is equal to $\mathcal{L}(\alpha, \lambda, \upsilon)$, which provides a lower bound on our original MIP (whether or not contiguity is imposed).

Now, we discuss how to safely fix variables to zero in the Hess model (3.1) using the heuristic upper bound UB and the Lagrangian. Let x^* be an optimal solution to the reduced Lagrangian problem (3.19) with objective value $\mathcal{L}(\alpha, \lambda, \upsilon)$. The associated set of centers is $S := \{v \in V \mid x^*_{vv} = 1\}$. If we were to tentatively fix $x_{ij} = 1$, the resulting Lagrangian bound $\mathcal{L}_{ij}(\alpha, \lambda, \upsilon)$ would be

$$\mathcal{L}_{ij}(\alpha, \lambda, \upsilon) = \begin{cases} \mathcal{L}(\alpha, \lambda, \upsilon) & \text{if } j \in S,\ i = j \\ \mathcal{L}(\alpha, \lambda, \upsilon) + \max\{0, \hat{w}_{ij}\} & \text{if } j \in S,\ i \neq j \\ \mathcal{L}(\alpha, \lambda, \upsilon) - \max\limits_{v \in S}\{W_v\} + W_j & \text{if } j \in V \setminus S,\ i = j \\ \mathcal{L}(\alpha, \lambda, \upsilon) - \max\limits_{v \in S}\{W_v\} + W_j + \max\{0, \hat{w}_{ij}\} & \text{if } j \in V \setminus S,\ i \neq j. \end{cases}$$

This value can be used to update the lower bound $LB_{ij} \leftarrow \max\{\mathcal{L}_{ij}(\alpha, \lambda, \upsilon), LB_{ij}\}$ on the objective value that would result from fixing $x_{ij} = 1$. Observe that $\mathcal{L}(\alpha, \lambda, \upsilon)$ and $\max\{W_v \mid v \in S\}$ do not depend on i nor j, meaning that they can be precomputed, stored, and then used to update all of the LB_{ij} values in time $\Theta(n^2)$.

The outer problem for the Lagrangian (for finding the best Lagrange multipliers) is controlled by Shor's r-algorithm. When it terminates with the final values LB_{ij}, we fix $x_{ij} = 0$ if $LB_{ij} > UB$.

3.4.3 Lagrangian-based variable fixing for contiguity models

We can fix even more variables x_{ij} to zero by exploiting contiguity. Ideally, we would add constraints to (3.18) requiring that the vertices $V_j(x)$ assigned to j induce a connected subgraph. In this case, the Lagrangian relaxation model could be solved by redefining W_j as the weight of a minimum-weight connected subgraph (MWCS) rooted at j (where the weight of vertex i is $\hat{w}_{ij}$) and again solving the problem (3.19). One issue is that the rooted MWCS problem is NP-hard. While this problem has been studied frequently lately (Álvarez-Miranda et al., 2013a,b; Álvarez-Miranda and Sinnl, 2017; Rehfeldt et al., 2019), and several research codes perform quite well on benchmark instances (Fischetti et al., 2017b; Gamrath et al., 2017; Rehfeldt and Koch, 2019), we choose not to use them given how frequently we would need to solve the rooted MWCS problem[5].

Instead, we consider a relaxed form of contiguity for which all LB_{ij} values can be updated in $O(n^2)$ time. This relaxed form of contiguity only enforces that an i,j-path exists within $G[V_j(x)]$ if x_{ij} is tentatively fixed to one. For this, define the weights q_u^j as below, representing the "extra" cost to use vertex u in an i,j-path in district $V_j(x)$. Observe that the "cost" $\hat{w}_{uj}$ of vertex u has already been accounted for in W_j if $u = j$ or $\hat{w}_{uj} \leq 0$, giving an extra cost of zero.

$$q_u^j = \begin{cases} 0 & \text{if } u = j \text{ or } \hat{w}_{uj} \leq 0 \\ \\ \hat{w}_{uj} & \text{otherwise.} \end{cases}$$

Again, let $\mathcal{L}(\alpha, \lambda, \upsilon)$ be the objective value of the Lagrangian relaxation model (3.19) and let S be the associated set of centers. Then, a lower bound on the *contiguity-constrained*

[5]Also, preliminary experiments with solving the rooted MWCS problem exactly showed little to no improvement—in terms of the number of variables fixed—over the simple and very quick procedure that we finally settled on.

Lagrangian relaxation model—when x_{ij} is tentatively fixed to one—is as follows.

$$\widetilde{\mathcal{L}}_{ij}(\alpha, \lambda, \upsilon) = \begin{cases} \mathcal{L}(\alpha, \lambda, \upsilon) + \text{dist}_{G,q^j}(i, j) & \text{if } j \in S \\ \\ \mathcal{L}(\alpha, \lambda, \upsilon) - \max\limits_{v \in S}\{W_v\} + W_j + \text{dist}_{G,q^j}(i, j) & \text{if } j \notin S. \end{cases}$$

Here, $\text{dist}_{G,q^j}(i, j)$ is the vertex-weighted distance from i to j, where vertex u has weight q^j_u. These distances $\text{dist}_{G,q^j}(\cdot, j)$ can be computed in time $O(n)$ with a planarity-exploiting single-source shortest path algorithm (Henzinger et al., 1997). Thus, all updates $LB_{ij} \leftarrow \max\{LB_{ij}, \widetilde{\mathcal{L}}_{ij}(\alpha, \lambda, \upsilon)\}$ can be computed in time $O(n^2)$. However, for ease of implementation, we use Dijkstra's algorithm, so our code takes time $O(n^2 \log n)$ to update the LB_{ij} values. As before, we fix $x_{ij} = 0$ if $LB_{ij} > UB$. Theorem 9 ensures that this is safe.

Theorem 9. *For every set of Lagrange multipliers $(\alpha, \lambda, \upsilon)$ we have $z^*_{ij} \geq \widetilde{\mathcal{L}}_{ij}(\alpha, \lambda, \upsilon)$, where z^*_{ij} is the optimal objective of the Hess model when x_{ij} is fixed to one and contiguity is imposed.*

Proof. Consider a set of Lagrange multipliers $(\alpha, \lambda, \upsilon)$, the corresponding $\hat{w}$ and W, and an optimal set of centers S for the Lagrangian (3.19). Then, let x^* be an optimal solution to the Hess model (3.1) when x_{ij} is fixed to one and contiguity is imposed, and let $S^* := \{v \in V \mid x^*_{vv} = 1\}$ be the associated set of centers. Observe that subset of vertices assigned to j induces a connected subgraph, so there is a path P from i to j whose vertices $u \in V(P)$ satisfy $x^*_{uj} = 1$.

Claim 7. $z^*_{ij} \geq \sum_{u \in V} \alpha_u + \sum_{v \in S^*} W_v + \text{dist}_{G,q^j}(i, j)$.

Proof. Using the notations $(a)^+ := \max\{0, a\}$ and $(a)^- := \min\{0, a\}$, observe that

$$\begin{aligned}
z^*_{ij} &= \sum_{u\in V}\sum_{v\in V} w_{uv}x^*_{uv} && \text{(3.20a)}\\
&\geq \sum_{u\in V}\sum_{v\in V} w_{uv}x^*_{uv} + \sum_{u\in V}\alpha_u\left(1 - \sum_{v\in V}x^*_{uv}\right) + \sum_{v\in V}|\lambda_v|\left(x^*_{vv} - \sum_{u\in V}\frac{p_u}{L}x^*_{uv}\right)\\
&\qquad + \sum_{v\in V}|\upsilon_v|\left(\sum_{u\in V}\frac{p_u}{U}x^*_{uv} - x^*_{vv}\right) && \text{(3.20b)}\\
&= \sum_{u\in V}\alpha_u + \sum_{u\in V}\sum_{v\in V}\hat{w}_{uv}x^*_{uv} && \text{(3.20c)}\\
&\geq \sum_{u\in V}\alpha_u + \sum_{v\in S^*}W_v + \operatorname{dist}_{G,q^j}(i,j). && \text{(3.20d)}
\end{aligned}$$

Here, inequality (3.20b) holds by the feasibility of x^* for the Hess model. Equality (3.20c) holds by the definition of $\hat{w}$. Finally, inequality (3.20d) holds by inequalities (3.21), which are shown below.

$$
\begin{aligned}
\sum_{u\in V}\sum_{v\in V}\hat{w}_{uv}x^*_{uv} &= \sum_{v\in V}\hat{w}_{vv}x^*_{vv} + \sum_{v\in V}\sum_{u\in V\setminus\{v\}}(\hat{w}_{uv})^- x^*_{uv} + \sum_{v\in V}\sum_{u\in V\setminus\{v\}}(\hat{w}_{uv})^+ x^*_{uv} && \text{(3.21a)}\\
&\geq \sum_{v\in V}\hat{w}_{vv}x^*_{vv} + \sum_{v\in V}\sum_{u\in V\setminus\{v\}}(\hat{w}_{uv})^- x^*_{uv} + \sum_{u\in V(P)\setminus\{j\}}(\hat{w}_{uj})^+ x^*_{uj} && \text{(3.21b)}\\
\text{(by } x^*_{uv}\leq x^*_{vv}\text{)} \quad &\geq \sum_{v\in V}\hat{w}_{vv}x^*_{vv} + \sum_{v\in V}\sum_{u\in V\setminus\{v\}}(\hat{w}_{uv})^- x^*_{vv} + \sum_{u\in V(P)\setminus\{j\}}(\hat{w}_{uj})^+ x^*_{uj} && \text{(3.21c)}\\
&= \sum_{v\in V}\left(\hat{w}_{vv} + \sum_{u\in V\setminus\{v\}}(\hat{w}_{uv})^-\right)x^*_{vv} + \sum_{u\in V(P)\setminus\{j\}}(\hat{w}_{uj})^+ x^*_{uj} && \text{(3.21d)}\\
\text{(by } W_v \text{ def.)} \quad &= \sum_{v\in V}W_v x^*_{vv} + \sum_{u\in V(P)\setminus\{j\}}(\hat{w}_{uj})^+ x^*_{uj} && \text{(3.21e)}\\
\text{(by } q^j_u \text{ def.)} \quad &= \sum_{v\in S^*}W_v + \sum_{u\in V(P)} q^j_u && \text{(3.21f)}\\
\text{(by } \mathrm{dist}_{G,q^j}(i,j) \text{ def.)} \quad &\geq \sum_{v\in S^*}W_v + \mathrm{dist}_{G,q^j}(i,j). && \text{(3.21g)}
\end{aligned}
$$

■

With Claim 7 established, we turn to the theorem. In the first case, suppose that $j \in S$. Then,

$$
\begin{aligned}
z^*_{ij} &\geq \sum_{u\in V}\alpha_u + \sum_{v\in S^*}W_v + \mathrm{dist}_{G,q^j}(i,j)\\
&\geq \sum_{u\in V}\alpha_u + \sum_{v\in S}W_v + \mathrm{dist}_{G,q^j}(i,j)\\
&= \mathcal{L}(\alpha,\lambda,\upsilon) + \mathrm{dist}_{G,q^j}(i,j) = \widetilde{\mathcal{L}}_{ij}(\alpha,\lambda,\upsilon),
\end{aligned}
$$

where the inequalities hold by Claim 7 and by the optimality of S for the Lagrangian (3.19), and the equalities hold by the definitions of $\mathcal{L}(\alpha, \lambda, \upsilon)$ and $\widetilde{\mathcal{L}}_{ij}(\alpha, \lambda, \upsilon)$.

In the other case, where $j \notin S$, there exists a vertex s that belongs to S but not to S^*. (Otherwise, $S \subseteq S^*$ and $j \in S^* \setminus S$, implying the contradiction $k = |S| < |S^*| = k$). Then,

$$\begin{aligned}
\sum_{v \in S^*} W_v &= \sum_{v \in (S^* \cup \{s\}) \setminus \{j\}} W_v - W_s + W_j \\
&\geq \sum_{v \in S} W_v - W_s + W_j \\
&\geq \sum_{v \in S} W_v - \max_{v \in S}\{W_v\} + W_j,
\end{aligned}$$

which, along with Claim 7, implies that

$$\begin{aligned}
z^*_{ij} &\geq \sum_{u \in V} \alpha_u + \sum_{v \in S^*} W_v + \mathrm{dist}_{G,q^j}(i,j) \\
&\geq \sum_{u \in V} \alpha_u + \sum_{v \in S} W_v - \max_{v \in S}\{W_v\} + W_j + \mathrm{dist}_{G,q^j}(i,j) \\
&= \mathcal{L}(\alpha, \lambda, \upsilon) - \max_{v \in S}\{W_v\} + W_j + \mathrm{dist}_{G,q^j}(i,j) = \widetilde{\mathcal{L}}_{ij}(\alpha, \lambda, \upsilon),
\end{aligned}$$

where the equalities hold by the definitions of $\mathcal{L}(\alpha, \lambda, \upsilon)$ and $\widetilde{\mathcal{L}}_{ij}(\alpha, \lambda, \upsilon)$. $\square$

3.5 Computational Experiments

In this section, we conduct an extensive computational study. Broadly speaking, the aim is to answer the questions: (i) Which model for imposing contiguity is the fastest in practice? (ii) Is this fastest model able to solve the large-scale instances encountered in practice?

Most experiments were performed on a machine with Intel Xeon E3-1270 v6 “Kaby Lake” 3.80 GHz CPU with 8 cores and 32 GB RAM. The MIP solver is Gurobi Optimizer 8.1.1. When solving the MIPs, default settings are used with the following exceptions: 8 threads maximum, 10 GB RAM maximum, concurrent method for the LP relaxation, and zero MIP gap tolerance. We invoke the `LazyConstraints` parameter when solving the CUT and LCUT models.

Recall that to initialize the Lagrangian procedure we use optimal dual multipliers from the LP relaxation of the Hess model. In this case, the barrier method is used to solve the LP. For four states with the most census tracts (FL, NY, TX, and CA) the size of this LP model exceeds available memory, in which case we solve the LP using a different machine that has dual Intel Xeon E5-2620 "Sandy Bridge" hex core 2.0 GHz CPUs and 256 GB RAM.

3.5.1 Data preparation

To compare the computational performance of the models, we needed test instances having the following input data: contiguity graph $G = (V, E)$, population vector p, bounds L and U, number of districts k, and the distance matrix d. This data is not directly provided by the Census (particularly the graph and distance matrix), and optimization researchers who have studied districting in the past did not make their data public. This is a significant barrier to entry for those who want to apply their methods to redistricting problems. Moreover, previous studies have focused on one or two states, making it impossible to know how well their methods would perform on another state or if the data were different (say, after a new Census).

For these reasons, we generate the requisite data ourselves for all 50 states using the 2010 Census numbers. So that future researchers can use the same data and also for purposes of transparency, we post the complete data set online at `https://lykhovyd.com/files/public/districting/`. The GitHub repository includes scripts used to convert the raw data from the USCB (2012) into formats convenient for our use.

We generate the input data at two different levels: county and census tract. There is an average of 62 counties per state and a maximum of 254 for Texas. Meanwhile, there is an average of 1,461 census tracts per state and a maximum of 8,057 in the case of California. The county-level instances are small enough that we can apply each of the contiguity formulations. This allows us to compare them and discern which of them has the best chance to handle large instances. Moreover, several states require congressional redistricting plans to split the fewest number of counties, and states such as Iowa and West Virginia split zero counties in their 2013 maps. This makes county-level instances practical in some cases. However, most states cannot redistrict at the county level. For example, Dallas County in Texas had a population of 2,368,139 after the 2010 Census, meaning that it needed to be split into (at least) 4 congressional districts in order to satisfy the rigid population bounds. Census tracts are designed to be relatively homogeneous and typically have between 2,500 and 8,000 people in them, which provides enough granularity to satisfy the population bounds. The tract-level instances are sufficiently large and challenging that roughly half of them can be solved by our techniques, allowing us to show their computational limits. Note, however, that no state has a law requiring the indivisibility of tracts, and many states perform districting at the block level, where n approaches one million. The exact techniques considered in this chapter cannot handle such instances, so we do not generate block-level data.

The data is constructed for each state separately, using a suitable map projection from the EPSG dataset (IOGP, 2019). The centroid of each county or tract is taken as its center, and Euclidean distances are measured between centroids. If two counties or tracts share a nontrivial border, we connect them by an edge in G. For some states, the graph G is disconnected (e.g., due to islands off a state's coast). In this case, we make it connected by adding a minimum-weight subset of (currently missing) edges via a straightforward extension of Kruskal's algorithm. We take the weight of an edge $\{i, j\}$ to be the distance between (the centroids) i and j. In this way, a *single* island tract will be made adjacent to its nearest tract from the coast. However, if there is a tightly packed *cluster* of islands sufficiently far from the coast, only one edge will be added connecting the islands to the coast; the other added edges will be internal to the island cluster.

Although rare, some tracts are themselves non-contiguous. For example, the tract in Massachusetts with GEOID 25023990003 consists of three disconnected pieces, each a body of water with zero population. In these cases, a district that is connected in the contiguity graph may not be contiguous on the map. In fact, this happened in our experiments. To address this issue, one could adjust the contiguity graph by creating a different node for each of the tract's pieces, or by merging the disconnected pieces into neighboring tracts (Duchin, 2020). However, as our intent in this chapter is to compare the performance of the contiguity formulations on realistic instances—and not to create "good" plans—we opt to keep the contiguity graphs as-is. In the future, this might not be a problem; the criteria specified for the 2020 Census state that "Census tracts must comprise a reasonably compact and contiguous land area" (Bureau of the Census, 2018).

The other parameters are set as follows, where the number k of congressional districts is known and set by reapportionment. We compute the ideal population $\bar{p} := \frac{1}{k}\sum_{i\in V} p_i$ of a district and allow a 1% deviation, setting $\hat{L} := 0.995(\bar{p})$ and $\hat{U} := 1.005(\bar{p})$, as was suggested in a redistricting competition held by reformers in Ohio (Altman and McDonald, 2018). We then round the population bounds (in the appropriate direction) to an integer, i.e., $L := \lceil \hat{L} \rceil$ and $U := \lfloor \hat{U} \rfloor$.

3.5.2 Experiments with heuristic and variable fixing

Table 3.1 reports our experience with the heuristic and variable fixing procedures from Section 3.4. For space considerations, only tract-level results are reported here in the paper. Tract-level results are also more interesting because this is where these procedures are most needed.

Reported under the Lagrangian columns are the lower bound obtained from the Lagrangian and the time in seconds spent by Shor's r-algorithm. As noted in Section 3.4, we initialize

Table 3.1: Heuristic and Lagrangian results. We report the number of tracts (n), the number of districts (k), the Lagrangian objective (LB) and time in seconds, as well as the heuristic objective (UB), time, and the percentage of the variables fixed to zero (fixed) when contiguity is (not) imposed—rounded to the nearest percent.

			Lagrangian		w/o contiguity			w/ contiguity		
State	n	k	LB	time	UB	time	fixed	UB	time	fixed
RI	244	2	228.33	0.09	228.42	0.21	100	228.42	0.23	100
NH	295	2	2,687.72	0.09	2,689.07	0.18	100	2,688.28	0.30	100
ID	298	2	53,913.93	0.22	53,930.05	0.15	100	53,916.36	0.20	100
HI	351	2	13,990.50	0.22	13,991.27	0.17	100	13,991.95	0.29	100
ME	358	2	7,713.30	0.30	7,716.04	0.16	100	7,716.04	0.22	100
WV	484	3	10,789.79	0.52	11,801.80	0.66	48	11,834.84	0.87	58
NM	499	3	31,575.28	0.53	31,598.10	1.72	95	31,608.19	1.91	95
NE	532	3	21,975.38	0.59	21,983.42	1.28	99	21,983.42	1.45	99
UT	588	4	22,029.31	0.81	22,035.62	2.03	97	22,037.00	2.50	97
MS	664	4	16,295.54	0.98	16,303.82	1.81	100	16,308.89	2.05	100
AR	686	4	15,490.50	1.02	15,565.41	3.85	96	15,569.97	4.09	97
NV	687	4	11,367.90	1.32	12,443.84	5.96	35	12,497.05	6.14	43
KS	770	4	22,654.34	1.43	22,786.63	3.08	82	22,786.69	3.87	85
IA	825	4	18,164.68	1.72	18,202.13	2.77	96	18,206.43	3.37	97
CT	833	5	1,420.41	1.68	1,422.13	4.50	99	1,422.13	4.74	99
OR	834	5	26,742.01	1.87	26,750.23	9.55	98	26,750.15	10.36	98
OK	1,046	5	19,106.02	3.34	19,132.60	3.60	99	19,131.99	4.35	99
SC	1,103	7	8,356.08	3.28	9,293.98	13.48	18	9,294.45	14.94	40
KY	1,115	6	14,793.25	2.90	14,833.72	10.25	95	14,835.16	11.27	96
LA	1,148	6	13,805.17	3.61	14,829.56	12.52	32	14,830.45	13.92	48
AL	1,181	7	13,058.24	3.86	13,510.26	17.54	49	13,510.26	18.08	68
CO	1,249	7	19,715.16	4.27	19,916.43	15.99	40	19,918.36	17.50	53
MN	1,338	8	24,207.22	4.86	24,220.52	184.95	89	24,224.19	217.11	87
MO	1,393	8	18,830.09	5.47	21,214.38	29.24	16	21,222.79	31.44	44
MD	1,406	8	5,079.56	5.56	5,084.47	26.79	94	5,084.53	29.75	95
WI	1,409	8	16,272.83	5.34	16,340.39	28.70	79	16,340.81	32.23	84
WA	1,458	10	12,376.66	5.70	12,604.21	47.42	53	12,609.09	71.92	61
MA	1,478	9	2,558.82	6.09	2,626.48	50.55	41	2,627.03	53.75	57
TN	1,497	9	10,783.27	6.05	13,092.08	48.49	8	13,092.08	51.55	44
IN	1,511	9	11,061.11	6.46	11,084.81	28.10	97	11,085.37	32.27	97
AZ	1,526	9	29,479.16	6.71	30,219.11	81.92	22	30,220.39	88.56	33
VA	1,907	11	12,836.67	11.22	13,814.08	379.21	20	13,815.25	391.63	49
GA	1,969	14	15,836.59	11.43	15,955.65	243.27	67	15,955.65	242.99	79
NJ	2,010	12	2,250.25	12.09	2,291.51	213.81	52	2,291.51	216.89	70
NC	2,195	13	14,416.66	14.29	14,679.00	331.61	52	14,680.02	350.92	77
MI	2,813	14	24,569.50	24.79	24,685.68	775.69	56	24,686.55	788.41	71
OH	2,952	16	11,520.54	26.59	11,908.25	1,936.37	25	11,911.64	2,056.53	65
IL	3,123	18	15,815.07	30.26	16,067.49	14,316.10	40	16,091.05	14,403.97	55
PA	3,218	18	11,424.45	32.59	11,799.41	6,674.80	38	11,806.00	6,721.78	66
FL	4,245	27	14,766.11	58.53	15,222.31	7,668.40	38	15,225.47	7,811.97	65
NY	4,919	27	15,405.98	77.60	16,700.05	13,848.68	7	16,700.81	15,120.01	45
TX	5,265	36	57,770.13	105.63	72,846.42	14,691.58	1	72,860.21	15,657.41	14
CA	8,057	53	26,238.24	218.99	27,603.66	22,122.05	28	-	-	-

Shor's r-algorithm with optimal dual multipliers from the LP relaxation of the Hess model. The time to solve this LP can be substantial—5 days for CA—so we precompute the LP dual multipliers and store them in the `ralg_warm` directory for reuse; the time to solve this LP is not included in the time given in Table 3.1. We see that if Shor's r-algorithm is limited to (at most) 100 iterations, it terminates in less than one minute for most instances. The last six columns report the upper bound obtained via the heuristic (without and with contiguity constraints), the time spent by the heuristic, and the percentage of the variables x_{ij} that are eventually fixed.

Generally speaking, the objective values of the heuristic solutions (contiguous vs. not) are similar. So, the price of contiguity appears small. (This is confirmed for *optimal* solutions later.) Recall that in our heuristic we solve a series of restricted MIPs. For speed considerations, we do not force the MIP solver to prove optimality for these restricted MIPs. This explains the perhaps counterintuitive observation that, when contiguity is enforced, the heuristic's objective value sometimes *improves* (e.g., for NH, ID, OR, and OK).

Another observation is that the variable fixing procedure is quite powerful, sometimes allowing us to fix approximately 100% of the variables. While this is most pronounced on the smaller instances (e.g., RI, NH, ID, HI, ME), some large instances are also amenable to fixing. Roughly 97% of the variables can be fixed for IN which has $n = 1,511$ tracts.

Finally, we observe that it can be quite helpful to exploit contiguity in the variable fixing procedure (Section 3.4.3). For example, consider the case of TN. Here, the heuristic's objective value does not change when contiguity is imposed, but the fixings increase from 8% to 44%. In another example, CO sees the fixings increase from 40% to 53% despite a degradation in objective value.

3.5.3 County-level results

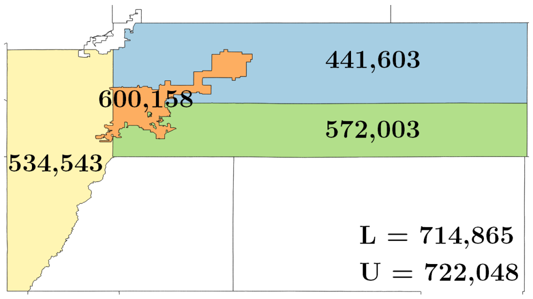
A majority of the county-level instances are infeasible. For example, consider Texas. Dallas County had a population of $p_v = 2,368,139$, which far exceeds the population limit $U = 701,980$ that we impose. Thus, Texas is obviously infeasible at the county level. This type of overt infeasibility where the population of a single vertex exceeds U is exhibited by 27 states. These, as well as the 7 trivial instances where $k = 1$, are uninteresting and are excluded from our county-level experiments.

Table 3.2: An initial classification of the 50 county-level instances

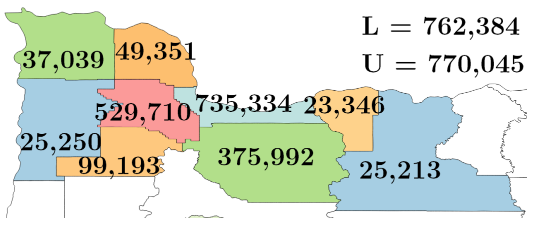
Class	#	States
Overt inf.	27	AZ, CA, CT, FL, GA, HI, IL, IN, KY, MA, MC, MI, MN, MO, NC, NJ, NV, NY, OH, PA, RI, TN, TX, UT, VA, WA, WI.
Trivial ($k = 1$)	7	AK, DE, MT, ND, SD, VT, WY.
Remaining	16	AL, AR, CO, IA, ID, KS, LA, ME, MS, NE, NH, NM, OK, OR, SC, WV.

This leaves 16 county-level instances. It turns out that each of them is MIP-feasible under the Hess model, although 4 of them become infeasible when contiguity is imposed (CO, NH, OR, SC), leaving 12 contiguity-feasible instances. The reasons why CO and OR are infeasible can be understood by inspecting the sub-maps depicted in Figure 3.8. For 4 instances, the optimal solution that is returned by the Hess model under the moment-of-inertia (MOI) objective happens to be contiguous (IA, ID, KS, MS). In the other 8 cases, the Hess solution that is found is not contiguous. Further, when contiguity is explicitly imposed, the optimal objective value increases, implying that for these states no optimal solution to the Hess model is contiguous (e.g., see Figure 3.9).

Table 3.3 reports the optimal objective values and solve times (in seconds) for the 16 remaining county-level instances. Note that the reported times include all operations (heuristic, Lagrangian fixing, model build, model solve) excluding read time and the time to get the initial Lagrange multipliers (see Section 3.4). More details can be found in the

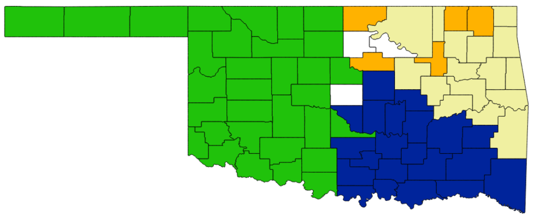


(a) Denver metro



(b) Portland metro

Figure 3.8: Side (a) shows that Colorado is infeasible at the county level because there is no feasible district containing Denver County (pop. 600,158). Side (b) shows that Oregon is infeasible at the county level because there is no feasible district containing Maltnomah County (pop. 735,334).


(a) Optimal solution w/o contiguity

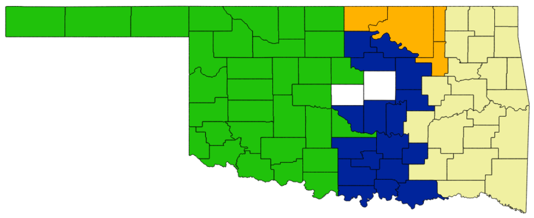

(b) Optimal solution w/ contiguity

Figure 3.9: Optimal county-level solutions for Oklahoma.

`results` directory in the GitHub repository. As the table shows, the Hess model is relatively easy for the MIP solver at the county level, with each being solved to optimality in under 10 seconds.

For the contiguity models, one observes that the MCF model is the slowest (attributable to its large size), while the others are reasonably competitive with each other. In some cases, however, LCUT edges out the competition. For example, LCUT edges out SHIR for Alabama (19 vs. 64 seconds). We attribute this time difference to LCUT being smaller and more nimble than SHIR. Meanwhile, LCUT edges out CUT for Colorado (2 seconds vs. TL). In fact, we find that LCUT proves infeasibility of Colorado at the root node of the branch-and-bound tree. To illustrate, recall Figure 3.8. Denver County cannot be paired with any adjacent counties without exceeding the population upper bound. Thus, the special case of the length-U a, b-separator inequalities from Remark 4 enforces that Denver County

Table 3.3: Times and objective values for solving the resulting MIPs at the county level. Infeasibilities are denoted by an objective value of ∞, while TL denotes that the one-hour time limit was reached.

			w/o contiguity		w/ contiguity				
State	n	k	obj	Hess	obj	MCF	SHIR	CUT	LCUT
NH	10	2	3,641.24	0.03	∞	0.23	0.12	0.12	0.08
ME	16	2	10,789.07	0.03	19,093.26	2.42	0.78	0.72	0.66
NM	33	3	32,847.05	0.07	32,944.25	0.63	0.11	0.10	0.10
OR	36	5	31,424.49	0.40	∞	55.80	0.59	3.08	0.32
ID	44	2	61,232.58	0.06	61,232.58	2.79	0.14	0.09	0.09
SC	46	7	12,479.94	8.67	∞	TL	TL	TL	TL
WV	55	3	11,844.76	0.82	12,000.62	62.16	1.68	0.86	0.84
LA	64	6	17,239.16	2.14	17,272.65	66.87	4.08	2.22	2.23
CO	64	7	35,466.64	2.46	∞	215.25	2.74	TL	1.94
AL	67	7	15,519.54	3.83	16,627.25	457.52	64.29	20.94	19.41
AR	75	4	16,525.07	0.53	16,543.15	29.04	1.62	0.96	0.96
OK	77	5	21,527.57	1.01	21,756.50	101.64	5.06	1.64	1.97
MS	82	4	16,142.04	0.41	16,142.04	8.31	0.54	0.49	0.49
NE	93	3	22,112.00	0.28	22,193.01	13.33	0.57	0.45	0.47
IA	99	4	17,748.05	0.85	17,748.05	29.23	1.47	0.87	0.87
KS	105	4	23,736.89	1.34	23,736.89	48.98	1.78	1.10	1.11

cannot be assigned to other counties, nor can other counties be assigned to it. So, Denver County must be in its own district. However, this conflicts with the population lower bound, proving infeasibility. We also observe that SHIR proves the infeasibility of Colorado at the root node—with the help of the MIP solver's presolve and cuts. When these features are turned off, branching is required to solve the SHIR model.

Surprisingly, South Carolina (with only 46 counties!) was left unsolved by all contiguity models after a one-hour time limit. Digging deeper, we find that each of its counties can be placed in a suitable district (satisfying contiguity and population balance), but that these districts cannot be pieced into a full districting plan. This provides a partial explanation why the contiguity models have more trouble proving the infeasibility of this instance than, say, Colorado. Another reason why the models might struggle is a sort of symmetry. To illustrate, observe that a partition $(V_1, V_2, \ldots, V_k)$ of the vertices can be represented in many ways in the formulation ($|V_1||V_2|\cdots|V_k|$ to be exact). Normally, we can distinguish between these solutions by their objective values (which differ based on the choice of centers). If any such feasible solution is discovered, then the other representations (with inferior objectives) would be pruned by bound. However, when the instance is infeasible there is no incumbent solution that can be used for pruning, unleashing the combinatorial explosion. If an instance is suspected to be infeasible, we can attempt to prove this by imposing a canonical center for a district, say, with the largest population, by fixing $x_{ij} = 0$ whenever $p_i > p_j$. In this way, we can prove the infeasibility of South Carolina with LCUT in three seconds and safely report the objective value in Table 3.3 as ∞. Applegate (2019) and Buchanan (2019) confirm the infeasibility of South Carolina using different methods.

3.5.4 Tract-level results

Now we report tract-level results for the Hess, SHIR, and CUT formulations. The formulation MCF is omitted because it is too large to handle tract-level instances, and LCUT is omitted because it performs nearly the same as CUT on tract-level instances. As before, we do not consider the seven trivial instances (where $k = 1$) in our experiments. The remaining 43 tract-level instances are feasible. Figure 3.10 gives an optimal solution for Oklahoma.

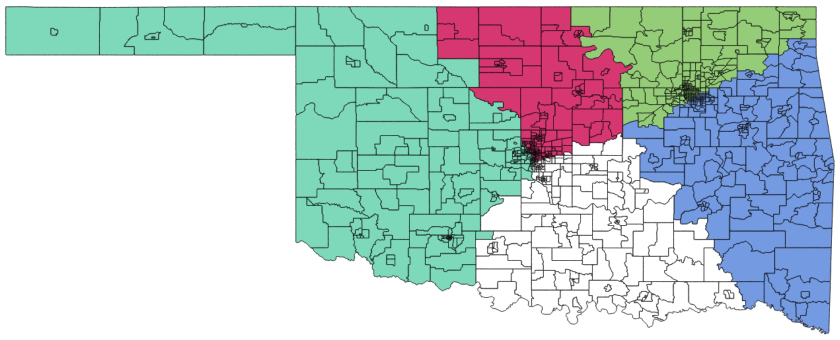


Figure 3.10: A tract-level solution for Oklahoma that is optimal for the Hess and CUT models.

Table 3.4 gives the results under a one-hour time limit. In every case, we apply the

Table 3.4: Results for solving the resulting MIPs at the tract level. For each formulation, we report the final lower and upper bounds at termination within a 3,600 second time limit, as reported by Gurobi. When the LP relaxation does not solve within the time limit, the time is reported as LPNS.

		Hess			SHIR			CUT	
State	LB	UB	time	LB	UB	time	LB	UB	time
RI	228.42	228.42	0.00	228.42	228.42	0.02	228.42	228.42	0.00
NH	2,688.28	2,688.28	0.03	2,688.28	2,688.28	0.13	2,688.28	2,688.28	0.01
ID	53,916.36	53,916.36	0.01	53,916.36	53,916.36	0.02	53,916.36	53,916.36	0.00
HI	13,991.27	13,991.27	0.01	13,991.63	13,991.63	0.15	13,991.63	13,991.63	0.01
ME	7,716.04	7,716.04	0.00	7,716.04	7,716.04[**]	0.02	7,716.04	7,716.04[**]	0.00
WV	11,026.27	11,801.80	TL	10,937.99	11,834.84	TL	11,053.92	11,834.84	TL
NM	31,598.10	31,598.10	24.79	31,603.15	31,603.15	216.76	31,603.15	31,603.15	44.78
NE	21,983.14	21,983.14	0.22	21,983.42	21,983.42	2.40	21,983.42	21,983.42	0.16
UT	22,035.37	22,035.37	9.57	22,037.00	22,037.00	44.09	22,037.00	22,037.00	16.85
MS	16,303.82	16,303.82	0.17	16,305.45	16,305.45	1.57	16,305.45	16,305.45	0.19
AR	15,563.52	15,563.52	17.72	15,569.97	15,569.97	119.47	15,569.97	15,569.97	14.99
NV	11,383.34	12,443.84	TL	11,368.70	12,497.05	TL	11,370.55	12,497.05	TL
KS	22,784.49	22,784.49	1,310.10	22,784.53	22,786.69	TL	22,786.69	22,786.69	660.25
IA	18,172.01	18,172.01	12.27	18,176.37	18,176.37	276.96	18,176.37	18,176.37	39.99
CT	1,422.13	1,422.13	8.39	1,422.13	1,422.13	32.21	1,422.13	1,422.13	6.96
OR	26,749.59	26,749.59	81.37	26,749.74	26,749.74	733.67	26,749.74	26,749.74	83.74
OK	19,107.72	19,107.72	2.09	19,107.72	19,107.72	17.94	19,107.87	19,107.87[*]	1.65
SC	8,357.73	9,293.98	TL	-	9,294.45	LPNS	8,360.04	9,294.45	TL
KY	14,833.72	14,833.72	244.93	14,835.16	14,835.16	2,272.44	14,835.16	14,835.16	2,354.22
LA	13,909.48	14,829.56	TL	-	14,830.45[**]	LPNS	13,909.56	14,830.45[**]	TL
AL	13,073.21	13,510.26	TL	-	13,510.26	LPNS	13,080.87	13,510.26	TL
CO	-	19,916.43	LPNS	-	19,918.36	LPNS	19,715.23	19,918.36	TL
MN	24,217.31	24,220.16	TL	24,210.23	24,223.84	TL	24,217.35	24,224.19	TL
MO	-	21,214.38	LPNS	-	21,222.79	LPNS	-	21,222.79	LPNS
MD	5,082.38	5,084.32	TL	5,080.71	5,084.53	TL	5,083.39	5,084.53	TL
WI	16,272.94	16,311.75	TL	-	16,340.81	LPNS	16,273.67	16,340.81	TL
WA	-	12,604.21	LPNS	-	12,609.09	LPNS	-	12,609.09	LPNS
MA	-	2,626.48	LPNS	-	2,627.03[**]	LPNS	-	2,627.03[**]	LPNS
TN	10,783.27	13,092.08	TL	-	13,092.08	LPNS	10,786.16	13,092.08	TL
IN	11,078.13	11,081.76	TL	11,075.06	11,081.76	TL	11,078.06	11,081.76	TL
AZ	-	30,219.11	LPNS	-	30,220.39	LPNS	-	30,220.39	LPNS
VA	-	13,814.08	LPNS	-	13,815.25	LPNS	12,836.85	13,815.25	TL
GA	15,836.60	15,955.65	TL	-	15,955.65	LPNS	-	15,955.65	LPNS
NJ	-	2,291.51	LPNS	-	2,291.51	LPNS	-	2,291.51	LPNS
NC	-	14,679.00	LPNS	-	14,680.02	LPNS	14,417.50	14,680.02	TL
MI	-	24,685.68	LPNS	-	24,686.55	LPNS	-	24,686.55	LPNS
OH	-	11,908.25	LPNS	-	11,911.64	LPNS	-	11,911.64	LPNS
IL	-	16,067.49	LPNS	-	16,091.05	LPNS	-	16,091.05	LPNS
PA	-	11,799.41	LPNS	-	11,806.00	LPNS	-	11,806.00	LPNS
FL	-	15,222.31	LPNS	-	15,225.47	LPNS	-	15,225.47	LPNS
NY	-	16,700.05	LPNS	-	16,700.81	LPNS	-	16,700.81	LPNS
TX	-	72,846.42	LPNS	-	72,860.21	LPNS	-	72,860.21	LPNS
CA	-	27,603.66	LPNS	-	-	LPNS	-	-	LPNS

[*] The objective values obtained for the CUT and SHIR models are inconsistent with each other. Suspecting this inconsistency was due to a numerical issue, we reran both models with the NumericFocus parameter set to 3. With this change, both models reported an objective value of 19,107.72.

[**] The best solution that was found is connected in the contiguity graph but not on the map (see Section 3.5.1).

heuristic from Section 3.4 and provide the resulting solution to Gurobi as a MIP start. We also use the associated upper bound in the Lagrangian-based procedure discussed in Section 3.4 to safely fix some variables to zero. As expected, the formulations solve most quickly when the number of tracts is small, and they generally struggle more and more as n increases. They all solve instances as big as Kentucky ($n = 1,115$), but solve none of the larger instances within a one-hour time limit. Somewhat surprisingly, the MIP solver even struggles with West Virginia ($n = 484$) and Nevada ($n = 687$)–whether or not contiguity is imposed. One explanation for this is that the Lagrangian-based reduced cost fixing is less effective on these instances (recall Table 3.1).

In some cases, Hess is noticeably faster than SHIR and CUT; for example, the times for Kentucky are 245 and 2,272 and 2,354 seconds, respectively. There are also cases where CUT is noticeably faster than Hess and SHIR; for example, the times for Kansas are 660 and 1,310 and $> 3,600$ seconds, respectively. Meanwhile, SHIR is consistently the slowest, performing worse than Hess in all cases, and worse than CUT in all but one case (Kentucky, by 82 seconds). The dominance of Hess over SHIR matches the experience of many researchers over the years who have observed that contiguity constraints tend to make problems more difficult. However, if using an alternative model for enforcing contiguity (CUT), this no longer seems to be true. In fact, the CUT and Hess models seem to have very similar performance. They solve the same 16 instances within a one-hour time limit, and their running times on these instances are on par with each other.

Another observation is the LP relaxations are often quite difficult to solve. Half of the SHIR LP relaxations could not be solved within the one-hour time limit, and one third of the CUT LP relaxations could not be solved. This is summarized in Table 3.5.

Out of curiosity, we decided to test the limits of the approach by running select states without a time limit. Table 3.6 reports five instances that eventually solved. Interestingly, the MIP solver could not solve the Hess model for Minnesota when it is told to obey a one-hour

Table 3.5: Summary of tract-level MIP results with a one-hour time limit.

Status	Hess	SHIR	CUT
# MIPs solved	16	15	16
# LPs solved (but not MIP)	11	6	13
# LPs not solved	16	22	14

time limit; however, when no time limit is imposed it solves in 2,205 seconds. The MIP solver appears to behave differently when told to obey a one-hour time limit.

Table 3.6: Tract-level MIPs that are solved when no time limit is imposed.

			Hess		CUT	
State	n	k	obj	time	obj	time
WV	484	3	11,801.26	27,056.92	11,834.84	28,984.58
MN	1,338	8	24,220.16	2,204.75	24,221.70	92,312.17
MD	1,406	8	5,084.32	36,553.32	5,084.44	75,539.58
WI	1,409	8	16,311.75	173,216.16	16,312.72	293,674.57
IN	1,511	9	11,081.72	10,570.75	11,081.72	13,435.36

3.6 Conclusion

Many researchers have stated that contiguity constraints make districting problems particularly difficult to solve. However, the experiments conducted in this chapter suggest otherwise; the exact same 21 tract-level instances are solved to optimality whether or not contiguity is imposed. One explanation for this phenomenon is that the compactness objective proposed by Hess et al. (1965) leads to *nearly* contiguous districting plans; a slight nudge is all that is needed to achieve "full" contiguity. Formulations based on graph cuts, like the CUT formulation of Oehrlein and Haunert (2017) and the newly proposed LCUT formulation, are well-suited to this task. Few inequalities are needed to prove optimality. Meanwhile, flow-based formulations can also perform well on smaller instances, but their size becomes a problem sooner as their LP relaxations are larger.

The biggest bottleneck is solving the root LP relaxations. The bedrock of the formulations considered in this chapter—the Hess formulation—grows quadratically with the number n of vertices. With Lagrangian reduced-cost fixing, many of these variables can be avoided, allowing half of the tract-level instances to be solved. With small tweaks to the implementation, other instances may be within reach, but entirely new ideas may be needed to solve the largest instances like Texas ($n = 5,265$) and California ($n = 8,057$).

We make no claims that the maps generated in our computational experiments are "good" or even legal. For example, our implementation does not explicitly consider the Voting Rights Act, political subdivisions, partisan makeup of the districts, nor any laws that vary by state. We consider one measure of compactness (the most prominent one in OR models), but there are many others in the literature. We hope that the work conducted in this chapter (including our publicly available source code and test instances) will provide a basis for further districting research.

CHAPTER IV

A LINEAR-SIZE FORMULATION FOR THE MINIMUM SPANNING TREE PROBLEM IN PLANAR GRAPHS[1]

In 2002, Williams (2002a) proposed an extended formulation for spanning trees of a planar graph. The formulation is remarkably small (using only linearly many variables and constraints) and remarkably strong (defining an integral polytope). This shows that the spanning tree polytope of a simple planar graph $G = (V, E)$ has extension complexity $O(n)$, where $n = |V|$ is the number of its vertices.

This result is frequently mentioned as an exemplary example of the power of extended formulations (Braun et al., 2016; Faenza et al., 2015; Iwata et al., 2016), and is a key ingredient in several subsequent extended formulations (Conforti et al., 2015; Kaibel et al., 2016; Fiorini et al., 2017). Indeed, a recent paper by Fiorini et al. (Fiorini et al., 2017), which gives size $O(g^{1/2}n^{3/2} + g^{3/2}n^{1/2})$ extended formulations for spanning trees in graphs of genus g, depends crucially on Williams' result. The spanning tree formulation and a subsequent formulation for vertex-induced connectivity by Williams (Williams, 2002b) are also used computationally (Xiao, 2006; Ahmadi and Martí, 2015; Wang and Önal, 2011; Kim and Xiao, 2017).

In this chapter, we show that Williams' spanning tree formulation is incorrect as stated. Specifically, we construct a binary feasible solution to Williams' spanning tree formulation that does not represent a spanning tree. Fortunately, a small tweak corrects the formulation. The change is to restrict the choice of the root vertices in the primal and dual spanning trees,

[1]Reprinted with permission from "A Note on "A linear-size zero-one programming model for the minimum spanning tree problem in planar graphs"" by H. Validi and A. Buchanan. *Networks*, 73(1): 135-142, 2019.

whereas Williams explicitly allowed them to be chosen arbitrarily. That this restriction on the roots results in a correct formulation was first observed by Pashkovich and Kaibel (and appears in Pashkovich's dissertation (Pashkovich, 2012)), but the result and its proof have not appeared outside of this dissertation, nor does the dissertation point out the error with Williams' original formulation. Additionally, we prove the converse statement: if their Root Rule is not followed, then Williams' formulation will (always) allow a solution that is not a spanning tree. This shows that the Root Rule is, in some sense, *the* way to fix Williams' formulation.

Then, we give a counterexample to a subsequent formulation from Williams for vertex-induced connected subgraphs of planar graphs, which was introduced in the context of acquiring contiguous parcels of land (Williams, 2002b). That this second formulation is incorrect as stated is perhaps unsurprising given that it relies on the correctness of the spanning tree formulation. Fortunately, the same Root Rule patches it.

Terminology and Notation

Williams' formulation uses the notion of the planar dual graph (defined below), which may have parallel edges and loops even when the graph coming from the original application does not. So, in an effort to keep the exposition uniform and general, all graphs considered here will be *multigraphs*, which are permitted to have parallel edges and loops. In this case, edges e are not uniquely identified by their endpoints u and v, so we will not write $e = \{u, v\}$, contra Williams (Williams, 2002a) and Pashkovich (Pashkovich, 2012).

In this note, $G = (V, E)$ will be a connected, undirected multigraph with vertex set V and edge set E. We call G the *primal* graph, or simply the "primal." Replacing each undirected edge of G by its directed counterpart(s) yields the *bidirected* primal graph $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$, or simply the "bidirected primal." When an edge is a loop in G, we intend for $\overleftrightarrow{G}$ to have only one associated directed edge. Undirected edges in G that have distinct endpoints result in two oppositely directed edges in $\overset{\leftrightarrow}{G}$.

We assume that G is embedded in the plane so that no two of its edges cross (i.e., a planar embedding). In this case, there is an associated graph $G^* = (V^*, E^*)$ called the *planar dual* of G, or simply the "dual." This dual graph G^* is created as follows[2]. Place one dual vertex inside each face of the embedding of G, including the exterior face. Place a dual edge across each primal edge, and connect it to the dual vertices representing the faces on either side of the primal edge. These two faces are identical when the primal edge is a bridge, yielding a loop in G^*. Figure 4.1 gives an example primal/dual pair of graphs.

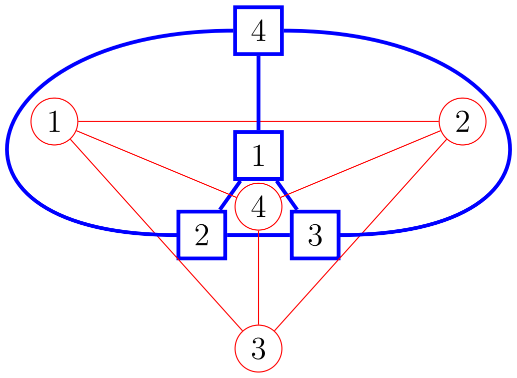


Figure 4.1: A planar embedding of "primal" G with round vertices, and its planar "dual" G^* with square vertices.

Note that G^* may have parallel edges and loops even when G does not. For example, this is true if G is a cycle with k edges, in which case G^* consists of two vertices with k parallel edges between them. Also, if G is a tree with k edges, then G^* consists of a single vertex with k loops.

Replacing each undirected edge of G^* by its directed counterpart(s) gives the bidirected dual graph $\overset{\leftrightarrow}{G^*} = (V^*, \overset{\leftrightarrow}{E^*})$, or simply the "bidirected dual." Figure 4.2 summarizes the notation.

As is standard, $\delta_H(i)$ denotes the subset of edges incident to vertex i in an undirected

[2]The notation G^* for the dual is common and is used, for example, by Tutte (Tutte, 1984) and Diestel (Diestel, 2010).

$$\begin{tikzcd} & \text{Undirected} & & \text{Directed} \\ \text{Primal} & G \arrow[d, leftrightarrow] \arrow[rr, "\text{bidirect}"] & & \overleftrightarrow{G} \\ \text{Dual} & G^* \arrow[rr, "\text{bidirect}"] & & \overleftrightarrow{G}^* \end{tikzcd}$$

Figure 4.2: The names of the multigraphs in this note.

graph H, and $\delta^-_H(i)$ denotes the subset of directed edges entering vertex i in a directed graph H. The function q_H maps an undirected edge to the set of its endpoints in graph H, as in $q_H(e) = \{u, v\}$. Note that $q_H(e)$ will be a singleton when e is a loop.

There is a natural bijection between the edges of the primal and dual, based on their crossings. Following Diestel (Diestel, 2010), the notation e will often refer to a primal edge, and e^* to its associated dual edge from the bijection.

4.1 Spanning Tree Formulation

Williams' formulation (Williams, 2002a) exploits the complementary nature of spanning trees in the primal and dual. For example, consider the primal spanning tree in Figure 4.3 indicated by the solid lines. The spanning tree drawn in the dual is the unique spanning tree that does not cross the edges of the primal spanning tree. Observe that exactly one edge is chosen from each pair of crossing edges. For example, the edge connecting primal vertices 1 and 4 is chosen but not the crossing edge that connects dual vertices 1 and 2. We will formalize these relationships in Lemma 10.

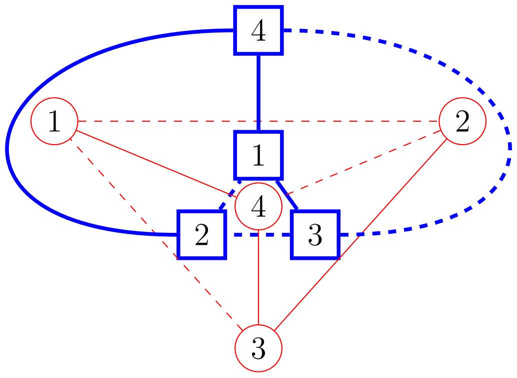


Figure 4.3: The complementary nature of primal and dual spanning trees. Solid edges are chosen and dashed edges are not.

Williams' formulation is as follows[3]. It models a spanning arborescence, which is a directed version of a spanning tree in which all edges are pointed away from a root vertex, in which case each vertex (besides the root) has one incoming edge. For each primal edge e and for each of its endpoints i, there is a binary variable $x_{e,i}$ representing the choice to select edge e and orient it towards vertex i. There are similar binary variables $y_{e^*,u}$ for dual edges e^* and endpoints u. Recall that e^* is the notation for the dual edge that crosses primal edge e.

[3]We make minor modifications. First, we model a spanning arborescence (instead of a spanning anti-arborescence) because arborescences are perhaps more common and also because this follows Pashkovich's dissertation. Second, Williams removes all edges pointing away from each terminus because the associated variables would end up being zero anyways in an anti-arborescence. Instead, like in Pashkovich's dissertation, we retain the edges pointing towards the roots. Third, Williams uses a variable x_{ij} for edge $e = (i, j)$, but due to the possibility for parallel edges, we instead use variables $x_{e,j}$, like in Pashkovich's dissertation, requiring us to refer to edge subsets $\delta_\cdot(\cdot)$ instead of vertex neighborhoods. Fourth, we introduce the notation $q_\cdot(\cdot)$ to correctly handle loops in the constraints (4.5).

Williams states that a primal root $r \in V$ and dual root $r^* \in V^*$ are to be picked arbitrarily.

$$\sum_{e \in \delta_G(i)} x_{e,i} = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.1}$$

$$\sum_{e \in \delta_G(i)} x_{e,i} = 0 \qquad i = r \tag{4.2}$$

$$\sum_{e^* \in \delta_{G^*}(u)} y_{e^*,u} = 1 \qquad \forall u \in V^* \setminus \{r^*\} \tag{4.3}$$

$$\sum_{e^* \in \delta_{G^*}(u)} y_{e^*,u} = 0 \qquad u = r^* \tag{4.4}$$

$$\sum_{i \in q_G(e)} x_{e,i} + \sum_{u \in q_{G^*}(e^*)} y_{e^*,u} = 1 \qquad \forall e \in E \tag{4.5}$$

$$x_{e,i} \in \{0,1\} \qquad \forall e \in \delta_G(i),\ i \in V \tag{4.6}$$

$$y_{e^*,u} \in \{0,1\} \qquad \forall e^* \in \delta_{G^*}(u),\ u \in V^*. \tag{4.7}$$

The indegree constraints (4.1), (4.2), (4.3), (4.4) ensure that the roots r and r^* have zero incoming edges and that all other vertices have one incoming edge. The crossing edge constraints (4.5) typically take the form $x_{e,i} + x_{e,j} + y_{e^*,u} + y_{e^*,v} = 1$, where $\{i, j\}$ and $\{u, v\}$ are the endpoints of e and e^*, respectively. One exception is when e is a loop, in which case the constraint will take the form $x_{e,i} + y_{e^*,u} + y_{e^*,v} = 1$. The other exception is when e is a bridge, meaning e^* will be a loop, and the constraint will take the form $x_{e,i} + x_{e,j} + y_{e^*,u} = 1$. However, in any feasible solution these loop variables will take a value of zero since the corresponding bridge variables will sum to one, so the loop variables can be removed from the formulation. If desired, the variables in constraints (4.2) and (4.4) can also be removed.

As given, the formulation has $4|E| - b - s$ variables, where b and s are the number of bridges and loops, respectively, in G. If $G = (V, E)$ is planar and *simple*, the number of variables is $O(n)$, where $n = |V|$. This is because simple planar graphs $G = (V, E)$ with at least three vertices satisfy $|E| \leq 3|V| - 6$, which holds by Euler's polyhedral formula.

Spanning tree counterexample. Figure 4.4 gives a counterexample to Williams' formulation (Williams, 2002a). Here, we have chosen $r = 4$ and $r^* = 4$ as the primal and dual roots, respectively. Observe that each primal and dual vertex (besides the roots) has precisely one incoming solid edge, so the indegree constraints (4.1) and (4.3) are satisfied. And, the roots have zero incoming solid edges, so constraints (4.2) and (4.4) are satisfied. Finally, the crossing edge constraints (4.5) are satisfied. Yet, the selected primal edges form a directed cycle on vertices 1, 2, and 3; they do not form a spanning arborescence.

Later in this chapter we give a general procedure for constructing such counterexamples when they exist; see the proof of Lemma 11 for the details.

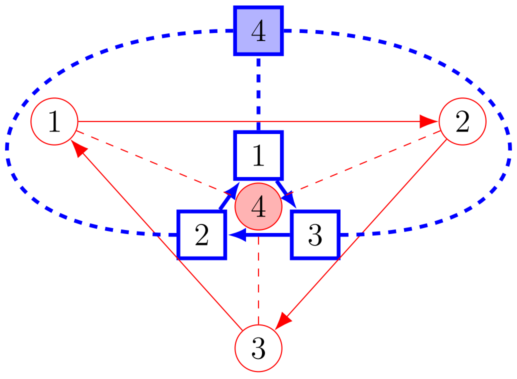


Figure 4.4: A counterexample to the spanning tree formulation of Williams (Williams, 2002a).

4.1.1 Spanning Tree Fix

To fix Williams' formulation, we can use the following Root Rule (Pashkovich, 2012).

Root Rule (Due to Pashkovich and Kaibel)**.** *Arbitrarily pick a face $r^* \in V^*$ as the dual root. Then, pick a vertex $r \in V$ from that same face to be the primal root.*

Figure 4.5 gives examples of good and bad choices for the primal and dual roots.

Lemma 9 (folklore)**.** *Let $G = (V, E)$ be a directed graph. The edge subset $\hat{E} \subseteq E$ is a spanning arborescence rooted at vertex $r \in V$ if and only if the following hold:*

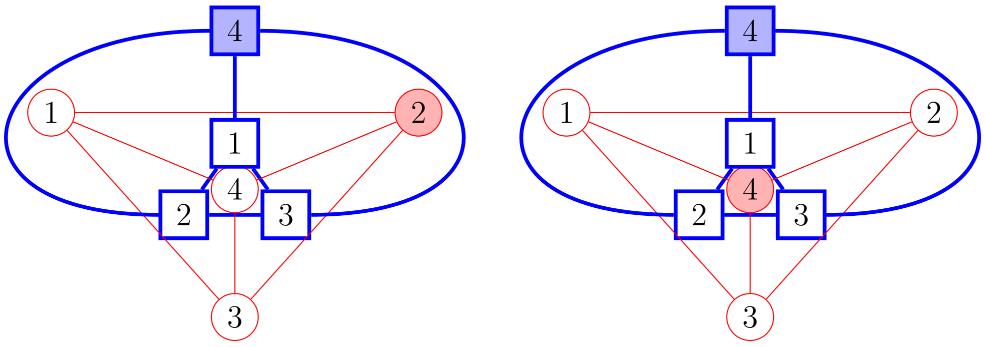


Figure 4.5: A good choice (left) and a bad choice (right) for the roots.

1. *for each non-root $i \in V \setminus \{r\}$, exactly one directed edge incoming to i belongs to $\hat{E}$, i.e., $|\delta^-_{\overleftrightarrow{G}}(u) \cap \hat{E}| = 1$;*
2. *no edge incoming to the root r belongs to $\hat{E}$, i.e., $|\delta^-_{\overleftrightarrow{G}}(r) \cap \hat{E}| = 0$;*
3. *the subgraph $(V, \hat{E})$ contains no directed cycles.*

Lemma 10 (see Theorem XI.6 of Tutte (Tutte, 1984))**.** *Suppose G is connected. The edge subset $T \subseteq E$ is the edge set of a spanning tree of G if and only if $T^* := \{e^* \mid e \in E \setminus T\}$ is the edge set of a spanning tree of G^*.*

With these two lemmata, it can be shown that the Root Rule fixes the formulation, which we prove for completeness. The idea behind the proof originates with Pashkovich and Kaibel (see Lemma 3.4 in Pashkovich (2012)).

Theorem 10 (Pashkovich and Kaibel)**.** *If the Root Rule is followed, then Williams' spanning tree formulation is correct.*

Proof. Suppose the Root Rule is followed. We are to show that a binary vector $\hat{x}$ represents a spanning arborescence of $\overleftrightarrow{G}$ rooted at r if and only if there exists a binary vector $\hat{y}$ such that $(\hat{x}, \hat{y})$ satisfies the formulation.

($\Longrightarrow$) Suppose that $\hat{x}$ represents a spanning arborescence of $\overleftrightarrow{G}$. Undirecting these edges gives a spanning tree of G with edge set T. By Lemma 10, $T^* := \{e^* \mid e \in E \setminus T\}$ is the edge set of a spanning tree of G^*. Directing the edges of T^* away from r^* gives a spanning arborescence of $\overleftrightarrow{G}^*$ rooted at r^*. Let $\hat{y}$ be its characteristic vector. Since $\hat{x}$ and $\hat{y}$ represent spanning arborescences rooted at r and r^*, respectively, and by Lemma 9, $(\hat{x}, \hat{y})$ satisfies the indegree constraints and binary restrictions from Williams' formulation. Finally, $(\hat{x}, \hat{y})$ satisfy constraints (4.5) since they represent orientations of the complementary spanning trees T and T^*.

($\Longleftarrow$) Suppose that $(\hat{x}, \hat{y})$ satisfies the formulation. We show that $\hat{x}$ represents a spanning arborescence of $\overleftrightarrow{G}$ by showing it satisfies the properties from Lemma 9. Since $\hat{x}$ satisfies the indegree constraints (4.1) and (4.2), properties 1 and 2 of Lemma 9 are satisfied. So, it suffices to show that $\hat{x}$ satisfies property 3, which we show via Pashkovich and Kaibel's infinite nested cycles.

For contradiction purposes, suppose $\hat{x}$ selects the edge set $\hat{C}_1$ of a directed cycle (and possibly other edges too). The cycle does not contain the primal root r by constraint (4.2). It creates two distinct regions in the planar embedding: the interior and the exterior of the cycle. Let R_1 be the region that does *not* contain the primal root r. By the Root Rule, the dual root r^* also does not belong to R_1. Now, the region R_1 contains at least one dual vertex, say $u \in V^* \setminus \{r^*\}$. By the indegree constraints (4.3), there is one dual edge e^* that is selected and oriented towards u, and by the crossing edge constraints (4.5) this edge cannot cross the cycle $\hat{C}_1$ into R_1. Thus, the other endpoint of e^*, say v, also belongs to R_1. Repeatedly tracing the directed edges backwards in this way shows that $\hat{y}$ selects at least one directed cycle $\hat{C}_1^*$ of $\overleftrightarrow{G}^*$ within R_1. This dual cycle $\hat{C}_1^*$ also creates two regions: its interior and exterior. Let the region that does not contain r (and thus does not contain r^*) be called R_1^*. By the same arguments as before, $\hat{x}$ selects a directed cycle of G that lies within R_1^*. Repeating this procedure gives an infinite sequence of nested primal/dual cycles $\hat{C}_1, \hat{C}_1^*, \hat{C}_2, \hat{C}_2^*, \dots$ that are edge disjoint. This contradicts that $\overleftrightarrow{G}$ and $\overleftrightarrow{G}^*$ have finitely many edges, implying that $\hat{x}$ satisfies property 3. So, the directed edges selected by $\hat{x}$ satisfy all properties from Lemma 9 and thus form a spanning arborescence of $\overleftrightarrow{G}$. $\square$

Observe that any choice of primal and dual roots satisfies the Root Rule when G is a tree or a cycle. First, planar embeddings of trees have only one face, meaning that there is only one choice for the dual root r^*, and all primal vertices lie on this face. Second, in the case of a cycle, there are two faces: interior and exterior. Again, all primal vertices lie on these faces, so either choice for the dual root will end up satisfying the Root Rule.

Remark 5. *If G is a tree or a cycle, then any choice of primal and dual roots satisfies the Root Rule.*

Lemma 11. *If G is neither a tree nor a cycle, then there is a choice of primal and dual roots such that Williams' spanning tree formulation is incorrect.*

Proof. Suppose that G is neither a tree nor a cycle. We select primal and dual roots r and r^* and construct a feasible solution $(\hat{x}, \hat{y})$ to the formulation such that $\hat{x}$ does not represent a spanning arborescence rooted at r.

Because G is not a tree (and by assumption throughout this note that G is connected), it contains an undirected cycle. Direct its edges into a directed cycle $\hat{C} \subset \overleftrightarrow{E}$. Create a breadth-first search (BFS) tree in G emanating from this cycle and let its directed edges be $\hat{T} \subset \overleftrightarrow{E}$. Let $r \in V$ be a leaf vertex in this BFS tree and let the directed edge pointing to r be $e_r \in \hat{T}$. (That r and e_r exist outside of this cycle holds because G is connected but is itself not a cycle.) Let $\hat{x}$ be the characteristic vector of $\hat{C} \cup \hat{T} \setminus \{e_r\}$. It can be observed that $\hat{x}$ satisfies constraints (4.1) and (4.2).

By undirecting the edges of $\hat{C} \cup \hat{T} \setminus \{e_r\}$ we obtain the edge subset $\hat{E} \subset E$. Let $\hat{E}^* := \{e^* \mid e \in E \setminus \hat{E}\}$. We argue that $\hat{E}^*$ induces a subgraph of G^* with two components: a tree and a 1-tree (i.e., a tree that has one extra edge). To see this, let e_c be an edge from the directed primal cycle $\hat{C}$. Let S be the spanning tree of G obtained by undirecting the edges of $(\hat{C} \setminus \{e_c\}) \cup \hat{T}$. Then, by Lemma 10, the dual edge subset $S^* := \{e^* \mid e \in E \setminus S\}$ is a spanning tree of G^*. Then, $S^* \setminus \{e^*_c\}$ is a forest containing two trees and $\hat{E}^* = (S^* \setminus \{e^*_c\}) \cup \{e^*_r\}$ induces a tree and a 1-tree.

Direct the edges of the 1-tree's cycle into a directed cycle $\hat{C}^*$, and let $\hat{T}^*$ be the directed edges of a BFS tree in $(V^*, \hat{E}^*)$ emanating from the cycle. Pick a vertex r^* from the tree component of $(V^*, \hat{E}^*)$ and direct the edges of the tree component away from r^*, giving the subset $\hat{T}^*_t$ of directed edges. Let $\hat{y}$ be the characteristic vector of $\hat{C}^* \cup \hat{T}^* \cup \hat{T}^*_t$. It can be observed that $\hat{y}$ satisfies constraints (4.3) and (4.4). Finally, constraints (4.5) are satisfied because $\hat{x}$ and $\hat{y}$ are orientations of the complementary edge subsets $\hat{E}$ and $\hat{E}^*$.

So, $(\hat{x}, \hat{y})$ satisfies all constraints of the formulation, but $\hat{x}$ does not represent a spanning arborescence of $\overleftrightarrow{G}$. Therefore, the formulation is incorrect under this choice of primal root r and dual root r^*. □

By Theorem 10, Remark 5, and Lemma 11, we have the following theorem.

Theorem 11. *Williams' spanning tree formulation is correct if and only if the Root Rule is followed.*

Williams (Williams, 2002a) notes that his formulation has a totally unimodular constraint matrix, implying that his formulation is integral. Pashkovich (Pashkovich, 2012) observes in a footnote that total unimodularity persists when G and G^* are multigraphs, as long as each loop in the undirected graphs results in one directed loop in the bidirected graphs. Since we are primarily concerned with the formulation's correctness, and not its integrality, we refer the reader to page 42 of Pashkovich (Pashkovich, 2012) for the proof.

4.2 Connected Subgraph Formulation

In a second paper, Williams (2002b) adapts his spanning tree formulation so that it selects contiguous parcels of land. In the graph context, a feasible solution is a subset $S \subseteq V$ of vertices that induces a connected subgraph $G[S]$. Again, he exploits planar graph duality, but the task of selecting only a *subset* of the *vertices* is different and uses different variables.

For each primal vertex $i \in V$, there is a variable u_i representing the decision whether or not to include it in S. There are two sets of primal edge variables, $x_{e,i}$ and $y_{e,i}$, that together select a spanning arborescence of $\overleftrightarrow{G}$. The variables $x_{e,i}$ select an arborescence that spans *the vertices selected by the u_i variables* (the "sub-arborescence"), and the variables $y_{e,i}$ represent the other edges of the spanning arborescence of $\overleftrightarrow{G}$. The variables $z_{e^*,u}$ represent a spanning arborescence of G^*. As before, $e^* \in E^*$ is the dual edge that crosses $e \in E$.

The formulation is as follows, where Williams again states that the primal and dual roots can be chosen arbitrarily. The first five constraints are identical to the spanning tree formulation, except that the selected primal edges are broken into two parts: $x_{e,i}$ and $y_{e,i}$. The last few constraints relate the vertex and edge decisions for the sub-arborescence[4].

[4]Originally, Williams imposed cardinality constraints on the number p of vertices (parcels) and the number $p - 1$ of edges. Instead, we write the more general constraint (4.15).

$$\sum_{e \in \delta(i)} (x_{e,i} + y_{e,i}) = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.8}$$

$$\sum_{e \in \delta(i)} (x_{e,i} + y_{e,i}) = 0 \qquad i = r \tag{4.9}$$

$$\sum_{e^* \in \delta(u)} z_{e^*,u} = 1 \qquad \forall u \in V^* \setminus \{r^*\} \tag{4.10}$$

$$\sum_{e^* \in \delta(u)} z_{e^*,u} = 0 \qquad u = r^* \tag{4.11}$$

$$\sum_{i \in q_G(e)} x_{e,i} + \sum_{i \in q_G(e)} y_{e,i} + \sum_{u \in q_{G^*}(e^*)} z_{e^*,u} = 1 \qquad \forall e \in E \tag{4.12}$$

$$\sum_{j \in q_G(e)} x_{e,j} \leq u_i \qquad \forall i \in q_G(e),\ e \in E \tag{4.13}$$

$$\sum_{e \in \delta(i)} x_{e,i} \leq u_i \qquad \forall i \in V \setminus \{r\} \tag{4.14}$$

$$\sum_{i \in V} u_i - \sum_{i \in V} \sum_{e \in \delta(i)} x_{e,i} = 1 \tag{4.15}$$

$$x_{e,i} \in \{0,1\} \qquad \forall e \in \delta_G(i),\ i \in V \tag{4.16}$$

$$y_{e,i} \in \{0,1\} \qquad \forall e \in \delta_G(i),\ i \in V \tag{4.17}$$

$$z_{e^*,u} \in \{0,1\} \qquad \forall e^* \in \delta_{G^*}(u),\ u \in V^* \tag{4.18}$$

$$u_i \in \{0,1\} \qquad \forall i \in V. \tag{4.19}$$

Connected subgraph counterexample. Figure 4.6 gives a planar embedding of a primal graph on 6 round vertices. The dual vertices are squares.

Figure 4.7 gives a counterexample to Williams' connected subgraph formulation (Williams, 2002b). Here, we have chosen $r = 6$ and $r^* = 3$ as the primal and dual roots, respectively. Observe that all constraints of the formulation above are satisfied, and yet the gray-filled primal vertices (representing vertices i with $u_i = 1$) induce a disconnected subgraph. The

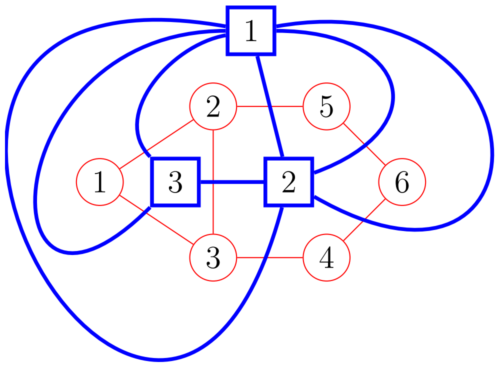


Figure 4.6: A planar embedding of a graph G and its dual G^*.

primal edges selected by the x variables (resp. y variables) are solid (resp. dashed). The dual edges selected by the z variables are solid.

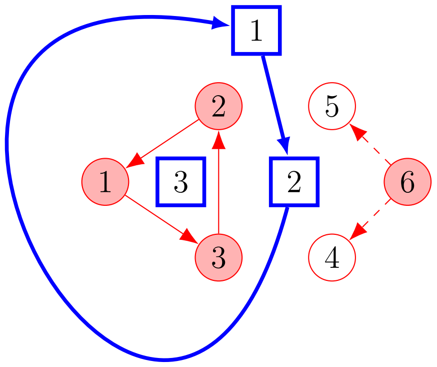


Figure 4.7: A counterexample to the connected subgraph formulation of Williams (2002b).
.

We note that even in Figure 1 of Williams' paper (Williams, 2002b) the choice of primal and dual roots violates the Root Rule, and so the x and y variables may not select a primal arborescence. However, the vertices selected by the u variables will luckily *induce* a connected subgraph for that instance. This is why we give a different counterexample—to illustrate that the selected vertices may be disconnected.

It can be shown that this formulation for vertex-induced connected subgraphs is correct when following the Root Rule. The proof of this is not difficult in light of Theorem 10 and is omitted.

4.3 Conclusion

In this chapter, we note that a rule (root rule) is not respected in William's spanning tree formulation for planar graphs. Thanks to Pashkovich (2012), we provide a fix for it. We also prove that if the root rule is not respected, then Williams' formulation always allows a solution that is not a spanning tree.

CHAPTER V

POLYHEDRAL STUDY OF k CONNECTED COMPONENTS[1]

5.1 Introduction

Imposing at most k connected subgraphs arises in interesting applications of network design and analysis problems: reserve network design problems (Jafari and Hearne, 2013, 2020; Önal et al., 2016; Jafari et al., 2017), and the max-p-regions problem (Duque et al., 2012). In each of these applications, one trie to find at most k connected subgraphs that satisfy specific properties. Wang (2015) addresses this problem as "an interesting and challenging task."

Given a simple graph $G = (V, E)$ and an integer k, our task is to write a "good" integer programming formulation for finding at most k connected components. Also, let $D = (V, A)$ be an orientation of graph G in which every edge $\{u, v\} \in E$ is replaced with exactly one directed edge $(u, v) \in A$ or $(v, u) \in A$. In this paper, we denote $n := |V|$, $m := |E|$, and the open neighborhood of vertex $v \in V$ as $N(v) := \{u \in V \mid \{v, u\} \in E\}$. Also, we define $c(G)$ and $\Delta(G)$ as the number of components and the maximum degree of graph G, respectively. For every vertex $v \in V$, we define $\deg_G(v)$ as the degree of vertex v in graph G, and d_v as the indegree value of vertex v in directed graph D. For any set S, we use $\pi(S)$ as a shorthand for $\sum_{i \in S} \pi_i$. Finally, we assume that 0-vertex and 1-vertex graphs are connected.

Definition 9. *The k-component polytope of graph $G = (V, E)$ is*

$$P_k(G) := \text{conv.hull}\left\{x^S \in \{0,1\}^n \;\middle|\; c(G[S]) \le k\right\},$$

[1]This chapter is based on work with Austin Buchanan (Validi and Buchanan, 2020c).

where x^S represents the characteristic vector of $S \subseteq V$.

Let x_i be the decision variable for each vertex $i \in V$, where setting this variable to one represents the decision to include i in set S. Our decision variables are x_i for each vertex $i \in V$, where setting this variable to one represents the decision to include i in our set S.

5.2 Basic Properties of $P_k(G)$

Proposition 10. *Consider a graph G with $q = c(G)$ components. The k-component polytope $P_k(G)$ is full-dimentional. Furthermore, for each $i \in V$,*

1. *$x_i \geq 0$ is facet-defining; and*
2. *$x_i \leq 1$ is facet-defining if and only if $k \geq 2$ or ($k = 1$ and $q = 1$).*

Proof. The points $\mathbf{0}$ and e_i for every $i \in V$ are affinely independent and show that $P_k(G)$ is full-dimentional. The n affinely independent points $\mathbf{0}$ and e_j for every $j \in V \setminus \{i\}$ satisfy $x_i \geq 0$ at equality. This proves the first statement.

Now we prove the second statement.

($\implies$) By the contrapositive, suppose $k = 1$ and $q \geq 2$. Then G is not connected and $x_i \leq 1$ does not induce a facet by Proposition 1 of Wang et al. (2017).

($\impliedby$) If $k = 1$ and $q = 1$, then G is connected and $x_i \leq 1$ is facet-defining by Proposition 1 of Wang et al. (2017). Now, suppose that $k \geq 2$. Without loss of generality, let $G' = (V', E')$ be the component that contains vertex i. Apply DFS algorithm from vertex i and let $(v_1^i, v_2^i, \dots, v_{n'}^i)$ be the order of visited vertices in the DFS procedure. For $u = 1, 2, \dots, n'$, points $\sum_{j=1}^{u} e_{v_j}$ form n' affinely independent points that satisfy $x_i \leq 1$ at equality. If $q = 1$, then we are done. Now, suppose $q \geq 2$. Then select a vertex v from $V \setminus V'$; i.e., $v \in V \setminus V'$. Then, unit vectors e_v's along with e_i form $n - n'$ points. This finishes the proof. □

Lemma 12. *Let $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ be a facet-defining inequality of $P_k(G)$. Then, $\pi_0 \geq 0$. The inequality is (a scalar multiple of) some nonnegativity bound $-x_j \leq 0$ if and only if $\pi_0 = 0$.*

Proof. Because the empty set is feasible, we have $\pi_0 \geq 0$.

($\Longrightarrow$) This is trivial.

($\Longleftarrow$) Suppose $\pi_0 = 0$. Then, for every $i \in V$, we must have $\pi_i \leq 0$ because each induced subgraph $G[\{i\}]$ is feasible. Also, suppose that at least two coefficients, say π_u and π_v, are negative. Then valid inequalities $\pi_u x_u \leq 0$ and $\sum_{i \in V \setminus \{u\}} \pi_i x_i \leq 0$ imply the valid inequality $\sum_{i \in V} \pi_i x_i \leq 0$. This contradicts the facet-defining assumption of the statement. □

Lemma 13. *Suppose graph G has components $G_1, G_2, \ldots, G_q$ with $q \geq k$, and let $\pi \in \mathbb{R}^n$ and $\pi_0 > 0$. Suppose that, for each $j \in [q]$, the inequality $\sum_{i \in V_j} \pi_i x_i \leq \pi_0$ is valid for $P_1(G_j)$. Then, the inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ is valid for $P_k(G)$.*

Proof. Suppose that each inequality $\sum_{i \in V_j} \pi_i x_i \leq \pi_0$ is valid for $P_1(G_j)$. Suppose that a subset of vertices S induces at most k components $G[S_1], G[S_2], \ldots, G[S_p]$ in graph G. So, we have

$$\pi(S) = \sum_{u=1}^{p} \pi(S_u) \leq \sum_{u=1}^{p} \pi_0 = p\pi_0 \leq k\pi_0.$$

Here, the first inequality holds by validity of $\sum_{i \in V_j} \pi_i x_i \leq \pi_0$ for $P_1(G_j)$. □

Theorem 12. *Suppose that G has components $G_1, G_2, \ldots, G_q$, and let $\pi \in \mathbb{R}^n$ and $\pi_0 > 0$. If each sub-inequality $\sum_{i \in V_j} \pi_i x_i \leq \pi_0$ induces a facet of $P_1(G_j)$ and if $q \geq k+1$, then the inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ induces a facet of $P_k(G)$.*

Proof. Because the inequality $\sum_{i \in V_j} \pi_i x_i \leq \pi_0$ induces a facet of $P_1(G_j)$ for each $G_j = (V_j, E_j)$, there are n_j affinely independent points that satisfy the inequality at equality. Let a_i^j be the i-th affinely independnet point of component j. Also, let A^j be a set of affinely independent points of component G_j. By Corollary 4.4 of Nemhauser and Trotter (1974), the following structure provides n affinely independent points that satisfy $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ at equality.

$$M = \left[\begin{array}{cccc|ccc}
\mathbf{0}_{n_1 \times n_2} & a_1^1 \mathbf{1}_{1 \times n_3} & \dots & \mathbf{A^1} & a_1^1 \mathbf{1}_{1 \times n_{k+2}} & \dots & a_1^1 \mathbf{1}_{1 \times n_q} \\
\mathbf{A^2} & \mathbf{0}_{n_2 \times n_3} & \dots & a_1^2 \mathbf{1}_{1 \times n_1} & a_1^2 \mathbf{1}_{1 \times n_{k+2}} & \dots & a_1^2 \mathbf{1}_{1 \times n_q} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
a_1^{k-1} \mathbf{1}_{1 \times n_2} & a_1^{k-1} \mathbf{1}_{1 \times n_3} & \dots & a_1^{k-1} \mathbf{1}_{1 \times n_1} & a_1^{k-1} \mathbf{1}_{1 \times n_{k+2}} & \dots & a_1^{k-1} \mathbf{1}_{1 \times n_q} \\
a_1^{k} \mathbf{1}_{1 \times n_2} & a_1^{k} \mathbf{1}_{1 \times n_3} & \dots & a_1^{k} \mathbf{1}_{1 \times n_1} & \mathbf{0} & \dots & \mathbf{0} \\
a_1^{k+1} \mathbf{1}_{1 \times n_2} & a_1^{k+1} \mathbf{1}_{1 \times n_3} & \dots & \mathbf{0}_{n_{k+1} \times n_1} & \mathbf{0} & \dots & \mathbf{0} \\
\hline
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{A}^{k+2} & \dots & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}^{q}
\end{array}\right]$$

Figure 5.1: Affinely independent points

□

Lemma 14. *Let $\pi \in \mathbb{R}^n$ and $\pi_0 > 0$ and suppose that the inequality $\sum_{i \in V} \pi_i x_i \leq \pi_0$ is valid for $P_1(G)$. If the inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ induces a facet of $P_k(G)$, then for every vertex $v \in V$ there exists a feasible tight set $S' \subseteq V$ such that $v \in S'$ and $\pi(S') = \pi_0$.*

Proof. To prove this, we first claim that for every vertex $v \in V$, there exists a feasible tight set $S \subseteq V$ such that $v \in S$ and $\pi(S) = k\pi_0$. Suppose not, then all points on face $F := P_k(G) \cap \{x \mid \pi^T x = k\pi_0\}$ also belong to the face $F' := P_k(G) \cap \{x \mid x_v = 0\}$. So, $F \subseteq F'$. Since $\mathbf{0} \in F'$ and $\mathbf{0} \notin F$, the inclusion is strict; i.e., $F \subset F'$. This shows that F is not a maximal face; so, it is not a facet. This contradicts the assumption that the inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ induces a facet of $P_k(G)$.

Because for every vertex $v \in V$, there is a feasible tight set $S \subseteq V$ such that $v \in S$ and $\pi(S) = k\pi_0$, $G[S]$ induces p components $G[S_1], G[S_2], \dots, G[S_p]$ with $p \leq k$ and $S = \uplus_{j \in [p]} S_j$. Because the inequality $\sum_{i \in V} \pi_i x_i \leq \pi_0$ is valid for $P_1(G)$, we have $\sum_{i \in S_j} \pi_i x_i \leq \pi_0$ for every $j \in [p]$. Finally, we have the following string of inequalities, where j_v denotes the component that contains vertex v.

$$k\pi_0 = \pi(S) = \sum_{j=1}^{p} \pi(S_j) = \sum_{j\in[p]\setminus\{j_v\}} \pi(S_j) + \pi(S_{j_v}) \leq p\pi_0 \leq k\pi_0.$$

Here, the first inequality holds by the fact that $\sum_{i\in S_j} \pi_i x_i \leq \pi_0$ for every $j \in [p]$. This chain of inequalities implies that $p = k$ and there exists a feasible tight set $S' := S_{j_v}$ such that $v \in S'$ and $\pi(S') = \pi_0$. □

Lemma 15 (Upper bound on lifting). *Suppose $\sum_{i\in V\setminus\{v\}} \pi_i x_i \leq k\pi_0$ induces a facet of $P_k(G-v)$ and $\sum_{i\in V\setminus\{v\}} \pi_i x_i \leq \pi_0$ is valid for $P_1(G-v)$, where $\pi_0 > 0$. Then, the lifting problem for v*

$$\zeta := \max_{S\subseteq V} \left\{ \sum_{i\in V\setminus\{v\}} \pi_i x_i^S \;\middle|\; x_v^S = 1 \text{ and } G[S] \text{ has at most } k \text{ components} \right\}$$

satisfies $\zeta \leq (\alpha(G[N(v)]) + k - 1)\,\pi_0$.

Proof. Let U' be an optimal solution for the lifting problem. Also, let $I = \{u_1, u_2, \ldots, u_s\}$ be a maximal independent set of $(G[N(v)])$, where $s = \alpha(G[N(v)])$. Then define $U := U' \setminus \{v\}$. Let G'_v be the component of $G[U']$ that contains v, let $U^1 = V(G'_v) \setminus \{v\}$ and let $U^2 = U \setminus U^1$. Partition U^1 into s subsets (some possibly empty) as follows. Let $U^1_1 \subseteq U^1$ be the set of vertices that are connected to u_1 by a path in $G[U^1]$. For every $p \in \{2, 3, \ldots, s\}$, let U^1_p be the vertices of $U^1 \setminus (U^1_1 \cup U^1_2 \cup \cdots \cup U^1_{p-1})$ that are connected to u_p by some path in $G[U^1]$. For every $p \in [s]$ because $G(U^1_p)$ is a connected subgraph of $G - v$, we have $\pi(U^1_p) \leq \pi_0$ by validity of $\sum_{i\in V\setminus\{v\}} \pi_i x_i \leq \pi_0$ for $P_1(G-v)$. Let $G[U^2_1], G[U^2_2], \ldots, G[U^2_q]$ be the components of $G[U^2]$, and see that $q \leq k-1$. Since $\sum_{i\in V\setminus\{v\}} \pi_i x_i \leq \pi_0$ is valid for $P_1(G-v)$, each component U_j^2 satisfies $\pi(U_j^2) \leq \pi_0$ for every $j \in [q]$. Thus,

$$\begin{aligned}
\zeta = \pi(U) &= \pi(U^1) + \pi(U^2) \\
&= \sum_{p=1}^{s} \left(\sum_{i \in U_p^1} \pi_i \right) + \sum_{j=1}^{q} \left(\sum_{i \in U_j^2} \pi_i \right) + \sum_{i \in V \setminus U} \pi_i x_i^U \\
&\leq s\pi_0 + (k-1)\pi_0 \\
&= \left(\alpha(G[N(v)]) + k - 1\right) \pi_0.
\end{aligned}$$

□

Lemma 16 (Lower bound on lifting)**.** *Suppose $\sum_{i \in V \setminus \{v\}} \pi_i x_i \leq k\pi_0$ induces a facet of $P_k(G-v)$ and $\sum_{i \in V \setminus \{v\}} \pi_i x_i \leq \pi_0$ is valid for $P_1(G-v)$, where $\pi_0 > 0$. Also, let $c(G-v)$ and t be the number of components of $G-v$ and $G[N(v)]$, respectively. Then, the lifting problem for v*

$$\zeta := \max_{S \subseteq V} \left\{ \sum_{i \in V \setminus \{v\}} \pi_i x_i^S \;\middle|\; x_v^S = 1 \text{ and } G[S] \text{ has at most } k \text{ components} \right\}$$

satisfies $\zeta \geq (k-1)\pi_0$ if v is isolated and

$$\zeta \geq \max\left\{k, t + \min\{k-1, c(G-v) - t\}\right\} \pi_0,$$

otherwise.

Proof. First suppose that v is isolated. Because the inequality $\sum_{i \in V \setminus \{v\}} \pi_i x_i \leq k\pi_0$ is facet-defining for $P_k(G-v)$, there exists a feasible tight set $H \subseteq V \setminus \{v\}$ for which $\pi(H) = k\pi_0$. Let $G[H_1], G[H_2], \ldots, G[H_p]$ be the connected subgraphs induced by H with $p \leq k$. By validity of $\sum_{i \in V \setminus \{v\}} \pi_i x_i \leq \pi_0$ for $P_1(G - v)$, we have

$$\pi(H_1) = \sum_{i \in V} \pi_i x^{H_1} \leq \pi_0. \tag{5.1}$$

Define $H' := (H \setminus H_1) \cup \{v\}$. H' also induces p connected components. So, we have

$$\pi(H' \setminus \{v\}) = \pi(H) - \pi(H_1) \geq \pi(H) - \pi_0 = k\pi_0 - \pi_0 = (k-1)\pi_0.$$

Here, the inequality holds by inequality (5.1). Hence, $\zeta \geq (k-1)\pi_0$ when v is isolated.

Now suppose that vertex v is not isolated. Consider a vertex $u \in N(v)$. Because the inequality $\sum_{i \in V \setminus \{v\}} \pi_i x_i \leq k\pi_0$ is facet-defining for $P_k(G - v)$, there exists a feasible tight set containing u that satisfies the inequality at equality. Hence, we have $\zeta \geq k\pi_0$ when v is not isolated.

Let $u_1, u_2, \ldots, u_t$ be vertices that connect vertex v to components $G[V_1], G[V_2], \ldots, G[V_t]$, respectively (i.e., for every $i \in [t]$, we have $N(v) \cap V_i = u_i$). By Lemma 14, for every $i \in [t]$, there exists a feasible tight set $S_i \subseteq V_i$ such that $u_i \in S_i$ and $\pi(S_i) = \pi_0$. Similarly, we can find connected components for every component of $G - v$ that is not connected to vertex v (there are $c(G - v) - t$ of them). This implies $\zeta \geq (t + \min\{k-1, c(G-v) - t\})\,\pi_0$. So, we have

$$\zeta \geq \max\{k, t + \min\{k-1, c(G-v) - t\}\}\,\pi_0.$$

Finally, we show that $\sum_{i \in V} \pi_i x_i \leq \pi_0$ is valid for $P_1(G)$ after lifting vertex v. If vertex v is isolated, then we have the following upper bound for π_v.

$$\pi_v = k\pi_0 - \zeta \leq k\pi_0 - (k-1)\pi_0 = \pi_0.$$

This means $\sum_{i\in V}\pi_i x_i \le \pi_0$ remains valid after lifting isolated vertex v. Now suppose vertex v is not isolated. Then, we claim that $\sum_{i\in V}\pi_i x_i \le \pi_0$ is always valid for $P_1(G)$ if $c(G-v)\ge k$ and $t=1$. If these two conditions hold, then we have $\zeta \ge k\pi_0$. So, let U be a connected component of graph $G[W]$ with $W := V_1\cup V_2\cup\cdots\cup V_t\cup\{v\}$. Then,

$$\begin{aligned}\sum_{i\in W}\pi_i x_i^U &= \pi_v x_v^U + \sum_{j\in[t]}\sum_{i\in V_j}\pi_i x_i^U\\ &\le (k\pi_0 - k\pi_0) + t\pi_0\\ &= \pi_0.\end{aligned}$$

For any $t\ge 2$, we claim that $\sum_{i\in V}\pi_i x_i \le \pi_0$ is always valid for $P_1(G)$ if $c(G-v)-t\ge k-1$. If this condition holds and $t\ge 2$, then we have $\zeta \ge (k+t-1)\pi_0$. So, let U be a connected component of graph $G[W]$ with $W := V_1\cup V_2\cup\cdots\cup V_t\cup\{v\}$. Then for any $t\ge 2$,

$$\begin{aligned}\sum_{i\in W}\pi_i x_i^U &= \pi_v x_v^U + \sum_{j\in[t]}\sum_{i\in V_j}\pi_i x_i^U\\ &\le k\pi_0 - (k+t-1)\pi_0 + t\pi_0\\ &= \pi_0.\end{aligned}$$

So, we have

$$\zeta \ge \begin{cases} (k-1)\pi_0 & \text{if } t=0\\ \\ (k+t-1)\pi_0 & \text{if } t\ge 1 \text{ and } c(G-v)-t\ge k-1.\end{cases}$$

□

Lemma 17. *Let $\sum_{i\in V}\pi_i x_i \le k\pi_0$ be a facet-defining inequality for $P_k(G)$. For any pair of vertices $u,v\in V$ with positive coefficients $\pi_u,\pi_v>0$, there is no path $P_{u,v}$ with $\pi(P_{u,v}) =$*

$\pi_u + \pi_v$.

Proof. Suppose there exist vertices $i, j \in V$ with $\pi_i, \pi_j > 0$ such that there is a path $\hat{P}_{ij}$ between them with $\pi(\hat{P}_{ij}) = \pi_i + \pi_j$. Since the inequality is facet-defining for $P_k(G)$, there are n affinely independent points $S_1, S_2, \ldots, S_n$ that satisfy the inequality at equality. At least one of these points, say S', contains exactly one of i and j. Without loss of generality, suppose S' contains i but not j. Let $S'' := S' \cup V(\hat{P}_{i,j})$. Because path $\hat{P}_{i,j}$ connects i to j, S'' also belongs to $P_k(G)$. But, we have

$$\sum_{w \in V} \pi_w x_w^{S''} = \sum_{w \in V} \pi_w x_w^{S'} + \pi(V(\hat{P}_{i,j}) \setminus \{i\}) = k\pi_0 + \pi_j > k\pi_0.$$

This is a contradiction. □

Lemma 18. *Suppose that the inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ induces a non-trivial facet, and define $U := \{i \in V \mid \pi_i > 0\}$. Then $|U| \geq k + 1$.*

Proof. For contradiction purposes, we consider the following three cases: (i) $|U| = 0$; (ii) $|U| = 1$; and (iii) $2 \leq |U| \leq k$.

In the first case, suppose $|U| = 0$. Note that $\pi_0 \geq 0$ by Lemma 12. Since no variable has a positive coefficient, π_0 should be zero; otherwise, no point in $P_k(G)$ satisfies the inequality at equality. Hence $\pi_0 = 0$. By Lemma 12, the inequality is a (scalar multiple of a) nonnegativity bound. This is a contradiction because the inequaity is a non-trivial facet.

In the second case, suppose that $U = \{v\}$ with $v \in V$. Then, $0 < \pi_v \leq k\pi_0$ because $G[U]$ induces at most k components and $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ is a valid inequality. Also, $\pi_v = k\pi_0$ because otherwise there is no point that satisfies the inequality at equality. So, the inequality $\pi_v x_v \leq k\pi_0$, where $\pi_v = k\pi_0$, is a valid inequality. Further, the inequality $0x_v + \sum_{i \in V \setminus \{v\}} \pi_i x_i \leq 0$ is valid because $\pi_i \leq 0$ for every $i \in V \setminus \{v\}$. If $\pi_i = 0$ for every $i \in V \setminus \{v\}$, then $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ is a scalar multiple of $x_v \leq 1$. If not, then there exists a vertex $j \in V \setminus \{v\}$ such that $\pi_j < 0$. Hence, the valid inequalities $\pi_v x_v \leq k\pi_0$ and

$0x_v + \sum_{i \in V \setminus \{v\}} \pi_i x_i \leq 0$ imply $\sum_{i \in V} \pi_i x_i \leq k\pi_0$. So, $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ is not a facet-defining inequality when $\pi_j < 0$ for a $j \in V \setminus \{v\}$.

In the third case, suppose $2 \leq |U| \leq k$. Inequality $\sum_{i \in V \setminus U} \pi_i x_i \leq 0$ is valid because $\pi_i \leq 0$ for every $i \in V \setminus U$. If $\pi_i = 0$ for every $i \in V \setminus U$, then $\sum_{i \in U} \pi_i x_i \leq k\pi_0$ is facet-defining and there is a feasible tight set W that satisfies it at equality. We claim that $U \subseteq W$; otherwise, there is a tight feasible set $U' \subset U$ for which $\pi(U') = k\pi_0$. This implies $\pi(U) > k$. Hence, $U \subseteq W$ and $\pi(U) = k\pi_0$. This means the scalar multiples of valid inequalities $x_v \leq 1$ for all $v \in U$ imply

$$\sum_{i \in U} \pi_i x_i \leq \sum_{i \in U} \pi_i = k\pi_0.$$

This is a contradiction. If there exists a vertex $j \in V \setminus U$ such that $\pi_j < 0$, then the scalar multiples of valid inequalities $x_v \leq 1$ for all $v \in U$ and $\sum_{i \in V \setminus U} \pi_i x_i \leq 0$ imply $\sum_{i \in V} \pi_i x_i \leq k\pi_0$. This means $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ is not a facet-defining inequality when $\pi_j < 0$ for a vertex $j \in V \setminus U$. $\square$

5.3 Separator Inequalities

Definition 10 (separator)**.** *In a graph $G = (V, E)$, a subset $C \subset V$ of vertices is said to be a separator for $S \subseteq V$ if no two vertices of S belong to the same component of $G - C$.*

For any $S \subseteq V$ with $|S| = k + 1$ and a separator $C \subseteq (V \setminus S)$ for S, separator inequality is defined as follows.

$$\text{(separator inequality)} \qquad x(S) - x(C) \leq k. \tag{5.2}$$

Also, define the polytope of separator inequalities ($Q_k(G)$) as follows.

$$Q_k(G) := \{x \in [0,1]^n \mid x \text{ satisfies all separator inequalities.}\}$$

Proposition 11. *The inequalities* (5.2) *and $x \in \{0,1\}^n$ correctly model the polytope $P_k(G)$.*

Proof. We show that the subgraph $G[U]$ induced by U has at most k components if and only if the characteristic vector x^U satisfies all constraints (5.2).

($\Longrightarrow$) By the contrapositive. Suppose that x^U violates some constraint (5.2) given by $x(S) - x(C) \leq k$, where C is a separator for S and $|S| = k+1$. Thus $x^U(S) - x^U(C) \geq k+1$. This happens only when $|U \cap S| = k+1$ (i.e., $S \subseteq U$) and $|U \cap C| = 0$ (i.e., $U \subseteq V \setminus C$). Because $U \subseteq V \setminus C$, each vertex $i \in S \subseteq U$ belongs to a different component of $G[U]$. Since $|S| = k+1$, $G[U]$ has at least $k+1$ components.

($\Longleftarrow$) By the contrapositive. Suppose that $G[U]$ has more than k components. Call them G_1, G_2, …, G_q with $q \geq k+1$. For each component G_i with $i \leq k+1$, pick a vertex v_i from $V(G_i)$. Let $S' = \{v_1, v_2, \ldots, v_{k+1}\}$ and $C' = V \setminus U$. Then, x^U violates constraints (5.2) for S' and C' since $x^U(S') - x^U(C') = (k+1) - 0 > k$. □

Lemma 19. *Let $C \subset V$ be a separator for $S \subseteq V$ with $|S| = k+1$. Then $x(S) \leq k$ induces a facet of $P(G - C)$.*

Proof. For any $v \in S$, let S_v be the component of $G - C$ that contains vertex v. Because inequality $x_v \leq 1$ induces a facet for $P_1(G[S_v])$, $x(S) \leq k$ induces a facet for $P(G - C)$ by Theorem 12. □

Lemma 20. *For a graph $G = (V, E)$ and an independent set $S \subseteq V$ with $|S| \geq k+1$, the inequality $x(S) \leq k$ induces a facet for $G[S]$.*

Proof. This follows by Theorem 12. □

Proposition 12 (cf. Wang et al. (2017))**.** *There is a facet-defining inequality of $P_k(G)$ with $\alpha(G)$ positive coefficients. None have more.*

Proof. The first claim holds by Lemma 20. The second claim holds by Lemma 17. $\square$

Theorem 13. *Consider a graph $G = (V, E)$. For every $S \subseteq V$ with $|S| = k + 1$ and a separator $C \subseteq (V \setminus S)$ for S,*

$$x(S) - x(C) \leq k \tag{5.3}$$

is facet-defining if and only if C is a minimal separator.

Proof. ($\Longrightarrow$) Suppose that C is not a separator. Then, there exist vertices $a, b \in S$ such that there is an a, b path P in $G - C$. Then, $G[S \cup V(P)]$ induces at most k connected components that is represented by x^*; however, we have

$$x^*(S) - x^*(C) = x^*(S) = k + 1 > k.$$

This means that inequality (5.3) is not valid. Now suppose that C is not minimal. This implies that there exists $c \in C$ such that $C' := C \setminus \{c\}$ is also a separator. So, $x(S) - x(C') \leq k$ is also valid. This means that inequality (5.3) is not facet-defining.

($\Longleftarrow$) Let C be a minimal separator, and $S := \{s_1, s_2, \dots, s_{k+1}\}$. Then define $S_1, S_2, \dots, S_{k+1}$ as follows.

$$S_1 := \{u \in V \mid u \text{ and } s_1 \text{ belong to the same component of } G - C\}$$
$$\vdots$$
$$S_{k+1} := \{u \in V \mid u \text{ and } s_{k+1} \text{ belong to the same component of } G - C\}$$
$$W := V \setminus (S_1 \cup S_2 \cup \cdots \cup S_{k+1} \cup C).$$

Let $Z := S_1 \cup S_2 \cup \cdots \cup S_{k+1}$. By Theorem 12, $x(S) \leq k$ is facet-defining for $P_k\left(G[Z]\right)$. Now, we claim that it is facet-defining for $P_k\left(G[Z \cup C]\right)$. So, we lift variables x_v for every $v \in C$. Let $C = \{v_1, v_2, \dots, v_q\}$, $C_0 = \emptyset$ and for every $j \in [q]$, let $C_j = \{v_1, v_2, \dots, v_j\}$. We show that for each $j \in \{0, 1, \dots, q\}$, the inequality $x(S) - x(C_j) \leq k$ is facet-defining for $P_k(G_j)$ with $G_j = G[Z \cup C_j]$. This is clearly true for $j = 0$. Assume that the statement holds for some $0 \leq j < q$. We show that it also holds for $j+1$. By assumption, the inequality $x(S) - x(C_j) \leq k$ is facet-defining for $P_k(G_j)$. Consider the problem of lifting variable $x_{v_{j+1}}$ into the inequality; i.e., solving

$$\zeta := \max_{T \subseteq Z \cup C_{j+1}} \left\{ x^T(S) - x^T(C_j) \;\middle|\; G[T] \text{ has at most } k \text{ components and } v_{j+1} \in T \right\}.$$

For any $x \in [0,1]^n$, we have

$$\sum_{i \in S} x_i - \sum_{u \in C_j} x_u \leq \sum_{i \in S} x_i \leq k+1. \tag{5.4}$$

Hence, $\zeta \leq k+1$. Since C is a minimal separator, there are $a, b \in S$ such that there exists a path from a to b in $G[(V \setminus C) \cup \{v_{j+1}\}]$. Let U be the vertices on this path. Then, $x^{U \cup S}$ is a feasible solution for the lifting problem and its objective value is calculated as follows.

$$\zeta \geq \sum_{i \in S} x_i^{U \cup S} - \sum_{u \in C_j} x_u^{U \cup S} = \sum_{i \in S} x_i^{U \cup S} = k+1. \tag{5.5}$$

This implies $\zeta \geq k+1$. So, $\zeta = k+1$ by inequalities (5.4) and (5.5). Finally, the inequality

$$(k - \zeta) x_{v_{j+1}} + \sum_{i \in S} x_i - \sum_{u \in C_j} x_u = \sum_{i \in S} x_i - \sum_{u \in C_{j+1}} x_u \leq k$$

induces a facet for $P_k(G_{j+1})$. Thus, the inequality $x(S) - x(C) \leq k$ induces a facet for $P_k(Z \cup C)$.

Finally we show that $x(S) - x(C) \leq k$ induces a facet for $P_k(G)$. For any $v \in W$, let $\sigma(v)$ be the length of shortest path from v to C. Order the vertices of $W = \{v_1, v_2, \dots, v_r\}$ such that $\sigma(s) \leq \sigma(t)$ for any $s, t \in [r]$ and $s \leq t$. Let $W_0 = \emptyset$; and for any $j \in [r]$, let $W_j = \{v_1, v_2, \dots, v_j\}$ and $H_j = G[(V \setminus W) \cup W_j]$. We employ induction to show that $x(S) - x(C) \leq k$ induces a facet for $P_k(H_j)$. The claim is clear for $j = 0$. Suppose that it holds for some $j \in \{0, 1, \dots, r-1\}$. We are to show that it holds for $j+1$. By the induction assumption, we know that $x(S) - x(C) \leq k$ induces a facet for H_j. Now, we define the lifting problem as follows.

$$\zeta := \max_{T \subseteq V(H_{j+1})} \left\{ x^T(S) - x^T(C) \;\middle|\; G[T] \text{ has at most } k \text{ components and } v_{j+1} \in T \right\}.$$

Let $U \subseteq V(H_{j+1})$ be a feasible solution to the lifting problem. If $x^U(S) - x^U(C) > k$, then all vertices of the set S belong to U. However, because U is a feasible solution and $G[U]$ has at most k connected components, there exists $q \in U \cap C$ and we have

$$x^U(S) - x^U(C) \leq x^U(S) - x^U_q \leq k.$$

So, we have $\zeta \leq k$.

To show the reverse inequality, let R_1 be the set of vertices in a shortest path from v_{j+1} to set C. Let $q \in C$ be the other endpoint of the path. Because R_1 forms the shortest path from v_{j+1} to set C, we have $R_1 \cap C = \{q\}$. Also, because C is minimal, there exists vertices $a, b \in S$ such that there is a path from a to b in $G[(V \setminus C) \cup \{q\}]$. Let $R_2 \subseteq V$ be the vertices on this path, and let $R = R_1 \cup R_2 \cup S$. Then, $G[R]$ induces exactly k connected components that contains v_{j+1}, a, b and $R \cap C = \{q\}$. Hence, x^R is a feasible solution for the lifting problem, and

$$x^R(S) - x^R(C) = x^R(S) - x_q^R = k.$$

Thus, $\zeta \geq k$, and this implies $\zeta = k$. So, the inequality $x(S) - x(C) \leq k$ induces a facet for $P_k(H_{j+1})$. Hence, it induces a facet for $P_k(H_r)$. $\square$

Lifting separator inequalities. Assume that $x(S) - x(C) \leq k$ induces a facet for $P_k(G-v)$. If we want to lift in vertex v such that the inequality is facet-defining for $P_k(G)$, then next theorem proves that the following algorithm can be run in linear time.

1. let $S_j = \{i \in V \mid i \text{ and } s_j \text{ belong to the same component of } G-C\}$ for every $j \in [k+1]$;
2. if $N(v) = \emptyset$, then return $\zeta = k - 1$;
3. if there exists at least two components S_i and S_j such that $S_i = S_j$, then return $\zeta = k + 1$;
4. if $S_1 \neq S_2 \neq \ldots \neq S_{k+1}$, then return $\zeta = k$.

Theorem 14. *The lifting algorithm lifts vertex v into a given separator inequality in linear time.*

Proof. The algorithm runs in linear time (based on the number of edges) because step one runs in linear time via BFS. If $N(v) = \emptyset$, then $\zeta = k - 1$ by Lemmata 15 and 16. Now, suppose that $N(v) \neq \emptyset$. The following two cases can happen.

1. Vertex v belongs to at least two components S_i and S_j. This means at least two components S_i and S_j are the same. Let P be a path between s_i and s_j in $G - C$. Then $U := S \cup V(P)$ maximizes ζ. So, we have

$$x^U(S) - x^U(C) \leq x^U(S) - 0 = k + 1.$$

This implies that $\zeta \leq k+1$. Also, because U is a feasible solution with objective value $k+1$ for the maximization problem, $\zeta \geq k+1$. Hence, $\zeta = k+1$.

2. Vertex v belongs to none of the components $S_1, S_2, \ldots, S_{k+1}$. This implies that C is a separator for S. Also, C is minimal because the separator inequality is facet-defining for $P_k(G-v)$. For contradiction purposes, suppose that there is a solution $U \subset V$ such that $x^U(S) - x^U(C) > k$ and $G[U]$ induces at most k connected components. Because C is minimal, there exists a vertex $c \in C$ such that there are two components S_i and S_j for which there is a path from s_i to s_j in $G[(V \setminus C) \cup \{c\}]$. This means U has a connected component containing s_i, s_j, c and v along with at most $k-1$ other components that contain s_u for every $u \in S \setminus \{i,j\}$. So , we have

$$x^U(S) - x^U(C) \leq x^U(S) - x^U_c = k.$$

This is a contradiction, and it implies that $\zeta \leq k$. Also, U is a feasible solution for the maximization problem; so, $\zeta \geq k$. Hence, $\zeta = k$.

□

5.3.1 Separation Complexity

Since there are exponentially many of the separator inequalities (5.2), we should use them carefully and adding them on-the-fly only as needed. That is, while solving our problem, we will encounter some possible solution $x^* \in \mathbb{R}^n$ and should determine whether x^* satisfies all separator inequalities. And if not, we are to provide a violated separator inequality.

Integer separation

To simplify the process, we may choose to separate only *integer* points $x^* \in \{0,1\}^n$. In this case, x^* will represent some vertex subset $S \subseteq V$. Let $S_1, S_2, \ldots, S_{k+1}$ be $k+1$ connected components that are induced by S. An integer separation procedure is provided in Algorithm 4.

Algorithm 4 IntegerSeparation(G, S)

```
 1: compute the connected components $G_1$, $G_2$, ..., $G_q$ of $G[S]$
 2: if $q \leq k$ then
 3:     return "none violated"
 4: $C \leftarrow \emptyset$
 5: let $T := \{v_i \mid i \in [k+1]\}$
 6: let $S' \leftarrow V_1 \cup V_2 \cup \cdots \cup V_{k+1}$
 7: while $S' \cup C \neq V$ do
 8:     let $C' := \{v \in (V \setminus S') \mid N(v) \cap S' \neq \emptyset\}$
 9:     for every $c \in C'$ do
10:         if $c$ neighbors only one component $V_i$ then
11:             $V_i \leftarrow V_i \cup \{c\}$
12:         else $C \leftarrow C \cup \{c\}$
13:     $S' \leftarrow V_1 \cup V_2 \cup \cdots \cup V_{k+1}$
14: return $x(T) - x(C) \leq k$
```

Proposition 13. *Algorithm 4 identifies a minimal violated separator inequality (when one exists) and can be implemented to run in time $O(m + na(n))$, where $a(.)$ denotes the inverse of Ackermann function.*

Proof. Minimality of the separator follows by the fact that each $c \in C$ neighbors at least two components. For time complexity, note that step one runs in $O(m)$ and steps 4–13 run in $O(na(n))$ (see Cormen et al. (2009)). □

Fractional separation

In this section, we prove that fractional separation is NP-hard for separator inequalities. To prove this, we first introduce Multiterminal Vertex Separator Problem as follows.

Problem: Multiterminal Vertex Separator.

Input: a simple graph $G = (V, E)$, a terminal set T and integer L.

Question: Is there a subset $C' \subseteq V$ such that $G - C'$ induces $|T|$ components ($|T| \geq 3$) each containing only one vertex from T and $|C'| < L$?

Problem: Separation Problem for Separator Inequalities.

Input: a simple graph $G = (V, E)$, a weight $x_v^* \in [0, 1]$ for each $v \in V$, and integer k.

Output: (if any exist) a vertex subset S of size $k + 1$ and an S-separator C such that $\sum_{i \in C} x_i^* < \sum_{i \in S} x_i^* - k$.

Theorem 15. *The separation problem is NP-complete for every $k \geq 2$.*

Proof. Membership in NP is obvious. The reduction is from an instance of Multiterminal Vertex Separator Problem which is NP-complete for $|T| \geq 3$ (Cornaz et al., 2019). Let $S := T$, $k := |T| - 1$, $x_i^* := 1$ for every $i \in S$, and $x_i^* := \frac{1}{L}$ for every $i \in V \setminus S$. We argue that (G, T, L) is a "yes" instance of Multiterminal Vertex Separator Problem if and only if (G, S, x^*, k) violates a separator inequality. Suppose that there is a violated separator inequality for some $C \subseteq V$. Then, $\frac{|C|}{L} = \sum_{i \in C} x_i^* < \sum_{i \in S} x_i^* - k = |S| - (|S| - 1) = 1$, i.e., $|C| < L$, and the instance of Multiterminal Vertex Separator Problem is a "yes". Finally, if there is a separator $C' \subseteq V$ such that $|C'| < L$ (or equivalently $\frac{|C'|}{L} < 1$), then $\sum_{i \in C'} x_i^* = \frac{|C'|}{L} < 1 = |T| - (|T| - 1) = \sum_{i \in T} x_i^* - k$. Hence, x^* violates the separator inequality. $\square$

A strong extended formulation for the separator inequalities

Let $\mathcal{I}$ be the family of all independent sets of size $k+1$ in graph G. Then, the polytope $F_k(G)$ is the set of all (x, f) that satisfies the following constraints. This formulation has a size of $O\left(\binom{n}{k+1}\binom{k+1}{2}2m\right)$.

$$\sum_{a,b\in U:a<b} f^{U,ab}(\delta^+(i)) \leq x_i \qquad \forall i \in V,\ \forall U \in \mathcal{I}$$

$$x(U) - \sum_{a,b\in U:a<b} \left(f^{U,ab}(\delta^+(a)) - f^{U,ab}(\delta^-(a))\right) \leq k \qquad \forall U \in \mathcal{I}$$

$$f^{U,ab}(\delta^+(i)) - f^{U,ab}(\delta^-(i)) = 0 \qquad \forall U \in \mathcal{I},\ \forall a,b \in U,\ a<b,\ \forall i \in V \setminus \{a,b\}$$

$$0 \leq x_i \leq 1 \qquad \forall i \in V$$

$$0 \leq f_{ij}^{U,ab} \leq 1 \qquad \forall (i,j) \in A,\ \forall U \in \mathcal{I},\ \forall a,b \in U,\ a<b.$$

Lemma 21. *$\operatorname{proj}_x(F_k(G)) \subseteq Q_k(G)$.*

Proof. Let $(\hat{x}, \hat{f}) \in F_k(G)$. Consider an arbitrary $U = \{u_1, u_2, \ldots, u_{k+1}\} \in \mathcal{I}$ and a corresponding separator $C \subseteq V \setminus U$. Without loss of generality, suppose that $U = \{1, 2, \ldots, k+1\}$. Let $U_1, U_2, \ldots, U_{k+1}$ be sets of vertices reachable from vertices $1, 2, \ldots, k+1$ in $G - C$, respectively. For every $a \in U$, let $R_a = V \setminus S_a$ with $S_a := U_a \cup C$. Also, for every $a, b \in U$

and $(i,j) \notin A$, let $\hat{f}_{ij}^{U,ab} = 0$. Then we have

$$\begin{aligned}
\hat{x}(U) - k &\le \sum_{a,b \in U : a<b} \left(\sum_{j \in V} \hat{f}_{aj}^{U,ab} - \sum_{j \in V} \hat{f}_{ja}^{U,ab} \right) \\
&= \sum_{a,b \in U : a<b} \left(\sum_{i \in S_a} \left(\sum_{j \in V} \hat{f}_{ij}^{U,ab} - \sum_{j \in V} \hat{f}_{ji}^{U,ab} \right) \right) \\
&= \sum_{a,b \in U : a<b} \left(\sum_{i \in S_a} \sum_{j \in S_a} (\hat{f}_{ij}^{U,ab} - \hat{f}_{ji}^{U,ab}) \right) + \sum_{a,b \in U : a<b} \left(\sum_{i \in S_a} \sum_{j \in R_a} (\hat{f}_{ij}^{U,ab} - \hat{f}_{ji}^{U,ab}) \right) \\
&= 0 + \sum_{a,b \in U : a<b} \left(\sum_{i \in S_a} \sum_{j \in R_a} (\hat{f}_{ij}^{U,ab} - \hat{f}_{ji}^{U,ab}) \right) \\
&= \sum_{a,b \in U : a<b} \left(\sum_{i \in C} \sum_{j \in R_a} (\hat{f}_{ij}^{U,ab} - \hat{f}_{ji}^{U,ab}) \right) \\
&\le \sum_{a,b \in U : a<b} \left(\sum_{i \in C} \sum_{j \in R_a} \hat{f}_{ij}^{U,ab} \right) \\
&= \sum_{i \in C} \left(\sum_{a,b \in U : a<b} \sum_{j \in V} \hat{f}_{ij}^{U,ab} \right) \\
&= \sum_{i \in C} \left(\sum_{a,b \in U : a<b} \hat{f}^{U,ab}(\delta^+(i)) \right) \\
&\le \sum_{i \in C} \hat{x}_i.
\end{aligned}$$

□

Lemma 22. *Suppose that $\sum_{i \in V} \pi_i x_i \le k\pi_0$ induces a facet for $P_k(G)$ and let $S := \{i \in V \mid \pi_i > 0\}$. If $|S| = k+1$, then $\pi_v = \pi_0$ for every $v \in S$.*

Proof. First, we claim $\pi(S) = (k+1)\pi_0$. For every $v \in S$, the inequality $\pi(S \setminus \{v\}) \le k\pi_0$ holds by validity of $\sum_{i \in V} \pi_i x_i \le k\pi_0$ for $P_k(G)$. We claim that $\pi(S \setminus \{v\}) = k\pi_0$ holds for every $v \in S$. Suppose not. Then, there exists $u \in S$ such that $\pi(S \setminus \{u\}) < k\pi_0$. This implies that any point that satisfy the inequality $\sum_{i \in V} \pi_i x_i \le k\pi_0$ at equality has $x_u^* = 1$. This means the facet-defining inequality $\sum_{i \in V} \pi_i x_i \le k\pi_0$ defines the same face as the inequality

$x_u \leq 1$ defines. This is a contradiction. Hence, for every $v \in S$, we have $\pi\left(S \setminus \{v\}\right) = k\pi_0$. This implies that

$$\sum_{v \in S} \pi\left(S \setminus \{v\}\right) = k\pi(S) = (k+1)k\pi_0.$$

This means $\pi(S) = (k+1)\pi_0$. Because $\pi\left(S \setminus \{v\}\right) = k\pi_0$ holds for every $v \in S$,

$$\pi_v = \pi(S) - \pi(S \setminus \{v\}) = (k+1)\pi_0 - k\pi_0 = \pi_0.$$

$\square$

Lemma 23. *If a facet-defining inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ for $P_k(G)$ has exactly $k+1$ positive coefficients, then it is a separator inequality.*

Proof. Let $\pi_{u_1}, \pi_{u_2}, \ldots, \pi_{u_{k+1}}$ be positive coefficients and define $S := \{u_1, u_2, \ldots, u_{k+1}\}$. By Lemma 22, $\pi_{u_1} = \pi_{u_2} = \cdots = \pi_{u_{k+1}} = \pi_0$. Now we define the following sets.

$$C = \{i \in V \mid \pi_i = -\pi_0\}$$

$$T = \{i \in V \mid -\pi_0 < \pi_i < 0\}$$

$$R = \{i \in V \mid \pi_i < -\pi_0\}.$$

First, we claim that $R = \emptyset$. Suppose not, then there is a vertex $v \in R$. Because $\pi_v < -\pi_0$, there is no point that contains v and satisfies the facet-defining inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ at equality. This contradicts the fact that $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ is a facet-defining inequality. Thus, $R = \emptyset$. So, we can write the facet-defining inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ as follows.

$$\sum_{i \in S} \pi_0 x_i - \sum_{i \in C} \pi_0 x_i + \sum_{i \in T} \pi_i x_i \leq k\pi_0. \tag{5.7}$$

We show that $C \cup T$ must be a separator. Suppose not, then there are two vertices in $S \subseteq V$

such that there is a path P between them in $G-(C \cup T)$ and $P \cup S$ induces at most k connected components. This means $\sum_{i \in V} \pi_i x_i^{S \cup P} \leq k\pi_0$; however, we find the following contradiction.

$$(k+1)\pi_0 = \sum_{i \in S} \pi_i = \sum_{i \in V} \pi_i x_i^{S \cup P} \leq k\pi_0.$$

Because inequality (5.7) is a facet-defining inequality, $C \cup T$ is a minimal separator.

Now we claim $T = \emptyset$. Suppose not. Because $C \cup T$ is a minimal separator, there exists a vertex $v \in T$ such that v belongs to the common neighbor set of at least two components of $G_1, G_2, \ldots, G_{k+1}$ in $G-(C \cup T)$. Define $U := V(G_1) \cup V(G_2) \cup \cdots \cup V(G_{k+1}) \cup \{v\}$. Then $G[U]$ induces at most k connected components; however,

$$\sum_{i \in V} \pi_i x_i^U = \sum_{i \in S} \pi_0 + \pi_v > (k+1)\pi_0 - \pi_0 = k\pi_0.$$

This contradicts the validity of $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ for $P_k(G)$. Thus, $T = \emptyset$ and inequality (5.7) is a separator inequality. $\square$

Theorem 16. *The separator inequalities $x(S) - x(C) \leq k$ and 0-1 bounds provide a perfect formulation of $P_k(G)$ if and only if $\alpha(G) \leq k+1$.*

Proof. ($\Longrightarrow$) By contrapositive, suppose that $\alpha(G) > k+1$. This means there exists an independent set $I \subseteq V$ with $|I| > k+1$ such that $x(I) \leq k$ induces a facet for $G[I]$ (by Lemma 20). By Proposition 12, there exists a facet-defining separator inequality with more than $k+1$ positive coefficients. Hence, the separator inequalities, with $|S| = k+1$, do not form all the facet-defining inequalities.

($\Longleftarrow$) Suppose that $\alpha(G) \leq k+1$. Consider an arbitrary facet-defining inequality $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ and let U be the set of variables with positive coefficients. By Proposition 12 and the assumption, we have $|U| \leq \alpha(G) \leq k+1$. We consider the following cases: (i)

$|U| \leq k$; and (ii) $|U| = k + 1$. In the first case, the separator inequalities do not induce non-trivial facets by Lemma 18. In the second case, the facet-defining inequality is a separator inequality by Lemma 23. □

Corollary 5. *For any fixed k if $\alpha(G) \leq k+1$, then the problem of finding a maximum weight subgraph that induces at most k connected components is solvable in polynomial time.*

Proof. By Theorem 16, separator inequalities $x(S) - x(C) \leq k$ and 0-1 bounds provide a perfect formulation of $P_k(G)$; however, its separation problem is hard for $k \geq 2$ by Theorem 15. This does not mean that we cannot solve LP relaxation in polynomial time because polytope $F_k(G)$ and 0-1 bounds provides a stronger formulation with polynomial size for any fixed k. □

5.4 Indegree Inequalities

For any direction d of edges E in graph G, we have the following valid indegree inequalities for imposing at most k connected components. Also, see Gröflin and Liebling (1981); Wang et al. (2017); Korte et al. (2012); Bley et al. (2017) for indegree inequalities when $k = 1$.

$$\sum_{i \in V} (1 - d_i) x_i \leq k.$$

In this section, we study the polytope of indegree inequalities that is defined as follows.

$$R_k(G) := \{x \in [0,1]^n \mid x \text{ satisfies all indegree inequalities.}\}$$

Lemma 24. *The indegree inequalities are valid for $P_k(G)$ for arbitrary graph G.*

Proof. Suppose that $S \subseteq V$ induces $q \leq k$ connected components. This means that the number edges with both endpoints in S are at least $|S| - q$. So, for any indegree vector

$d \in \mathbb{Z}_+^n$, we have $\sum_{i \in S} d_i \geq |S| - q$. This implies

$$\sum_{i \in V} (1 - d_i) x_i^S = |S| - \sum_{i \in S} d_i \leq |S| - (|S| - q) = q \leq k.$$

□

Lemma 25. *Suppose that $S \subseteq V$ induces at most k connected components of G. Then, S is a feasible tight set for an indegree inequality if and only if*

1. *S induces a forest of k trees in G; and*
2. *each edge of E having exactly one endpoint in S is oriented out of S.*

Proof. ($\Longleftarrow$) Because of the first condition, the number of edges with both endpoints in S is exactly $|S| - k$. Also because of the second condition, we have $\sum_{i \in S} d_i = |S| - k$. Hence,

$$\sum_{i \in V} x_i^S - \sum_{i \in V} d_i x_i^S = |S| - (|S| - k) = k.$$

($\Longrightarrow$) It is easy to show that if S induces less than k trees, or if one of the edges directed towards S, then $\sum_{i \in V} x_i^S - \sum_{i \in V} d_i x_i^S$ is at most $k - 1$ which implies that S is not tight. □

Lemma 26. *If the number of zero-degree vertices of an orientation $D = (V, A)$ is less than or equal to k, then the corresponding indegree inequality induces a trivial facet of $P_k(G)$.*

Proof. If the number of zero-degree vertices of an orientation D is less than or equal to k, then the number of variables with positive coefficients is also less than or equal to k. By Lemma 18, the indegree inequality induces a trivial facet of $P_k(G)$. □

Lemma 27. *If there are two directed $s - t$ walks in an orientation $D = (V, A)$ of G, then the corresponding indegree inequality does not induce a facet of $P_k(G)$.*

Proof. For the indegree inequalities to induce a facet, there must be a feasible tight set S containing t. We argue that the vertices along the two $s-t$ walks must belong to S. Suppose not. Then there is at least one directed edge that is not directed out of S. By the second condition of Lemma 25, S is not tight. Hence, all vertices on these two $s-t$ walks belong to S. However, this implies that S induces a cycle in G and this is in contradiction with the first condition of Lemma 25. Thus, no such S exists and indegree inequality cannot induce a facet. $\square$

Lemma 28. *Consider an orientation $D = (V, A)$ of graph G with q components and $q \leq k$. Also let Z be the set of zero-degree vertices and $z := |Z| \geq k+1$. If there is a vertex $v \in V \setminus Z$ that is reachable by at least $z-k+2$ of zero-degree vertices, then the corresponding indegree inequality does not induce a facet of $P_k(G)$.*

Proof. Suppose there is a vertex $v \in V \setminus Z$ that is reachable by at least $z-k+2$ of zero-degree vertices. By second property of Lemma 25, any tight point S that includes vertex v must contain at least $z-k+2$ zero-indegree vertices. This means that we can construct at most $k-1$ trees when vertex v is chosen. Then by the first condition of Lemma 25, there is no feasible tight set in orientation D that contains vertex v and satisfies the inequality at equality. $\square$

Theorem 17. *The indegree inequality corresponding to an orientation $D = (V, A)$ of G induces a facet of $P_k(G)$ if and only if*

1. *the set of zero-indegree vertices (set Z) has a size of z, where $z \geq k+1$,*

2. *for every $u, v \in V$, there is at most one directed $u-v$ walk in D; and*

3. *for every $v \in V \setminus Z$, there are at most $z-k+1$ zero-degree vertices that can find a directed path toward v.*

Proof. ($\Longrightarrow$) This holds by Lemmata 26–28.

($\Longleftarrow$) Because $z \geq k+1$, seed inequality $\sum_{j \in Z} x_j \leq k$ is facet-defining for $P_k(G[Z])$ by Lemma 20.

Now, we claim that the inequality is facet-defining for $P_k(G)$. In this step, we employ the topological ordering of all vertices in set $V \setminus Z$ for lifting purposes. Consider the lifting problem of vertex v in the topological ordering. Define D_v as follows.

$$D_v := \{v\} \cup \{u \in V \mid \textit{there is a directed } u - v \textit{ path in } D\}.$$

We claim that $G[D_v]$ is a tree. Suppose not. Then there is an undirected cycle (V', E') in $G[D_v]$ such that each vertex $u \in V'$ can find a directed path toward vertex v in the induced subgraph $G[D_v]$. This implies that there exist a pair of vertices $a, b \in V'$ such that there are two directed $a - b$ walks in (V', E'). This means there are at least two directed $a - b$ walks in $G[D_v]$. This contradicts the second assumption.

Let z_v be the number of zero-degree vertices that find a path toward vertex v. By the third assumption, $z - z_v \geq k - 1$. This implies that one can form a set of zero-indegree vertices S_v such that it contains exactly $k-1$ zero-indegree vertices and none of them belongs to set D_v.

Finally, we define forest $F_v := S_v \cup D_v$ with exactly k components. Hence, we have a lower bound for ζ as follows.

$$\begin{aligned}\zeta \geq \sum_{i \in F_v \setminus \{v\}} (1 - d_i) &= (|F_v| - 1) - (|E(G[F_v])| - d_v) \\ &= (|F_v| - 1) - (|F_v| - k - d_v) = (k-1) + d_v.\end{aligned}$$

By Lemma 15, we have $\zeta \leq (k-1) + \alpha(G[N(v) \cap V(F_v)]) = (k-1) + d_v$. This finishes the proof. $\square$

5.4.1 Lifting indegree inequalities

Theorem 18 (Wang et al. (2017)). *Lifting a vertex into an indegree inequality is strongly NP-hard. This holds even when the graph is bipartite and 2-degenerate.*

5.4.2 Separating indegree inequalities

Theorem 19 (Wang et al. (2017)). *There is a linear time algorithm that finds most-violated indegree inequality.*

Proof. See section 4.3 of Wang et al. (2017) for the algorithm and its complexity proof. □

5.4.3 A linear-size extended formulation for the indegree inequalities

In this section, we provide an extended formulation of size $O(m+n)$ for the polytope of the indegree inequalities and non-negativity bounds.

$$y_e - x_v \leq 0 \quad \textit{and} \quad y_e - x_u \leq 0 \qquad \forall e = \{u,v\} \in E \tag{5.8}$$

$$\sum_{i \in V} x_i - \sum_{e \in E} y_e \leq k \tag{5.9}$$

$$y_e \geq 0 \qquad \forall e \in E \tag{5.10}$$

$$x_i \leq 1 \qquad \forall i \in V. \tag{5.11}$$

Theorem 20. *The polytope of indegree inequalities and non-negativity bounds admits an extended formulation of size $O(m+n)$.*

Proof. The proof is similar to the proof of Theorem 9 in Wang et al. (2017). □

Lemma 29. *Let $G = (V,E)$ be a simple graph and k be a positive integer with $k \geq 2$. Suppose that every independent set S with $|S| \geq k+1$ hits every cycle at least twice and satisfies*

$T \subseteq S$, where $T := \{v \in V \mid \deg_G(v) > 2\}$. If a feasible set U is tight for a facet-defining inequality $\sum_{i \in V} \pi_i x_i \leq k$, then

1. *$G[U]$ has exactly k components, and*

2. *$G[U]$ is a forest.*

Proof. First, we claim that no component of $G[U]$ is a cycle. Let p be the number of positive coefficients of the facet-defining inequality. Because the facet-defining inequality is non-trivial, $k + 1 \leq p$ by Lemma 18. Also, $p \leq \alpha(G)$ by Proposition 12. So, $k + 1 \leq \alpha(G)$. For contradiction purposes, suppose that $G[U]$ contains a cycle $\hat{C}$. Because every independent set with size of at least $k + 1$ hits every cycle at least twice, $\alpha(\hat{C}) \geq 2$. Because the facet-defining inequality has at least $k + 1$ positive coefficients and no two adjacent vertices have positive coefficients by Lemma 17, there is a non-trivial facet-defining inequality in which the coefficients of two vertices of $\hat{C}$ is positive. By Lemma 17, $\hat{C}$ contains at least two vertices, say w and w', with negative coefficient. So, $\pi(U \setminus \{w\}) > k$ while the number of connected components does not change. This is a contradiction.

Finally, we show that $G[U]$ has exactly k components. Note that $G[U]$ has at most k components by correctness of $P_k(G)$. So, it suffices to show that U induces at least k components. For contradiction purposes, suppose that U induces q components with $q < k$. We proved that no component of $G[U]$ is a cycle. So, each component is a tree. For any component W, none of its endpoints has negative coefficient; otherwise, one can remove it and the facet-defining inequality is violated while the number of components does not change. Without loss of generality, we can reduce each component W to W' such that endpoints of W' are positive. Let U' be the union of W' components. Then, $G[U']$ also has q components with $q < k$.

If $G[U']$ contains q isolated vertices, then because the facet-defining inequality is non-trivial and contains at least $k + 1$ positive coefficients, we can add another vertex, say u with

$\pi_u > 0$, to set U' and keep number of connected components less than or equal to k. However, $\pi(U' \cup \{u\}) > k$. This is a contradiction. If $G[U']$ contains a component with two positive endpoints, then by Lemma 17 there exists a vertex w with $\pi_w < 0$ between the endpoints. Because every vertex $v \in V$ with $\deg_G(v) > 2$ belongs to any independent set of size at least $k+1$ and the facet-defining inequality has at least $k+1$ positive coefficients, the coefficient of vertex v is always positive by Lemma 17. So, $w \neq v$ and $\deg_G(w) \leq 2$. Hence we can remove vertex w and keep number of connected components less than or equal to k. However, $\pi(U' \setminus \{w\}) > k$. This is a contradiction. □

Lemma 30. *Let $G = (V, E)$ be a simple graph and k be a positive integer with $k \geq 2$. Suppose that every independent set S with $|S| \geq k+1$ hits every cycle at least twice and satisfies $T \subseteq S$, where $T := \{v \in V \mid \deg_G(v) > 2\}$. Also, suppose $\sum_{i \in V} \pi_i x_i \leq k$ defines a non-trivial facet for $P_k(G)$. For every edge $e = \{i, j\} \in E$, either*

1. *$i \in S$ or $j \notin S$ for every feasible tight set S, or*

2. *$i \notin S$ or $j \in S$ for every feasible tight set S.*

Proof. For contradiction purposes, suppose that there is an edge $\hat{e} = \{u, v\}$ for which there exists (i) a feasible tight set S_1 with $u \notin S_1$ and $v \in S_1$ and (ii) a feasible tight set S_2 with $u \in S_2$ and $v \notin S_2$. First, note that π_u and π_v are non-positive; otherwise, one can add u to S_1 (or v to S_2) and this violates the validity of $\sum_{i \in V} \pi_i x_i \leq k$. Now, we claim that $\pi_u = \pi_v = 0$. Otherwise, one can remove them without changing the number of connected components; however, this violates the validity of $\sum_{i \in V} \pi_i x_i \leq k$. Trace a path from v inside S_1 until it hits a vertex v' of positive weight. Let v'' be the last vertex of this path with non-positive coefficient. None of the interior vertices on this $v - v''$ path can have negative weight. Otherwise, one can remove the negative tail while the number of components does not change; however, this violates the validity of $\sum_{i \in V} \pi_i x_i \leq k$. Similarly, trace a path from u inside S_2 until it hits a vertex u' of positive weight. Let u'' be the last vertex of this path with non-positive coefficient. Similar to the previous argument, we claim that none of the interior vertices on this $u - u''$ path can have negative weight. However, this is in contradiction with Lemma 17.

Finally, it suffices to show that u' and v' are distinct vertices. It is clear when graph G is a tree. If graph G contains a cycle, then the cycle has at least two distinct vertices with positive coefficients because (i) every independent set of size at least $k+1$ hits the cycle at least twice; (ii) the non-trivial facet-defining inequality has at least $k+1$ positive coefficients by Lemma 18; and (iii) no pair of adjacent vertices has positive coefficients by Lemma 17. □

Proposition 14. *$P_k(G) = [0,1]^n$ if and only if $\alpha(G) \leq k$.*

Proof. ($\Longrightarrow$) By contrapositive, suppose $\alpha(G) > k$. Then, let S be an independent set of size $k+1$. Then $x(S) \leq k$ induces a facet of $P_k(G[S])$ by Lemma 20. This seed inequality can be lifted to get a facet-defining inequality of $P_k(G)$ which is not a scalar multiple of a 0-1 bound.

($\Longleftarrow$) Let S be any subset of vertices. Because $\alpha(G) \leq k$, subgraph $G[S]$ induces at most k components. So, $x^S \in P_k(G)$ for any $S \subseteq V$. This implies $P_k(G) = \text{conv.hull}(\{0,1\}^n) = [0,1]^n$. □

Theorem 21. *Let $G = (V, E)$ be a simple graph and k be a positive integer with $k \geq 2$. The indegree inequalities and 0-1 bounds provide a perfect formulation for $P_k(G)$ if and only if every independent set S with $|S| \geq k+1$ hits every cycle at least twice and satisfies $T \subseteq S$, where $T := \{v \in V \mid \deg_G(v) > 2\}$.*

Proof. ($\Longrightarrow$) First, if $P_k(G)$ contains only 0-1 bounds, then $\alpha(G) \leq k$ by Proposition 14.

Now by contrapositive, suppose that (i) $\alpha(G) \geq k+1$; and (ii) there is an independent set $\hat{S}$ with $|\hat{S}| \geq k+1$ such that either of the followings occurs:

1. S hits a cycle $\hat{C}$ at most once; or

2. there is a vertex $v \in (V \setminus \hat{S})$ such that $\deg_G(v) > 2$.

Consider the first case. Then there is no facet-defining inequality by the second condition of Theorem 17.

Now consider the second case. Then we have $\alpha(G - v) \geq k + 1$. Let u_1, u_2, and u_3 be three neighbors of vertex v. Also define $U := \{v, u_1, u_2, u_3\}$. Then we have the following cases:

1. $G[U]$ contains a cycle C_3. Then no orientation $D = (V, A)$ of G can define a facet-defining indegree inequality by the second condition of Theorem 17.

2. $G[U]$ contains no cycle. Because $\alpha(G - v) \geq k + 1$, there exists an independent set I of size $k + 1$ such that $(U \setminus \{v\}) \subseteq I$. Let $\hat{C}$ be a minimal I-separator. Then, $v \in \hat{C}$ and the separator inequality $X(I) - x(\hat{C}) \leq k$ is facet-defining for $P_k(G)$. However, no orientation $\hat{D}$ of G can make this facet-defining inequality.

($\Longleftarrow$) First, suppose that $\alpha(G) \leq k$. Then by Proposition 14, 0-1 bounds provide a perfect formulation for $P_k(G)$.

Now suppose $\alpha(G) \geq k + 1$ and the second condition holds. Let $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ be a non-trivial facet-defining inequality in the full description of $P_k(G)$. By Lemma 18, the number of variables with positive coefficients must be greater than k in the facet-defining inequality. Because $\mathbf{0} \in P_k(G)$, we have $\pi_0 \geq 0$. Because $\sum_{i \in V} \pi_i x_i \leq k\pi_0$ is a non-trivial facet-defining inequality, $\pi_0 > 0$. So, we can divide both sides of the inequality by π_0. Hence, $\sum_{i \in V} \pi'_i x_i \leq k$ is also facet-defining with $\pi'_i := \frac{\pi_i}{\pi_0}$ for every $i \in V$.

By Lemma 30, for every edge $e \in E$, at most one of its endpoints is included in a feasible tight set among all existing feasible tight sets. For every edge $e = \{u, v\} \in E$, orient it from u to v if there is a feasible tight set S with $u \in S$ and $v \notin S$. Let d be a corresponding vector of this orientation. Then for any feasible tight set S, we have

$$\sum_{i\in S}(1-d_i) = |S| - \sum_{i\in S} d_i = |S| - (|S|-k) = k.$$

Here, the second equation holds because S induces a forest of k trees by Lemma 29. Hence, every feasible tight point of $\sum_{i\in V}\pi'_i x_i = k$ lies on the face of the above indegree inequality. So, $\sum_{i\in V}\pi'_i x_i = k$ is the indegree inequality. $\square$

5.5 Conclusion

In this chapter, we studied two polytopes for finding at most k connected components: (i) separator inequalities, and (ii) indegree inequalities. Although both of these polytopes are studied for $k=1$ (Wang et al., 2017), we considered a generalization of these polytopes for any positive integer k. We also identified when these inequalities are facet-defining. Finally, we provided full characterizations for both classes of inequalities. An interesting future work can be conducting polyhedral studies on other generalizations of connectivity polytopes.

CHAPTER VI

SUMMARY AND FUTURE WORK

In Chapter II, we propose a cut formulation (with exponentially many constraints) for the latency-constrained connected dominating set problem. In this formulation, we employ length-bounded cuts to impose (i) connectivity, and (ii) the maximum desired length of communication between any two nodes in a telecommunication network. We also introduce a ploynomial-size formulation for the problem. Our experiments show the superiority of the cut formulation over the polynomial-size one. As a future work, we are trying to find similar length-bounded cuts for some other network design problems (e.g., network design problem with relays) in the vertex-and-edge space.

In Chapter III, we employ $\mathcal{P}_{\text{HESS}}$ formulation to obtain compact districts with respect to the moment-of-inertia. Because $\mathcal{P}_{\text{HESS}}$ formulation does not guarantee connectivity, we add two cut-based and two flow-based sets of constraints to $\mathcal{P}_{\text{HESS}}$ formulation for imposing connectivity. We show that the cut-based formulations are at least as strong as the flow-based ones. As a future work, we are working on formulations that (i) are smaller than Hess in number of variables, and (ii) employ other compactness objective functions (e.g., cut-edges or Polsby-Popper).

In Chapter IV, we identify an error in a well-known extended formulation for the minimum spanning tree problem in planar graphs. We also mention how Pashkovich's root rule (Pashkovich, 2012) can fix the formulation. As a future work, we are working on an extended formulation for the minimum spanning forest with k sub-trees, where $k \in \mathbb{Z}_+$ and $k \geq 2$. We can employ the new light extended formulation in some network design problems (e.g., political districting); however, these kinds of formulations suffer from symmetry. So, we are also working on symmetry breaking methods in edge space.

In Chapter V, we generalize separator and indegree inequalities for imposing at most k connected components. We conduct a polyhedral study on both classes of inequalities and identify when these inequalities are facet-defining. Also, we introduce conditions under which these inequalities provide perfect formulations for the polytope of k connected components. For future work, one can investigate other interesting classes of valid inequalities for imposing at most k connected components.

REFERENCES

W. T. Adler and S. S.-H. Wang. Response to Cho and Liu, "Sampling from complicated and unknown distributions: Monte Carlo and Markov chain Monte Carlo methods for redistricting". *Physica A: Statistical Mechanics and its Applications*, 516:591–593, 2019.

H. Ahmadi and J. R. Martí. Mathematical representation of radiality constraint in distribution system reconfiguration problem. *International Journal of Electrical Power & Energy Systems*, 64:293–299, 2015.

N. Ahn and S. Park. An optimization algorithm for the minimum k-connected m-dominating set problem in wireless sensor networks. *Wireless Networks*, 21(3):783–792, 2015.

N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177, 2006.

M. Altman. The computational complexity of automated redistricting: Is automation the answer? *Rutgers Computer & Tech. LJ*, 23:81, 1997.

M. Altman. Traditional districting principles: Judicial myths vs. reality. *Social Science History*, 22(2):159–200, 1998.

M. Altman and M. McDonald. The promise and perils of computers in redistricting. *Duke J. Const. L. & Pub. Pol'y*, 5:69, 2010.

M. Altman and M. McDonald. Redistricting by formula: An Ohio reform experiment. *American Politics Research*, 46(1):103–131, 2018.

M. Altman, M. P. McDonald, et al. BARD: Better automated redistricting. *Journal of Statistical Software*, 42(4):1–28, 2011.

E. Álvarez-Miranda and M. Sinnl. A relax-and-cut framework for large-scale maximum weight connected subgraph problems. *Computers & Operations Research*, 87:63–82, 2017.

E. Álvarez-Miranda, I. Ljubić, and P. Mutzel. The rooted maximum node-weight connected subgraph problem. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 300–315. Springer, 2013a.

E. Álvarez-Miranda, I. Ljubić, and P. Mutzel. The maximum weight connected subgraph problem. In *Facets of Combinatorial Optimization*, pages 245–270. Springer, 2013b.

D. Applegate. Personal communication, 2019.

T. S. Arrington. Redistricting in the US: A review of scholarship and plan for future research. In *The Forum*, volume 8. De Gruyter, 2010.

O. Arslan, O. E. Karaşan, A. R. Mahjoub, and H. Yaman. A branch-and-cut algorithm for the alternative fuel refueling station location problem with routing. *Transportation Science*, 53(4):1107–1125, 2019.

O. Arslan, O. Jabali, and G. Laporte. A flexible, natural formulation for the network design problem with vulnerability constraints. *INFORMS Journal on Computing*, 32(1):120–134, 2020.

C. Backes, A. Rurainski, G. Klau, O. Müller, D. Stöckel, A. Gerasch, J. Küntzer, D. Maisel, N. Ludwig, M. Hein, et al. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic Acids Research*, 40(6):e43, 2012.

G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella. Length-bounded cuts and flows. *ACM Transactions on Algorithms (TALG)*, 7(1):4, 2010.

M. Bailly-Bechet, C. Borgs, A. Braunstein, J. Chayes, A. Dagkessamanskaia, J.-M. François, and R. Zecchina. Finding undetected protein associations in cell signaling by belief propagation. *Proceedings of the National Academy of Sciences*, 108(2):882–887, 2011.

J. E. Beasley. Lagrangean heuristics for location problems. *European Journal of Operational Research*, 65(3):383–399, 1993.

A. Bley, I. Ljubić, and O. Maurer. A node-based ILP formulation for the node-weighted dominating steiner problem. *Networks*, 69(1):33–51, 2017.

E. Boehm. Gerrymandering is out of control. *Reason*, March 27, 2018. URL `https://reason.com/archives/2018/03/27/gerrymandering-is-out-of-control`.

B. Bozkaya, E. Erkut, and G. Laporte. A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research*, 144(1):12–26, 2003.

G. Braun, S. Fiorini, and S. Pokutta. Average case polyhedral complexity of the maximum stable set problem. *Mathematical Programming*, 160(1-2):407–431, 2016.

O. Briant and D. Naddef. The optimal diversity management problem. *Operations Research*, 52(4):515–526, 2004.

C. G. Broyden. The convergence of a class of double-rank minimization algorithms: 1. General considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

A. Buchanan. Personal communication, 2019.

A. Buchanan, J. S. Sung, V. Boginski, and S. Butenko. On connected dominating sets of restricted diameter. *European Journal of Operational Research*, 236(2):410–418, 2014.

A. Buchanan, J. S. Sung, S. Butenko, and E. L. Pasiliao. An integer programming approach for fault-tolerant connected dominating sets. *INFORMS Journal on Computing*, 27(1): 178–188, 2015.

C. S. Bullock III. *Redistricting: The most political activity in America*. Rowman & Littlefield Publishers, 2010.

Bureau of the Census. Census tracts for the 2020 census–final criteria. *Federal Register*, 83 (219):56277–56284, 2018.

S. Butenko, X. Cheng, C. A. Oliveira, and P. M. Pardalos. A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks. In *Recent Developments in Cooperative Control and Optimization*, pages 61–73. Springer, 2004.

E. A. Cabral, E. Erkut, G. Laporte, and R. A. Patterson. The network design problem with relays. *European Journal of Operational Research*, 180(2):834–844, 2007.

E. A. Cabral, E. Erkut, G. Laporte, and R. Patterson. Wide area telecommunication network design: application to the Alberta SuperNet. *Journal of the Operational Research Society*, 59(11):1460–1470, 2008.

F. Caro, T. Shirabe, M. Guignard, and A. Weintraub. School redistricting: Embedding GIS tools with integer programming. *Journal of the Operational Research Society*, 55(8): 836–849, 2004.

R. Carvajal, M. Constantino, M. Goycoolea, J. P. Vielma, and A. Weintraub. Imposing connectivity constraints in forest planning models. *Operations Research*, 61(4):824–836, 2013.

C.-Y. Chen and K. Grauman. Efficient activity detection with max-subgraph search. In *2012*

IEEE Conference on Computer Vision and Pattern Recognition, pages 1274–1281. IEEE, 2012.

S. Chen, I. Ljubić, and S. Raghavan. The regenerator location problem. *Networks*, 55(3): 205–220, 2010.

S. Chen, I. Ljubić, and S. Raghavan. The generalized regenerator location problem. *INFORMS Journal on Computing*, 27(2):204–220, 2015.

W. K. T. Cho and Y. Y. Liu. Sampling from complicated and unknown distributions: Monte Carlo and Markov chain Monte Carlo methods for redistricting. *Physica A: Statistical Mechanics and its Applications*, 506:170–178, 2018.

S. Chopra and M. R. Rao. The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64(1-3):209–229, 1994.

V. Cohen-Addad, P. N. Klein, and N. E. Young. Balanced centroidal power diagrams for redistricting. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 389–396. ACM, 2018.

M. Conforti, V. Kaibel, M. Walter, and S. Weltge. Subgraph polytopes and independence polytopes of count matroids. *Operations Research Letters*, 43(5):457–460, 2015.

J. Conrad, C. Gomes, W.-J. van Hoeve, A. Sabharwal, and J. Suter. Wildlife corridors as a connected subgraph problem. *Journal of Environmental Economics and Management*, 63 (1):1–18, 2012.

W. Cook. Computing in combinatorial optimization. In *Computing and Software Science*, pages 27–47. Springer, 2019.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

D. Cornaz, Y. Magnouche, A. R. Mahjoub, and S. Martin. The multi-terminal vertex separator problem: Polyhedral analysis and branch-and-cut. *Discrete Applied Mathematics*, 256:11–37, 2019.

M. S. Daskin and E. L. Tucker. The trade-off between the median and range of assigned demand in facility location models. *International Journal of Production Research*, 56(1-2): 97–119, 2018.

W. C. Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1 (1):1–17, 1991.

D. DeFord, M. Duchin, and J. Solomon. Recombination: A family of Markov chains for redistricting. *arXiv preprint arXiv:1911.05725*, 2019.

P. G. Di Cortona, C. Manzi, A. Pennisi, F. Ricca, and B. Simeone. *Evaluation and optimization of electoral systems*, volume 1 of *Monographs on Discrete Mathematics and Applications*. SIAM, 1999.

R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, fourth edition, 2010.

B. Dilkina and C. Gomes. Solving connected subgraph problems in wildlife conservation. In A. Lodi, M. Milano, and P. Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 102–116. Springer, 2010.

I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 624–633. ACM, 2014.

M. Dittrich, G. Klau, A. Rosenwald, T. Dandekar, and T. Müller. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2008.

A. Drexl and K. Haase. Fast approximation methods for sales force deployment. *Management Science*, 45(10):1307–1323, 1999.

D.-Z. Du and P.-J. Wan. *Connected Dominating Set: Theory and Applications*, volume 77 of *Springer Optimization and Its Applications*. Springer, New York, 2013.

M. Duchin. Political geometry: voting districts, "compactness," and ideas about fairness. MAA-AMS-SIAM Gerald and Judith Porter Public Lecture given at the 2018 Joint Mathematics Meetings, 2018. Video of talk available at `https://www.youtube.com/watch?v=VddL0evo7QY`.

M. Duchin. Personal communication, 2020.

M. Duchin, T. Gladkova, E. Henninger-Voss, B. Klingensmith, H. Newman, and H. Wheelen. Locating the representational baseline: Republicans in Massachusetts. *Election Law Journal: Rules, Politics, and Policy*, 18(4):388–401, 2019.

J. C. Duque, R. L. Church, and R. S. Middleton. The p-regions problem. *Geographical Analysis*, 43(1):104–126, 2011.

J. C. Duque, L. Anselin, and S. J. Rey. The max-p-regions problem. *Journal of Regional Science*, 52(3):397–419, 2012.

W. L. Eastman. *Linear programming with pattern constraints*. PhD thesis, Harvard University, 1958.

J. Edmonds and E. L. Johnson. Matching: A well-solved class of integer linear programs. In R. Guy, H. Hanani, N. Sauer, and Schönheim, editors, *Proceedings Calgary International*

Conference on Combinatorial Structures and Their Applications, Combinatorial structures and their applications, pages 89–92. Gordon and Breach, 1970.

Y. Faenza, S. Fiorini, R. Grappe, and H. R. Tiwary. Extended formulations, nonnegative factorizations, and randomized communication protocols. *Mathematical Programming*, 153 (1):75–94, 2015.

N. Fan and J.-P. Watson. Solving the connected dominating set problem and power dominating set problem by integer programming. In G. Lin, editor, *Combinatorial Optimization and Applications*, volume 7402 of *Lecture Notes in Computer Science*, pages 371–383. Springer, 2012.

U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45 (4):634–652, 1998.

B. Fifield, M. Higgins, K. Imai, and A. Tarr. A new automated redistricting simulator using Markov chain Monte Carlo. *Work. Pap., Princeton Univ., Princeton, NJ*, 2015.

S. Fiorini, T. Huynh, G. Joret, and K. Pashkovich. Smaller extended formulations for the spanning tree polytope of bounded-genus graphs. *Discrete & Computational Geometry*, 57 (3):757–761, 2017.

M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning Benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2016.

M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017a.

M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017b.

R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3): 317–322, 1970.

L. R. Ford Jr and D. R. Fulkerson. *Flows in networks*. Princeton university press, 1962.

T. Fujie. The maximum-leaf spanning tree problem: Formulations and facets. *Networks*, 43 (4):212–223, 2004.

G. Gamrath, T. Koch, S. J. Maher, D. Rehfeldt, and Y. Shinano. SCIP-Jack–a solver for STP and variants with parallelization extensions. *Mathematical Programming Computation*, 9 (2):231–296, 2017.

M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

R. S. Garfinkel and G. L. Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science*, 16(8):B–495, 1970.

B. Gendron, A. Lucena, A. S. da Cunha, and L. Simonetti. Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing*, 26(4):645–657, 2014.

S. Gentry, E. Chow, A. Massie, and D. Segev. Gerrymandering for justice: Redistricting US liver allocation. *Interfaces*, 45(5):462–480, 2015.

S. Goderbauer and J. Winandy. Political districting problem: Literature review and discussion with regard to federal elections in Germany, 2018. URL `https://www.or.rwth-aachen.de/files/research/repORt/LitSurvey_PoliticalDistricting__Goderbauer_Winandy_20181024.pdf`.

M. X. Goemans and Y.-S. Myung. A catalog of Steiner tree formulations. *Networks*, 23(1): 19–28, 1993.

R. Gopalan, S. O. Kimbrough, F. H. Murphy, and N. Quintus. The Philadelphia districting contest: Designing territories for city council based upon the 2010 census. *Interfaces*, 43 (5):477–489, 2013.

H. Gröflin and T. M. Liebling. Connected and alternating vectors: Polyhedra and algorithms. *Mathematical Programming*, 20(1):233–244, 1981.

B. Grofman. Criteria for districting: A social science perspective. *UCLA L. Rev.*, 33:77, 1985.

M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992a.

M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992b.

M. Grötschel, C. L. Monma, and M. Stoer. Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM Journal on Optimization*, 2(3):474–504, 1992c.

M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition, 1993.

M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. *Handbooks in Operations Research and Management Science*, 7:617–672, 1995.

K. Gue. Very high density storage systems. *IIE Transactions*, 38(1):79–90, 2006.

D. Guo and H. Jin. iRedistrict: Geovisual analytics for redistricting optimization. *Journal of Visual Languages & Computing*, 22(4):279–289, 2011.

M. Á. Gutiérrez-Andrade, E. A. Rincón-García, S. G. de-los Cobos-Silva, P. Lara-Velázquez, R. A. Mora-Gutiérrez, and A. Ponsich. Simulated annealing and artificial bee colony for the redistricting process in Mexico. *INFORMS Journal on Applied Analytics*, 49(3):189–200, 2019.

T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

J. G. Hebert, M. E. Vandenberg, and P. Smith. *The Realist's Guide to Redistricting: Avoiding the Legal Pitfalls*. American Bar Association, 2010.

M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1(1):6–25, 1971.

M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.

S. Hess, J. Weaver, H. Siegfeldt, J. Whelan, and P. Zitlau. Nonpartisan political redistricting by computer. *Operations Research*, 13(6):998–1006, 1965.

M. Hojati. Optimal political districting. *Computers & Operations Research*, 23(12):1147–1161, 1996.

F. K. Hwang, D. S. Richards, P. Winter, and P. Widmayer. The Steiner tree problem, Annals of Discrete Mathematics, volume 53. *ZOR-Methods and Models of Operations Research*, 41 (3):382, 1995.

R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.

R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512–530, 2001.

IOGP. Epsg dataset, 2019. `http://www.epsg.org/`.

S. Iwata, N. Kamiyama, N. Katoh, S. Kijima, and Y. Okamoto. Extended formulations for sparsity matroids. *Mathematical Programming*, 158(1-2):565–574, 2016.

N. Jafari and J. Hearne. A new method to solve the fully connected reserve network design problem. *European Journal of Operational Research*, 231(1):202–209, 2013.

N. Jafari and J. Hearne. Corrigendum to 'A new method to solve the fully connected reserve network design problem' [European Journal of Operational Research, 231 (1), 2013, 202–209]. *European Journal of Operational Research*, 2020.

N. Jafari, B. L. Nuse, C. T. Moore, B. Dilkina, and J. Hepinstall-Cymerman. Achieving full connectivity of sites in the multiperiod reserve network design problem. *Computers & Operations Research*, 81:119–127, 2017.

M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. *Combinatorial Optimization, Dimacs*, 20:111–152, 1995.

S. Kahruman-Anderoglu, A. Buchanan, S. Butenko, and O. A. Prokopyev. On provably best construction heuristics for hard combinatorial optimization problems. *Networks*, 67(3): 238–245, 2016.

V. Kaibel, J. Lee, M. Walter, and S. Weltge. Extended formulations for independence polytopes of regular matroids. *Graphs and Combinatorics*, 32(5):1931–1944, 2016.

H. Kaplan and Y. Nussbaum. Maximum flow in directed planar graphs with vertex capacities. *Algorithmica*, 61(1):174–189, 2011.

F. Kappel and A. V. Kuntsevich. An implementation of Shor's r-algorithm. *Computational Optimization and Applications*, 15(2):193–205, 2000.

H. Kerivin and A. R. Mahjoub. Design of survivable networks: A survey. *Networks: An International Journal*, 46(1):1–21, 2005.

M. Kim and N. Xiao. Contiguity-based optimization models for political redistricting problems. *International Journal of Applied Geospatial Research (IJAGR)*, 8(4):1–18, 2017.

M. J. Kim. Give-and-take heuristic model to political redistricting problems. *Spatial Information Research*, 27:539–552, 2019.

D. M. King, S. H. Jacobson, E. C. Sewell, and W. K. T. Cho. Geo-graphs: An efficient model for enforcing contiguity and hole constraints in planar graph partitioning. *Operations Research*, 60(5):1213–1228, 2012.

D. M. King, S. H. Jacobson, and E. C. Sewell. Efficient geo-graph contiguity and hole algorithms for geographic zoning and dynamic plane graph partitioning. *Mathematical Programming*, 149(1-2):425–457, 2015.

D. M. King, S. H. Jacobson, and E. C. Sewell. The geo-graph in practice: creating United States congressional districts from census blocks. *Computational Optimization and Applications*, 69(1):25–49, 2018.

Y. Kong, Y. Zhu, and Y. Wang. A center-based modeling approach to solve the districting problem. *International Journal of Geographical Information Science*, 33(2):368–384, 2019.

B. Korte, L. Lovász, and R. Schrader. *Greedoids*, volume 4. Springer Science & Business Media, 2012.

M. S. Krishnamoorthy and B. Krishnamurthy. Fault diameter of interconnection networks. *Computers & Mathematics with Applications*, 13(5):577–582, 1987.

J. Lacki, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Single source–all sinks max flows in planar digraphs. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 599–608. IEEE, 2012.

A. Land and A. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, 28:497–520, 1960.

M. Leitner, I. Ljubić, M. Luipersbeck, and M. Sinnl. A dual ascent-based branch-and-bound framework for the prize-collecting steiner tree and related problems. *INFORMS Journal on Computing*, 30(2):402–420, 2018.

M. Leitner, I. Ljubić, M. Riedler, and M. Ruthmair. Exact approaches for network design problems with relays. *INFORMS Journal on Computing*, 31(1):171–192, 2019.

H. A. Levin and S. A. Friedler. Automated congressional redistricting. *Journal of Experimental Algorithmics (JEA)*, 24(1):1–10, 2019.

D. Li, H. Du, P.-J. Wan, X. Gao, Z. Zhang, and W. Wu. Construction of strongly connected dominating sets in asymmetric multihop wireless networks. *Theoretical Computer Science*, 410(8-10):661–669, 2009.

X. Li and Y. P. Aneja. Regenerator location problem: Polyhedral study and effective branch-and-cut algorithms. *European Journal of Operational Research*, 257(1):25–40, 2017.

Y. Li, D. Kim, F. Zou, and D.-Z. Du. Constructing connected dominating sets with bounded diameters in wireless networks. In *International Conference on Wireless Algorithms, Systems and Applications (WASA 2007)*, pages 89–94. IEEE, 2007.

A. Linskey. Foes of congressional map meet target. *The Baltimore Sun*, July 11, 2012. URL `http://articles.baltimoresun.com/2012-07-11/news/bs-md-congressional-redistrict-20120711_1_congressional-map-new-map-state-board`.

Y. Y. Liu, W. K. T. Cho, and S. Wang. PEAR: A massively parallel evolutionary computation approach for political redistricting optimization and analysis. *Swarm and Evolutionary Computation*, 30:78–92, 2016.

I. Ljubić, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming*, 105(2-3):427–449, 2006.

D. Lokshtanov, D. Marx, S. Saurabh, et al. Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS*, 3(105), 2013.

L. Lovász, V. Neumann-Lara, and M. Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9(4):269–276, 1978.

A. Lucena, N. Maculan, and L. Simonetti. Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Science*, 7(3):289–311, 2010.

C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.

E. Lykhovyd. Efficient implementation of Shor's r-algorithm using MKL, 2019. URL `https://github.com/zhelih/ralg`.

H. M. Markowitz and A. S. Manne. On the solution of discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 84–110, 1957.

R. K. Martin. Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10(3):119–128, 1991.

A. Mehrotra, E. L. Johnson, and G. L. Nemhauser. An optimization based heuristic for political districting. *Management Science*, 44(8):1100–1114, 1998.

S. Miller. The problem of redistricting: The use of centroidal Voronoi diagrams to build unbiased congressional districts. *Senior project, Whitman College*, 2007.

C. L. Monma and D. F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4):531–541, 1989.

J. Moody and D. White. Structural cohesion and embeddedness: A hierarchical concept of social groups. *American Sociological Review*, pages 103–127, 2003.

M. J. Morgan and V. Grout. Finding optimal solutions to backbone minimisation problems using mixed integer programming. In *Proceedings of the 7th International Network Conference (INC 2008)*, pages 53–64, 2008.

D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 276–287. Springer, 2012.

F. H. Murphy, S. W. Hess, and C. G. Wong-Martinez. Politics. In *Encyclopedia of Operations Research and Management Science*, pages 1137–1141. Springer, 2013.

NCSL. Redistricting criteria. `http://www.ncsl.org/research/redistricting/redistricting-criteria.aspx`, 2019. Accessed: 2019-06-20.

G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6(1):48–61, 1974.

R. G. Niemi, B. Grofman, C. Carlucci, and T. Hofeller. Measuring compactness and the role of a compactness standard in a test for partisan and racial gerrymandering. *The Journal of Politics*, 52(4):1155–1181, 1990.

J. Oehrlein and J.-H. Haunert. A cutting-plane method for contiguity-constrained spatial aggregation. *Journal of Spatial Information Science*, 2017(15):89–120, 2017.

B. Olson. Impartial automatic redistricting. `https://bdistricting.com/2010/`, 2019. Accessed: 2019-06-21.

H. Önal, Y. Wang, S. T. Dissanayake, and J. D. Westervelt. Optimal design of compact and functionally contiguous conservation management areas. *European Journal of Operational Research*, 251(3):957–968, 2016.

J. B. Orlin. Max flows in $O(nm)$ time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 765–774. ACM, 2013.

K. Pashkovich. *Extended formulations for combinatorial polytopes.* PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2012.

T. Polzin and S. V. Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112(1-3):241–261, 2001.

R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.

D. Rehfeldt and T. Koch. Combining NP-hard reduction techniques and strong heuristics in an exact algorithm for the maximum-weight connected subgraph problem. *SIAM Journal on Optimization*, 29(1):369–398, 2019.

D. Rehfeldt, T. Koch, and S. J. Maher. Reduction techniques for the prize collecting Steiner tree problem and the maximum-weight connected subgraph problem. *Networks*, 73(2): 206–233, 2019.

M. G. Resende and R. F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88, 2004.

F. Ricca and B. Simeone. Local search algorithms for political districting. *European Journal of Operational Research*, 189(3):1409–1426, 2008.

F. Ricca, A. Scozzari, and B. Simeone. Weighted Voronoi region algorithms for political districting. *Mathematical and Computer Modelling*, 48(9-10):1468–1477, 2008.

F. Ricca, A. Scozzari, and B. Simeone. Political districting: From classical models to recent approaches. *Annals of Operations Research*, 204(1):271–299, 2013.

L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524. ACM, 2013.

H. Salemi and A. Buchanan. Parsimonious formulations for low-diameter clusters. *Mathematical Programming Computation*, 2020. To appear.

A. Sassano. On the facial structure of the set covering polytope. *Mathematical Programming*, 44(1-3):181–202, 1989.

O. Schaudt. On dominating sets whose induced subgraphs have a bounded diameter. *Discrete Applied Mathematics*, 161(16):2647–2652, 2013.

A. A. Schoone, H. L. Bodlaender, and J. Van Leeuwen. Diameter increase caused by edge deletion. *Journal of Graph Theory*, 11(3):409–427, 1987.

A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, Berlin, 2003.

T. Shirabe. A model of contiguity for spatial unit allocation. *Geographical Analysis*, 37(1): 2–16, 2005.

T. Shirabe. Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design*, 36(6):1053–1066, 2009.

N. Z. Shor. *Minimization methods for non-differentiable functions*, volume 3 of *Springer Series in Computational Mathematics*. Springer, 1985.

L. Simonetti, A. S. Da Cunha, and A. Lucena. The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In *Network Optimization*, pages 162–169. Springer Berlin Heidelberg, 2011.

L. Svec, S. Burden, and A. Dilley. Applying Voronoi diagrams to the redistricting problem. *The UMAP Journal*, 28(3):313–329, 2007.

R. Swamy, D. M. King, and S. H. Jacobson. A case for transparency in the design of political districts, 2019a. Working paper.

R. Swamy, D. M. King, and S. H. Jacobson. Multi-objective optimization for political districting: A scalable multilevel approach, 2019b. URL `http://www.optimization-online.org/DB_FILE/2019/03/7123.pdf`. Working paper.

M. Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, 3(69):330–353, 2004.

W. T. Tutte. *Graph Theory*, volume 21 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1984.

USCB. TIGER/Line shapefiles, 2012. `https://www2.census.gov/geo/tiger/TIGER2010DP1/Profile-County_Tract.zip`.

H. Validi and A. Buchanan. A note on "A linear-size zero-one programming model for the minimum spanning tree problem in planar graphs". *Networks*, 73(1):135–142, 2019a.

H. Validi and A. Buchanan. The optimal design of low-latency virtual backbones. *INFORMS Journal on Computing*, 2019b. To appear.

H. Validi and A. Buchanan. Imposing connectivity with multiterminal cuts. In progress, 2019c.

H. Validi and A. Buchanan. Polyhedral study of k connected components. In progress, 2020c.

H. Validi, A. Buchanan, and E. Lykhovyd. Imposing contiguity constraints in political districting models. 2020. Optimization Online.

A. Veremyev and V. Boginski. Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European Journal of Operational Research*, 218 (2):316–326, 2012.

W. Vickrey. On the prevention of gerrymandering. *Political Science Quarterly*, 76(1):105–110, 1961.

S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *CVPR 2011*, pages 1401–1408. IEEE, 2011.

S. Voß. Steiner tree problems in telecommunications. In *Handbook of optimization in telecommunications*, pages 459–492. Springer, 2006.

Y. Wang. *Connectivity Constraints in Network Analysis*. PhD thesis, Texas A&M University, 2015.

Y. Wang, A. Buchanan, and S. Butenko. On imposing connectivity constraints in integer programs. *Mathematical Programming*, 166(1-2):241–271, 2017.

Y.-c. Wang and H. Önal. Designing connected nature reserve networks using a graph theory approach. *Acta Ecologica Sinica*, 31(5):235–240, 2011.

S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*, volume 8 of *Strutural Analysis in the Social Sciences*. Cambridge University Press, New York, 1994.

J. C. Williams. A linear-size zero-one programming model for the minimum spanning tree problem in planar graphs. *Networks*, 39(1):53–60, 2002a.

J. C. Williams. A zero-one programming model for contiguous land acquisition. *Geographical Analysis*, 34(4):330–349, 2002b.

N. Xiao. An evolutionary algorithm for site search problems. *Geographical Analysis*, 38(3): 227–247, 2006.

M. Xie and M. Haenggi. Towards an end-to-end delay analysis of wireless multihop networks. *Ad Hoc Networks*, 7(5):849–861, 2009.

J. Xu. *Topological structure and analysis of interconnection networks*. Kluwer, 2001.

M. Xu, J.-M. Xu, and X.-M. Hou. Fault diameter of Cartesian product graphs. *Information Processing Letters*, 93(5):245–248, 2005.

H. P. Young. Measuring the compactness of legislative districts. *Legislative Studies Quarterly*, 13(1):105–115, 1988.

N. Zhang, I. Shin, F. Zou, W. Wu, and M. T. Thai. Trade-off scheme for fault tolerant connected dominating sets on size and diameter. In *Proceedings of the 1st ACM international* *workshop on Foundations of wireless ad hoc and sensor networking and computing*, pages 1–8. ACM, 2008.

Y. Zhong, M. Haenggi, F.-C. Zheng, W. Zhang, T. Q. Quek, and W. Nie. Towards a tractable delay analysis in ultradense networks. *IEEE Communications Magazine*, page 104, 2017.

A. A. Zoltners and P. Sinha. Sales territory alignment: A review and model. *Management Science*, 29(11):1237–1256, 1983.

VITA

Hamidreza Validi

Candidate for the Degree of

Doctor of Philosophy

Dissertation: IMPOSING CONNECTIVITY IN NETWORK DESIGN PROBLEMS

Major Field: Industrial Engineering and Management

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Industrial Engineering and Management at Oklahoma State University, Stillwater, Oklahoma in June, 2020

Completed the requirements for the Master of Science in Industrial Engineering at Sharif University of Technology, Tehran, Iran in 2014

Completed the requirements for the Bachelor of Science in Industrial Engineering at Ferdowsi University of Mashhad, Mashhad, Iran in 2012

Experience:

Hamidreza Validi is Ph.D. candidate at Industrial Engineering and Management school of Oklahoma State University. He is broadly interested in combinatorial optimization, integer programming, and network optimization. Hamid's previous work has been published in *Networks* and *INFORMS Journal on Computing.* He was a summer fellow at Voting Rights Data Institute 2019. He has taught IEM 3103 in Fall 2018 and Fall 2019.