INTEGER PROGRAMMING FORMULATIONS FOR DISTANCE-CONSTRAINED NETWORK PROBLEMS

By

HOSSEINALI SALEMI

Bachelor of Science in Industrial and Systems
Engineering
Amirkabir University of Technology
Tehran, Iran
2013

Master of Science in Management Engineering
Politecnico di Milano
Milan, Italy
2015

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

INTEGER PROGRAMMING FORMULATIONS FOR DISTANCE-CONSTRAINED NETWORK PROBLEMS

Dissertation Approved:

Dr. Austin Buchanan

Dissertation Advisor

Dr. Balabhaskar Balasundaram

Dr. Sunderesh Heragu

Dr. Mahdi Asgari

Dedicated to my parents and beloved wife.

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

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude and appreciation to my advisor, Dr. Austin Buchanan, for his unwavering support, valuable advice, and constant encouragement during my PhD. He has been one of the most influential professors I have ever had. I have learned a lot from his expertise, knowledge, and behavior. This dissertation would have not been possible without his invaluable contribution, insightful suggestions, and guidance.

I would like to thank my committee members Dr. Balasundaram, Dr. Heragu, and Dr. Asgari for their constructive comments and persistent help throughout my doctoral research.

I also wish to thank all the faculty and staff of the School of Industrial Engineering & Management at Oklahoma State University.

Many thanks to my friends Hamidreza Validi, Hao Pan, and Yajun Lu whose friendship has been valuable to me.

Finally, my deepest appreciation goes to to my lovely wife, Mahshid, and my parents for their tremendous support and enormous help during all these years. It would have not been possible to go through these eventful times without their assistance and companionship.

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

Name: HOSSEINALI SALEMI

Date of Degree: June, 2020

Title of Study: INTEGER PROGRAMMING FORMULATIONS FOR DISTANCE-CONSTRAINED NETWORK PROBLEMS

Major Field: INDUSTRIAL ENGINEERING AND MANAGEMENT

Abstract: Network-based models (graphs) are among the most powerful tools for representing relationships and communications between different elements of a system in science, engineering, and business. In each of these scientific fields, nodes of a network represent distinct objects, and edges show relationships or similarities between them. Due to the ability of networks to model discrete objects in many different fields, network design and analysis have encountered an explosive growth during recent years. The presence of distance constraints in various network design and analysis problems such as detecting clusters and critical nodes, has been problematic and challenging for many researchers. These NP-hard combinatorial optimization problems are among the popular topics in network design and analysis and are challenging to be solved—especially when the number of nodes and/or edges of a network is large. Towards this end, we need to develop exact methods to identify low-diameter clusters and critical nodes of a network efficiently. In this dissertation, we introduce new integer programming formulations, algorithms, preprocessing techniques, and decomposition methods that allow us to find low-diameter clusters and critical nodes in relatively large networks.

Contents

Chapter Page

List of Tables

Table Page

List of Figures

Chapter I

Introduction

Network-based models (graphs) are among the most powerful tools for representing relationships and communications between different elements of a system in science, engineering, and business. In each of these scientific fields, nodes of a network represent distinct objects, and edges show relationships or similarities between them. Network models are widely used in areas including but not limited to computer science, electrical and power systems, social science, transportation and distribution, physics and chemistry, economics, biology, telecommunication, stock market, and public health. Moreover, data of a system can be represented by nodes of a network and, if some data points share similarities, the associated nodes are connected by edges. Nowadays, in the world with growing amounts of Big Data, the necessity of using these models is evident to everyone. In fact, we need network-based models, techniques, and algorithms that enable us to mine these substantial amounts of data and extract hidden knowledge from them. Due to the ability of networks to model discrete objects in many different fields, network design and analysis have encountered an explosive growth during recent years. Indeed, researchers have studied and analyzed electric networks, social networks, biological networks, transportation networks, and others.

The presence of connectivity, distance, or diameter constraints in various network design and analysis problems has been problematic and challenging for many researchers. Examples are detecting clusters and critical nodes in a network. These NP-hard problems are among the popular topics in network design and analysis and are challenging to be solved—especially when the number of nodes and/or edges of a network is large. Towards this end, we need to develop exact methods to identify clusters and critical nodes of a network efficiently. In this dissertation, we introduce new Integer Programming (IP) formulations, algorithms, and methodologies that allow us to solve relatively large instances of these problems. Before proceeding with the definitions of these combinatorial problems and their applications, we first start with graph notations and terminologies and then briefly discuss the importance of "distance" in network design and analysis.

1.1 Graph Notations

Consider a simple, edge-weighted graph $G = (V, E)$ with vertex set V and edge set $E \subseteq \binom{V}{2}$, where $\binom{V}{2} := \{\{u, v\} \mid u, v \in V,\ u \neq v\}$. We often let $n := |V|$ and $m := |E|$. Denote by

$$N_G(v) := \{u \in V \mid \{u, v\} \in E\},$$

the neighbors of v in G; its cardinality is the degree denoted $\deg_G(v) := |N_G(v)|$. The subset of edges incident to v is denoted

$$\delta_G(v) := \{\{u, v\} \in E \mid u \in N_G(v)\}.$$

The weight w_e of each edge $e \in E$ is assumed to be nonnegative. The length of a path is the sum of its edges' weights. The distance from node a to node b in graph G is the length of a shortest path from a to b and is denoted $\text{dist}_G(a, b)$. In some cases, distances are "hop-based," meaning that each edge has weight one, and the length of a path is equal to its number of edges. The diameter of G is the maximum of these pairwise distances and is denoted

$$\text{diam}(G) := \max \left\{ \text{dist}_G(a, b) \;\middle|\; \{a, b\} \in \binom{V}{2} \right\}.$$

The subgraph of G induced by the vertex subset $S \subseteq V$ is denoted by $G[S] := (S, E(S))$, where $E(S) := E \cap \binom{S}{2}$ denotes the subset of edges with both endpoints in S. For the vertex subset $S \subseteq V$, let $G - S := G[V \setminus S]$ represent the subgraph obtained by removing the vertices of S (and any incident edges). Similarly, for the edge subset $F \subseteq E$, let $G - F := (V, E \setminus F)$. The k-th power $G^k = (V, E^k)$ of graph $G = (V, E)$ has the same vertex set as G and edge set

$$E^k := \left\{ \{i, j\} \in \binom{V}{2} \;\middle|\; \text{dist}_G(i, j) \leq k \right\}.$$

Observe that this definition applies regardless of whether distances are hop-based or edge-weighted. A subset $S \subseteq V$ of vertices is a clique if $E(S) = \binom{S}{2}$. A vertex v is simplicial if its neighborhood $N(v)$ is a clique.

1.2 Connectivity and Distance

Intuitively, a cluster in a network is a subset of nodes that are densely connected to each other and sparsely connected to the nodes of other clusters. Clique, which is a subset of pairwise adjacent nodes in a network, is an ideal case for a cluster because each node is directly connected (by an edge) to all other nodes in the cluster. An example for a clique is two friends in Facebook who have a mutual friend. These three friends form a clique. However, the definition of clique is too restrictive in practice (for example due to possible errors resulting in missing edges.) These types of restrictions have convinced and motivated researchers to relax the definition of clique to a subset of vertices of diameter at most k, yielding a k-club. Although the k-club definition allows vertices to be nonadjacent (when $k \geq 2$), it guarantees that all vertices in the k-club are connected and their pairwise distance is at most k. It should be emphasized here that both clique and k-club models consider *distance* between nodes as a key metric in identifying low-diameter clusters in a network.

Moreover, in social networks, if two connected nodes are far apart from each other, they might be considered a disconnected pair because they may not be able to communicate desirably or quickly if the connecting path is too long (Borgatti, 2006). The velocity and quality of communications between different individuals depend on their actual distance. The shorter the communication path, the better quality for the communication. This argument holds for telecommunication applications as well. Take for instance the case in which there are two same-size telecommunication networks: a star and a path. While they have the same number of connected vertex pairs, the star might be more desirable, because each vertex is at most two hops away from all other vertices, which is not the case in the path (see Figure 1.1.) Other examples are flow network applications where it is necessary that perishable commodities do not flow through too many edges on their way to their destinations (Mahjoub and McCormick, 2010), and biological networks where distances between nodes affect their chemical interactions (Aringhieri et al., 2019). In these and many other real-world applications, *distance* between nodes plays a major role when connectivity is considered.

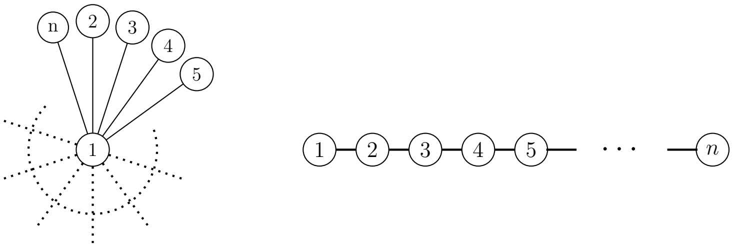


Figure 1.1: The left graph is a star S_n of n vertices. The right graph is a path P_n of n vertices. Although the number of connected vertex pairs in both graphs are equal, the maximum distance between all pairs of nodes in S_n is 2, while this number is $n-1$ for P_n. As a result, the star might be more desirable for decision-makers.

1.3 Cluster Detection

Cluster detection has been widely applied to areas such as social sciences, business and marketing, computer science, biology, bioinformatics, pattern recognition, and machine learning, among others. Cluster detection is also used in data mining and statistical data analysis. A "good" cluster is a one that has a small diameter. For example, possessing a small diameter is key in social network analysis, where a cluster represents a group of people who can quickly communicate with each other. One possible way to find clusters is to use clique and k-club models. See Figure 1.2 for an illustration of k-clubs. The notion of a k-club was originally introduced in sociology (Alba, 1973; Mokken, 1979), and has found applications in the analysis of biological networks (Balasundaram et al., 2005), as well as in text mining, terrorist networks, and network security (Shahinpour and Butenko, 2013b). Essentially the same type of distance or diameter constraint appears in political districting and wildlife reserve design applications where compactness is key. In Chapter II, we discuss how to impose these length bounded constraints efficiently and how to identify maximum k-clubs in relatively large networks.

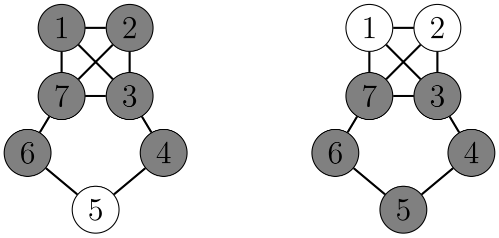


Figure 1.2: The gray vertices on the right form a 2-club. The gray vertices on the left do not, as vertex 4 and vertex 6 are distance 3 from each other in the induced subgraph.

1.4 Critical Nodes Detection

Networks are vulnerable to disruptive events caused by natural disasters or malicious attacks. In order to have a better assessment of network vulnerability, one might identify the nodes that are most important to its connectivity. These nodes, which are accountable for network cohesion, are often referred to as critical nodes. The critical nodes of a network are typically a small subset of vertices whose deletion maximally reduces some connectivity metric or equivalently results in maximum graph fragmentation (Arulselvan et al., 2009). The problem of optimally identifying critical nodes is known as critical node detection or critical node problem (CNP), which belongs to the intersection of network interdiction problems and graph theory research (Shen and Smith, 2012). Decision-makers might have two different motives for identifying critical nodes: to derive efficient defensive strategies for reducing impacts of possible disruptions; or to design effective offensive strategies with the most possible damage (Walteros et al., 2019). As an example, consider security issues for decentralized types of wireless networks, e.g., ad hoc networks, where targeting few nodes results in huge fragmentation. In such networks, one can find critical nodes to increase the network security or to deteriorate its connectivity (Lalou et al., 2018). Another case is network immunization where the objective is to reduce transmission of a virus with the constraint that not all members of the network can be immunized due to high costs of mass vaccination. One possible solution is to vaccinate the critical nodes, that is, the key humans on whom the network depends substantially to maintain its connectivity (Borgatti, 2006; Arulselvan et al., 2009). A further example is identifying critical nodes in transportation networks. Failure of these nodes cause disruptions in transportation, especially when a disaster occurs in an area. Identification of critical nodes informs decision-makers how to safeguard against possible damages and plan emergency evacuations (Vitoriano et al., 2011). The problem of detecting critical nodes, along with its similar versions, has been interesting to many researchers over the years. Bavelas (1948) and Freeman (1978), who consider the centrality of nodes as a measure of significance in social networks and human communications are among the first researchers in this area. In recent years, CNP has found numerous applications in areas such as network immunization (Cohen et al., 2003), transportation (Kutz, 2004), epidemic control (Tao et al., 2006), telecommunication (Commander et al., 2007), evacuation (Matisziw and Murray, 2009), and biology (Boginski and Commander, 2009). In general, CNP can be applied to all areas in which networks are subject to random failures/errors, natural disasters, and attacks. Borgatti (2006) shows that the criteria under which a node is considered to be critical depend on the role that the node plays in the network; as a result, a critical node under a specific measure might not be critical in terms of another metric. Consequently, based on how network connectivity is defined, there exist different classes of CNP. Shen et al. (2012) and Shen and Smith (2012) consider maximizing the number of network components and minimizing the largest component size in the remaining graph as two different metrics. Arulselvan et al. (2009), Addis et al. (2013), and Veremyev et al. (2014) focus on minimizing the total number of connected node pairs as a connectivity reduction criterion.

Considering whether two nodes of a network are connected or not (via a path) as the only network connectivity metric does not address many practical problems (Veremyev et al., 2015). As discussed in Section 1.2, distance between nodes should be taken into account in many real-world cases to have a meaningful analysis of network connectivity. These observations motivate researchers to consider distances between vertices of a network in order to find the most critical nodes. In Chapters III and IV, we discuss how to identify distance-based critical nodes of a network. Next, we briefly provide our contributions in this dissertation.

1.5 Our Contributions

In Chapter II, we propose formulations and techniques to identify tightly knit clusters, modeled by k-clubs. We propose new path-like and cut-like integer programming formulations for detecting these low-diameter subgraphs. They simplify, generalize, and/or dominate several previously existing formulations. Our best-performing formulation uses only node variables (quite unlike previous formulations) and imposes the diameter-at-most-k constraints via an exponentially large class of cut-like inequalities. A relatively simple implementation of the cut-like formulation easily outperforms previous approaches, solving dozens of instances of the maximum k-club problem in a second or two that would take hours by other formulations. Moreover, the cut-like formulation is more general in the sense that it applies even when distances are not measured in terms of hops. While we consider only the k-club problem, the proposed techniques may also be useful in other applications where compact solutions are key (e.g., political districting and wildlife reserve design).

In Chapter III, we propose new path-like and thin integer programming formulations to model a class of critical node problems. In this class, we want to delete at most b nodes of a network in order to minimize the number of node pairs that remain connected by a path of length at most k. Our new formulations directly apply under hop-based and edge-weighted distances, as opposed to an existing recursive formulation. We compare the strength of these three formulations by introducing the notion of partial dominant of a polyhedron. While the thin formulation has exponentially many constraints, we show that the integer and fractional separation of violated inequalities can be done in polynomial time under hop-based distances. When distances are edge-weighted, we show that integer separation still takes polynomial time, although it is NP-hard to solve the fractional separation. In addition, we propose a more general preprocessing procedure that, on average fixes three times as many variables than before. The new proposed formulations and a preprocessing method enable us to solve larger instances than previous works.

In Chapter IV, we propose a Benders decomposition algorithm with branch-and-cut implementations of it to solve the (distance-based) critical node problem. In this chapter, we show how to identity Benders optimality cuts in a combinatorial manner instead of solving the dual subproblem as an LP. We also provide techniques to accelerate the convergence of the Benders decomposition method.

Finally, we conclude in Chapter V and provide ideas for future works.

Chapter II

Parsimonious Formulations for Low-diameter Clusters

This chapter is based on work with Austin Buchanan (Salemi and Buchanan, 2020a)[1]. Cluster detection is a common problem encountered in network analysis, and an oft-required property of a "good" cluster is that it have a small diameter. A clique in a graph, which is a subset $S \subseteq V$ of vertices that induces a subgraph of diameter at most 1, is an ideal cluster. Relaxing this definition to "induces a subgraph of diameter at most k" yields a k-club.

Definition 1 (k-club, essentially due to Mokken (1979))**.** *A subset $S \subseteq V$ of vertices in a graph $G = (V, E)$ is called a k-club if* $\operatorname{diam}(G[S]) \leq k$.

Effectively enforcing these diameter-at-most-k (i.e., k-club) constraints in an integer program (IP) has proven difficult for researchers. Notable techniques include introducing a binary variable for (the interior of) each path of length at most k (Bourjolly et al., 2002; Wotzlaw, 2014) and linearizing multilinear 0-1 formulations (Veremyev and Boginski, 2012; Veremyev et al., 2015). A naïve implementation of the path-based formulation quickly becomes impractical as the value of k increases, since its number of variables is $\Theta(n^{k-1})$, under hop-based distances (Wotzlaw, 2014)—or size $\Theta(n^{k+1})$ if the path variables include the path's endpoints (Bourjolly et al., 2002). Any practical approach based on it would require a complicated branch-and-price implementation even for moderate values of k. The linearized multilinear 0-1 formulations of Veremyev et al. (Veremyev and Boginski, 2012; Veremyev et al., 2015) use $\Theta(kn^2)$ variables and typically solve real-life 200-node instances of

[1] Reprinted with permission from "Parsimonious formulations for low-diameter clusters" by H. Salemi and A. Buchanan. *Mathematical Programming Computation*, (2020): 1-36, Copyright 2020 by Springer.

the maximum k-club problem with $k = 4$ in ten minutes, but routinely fail to solve 300-node instances with $k = 5$ in under one hour (Moradi and Balasundaram, 2018). One limitation of these formulations is that they cannot be used to solve very large instances (with thousands of nodes), as the variables will number in the millions. Also, they only apply when distances are measured in terms of hops (cf. the pseudopolynomial formulation of Veremyev et al. (2015)).

In this chapter, we propose new *path-like* and *cut-like* formulations that generalize or improve upon several previously existing formulations for k-clubs. We include "-like" in their names to emphasize that they are not based on the usual notions of paths and cuts. Instead, they are based on length-bounded connectors and length-bounded separators, which are defined below. Examples to illustrate the definitions are given in Figure 2.1.

Definition 2 (Length-k a, b-connector)**.** *A subset $C \subseteq V \setminus \{a, b\}$ of vertices is called a length-k a, b-connector in an edge-weighted graph $G = (V, E)$ if $\text{dist}_{G[C \cup \{a,b\}]}(a, b) \leq k$.*

Definition 3 (Length-k a, b-separator)**.** *A subset $S \subseteq V \setminus \{a, b\}$ of vertices is called a length-k a, b-separator in an edge-weighted graph $G = (V, E)$ if $\text{dist}_{G-S}(a, b) > k$.*

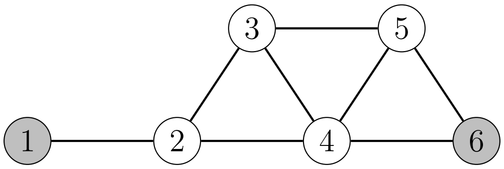


Figure 2.1: Under hop-based distances, the vertex subset $\{2, 4\}$ is a length-3 $1, 6$-connector, and the vertex subset $\{4\}$ is a length-3 $1, 6$-separator.

The path-like formulation that we propose generalizes the folklore $k = 2$ common neighbor formulation (see constraints (2.11)), simplifies and dominates the chain formulations of Bourjolly et al. (2002) and Wotzlaw (2014), and generalizes the $k = 3$ neighborhood F_N formulation of Almeida and Carvalho (2012). We observe that, when distances are hop-based, the path-like formulation has $\mathcal{O}(m^{(k-1)/2})$ variables when k is an odd constant and $\mathcal{O}(nm^{(k-2)/2})$ variables when k is an even constant. In particular, the formulation has size $\mathcal{O}(n+m)$ when $k=3$ and size $\mathcal{O}(nm)$ when $k=4$. This makes the path-like formulation a reasonable option when $k \leq 4$, distances are hop-based, and the graph is sparse.

However, we prefer the cut-like formulation for a number of reasons. First, it uses just n binary variables, regardless of the edges' weights or the value of k. This allows us to apply it to large instances and to instances having non-unit edge lengths. Second, it is conceptually simple, being defined by a single class of constraints:

$$\text{(length-}k\ a,b\text{-separator inequality)} \qquad x_a + x_b \leq 1 + x(S). \tag{2.1}$$

Here, x_i is a binary variable representing the decision to include vertex i in the k-club, and $x(S)$ is shorthand for $\sum_{i \in S} x_i$. These inequalities are written for every pair $\{a, b\}$ of nonadjacent vertices and every length-k a,b-separator $S \subset V \setminus \{a, b\}$. Third, our implementation of the cut-like formulation handily outperforms all previous approaches for the maximum k-club problem, solving dozens of instances in seconds that take hours by other formulations. This is despite the fact that this formulation can have exponentially many inequalities (for which the separation problem is generally hard). Other notable properties of the cut-like formulation include: it is stronger than the path-like formulation; it generalizes the folklore $k=2$ *common neighbor* formulation; and it generalizes the $k=3$ *node cut set* formulation F_S of Almeida and Carvalho (2012).

Though we focus on the maximum k-club problem in this chapter, our broader intent is to illustrate the potential of using "cut-like" formulations for diameter-constrained problems, e.g., arising in political districting or wildlife reserve design. The k-club problem serves as a well-studied, stylized problem on which to test the approach.

2.1 Literature Review

Mokken (1979) defined a k-club as an inclusionwise maximal subset of vertices whose induced subgraph has diameter at most k (in terms of hops). Nowadays, the maximality condition is usually dropped from the definition. Still, much of the literature on k-clubs is devoted to finding large k-clubs in graphs, particularly those that are *maximum*. The maximum k-club problem is known to be NP-hard (Bourjolly et al., 2002), even in graphs of diameter $k+1$ (Balasundaram et al., 2005). Further, the problem of testing whether a given k-club is inclusionwise maximal is coNP-complete for every $k \geq 2$ (Mahdavi Pajouh and Balasundaram, 2012). For every $k \geq 2$, the maximum k-club problem is approximable within a factor of $\lceil n^{1/2} \rceil$ and essentially no better (Asahiro et al., 2018). For more, see the survey of Shahinpour and Butenko (2013b).

To verify that a subset $S \subseteq V$ of vertices is a k-club, one can create the subgraph $G[S]$ induced by S and then perform a single-source shortest paths computation from each node v to ensure that all other nodes are no farther than k away. When distances are hop-based this takes time $\mathcal{O}(|S|m) = \mathcal{O}(nm)$ by BFS. When each edge length is a positive integer, this can be done in time $\mathcal{O}(|S|m + |S|^2 \log\log |S|) = \mathcal{O}(nm + n^2 \log\log n)$ by Thorup (2004). Under the strong exponential time hypothesis (SETH) of Impagliazzo et al. (Impagliazzo et al., 2001; Impagliazzo and Paturi, 2001), this is essentially best-possible, even in the simplest nontrivial case of 2-clubs, see Theorem 1. SETH is an unproven complexity assumption that is stronger than P≠NP, and, while not everyone believes that it is true, disproving it would be a breakthrough and imply faster algorithms for many problems.

Theorem 1 (Roditty and Vassilevska Williams (2013))**.** *If SETH holds, then for every $\varepsilon > 0$ there is no time $\mathcal{O}(m^{2-\varepsilon})$ algorithm for checking whether a connected graph G has $\operatorname{diam}(G) \leq 2$.*

Previously, we mentioned the integer programming formulations for k-clubs given by Bourjolly et al. (2002) (cf. Wotzlaw (2014)). Another recent approach by Moradi and Balasundaram (2018), cf. Lu et al. (2018), is to:

- initialize a formulation with inequalities of the form $x_a + x_b \leq 1$, where $\text{dist}_G(a, b) > k$;
- as necessary, cut off infeasible 0-1 vectors x^* using the constraint:

$$\sum_{i \in V:\ x_i^* = 0} x_i + \sum_{i \in V:\ x_i^* = 1} (1 - x_i) \geq 1. \tag{2.2}$$

This is similar in spirit to what we propose, except that our length-k a, b-separator inequalities (2.1) exploit the problem's structure and are naturally stronger than the canonical hypercube cuts (2.2) which only cut off x^*.

Almeida and Carvalho (2012) compare three different formulations for the 3-club problem: (i) the *chain* formulation of Bourjolly et al. (2002) which uses a variable for each path of length at most 3; (ii) a so-called *neighborhood* formulation (F_N) which imposes linearized versions of the following constraints for every pair $\{a, b\}$ of nonadjacent nodes:

$$\text{(constraints of F\_N)} \qquad x_a + x_b \leq 1 + \sum_{c \in N(a) \cap N(b)} x_c + \sum_{\{i,j\} \in E_{ab}} x_i x_j$$

where E_{ab} is the subset of edges $\{i, j\} \in E$ for which $i \in N(a) \setminus N(b)$ and $j \in N(b) \setminus N(a)$; and (iii) a so-called *node cut set* formulation (F_S) based on the exponential class of constraints:

$$\text{(constraints of F\_S)} \qquad x_a + x_b \leq 1 + \sum_{c \in N(a) \cap N(b)} x_c + \sum_{i \in S_{ab}} x_i$$

where $S_{ab} \subseteq V$ is a vertex cover of the graph induced by the edge set E_{ab}. We will see that the formulations F_N and F_S are the special cases of our path-like and cut-like formulations, respectively, when $k = 3$.

In a number of applications, one requires the selected vertices to induce a connected subgraph, but with no specific bound on its diameter. This can be formulated using the a, b-separator inequalities $x_a + x_b \leq 1 + x(S)$, where $S \subseteq V \setminus \{a, b\}$ is an a, b-separator, i.e., there is no path from a to b in $G - S$. These inequalities have been studied in detail by Wang et al. (2017). Approaches based on a, b-separator inequalities have outperformed flow-based formulations for imposing induced connectivity (Carvajal et al., 2013; Buchanan et al., 2015; Fischetti et al., 2017a), and thus it is perhaps not surprising that our distance-constrained generalization performs well. The *separation problem* for the a, b-separator inequalities is polynomial-time reducible to max flow (see, e.g., Fischetti et al. (2017a)), implying that one can optimize over the corresponding LP relaxation in polynomial time (Grötschel et al., 1993). We will see that the separation problem for our cut-like formulation is similarly reducible to max flow when $k \in \{2, 3, 4\}$ but that separation is hard when $k \geq 5$.

2.2 A Recursive Formulation

Here, we provide a "recursive" integer programming formulation for k-clubs in a directed graph $D = (V, A)$ when $k \geq 3$ and distances are hop-based, similar to that of Veremyev and Boginski (2012)[2]. If the input graph is undirected, first replace each undirected edge $\{u, v\}$ by its directed counterparts (u, v) and (v, u).

Denote by $N^-(i) := \{j \in V \mid (j, i) \in A\}$ the set of incoming neighbors to node i. The binary variable x_i represents the decision to include vertex i in the k-club. The binary variable y_{ij}^t equals one if and only if there exists a path from i to j of length exactly t whose vertices (including i and j) belong to the chosen k-club. This variable is only defined when $t \geq 1$ and should not be confused with y_{ij} raised to the t-th power. In the formulation, we should write constraints that impose the following condition:

$$y_{iv}^t = 1 \iff x_v = 1 \text{ and there exists } j \in N^-(v) \text{ such that } y_{ij}^{t-1} = 1.$$

In words, there is a path (across k-club nodes) from i to v of length t if and only if (i) node v

[2] Later, we use this "recursive" formulation in our implementation to solve the maximum k-club problem.

belongs to the k-club, and (ii) there is a path (using only k-club nodes) of length $t-1$ from node i to some incoming neighbor j of v.

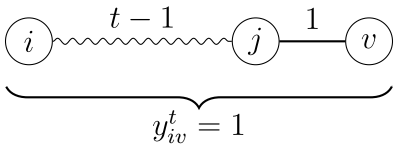


Figure 2.2: An illustration to explain the variable y_{ij}^t.

When $t \geq 2$, this equivalence can be formulated as follows.

$$(\Longleftarrow) \qquad y_{ij}^{t-1} + x_v \leq y_{iv}^t + 1 \qquad \forall j \in N^-(v)$$

$$(\Longrightarrow) \qquad y_{iv}^t \leq x_v \text{ and } y_{iv}^t \leq \sum_{j \in N^-(v)} y_{ij}^{t-1}.$$

For the case $t = 1$, we want to impose that $y_{ij}^1 = 1$ if and only if $x_i = x_j = 1$ and $(i,j) \in A$. This can be formulated as follows, where the variable y_{ij}^1 is only defined when $(i,j) \in A$.

$$(\Longleftarrow) \qquad x_i + x_j \leq y_{ij}^1 + 1$$

$$(\Longrightarrow) \qquad y_{ij}^1 \leq x_i \quad \text{and} \quad y_{ij}^1 \leq x_j.$$

So far, the constraints impose that the y_{ij}^t variables take their intended values. Now, to enforce that the selected vertices form a k-club, if vertices i and j are both selected, then there must be an i,j-path of length $\leq k$, i.e.,

$$x_i + x_j \leq 1 + \sum_{t=1}^{k} y_{ij}^t.$$

Observe that this constraint can easily be modified to impose different distance requirements depending on i and j. In summary, the formulation is as follows, where $T_{\geq 2} := \{2, \dots, k\}$.

$$
\begin{aligned}
& x_i + x_j \leq y_{ij}^1 + 1 && (i,j) \in A && \text{(2.3a)}\\
& y_{ij}^1 \leq x_i && (i,j) \in A && \text{(2.3b)}\\
& y_{ij}^1 \leq x_j && (i,j) \in A && \text{(2.3c)}\\
& y_{ij}^{t-1} + x_v \leq y_{iv}^t + 1 && i \in V \setminus \{j,v\},\ (j,v) \in A,\ t \in T_{\geq 2} && \text{(2.3d)}\\
& y_{iv}^t \leq x_v && i \in V \setminus \{v\},\ v \in V,\ t \in T_{\geq 2} && \text{(2.3e)}\\
& y_{iv}^t \leq \sum_{j \in N^-(v)} y_{ij}^{t-1} && i \in V \setminus \{v\},\ v \in V,\ t \in T_{\geq 2} && \text{(2.3f)}\\
& x_i + x_j \leq 1 + \sum_{t=1}^{k} y_{ij}^t && i \in V \setminus \{j\},\ j \in V && \text{(2.3g)}\\
& x_i \in \{0,1\} && i \in V && \text{(2.3h)}\\
& y_{ij}^t \in \{0,1\} && i \in V \setminus \{j\},\ j \in V,\ t \in \{1, \dots, k\}. && \text{(2.3i)}
\end{aligned}
$$

Since MIP solvers use sparse matrix representation, the number of nonzeros in the formulation is more indicative of its size than the quantity obtained by multiplying the number of variables by the number of constraints.

Theorem 2. *The above is a correct formulation for k-clubs in digraphs (under hop-based distances) and has $\mathcal{O}(kn^2)$ variables, $\mathcal{O}(knm)$ constraints, and $\mathcal{O}(knm)$ nonzeros.*

Not all of these variables and constraints may be necessary. For example, if the input graph is undirected we can assume $y_{ij}^t = y_{ji}^t$. If desired, the user can impose the constraints $y_{ij}^t = y_{ji}^t$ when implementing the formulation (as we do), and the solver will perform the substitutions in presolve.

In a later work, Veremyev et al. (2015) made a small change to the variables' definitions, defining them for paths of length *at most* k, instead of for paths of length *exactly* k. They also strengthened the formulation by disaggregating some big-M constraints, which was found to perform better computationally despite the increase in the number of constraints. (This is perhaps unsurprising given that this improvement in strength came at essentially no cost to the formulation's size measured with respect to the number of nonzeros.)

2.3 Our Contributions

In Section 2.4, we introduce the path-like formulation. It generalizes the folklore $k = 2$ common neighbor formulation, simplifies and dominates the chain formulations of Bourjolly et al. (2002) and Wotzlaw (2014), and generalizes the $k = 3$ neighborhood F_N formulation of Almeida and Carvalho (2012). We also detail the formulation for the $k = 4$ case.

In Section 2.5, we propose the cut-like formulation, which is the first nontrivial formulation for k-club that uses n variables, and prove its correctness—even when distances are not hop-based. It generalizes the folklore $k = 2$ common neighbor formulation and the $k = 3$ node cut set formulation F_S of Almeida and Carvalho (2012). We provide the exact conditions under which the formulation's constraints induce facets, and show that the cut-like formulation is stronger than the path-like formulation for every $k \geq 3$, generalizing results of Almeida and Carvalho (2012) who showed that F_S is stronger than F_N (i.e., the $k = 3$ case).

In Section 2.5.3, we examine some complexity issues relating to the cut-like formulation. Namely, we observe that the associated separation problem is polynomial-time solvable when $k \in \{2, 3, 4\}$ (if distances are measured in terms of hops) and prove NP-hardness for every $k \geq 5$. Note that Almeida and Carvalho (2012) never addressed the complexity of separation for their F_S formulation (i.e., the $k = 3$ case of our cut-like formulation) and resorted to heuristic separation in their implementation[3]. The polynomiality of separation for the case $k = 4$ is intimately linked with observations of Lovász et al. (1978) on length-bounded cuts. We also remark that these flow-based separation routines immediately lead to polynomial-size

[3] In later work, Almeida and Carvalho (2014) show that their node cut set formulation F_S admits a size $\mathcal{O}(n^4)$ extended formulation which they call F_EC, but the separation problem for F_S is never explicitly discussed.

(but impractical) extended formulations when $k \in \{3, 4\}$.

In Section 2.6, we perform computational experiments. They demonstrate the superiority of the cut-like formulation over all other formulations, including the path-like formulation and the compact formulations of Veremyev et al. (Veremyev and Boginski, 2012; Veremyev et al., 2015). In many cases, the differences in running time are dramatic, with the cut-like formulation taking a second or two and the other approaches taking hours. This happens on both real-life and synthetic testbeds that have been considered in the previous literature on the maximum k-club problem. Taking inspiration from Moradi and Balasundaram (2018), we also propose a decomposition procedure to handle the exorbitant number of conflict constraints generated for some of the larger instances. Our code is publicly available (Salemi and Buchanan, 2019).

Finally, we conclude in Section 2.7.

2.4 The Path-Like Formulation

Here we introduce the path-like formulation, which improves upon the chain formulation of Bourjolly et al. (2002) (cf. Wotzlaw (2014)) and generalizes the $k = 3$ neighborhood formulation F_N of Almeida and Carvalho (2012). For completeness, we briefly review the previous formulations.

Chain Formulation. The chain formulation of Bourjolly et al. (2002) has a binary variable y_P for each chain (or path) of length at most k and is a follows.

$$\max \sum_{i \in V} x_i \tag{2.4a}$$

$$x_a + x_b \leq 1 + \sum_{P \in P_{ab}^k} y_P \qquad \forall \{a,b\} \in \binom{V}{2} \setminus E \tag{2.4b}$$

$$y_P \leq x_i \qquad \forall i \in V(P),\ \forall P \tag{2.4c}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{2.4d}$$

$$y_P \in \{0,1\} \qquad \forall P. \tag{2.4e}$$

Here, P_{ab}^k is the collection of paths of length at most k between nodes a and b, and $\forall P$ is shorthand for all paths of length at most k. Constraints (2.4b) ensure that, for every pair of nonadjacent vertices a and b in the chosen k-club, there is at least one path between them of length at most k. Constraints (2.4c) ensure that these paths can be crossed only when all of their nodes are selected in the k-club.

Observe that a path consisting of k edges crosses $k-1$ interior nodes and two endpoints for a total of $k+1$ nodes. Thus, the number of path variables are of the order $\mathcal{O}(n^{k+1})$ when distances are hop-based. And, when G is complete, the number of paths is indeed $\Omega(n^{k+1})$. However, as observed by Wotzlaw (2014), the path variables can be defined with respect to the interior nodes only, giving size $\mathcal{O}(n^{k-1})$. For example, if G is the path graph 1-2-3-4 with an additional leaf node 5 attached to node 3 and $k = 3$, then the path 2-3 can connect the a,b-pairs $\{1,4\}$ as well as $\{1,5\}$; there is no need to define two variables for the paths 1-2-3-4 and 1-2-3-5.

It is unclear whether Bourjolly et al. intended for these path variables to include their endpoints, but later papers claim that the chain formulation has size $\mathcal{O}(n^{k+1})$ (see Veremyev and Boginski (2012); Almeida and Carvalho (2012)), and to our knowledge Wotzlaw was the first to explicitly state a size bound of $\mathcal{O}(n^{k-1})$, albeit in a MAX-SAT formulation.

Path-like formulation. In the "path-like" formulation, we actually define variables for length-bounded connectors (and not for paths). We will see that this yields several benefits. The formulation is as follows, where y_C is a binary variable denoting whether to choose the connector $C \subset V$, and C^k_{ab} is the collection of all *minimal* length-k a,b-connectors.

$$\max \sum_{i \in V} x_i \tag{2.5a}$$

$$x_a + x_b \leq 1 + \sum_{C \in C^k_{ab}} y_C \qquad \forall \{a,b\} \in \binom{V}{2} \setminus E \tag{2.5b}$$

$$y_C \leq x_i \qquad \forall i \in C,\ \forall C \tag{2.5c}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{2.5d}$$

$$y_C \in \{0,1\} \qquad \forall C. \tag{2.5e}$$

Here, $\forall C$ is shorthand for $\forall C \in \bigcup C^k_{ab}$ where the union is over $\{a,b\} \in \binom{V}{2} \setminus E$. We note that this formulation applies when distances are edge-weighted, although it seems that little can be said about its size in this case.

There are several advantages of defining variables for length-bounded connectors (instead of for paths) as follows.

- there is no confusion regarding whether the "endpoints" are part of the variable's definition or not;
- the order in which the vertices are visited in a path is stricken from the variable definition, resulting in fewer variables;
- it allows us to define variables only for *minimal* connectors, which again reduces the formulation's size.

In fact, the restriction to *minimal* connectors is the key advantage of the $k = 3$ neighborhood formulation F_N of Almeida and Carvalho (2012) over the chain formulation (and over its no-endpoints variant).

Proposition 1. *Suppose distances are hop-based, G is connected, and $k \geq 2$ is a constant. Then, the number of variables in the path-like formulation is:*

- $O\left(m^{(k-1)/2}\right)$ *when k is odd, and*
- $O\left(nm^{(k-2)/2}\right)$ *when k is even.*

And, there are graphs requiring $\Omega(n^{k-1})$ variables.

Proof. Since distances are hop-based, every minimal length-k a,b-connector $C \in C_{ab}^k$ induces an a,b-path graph, say $a = v_0\text{-}v_1\text{-}v_2\text{-}\cdots\text{-}v_q = b$, where $q \leq k$. For such a connector C, define $f(C)$ as follows

$$f(C) := \begin{cases} \left(\emptyset, \left\{\{v_1, v_2\}, \cdots, \{v_{q-2}, v_{q-1}\}\right\}\right) & \text{if } |C| \geq 2 \text{ is even } (q \text{ odd}) \\ \left(v_1, \left\{\{v_2, v_3\}, \cdots, \{v_{q-2}, v_{q-1}\}\right\}\right) & \text{if } |C| \geq 1 \text{ is odd } (q \text{ even) and } v_1 < v_{q-1} \\ \left(v_{q-1}, \left\{\{v_1, v_2\}, \cdots, \{v_{q-3}, v_{q-2}\}\right\}\right) & \text{if } |C| \geq 1 \text{ is odd } (q \text{ even) and } v_1 > v_{q-1}. \end{cases}$$

Observe that the function f maps a connector C to an ordered pair (v_C, E_C) where $v_C \in V \cup \{\emptyset\}$ and $E_C \subseteq E$. See that, when $|C|$ is even, f maps C to $(q-1)/2$ edges; when $|C|$ is odd, f maps C to a vertex and $(q-2)/2$ edges. Define $\mathcal{C} := \bigcup C_{ab}^k$ where the union is over $\{a,b\} \in \binom{V}{2} \setminus E$ and $F := \{f(C) \mid C \in \mathcal{C}\}$. By assumption that G is connected and k is a constant, $nk = \mathcal{O}(m)$. Then,

$$|\mathcal{C}| = |F| \leq \sum_{\substack{q=2 \\ (q \text{ even})}}^{k} n\binom{|E|}{(q-2)/2} + \sum_{\substack{q=3 \\ (q \text{ odd})}}^{k} \binom{|E|}{(q-1)/2}.$$

So, when $k \geq 3$ is odd,

$$|\mathcal{C}| \leq \frac{k-1}{2} n \binom{|E|}{(k-3)/2} + \frac{k-1}{2} \binom{|E|}{(k-1)/2} = \mathcal{O}\left(k \binom{|E|}{(k-1)/2} \right) = \mathcal{O}(m^{(k-1)/2}),$$

and when $k \geq 2$ is even,

$$|\mathcal{C}| \leq \frac{k}{2} n \binom{|E|}{(k-2)/2} + \frac{k-2}{2} \binom{|E|}{(k-2)/2} = \mathcal{O}\left(nk \binom{|E|}{(k-2)/2} \right) = \mathcal{O}(nm^{(k-2)/2}).$$

Since the number of variables in the path-like formulation is $n + |\mathcal{C}|$, the first claim holds.

The following construction shows the second claim. Consider a graph with the vertex set $V = \{a\} \cup V_1 \cup V_2 \cup \cdots \cup V_{k-1} \cup \{b\}$, where each V_i has $\frac{n-2}{k-1}$ vertices. Connect a to all vertices of V_1, each vertex of V_1 to all vertices of V_2, ..., and each vertex of V_{k-1} to b. Picking one vertex v_i from each V_i gives a minimal length-k a,b-connector $\{v_1, v_2, \ldots, v_{k-1}\}$. So, number of minimal length-k a,b-connectors is at least $(\frac{n-2}{k-1})^{k-1}$, which is $\Omega(n^{k-1})$ when k is fixed. □

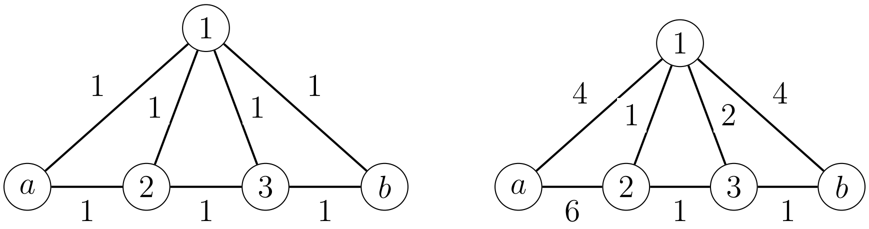


Figure 2.3: Examples of collection C_{ab}^k of minimal length-k a,b-connectors. On the left, $C_{ab}^3 = \{\{1\}, \{2, 3\}\}$. On the right, $C_{ab}^7 = \{\{1, 3\}\}$. Edge weights are given next to the edges.

Remark 1. *Figure 2.3 shows that if distances are* not *hop-based, a minimal length-k a,b-connector might not induce a path graph.*

To better illustrate the reason for using *connectors* that are *minimal*, consider the graphs obtained by removing an edge $e = \{a, b\}$ from a complete graph, i.e., $K_n - e$, and suppose distances are hop-based. For these graphs, the formulation of Wotzlaw (2014) would use a variable for each of the $\binom{n-2}{k-1}(k-1)!$ paths of length k that connect a and b (not to mention the variables for the shorter paths). If the variables were defined for length-k a, b-connectors (instead of for paths), this number would reduce to $\binom{n-2}{k-1}$. Enforcing that the connectors be *minimal* further reduces the number of auxiliary variables to $n-2$. Thus, these graphs $K_n - e$ provide examples where the path-like formulation is much smaller than the formulation of Wotzlaw (2014).

2.4.1 The hop-based case $k = 3$

Here we detail the path-like formulation for the hop-based case $k = 3$, showing that it is essentially the neighborhood formulation F_N of Almeida and Carvalho (2012). A similar analysis shows that it generalizes the folklore $k = 2$ common neighbor formulation.

To flesh out the formulation, we only need to identify the minimal length-3 a, b-connectors C^3_{ab}. So, suppose that vertices a and b are nonadjacent. Observe that if $v \in N(a) \cap N(b)$, then $\{v\}$ is a minimal length-3 a, b-connector. And, if $\{u, v\} \in E$ and $u \in N(a) \setminus N(b)$ and $v \in N(b) \setminus N(a)$, then $\{u, v\}$ is a minimal length-3 a, b-connector. Finally, there are no others. Thus, letting

$$E_{ab} := \{\{u, v\} \in E \mid u \in N(a) \setminus N(b),\ v \in N(b) \setminus N(a)\},$$

we can write the constraints (2.5b) as

$$x_a + x_b \leq 1 + \sum_{v \in N(a) \cap N(b)} y_{\{v\}} + \sum_{e \in E_{ab}} y_e. \tag{2.6}$$

We can add constraints of the following form to the path-like formulation without changing the feasible region in the space of x variables.

$$\sum_{i \in C} x_i \leq (|C| - 1) + y_C. \tag{2.7}$$

Observing that constraints (2.7) and (2.5c) will force $x_v = y_{\{v\}}$ for each $v \in N(a) \cap N(b)$, the path-like formulation with constraints (2.7) reduces to[4]:

$$\max \sum_{i \in V} x_i \tag{2.8a}$$

$$x_a + x_b \leq 1 + \sum_{v \in N(a) \cap N(b)} x_v + \sum_{e \in E_{ab}} y_e \qquad \forall \{a,b\} \in \binom{V}{2} \setminus E \tag{2.8b}$$

$$x_u + x_v \leq 1 + y_e \qquad \forall e = \{u,v\} \in E \tag{2.8c}$$

$$y_e \leq x_v \qquad \forall v \in e,\ \forall e \in E \tag{2.8d}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{2.8e}$$

$$y_e \in \{0,1\} \qquad \forall e \in E. \tag{2.8f}$$

This is the neighborhood formulation F_N of Almeida and Carvalho (2012).

2.4.2 The hop-based case $k = 4$

Here we detail the hop-based case $k = 4$ of the path-like formulation. For every pair $\{a, b\}$ of nonadjacent vertices, we partition the vertices of the graph $G = (V, E)$ into sets $V_{ij} = V_{ij}(a, b)$ as follows.

$$V_{ij}(a,b) := \{v \in V \mid \operatorname{dist}_G(a,v) = i,\ \operatorname{dist}_G(v,b) = j\}. \tag{2.9}$$

Observe that if some vertex v belongs to a set V_{ij} with $i + j > 4$, then it cannot belong to a minimal length-4 a,b-connector. Thus, every minimal length-4 a,b-connector is a subset of $V_{11} \cup V_{12} \cup V_{21} \cup V_{13} \cup V_{22} \cup V_{31}$, as depicted in Figure 2.4.

Every minimal length-4 a,b-connector $C \in \mathcal{C}^4_{ab}$ is one of the following six types.

[4] One possible caveat is that not all edges will be minimal length-3 a,b-connectors, meaning that the path-like formulation may in fact have fewer variables and constraints than F_N.

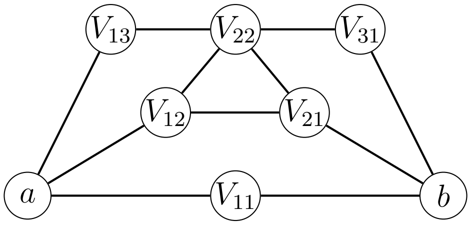


Figure 2.4: Vertices (and edges) within minimal length-4 a, b-connectors.

1. $\{i\}$ where $i \in V_{11}$;

2. $\{p, q\}$ where $p \in V_{12}$, $q \in V_{21}$, $pq \in E$;

3. $\{p, v, q\}$ where $p \in V_{12}$, $v \in V_{22}$, $q \in V_{21}$, $pv \in E$, $vq \in E$, $pq \notin E$;

4. $\{p, v, w\}$ where $p \in V_{12}$, $v \in V_{22}$, $w \in V_{31}$, $pv \in E$, $vw \in E$;

5. $\{u, v, q\}$ where $u \in V_{13}$, $v \in V_{22}$, $q \in V_{21}$, $uv \in E$, $vq \in E$;

6. $\{u, v, w\}$ where $u \in V_{13}$, $v \in V_{22}$, $w \in V_{31}$, $uv \in E$, $vw \in E$.

Moreover, for a particular $\{a, b\}$ pair, all minimal length-4 a, b-connectors can be enumerated in time $\mathcal{O}(nm)$ using linear space. The interested reader can consult our C++ implementation for the details (Salemi and Buchanan, 2019).

Remark 2. *By Proposition 1, the path-like formulation for 4-club has $\mathcal{O}(mn)$ variables and constraints when distances are hop-based.*

2.5 The Cut-Like Formulation

Here we introduce the cut-like formulation for k-club, generalizing the $k = 2$ common neighbor formulation and the $k = 3$ node cut set formulation F_S of Almeida and Carvalho (2012). It has only n variables, and its constraints are based on length-k a, b-separators.

Recall that a subset $S \subseteq V \setminus \{a, b\}$ of vertices in a graph $G = (V, E)$ is called a length-k a, b-separator if the distance from node a to node b in the graph $G - S$ is greater than k. In other words, each a, b-path of length $\leq k$ crosses at least one vertex of S. See Figure 2.1 for an example.

Cut-like formulation. As before, there is a binary variable x_i representing the decision to include vertex $i \in V$ in the k-club.

$$\max \sum_{i \in V} x_i \tag{2.10a}$$

$$x_a + x_b \leq 1 + x(S) \qquad \forall (a, b, S) \tag{2.10b}$$

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

Here, $\forall (a, b, S)$ is shorthand for all nonadjacent vertices a and b and all length-k a, b-separators $S \subseteq V \setminus \{a, b\}$. Naturally, it is sufficient to consider only *minimal* length-k a, b-separators.

In the hop-based case $k = 2$, observe that the only minimal length-k a, b-separator is $N(a) \cap N(b)$ so constraints (2.10b) reduce to the folklore $k = 2$ constraints:

$$x_a + x_b \leq 1 + x(N(a) \cap N(b)) \qquad \forall \{a, b\} \in \binom{V}{2} \setminus E. \tag{2.11}$$

Theorem 3. *The cut-like formulation is correct, even when distances are edge-weighted.*

Proof. We show that the vertex subset $K \subseteq V$ is a k-club if and only if its characteristic vector $x^K \in \{0, 1\}^n$ satisfies all constraints (2.10b).

($\Longrightarrow$) By the contrapositive. Let $K \subseteq V$ and suppose $x_a^K + x_b^K > 1 + \sum_{i \in S} x_i^K$ for some length-k a, b-separator $S \subseteq V \setminus \{a, b\}$. This implies $a, b \in K$ and $|S \cap K| = 0$. Since S is a length-k a, b-separator, $\text{dist}_{G-S}(a, b) > k$. Since $G[K]$ is a subgraph of $G - S$, $\text{dist}_{G[K]}(a, b) \geq \text{dist}_{G-S}(a, b)$. Thus, $\text{diam}(G[K]) \geq \text{dist}_{G[K]}(a, b) \geq \text{dist}_{G-S}(a, b) > k$, so K is not a k-club.

($\Longleftarrow$) By the contrapositive. Suppose that $K \subseteq V$ is not a k-club. That is, there exist vertices $a, b \in K$ such that $\text{dist}_{G[K]}(a,b) > k$. Then, $S := V \setminus K$ is a length-k a,b-separator in G. So, x^K violates the length-k a,b-separator inequality (2.10b) since $x^K_a + x^K_b = 2 > 1 + 0 = 1 + \sum_{i \in S} x^K_i$. □

2.5.1 Facet Characterization

Here we provide the exact conditions under which the length-k a,b-separator inequality (2.10b) induces a facet of the k-club polytope $\text{CLUB}_k(G) = \text{CLUB}_k(G,w)$ of graph G. To our knowledge, this is the first known nontrivial facet of the *edge-weighted* k-club polytope, besides the folklore inequalities mentioned in Proposition 2.

Proposition 2 (folklore, $k = 2$ by Balasundaram et al. (2005))**.** *If $S \subseteq V$ is a subset of vertices whose pairwise distances are greater than k, then $x(S) \leq 1$ is valid for* $\text{CLUB}_k(G)$*. Moreover, if no proper superset of S satisfies this property, then $x(S) \leq 1$ induces a facet.*

We will take as given that $\text{CLUB}_k(G)$ is full-dimensional, which is well-known (say, because it contains the zero vector and the unit vectors e_i). For more on k-club polyhedra, consult Balasundaram et al. (2005); Balasundaram (2007); Mahdavi Pajouh et al. (2016).

Lemma 1 (A way to lift)**.** *Let $S \subseteq V \setminus \{a,b\}$ be a length-k a,b-separator in graph $G = (V,E)$. If vertex $d \in V \setminus S$ satisfies properties 1 and 2 below, then $x_a + x_b + x_d - x(S) \leq 1$ is valid for* $\text{CLUB}_k(G)$*.*

1. $\text{dist}_{G-S}(a,d) > k$ *and* $\text{dist}_{G-S}(b,d) > k$*;*

2. *for every $s \in S$, at least one of the following holds:*

   (a) $\text{dist}_{G_s}(d,s) + \text{dist}_{G_s}(s,a) > k$*;*

   (b) $\text{dist}_{G_s}(d,s) + \text{dist}_{G_s}(s,b) > k$*;*

   where $G_s := G - (S \setminus \{s\})$.

Proof. By the contrapositive. Suppose there is a k-club $K \subseteq V$ with $x^K_a + x^K_b + x^K_d - x^K(S) \geq 2$, where x^K is the characteristic vector of K.

In the first case, suppose $x^K(S) = 0$. This implies that at least 2 vertices of $\{a, b, d\}$ belong to K, and a and b cannot both belong to K since S is a length-k a, b-separator. So, without loss, suppose that $a, d \in K$ and $b \notin K$. Then,

$$\text{dist}_{G-S}(a, d) \leq \text{dist}_{G[K]}(a, d) \leq k.$$

The first inequality holds because $G - S$ is a supergraph of $G[K]$, and the second inequality holds because K is a k-club that contains a and d. This shows that property 1 fails.

In the other case, $x^K(S) \geq 1$. Pick $s \in K \cap S$ arbitrarily. Since $x^K_a + x^K_b + x^K_d - x^K(S) \geq 2$, we have $a, b, d \in K$ and so $K \cap S = \{s\}$. Now, if $\text{dist}_{G-S}(a, d) \leq k$ or $\text{dist}_{G-S}(b, d) \leq k$ then property 1 fails, in which case we are done. So, suppose that $\text{dist}_{G-S}(a, d) > k$ and $\text{dist}_{G-S}(b, d) > k$. By straightforward properties of distances, the following inequalities hold.

$$\text{dist}_{G_s}(d, a) \leq \text{dist}_{G_s}(d, s) + \text{dist}_{G_s}(s, a) \tag{2.12}$$

$$\text{dist}_{G_s}(d, b) \leq \text{dist}_{G_s}(d, s) + \text{dist}_{G_s}(s, b). \tag{2.13}$$

We claim that these inequalities (2.12) and (2.13) hold at equality. Suppose not. Then at least one of them holds strictly; without loss, let it be the former, i.e., $\text{dist}_{G_s}(d, a) < \text{dist}_{G_s}(d, s) + \text{dist}_{G_s}(s, a)$. This implies that $\text{dist}_{G_s}(d, a) = \text{dist}_{G-S}(d, a)$, in which case we arrive at the contradiction

$$k < \text{dist}_{G-S}(d, a) = \text{dist}_{G_s}(d, a) \leq \text{dist}_{G[K]}(d, a) \leq k.$$

Here, the inequality near the middle holds because G_s is a supergraph of $G[K]$, and the last inequality holds because K is a k-club that contains a and d. This shows that inequalities (2.12) and (2.13) hold at equality, and so the following holds.

$$\begin{aligned}\operatorname{dist}_{G_s}(d,s)+\operatorname{dist}_{G_s}(s,a)=\operatorname{dist}_{G_s}(d,a)\leq \operatorname{dist}_{G[K]}(d,a)\leq k\\ \operatorname{dist}_{G_s}(d,s)+\operatorname{dist}_{G_s}(s,b)=\operatorname{dist}_{G_s}(d,b)\leq \operatorname{dist}_{G[K]}(d,b)\leq k.\end{aligned}$$

This shows that property 2 fails. □

Theorem 4. *The length-k a,b-separator inequality $x_a+x_b\leq 1+x(S)$ induces a facet of the k-club polytope $\mathrm{CLUB}_k(G)$ of G if and only if:*

1. *S is a minimal length-k a,b-separator; and*
2. *no vertex $d\in V\setminus S$ satisfies properties 1 and 2 of Lemma 1.*

Proof. ($\implies$) Suppose that $x_a+x_b-x(S)\leq 1$ is facet-defining. If S is not minimal (i.e., there is a $S'\subsetneq S$ that is also a length-k a,b-separator), then the valid inequalities $x_a+x_b-x(S')\leq 1$ and $-x(S\setminus S')\leq 0$ imply $x_a+x_b-x(S)\leq 1$, so it cannot induce a facet. So, S is minimal. Similarly, if some vertex d satisfies properties 1 and 2 of Lemma 1, we can write the valid inequalities $x_a+x_b+x_d-x(S)\leq 1$ and $-x_d\leq 0$ which imply $x_a+x_b-x(S)\leq 1$, so it cannot induce a facet.

($\impliedby$) Suppose that S is a minimal length-k a,b-separator and that no vertex $d\in V\setminus S$ satisfies properties 1 and 2 of Lemma 1. We already know that the inequality $x_a+x_b-x(S)\leq 1$ is valid. So, to show that it is facet-defining, we will give n different k-clubs whose characteristic vectors are affinely independent and belong to the face where the inequality $x_a+x_b-x(S)\leq 1$ holds at equality. To do so, we will need some notations. Define

$$D:=\{v\in V\setminus S \mid \operatorname{dist}_{G-S}(a,v)>k,\ \operatorname{dist}_{G-S}(b,v)>k\}.$$

We refer to paths that start at vertex $v\in V$ and end at some vertex of $U\subseteq V$ as *v-U paths.* A *shortest v-U path* has the shortest length among all v-U paths. Finally, denote by $\operatorname{length}(P)$, hops(P), and $V(P)$ as the (edge-weighted) length, number of edges, and the vertex set of path P, respectively.

We construct n different k-clubs as follows. See Figure 2.5 for an illustration.

- For every vertex $v \in V \setminus (S \cup D)$, do the following. Let $R = \{a, b\}$. Among the shortest v-R paths in graph $G - S$, pick a path P_v having the fewest number of edges. Then, order the vertices of $V \setminus (S \cup D)$ as $(v_1, v_2, ..., v_q)$, where $q := n - |S| - |D|$, so that:

    1. $\text{length}(P_{v_1}) \leq \text{length}(P_{v_2}) \leq \cdots \leq \text{length}(P_{v_q})$, and
    2. if $i < j$ and $\text{length}(P_{v_i}) = \text{length}(P_{v_j})$, then $\text{hops}(P_{v_i}) \leq \text{hops}(P_{v_j})$.

  This can be obtained by sorting the vertices $v \in V \setminus (S \cup D)$ by the lengths of their paths P_v, breaking any ties by hops(P_v). This gives the k-club denoted by $K_i := V(P_{v_i})$.

- For every vertex $s \in S$, do the following. Consider a shortest a-b path P_s in graph $G_s := G - (S \setminus \{s\})$. Such a path exists and has length at most k by minimality of S. This gives the k-club $V(P_s)$.

- For every vertex $d \in D$, do the following. Let $S_d \subseteq S$ be the set of vertices violating property 2 of Lemma 1. Among the shortest d-S_d paths in graph G, pick a path P_d that has the fewest number of edges. Then, order the vertices of D as $(d_1, d_2, \ldots, d_t)$, where $t := |D|$, so that:

    1. $\text{length}(P_{d_1}) \leq \text{length}(P_{d_2}) \leq \cdots \leq \text{length}(P_{d_t})$, and
    2. if $i < j$ and $\text{length}(P_{d_i}) = \text{length}(P_{d_j})$, then $\text{hops}(P_{d_i}) \leq \text{hops}(P_{d_j})$.

  This can be obtained by sorting the vertices $d \in D$ by the lengths of their paths P_d, breaking any ties by hops(P_d). This gives the k-club denoted by $T_i := V(P_{d_i}) \cup V(P_s)$, where s is the endpoint of P_{d_i} that belongs to S, and P_s is defined as above.

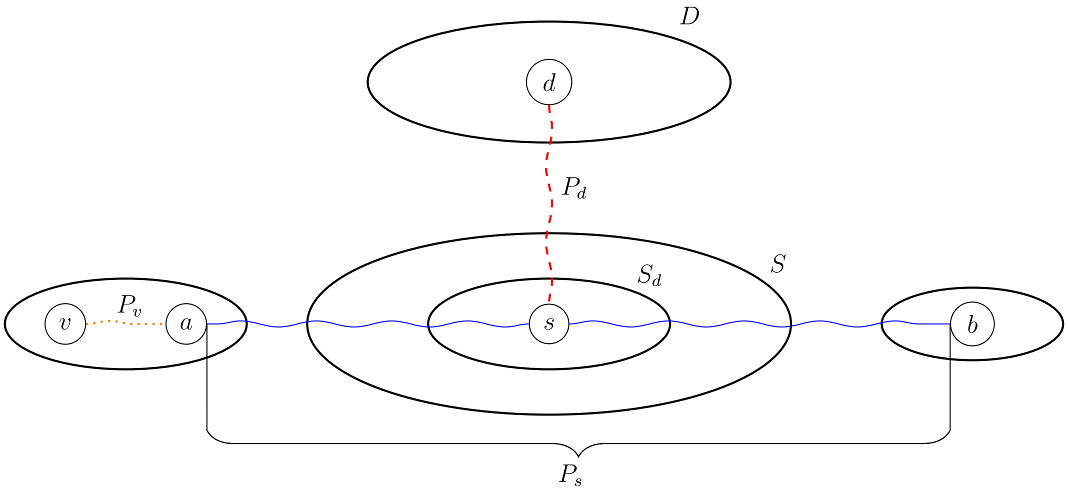


Figure 2.5: Construction of n different k-clubs in the proof of Theorem 4.

We claim that the characteristic vectors of the aforementioned k-clubs each belong to the face where the inequality $x_a + x_b - x(S) \leq 1$ holds at equality. This is true for the k-clubs K_i as they contain precisely one vertex from $R = \{a, b\}$ and no vertices from S. This is also true for the k-clubs $V(P_s)$ and T_i as they contain both a and b, as well as one vertex from S.

We claim that the characteristic vectors of the aforementioned k-clubs are linearly independent and thus affinely independent. To see this, it is enough to show that these (column) vectors, arranged from left to right in the order in which they were described, form an upper triangular matrix with ones on the main diagonal, as depicted in Figure 2.6. (The entries of each column are to be arranged so that the first q entries correspond to $(v_1, v_2, \ldots, v_q)$, the next $|S|$ entries correspond to the vertices of S, and the last $|D|$ entries correspond to $(d_1, d_2, \ldots, d_t)$.) By our construction of the k-clubs, it can be seen that the center submatrix is the identity matrix and that the three submatrices near the lower-left corner are zero matrices. So, all that is left to show is that the upper-left and lower-right submatrices are themselves upper triangular.

To prove that the upper-left submatrix is upper triangular, we are to show that each k-club K_i is a subset of $\{v_1, v_2, \ldots, v_i\}$. Suppose that this is not true for some k-club K_i.

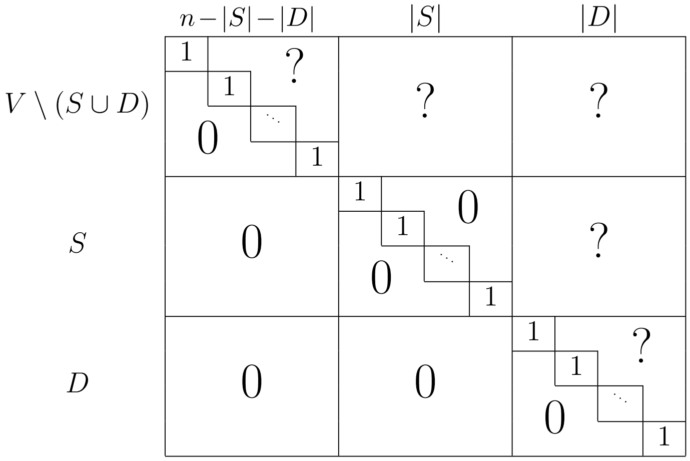


Figure 2.6: A matrix whose columns represent the k-clubs in the proof of Theorem 4.

Then, K_i contains a vertex $v_j \in V \setminus (S \cup D)$ with $i < j$. By construction of K_i, this implies that vertex v_j is on a shortest v_i-R path in $G - S$, and by the construction of the ordering $(v_1, v_2, \ldots, v_q)$, this implies that $\text{length}(P_{v_i}) \leq \text{length}(P_{v_j})$. Let P'_{v_j} be the subpath of P_{v_i} that starts at v_j and ends at (a vertex of) R. Then,

$$\text{length}(P_{v_i}) \leq \text{length}(P_{v_j}) \leq \text{length}(P'_{v_j}) \leq \text{length}(P_{v_i}).$$

Here, the middle inequality holds by definition of P_{v_j}, and the last inequality holds because P'_{v_j} is a subpath of P_{v_i} and edges have nonnegative weights. Thus,

$$\text{length}(P_{v_i}) = \text{length}(P_{v_j}) = \text{length}(P'_{v_j}),$$

which in turn implies that

$$\text{hops}(P_{v_i}) \leq \text{hops}(P_{v_j}) \leq \text{hops}(P'_{v_j}) < \text{hops}(P_{v_i}).$$

Here, the first inequality holds by the construction of the ordering in addition to the observation that $\text{length}(P_{v_i}) = \text{length}(P_{v_j})$, the middle inequality holds by definition of P_{v_j} together with the fact that length$(P_{v_j}) = \text{length}(P'_{v_j})$, and the last inequality holds because P'_{v_j} is a proper subpath of P_{v_i}. This has given us the contradiction hops$(P_{v_i}) < \text{hops}(P_{v_i})$, which means that our assumption that $v_j \in K_i$ cannot hold. Thus, it is true that $K_i \subseteq \{v_1, v_2, \ldots, v_i\}$, and so the upper-left submatrix is upper triangular.

To prove that the lower-right submatrix is upper triangular, we are to show that each k-club T_i is a subset of $V \setminus \{d_{i+1}, \ldots, d_t\}$. Suppose that this is not true for some k-club T_i. Then, T_i contains a vertex $d_j \in D$ with $i < j$. By construction of T_i, this implies that vertex d_j is on a shortest d_i-S_{d_i} path in G, and by the construction of the ordering $(d_1, d_2, \ldots, d_t)$, this implies that length$(P_{d_i}) \leq \text{length}(P_{d_j})$. Let P'_{d_j} be the subpath of P_{d_i} that starts at d_j and ends at (a vertex of) S_{d_i}. Then,

$$\text{length}(P_{d_i}) \leq \text{length}(P_{d_j}) \leq \text{length}(P'_{d_j}) \leq \text{length}(P_{d_i}).$$

Here, the middle inequality holds by definition of P_{d_j}, and the last inequality holds because P'_{d_j} is a subpath of P_{d_i} and edges have nonnegative weights. Thus,

$$\text{length}(P_{d_i}) = \text{length}(P_{d_j}) = \text{length}(P'_{d_j}),$$

which in turn implies that

$$\text{hops}(P_{d_i}) \leq \text{hops}(P_{d_j}) \leq \text{hops}(P'_{d_j}) < \text{hops}(P_{d_i}).$$

Here, the first inequality holds by the construction of the ordering in addition to the observation that length$(P_{d_i}) = \text{length}(P_{d_j})$, the middle inequality holds by definition of P_{d_j} together with the fact that length$(P_{d_j}) = \text{length}(P'_{d_j})$, and the last inequality holds because P'_{d_j} is a proper subpath of P_{d_i}. This has given us the contradiction hops$(P_{d_i}) < \text{hops}(P_{d_i})$, which means that our assumption that $d_j \in T_i$ cannot hold. Thus, it is true that $T_i \subseteq V \setminus \{d_{i+1}, \ldots, d_t\}$, and so the lower-right submatrix is upper triangular. This concludes the proof. □

2.5.2 Formulation Strength

Here we show that the cut-like formulation is at least as strong as the path-like formulation, generalizing the $k = 3$ result of Almeida and Carvalho (2012). When $k \geq 3$, we also characterize the graphs for which the cut-like formulation is integral as those with no 3-vertex independent set.

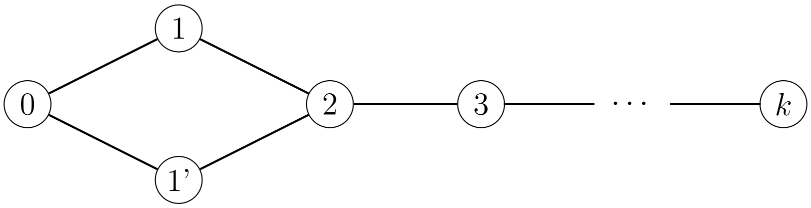


Figure 2.7: When $k \geq 3$, setting $x_0^* = x_k^* = 1$ and $x_i^* = \frac{1}{2}$ for all other nodes is feasible for the path-like formulation but not for the cut-like formulation. Indeed, the length-k a,b-separator inequality $x_0 + x_k \leq 1 + x_2$ is violated.

Proposition 3. *The cut-like formulation is (always) at least as strong as the path-like formulation. This inclusion is strict for the hop-based cases $k \geq 3$. They are equally strong in the hop-based case $k = 2$.*

Proof. First we show that the cut-like formulation is always at least as strong as the path-like formulation. Suppose that x^* satisfies the LP relaxation of the cut-like formulation. For each minimal length-k a,b-connector $C \in \mathcal{C}_{ab}^k$, let i_C be a minimum-weight vertex of C, i.e., $i_C \in C$ and $x_{i_C}^* = \min\{x_i^* \mid i \in C\}$, and set $y_C^* = x_{i_C}^*$. We show that (x^*, y^*) satisfies the LP relaxation of the path-like formulation. Obviously, (x^*, y^*) satisfies the constraints (2.5c) and the 0-1 bounds. So, consider constraint (2.5b) for some nonadjacent vertices a and b. Observe that $S := \cup_{C \in \mathcal{C}_{ab}^k} \{i_C\}$ is a length-k a,b-separator, so $x_a^* + x_b^* \leq 1 + \sum_{s \in S} x_s^*$. So, constraint (2.5b) is satisfied as

$$x_a^* + x_b^* \leq 1 + \sum_{s \in S} x_s^* \leq 1 + \sum_{C \in C_{ab}^k} x_{i_C}^* = 1 + \sum_{C \in C_{ab}^k} y_C^*.$$

Note that some x_v^* may make multiple appearances in the sum $\sum_{C \in C_{ab}^k} x_{i_C}^*$ but only once in the sum $\sum_{s \in S} x_s^*$, hence the middle inequality.

The equality of the path-like and cut-like LP relaxations is easy to see in the hop-based case $k = 2$, since the IP formulations themselves are equivalent. Figure 2.7 shows that the inclusion can be strict when $k \geq 3$. □

Recall that the independence number of a graph G, denoted by $\alpha(G)$, is the size of a maximum independent set in G.

Proposition 4. *Under hop-based distances and $k \geq 3$, the cut-like formulation is integral for graph G if and only if G has no 3-vertex independent set (i.e., $\alpha(G) \leq 2$).*

Proof. Suppose $k \geq 3$ and let $Q_k(G)$ be the LP relaxation of the cut-like formulation (including the 0-1 bounds). Without loss of generality, suppose that $k \leq n - 1$. Since the formulation is correct, we are proving the statement that $Q_k(G) = \text{CLUB}_k(G)$ if and only if $\alpha(G) \leq 2$. Recall that $Q_k(G)$ and $\text{CLUB}_k(G)$ are both full dimensional and thus have unique half-space representations up to scalar multiples.

($\implies$) By the contrapositive. Suppose there is an independent set I with three vertices. Then, the inequality $x(I) \leq 1$ induces a facet of $\text{CLUB}_k(G[I])$. Lifting this seed inequality shows that $\text{CLUB}_k(G)$ has a facet-defining inequality of the form $x(I) + \sum_{i \in V \setminus I} \pi_i x_i \leq 1$ with at least three positive coefficients. But, this inequality is not part of the definition of $Q_k(G)$, so $Q_k(G) \neq \text{CLUB}_k(G)$.

($\impliedby$) Suppose $\alpha(G) \leq 2$. Observe that the subgraph $G[S]$ induced by a vertex set S is either disconnected or has $\text{diam}(G[S]) \leq 3$. (Otherwise, if $G[S]$ is connected with diam($G[S]$) ≥ 4, then a 3-vertex independent set $\{v_0, v_2, v_4\}$ can be obtained from a diameter-inducing path v_0-v_1-v_2-v_3-v_4-$\cdots$ of $G[S]$.) Thus, $\text{CLUB}_{n-1}(G) = \text{CLUB}_3(G)$. Wang et al. (2017) have shown that $\text{CLUB}_{n-1}(G) = Q_{n-1}(G)$ when $\alpha(G) \leq 2$. So,

$$\text{CLUB}_3(G) \subseteq \text{CLUB}_k(G) \subseteq Q_k(G) \subseteq Q_{n-1}(G) = \text{CLUB}_{n-1}(G) = \text{CLUB}_3(G),$$

and thus $Q_k(G) = \text{CLUB}_k(G)$. □

2.5.3 Separation Problem and Extended Formulations

Here we discuss the separation problem for the length-k a, b-separator inequalities (2.10b), particularly for the case that distances are measured in hops. Since there are only $\mathcal{O}(n^2)$ minimal inequalities when $k = 2$, we will ignore that case. We observe that separation is polynomial-time solvable for $k \in \{3, 4\}$ by reduction to min-cut, and show that separation is hard for each $k \geq 5$. By the equivalence of separation and optimization (Grötschel et al., 1993), we can optimize over the LP relaxations of the cut-like formulation in polynomial time when $k \in \{3, 4\}$, but this is hard when $k \geq 5$.

By standard arguments, the cut-like formulation admits a polynomial-size extended formulation when $k \in \{3, 4\}$. This follows because the separation problem can be written as a (particular) min-cut problem, which can be solved via a linear program. By techniques of Martin (1991), this immediately gives the polynomial-size extended formulations. However, the resulting formulations are too large to be practical, so we do not discuss them further. The interested reader is invited to consult Lovász et al. (1978) or Xu (2001) for more details about the reduction to min-cut.

The separation problem for the cut-like formulation can be stated as follows.

Problem: Separation for length-k a, b-separator inequalities (2.10b).

Input: A graph $G = (V, E)$, vertex weights $x^* \in [0, 1]^n$, positive integer k.

Output: (if any exist) nonadjacent vertices $a, b \in V$ and a length-k a,b-separator $S \subseteq V \setminus \{a, b\}$ such that $x^*_a + x^*_b - \sum_{i \in S} x^*_i > 1$.

This problem is closely related to the Length-Bounded Node-Cut problem, which is known to be NP-hard for each $L \geq 5$ and easy for $L \in \{2, 3, 4\}$. Hardness for $L \geq 5$ was shown by Baier et al. (2010), and polynomiality when $L = 4$ essentially follows by Lovász et al. (1978). The optimization version of this problem is given below, and the associated decision problem will be styled Length-Bounded Node Cut.

Problem: Length-Bounded Node-Cut.

Input: A graph $G = (V, E)$, nonadjacent nodes p and q, a weight $w_v \geq 0$ for each $v \in V \setminus \{p, q\}$, and a positive integer L.

Output: A vertex subset $S \subseteq V \setminus \{p, q\}$ of minimum weight $\sum_{i \in S} w_i$ such that $\text{dist}_{G-S}(p, q) > L$ (with respect to hop-based distances).

Separation is hard for $k \geq 5$

Here, we show that it is hard to separate the length-k a,b-separator inequalities when $k \geq 5$ and distances are hop-based. As an easy consequence, separation is hard for all $k > 0$ under weighted distances.

Theorem 5. *For each $k \geq 5$, it is coNP-complete to determine whether a given $x^* \in \mathbb{R}^n$ satisfies all length-k a,b-separator inequalities* (2.10b)*.*

Proof. Membership in coNP is clear, as a suitable witness for a "no" instance is given by nonadjacent nodes a and b and a length-k a,b-separator $S \subseteq V \setminus \{a, b\}$. Consider an instance of Length-Bounded Node-Cut given by graph $G = (V, E)$, nodes p and q, and distance threshold $L \geq 5$ and a target weight W for the cut S. We suppose that the node weights w_v are all equal to one. This variant of Length-Bounded Node-Cut is NP-hard (Baier et al., 2010).

Let $k = L$ and $n = |V|$. We construct an $x^* \in [0,1]^n$ that violates a length-k a, b-separator inequality (2.10b) for G if and only if G has a length-L-bounded node-cut of size W. Namely, let $x^*_p = x^*_q = \frac{2n+1}{4n}$ and $x^*_i = \frac{1}{2n(W+1)}$ for all other nodes i of G.

($\Longleftarrow$) Suppose G has a length-L-bounded node-cut $S' \subseteq V$ of size W. Then x^* violates the inequality (2.10b) for $a = p$, $b = q$, and $S = S'$, since

$$\begin{aligned} x^*_a + x^*_b - \sum_{i \in S} x^*_i &= x^*_p + x^*_q - \sum_{i \in S'} x^*_i \\ &= \frac{2n+1}{4n} + \frac{2n+1}{4n} - W\left(\frac{1}{2n(W+1)}\right) \\ &= 1 + \frac{1}{2n(W+1)} > 1. \end{aligned}$$

($\Longrightarrow$) Suppose that x^* violates a length-k a, b-separator inequality (2.10b) with length-k a, b-separator $S \subseteq V \setminus \{a, b\}$. Then no vertex of $V \setminus \{p, q\}$ can belong to $\{a, b\}$ since otherwise

$$x^*_a + x^*_b - \sum_{i \in S} x^*_i \leq x^*_a + x^*_b \leq \frac{2n+1}{4n} + \frac{1}{2n(W+1)} \leq 1,$$

and the inequality (2.10b) is satisfied. Thus, $\{a, b\} = \{p, q\}$. Then,

$$\begin{aligned} 1 < x^*_a + x^*_b - \sum_{i \in S} x^*_i &= x^*_p + x^*_q - \sum_{i \in S} x^*_i \\ &= \frac{2n+1}{4n} + \frac{2n+1}{4n} - |S|\left(\frac{1}{2n(W+1)}\right). \end{aligned}$$

So, $|S| < W + 1$, and S is a length-L-bounded node-cut of size $\leq W$. □

Separation is easy for $k \in \{2, 3, 4\}$

As noted above, the Length-Bounded Node-Cut problem is polynomial-time solvable for $L \in \{2, 3, 4\}$. The cases $L = 2$ and $L = 3$ can be considered folklore, and the case $L = 4$ was essentially shown by Lovász et al. (1978) by reduction to min-cut. The min-cut instance that is created has linear size with respect to the input graph, so Length-Bounded Node-Cut with $L = 4$ can be solved in time $\mathcal{O}(mn)$ (Orlin, 2013).

Thus, to solve the separation problem for the length-k a, b-separator inequalities when given $x^* \in [0,1]^n$, one can solve, for each pair $\{a, b\}$ of nonadjacent vertices, a Length-Bounded Node-Cut problem to find a minimum-weight S and then check whether $x^*_a + x^*_b - \sum_{i \in S} x^*_i > 1$. Since there are $\binom{n}{2} - m$ a, b-pairs, the total running time would be $\mathcal{O}(mn^3)$.

There are some ways to speed up separation in practice. For example, if $x^*_a + x^*_b \leq 1$, then certainly all length-k a, b-separator inequalities will be satisfied. The same holds if $x^*_a + x^*_b \leq 1 + x^*(N(a) \cap N(b))$ since every length-k a, b-separator has $N(a) \cap N(b)$ as a subset. However, we are unaware of a way to achieve a better worst-case running time. For these and other reasons (e.g., poor initial performance, simplicity of the approach, ease of implementation), we do not employ fractional separation in our implementation. As such, we do not bother detailing the Length-Bounded Node-Cut algorithms for $L \in \{3, 4\}$ that are implied by Lovász et al. (1978).

2.6 Computational Experiments

Here we evaluate the performance of the path-like and cut-like formulations with that of existing formulations and approaches for solving the maximum k-club problem.

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 are implemented in Microsoft Visual Studio 2015 in C++ for Gurobi version 7.5.1. We impose a time limit of 3600 seconds on each instance and set the `method` parameter to concurrent.

First, we propose a heuristic for maximum k-club that is based on the k-clique and DROP heuristic of Bourjolly et al. (2000). Then, we describe a preprocessing procedure. Next, due to the exponential number of constraints defining the cut-like formulation, we explain our implementation of it. Then, we compare running times for the different formulations on test instances considered by Shahinpour and Butenko (2013a) and Moradi and Balasundaram (2018). We also propose a decomposition procedure to handle the exorbitant number of conflicts generated for some of the larger instances. Lastly, we compare running times on the synthetic instances considered by Veremyev and Boginski (2012) and Moradi and Balasundaram (2018). For simplicity, we limit ourselves to hop-based distances in the experiments. Accordingly, all discussion (for heuristics, preprocessing, and separation) concerns only the hop-based case, although most, if not all, of the ideas straightforwardly extend to the edge-weighted case.

2.6.1 Heuristic and Preprocessing

The heuristic used in our implementation is based on the k-clique and DROP heuristic of Bourjolly et al. (2000). Recall the definition of a (distance) k-clique.

Definition 4 (k-clique)**.** *A subset $S \subseteq V$ of vertices in a graph $G = (V, E)$ is called a (distance) k-clique if, for every two vertices $i, j \in S$, $\mathrm{dist}_G(i, j) \leq k$.*

The following pseudocode for k-clique and DROP uses two notions that we have not discussed yet. The first is the k-th power $G^k = (V^k, E^k)$ of a graph $G = (V, E)$, which has the same vertex set, i.e., $V^k = V$, and two nodes u, v in G^k are adjacent if their distance $\mathrm{dist}_G(u, v)$ in G is at most k. Thus, a k-clique in G is equivalent to a clique in G^k. The second is the k-hop neighborhood $N^k_G(v) = \{w \in V \mid \mathrm{dist}_G(v, w) \leq k\}$ of a vertex v. Observe that v belongs to $N^k_G(v)$, assuming $k \geq 0$.

k-clique and DROP (Bourjolly et al. (2000)):

1. find a maximum k-clique S in G (i.e., a maximum clique in G^k);

2. while S is not a k-club in G do
   - pick $v \in S$ with the fewest nearby nodes $|N^k_{G[S]}(v) \cap S|$ in $G[S]$;
   - $S \leftarrow S \setminus \{v\}$;
3. return S.

One possible problem is that step 1 of k-clique and DROP requires us to find a maximum k-clique in G, which is NP-hard, even to approximate (Asahiro et al., 2018). This motivates our modification, which is to heuristically find a large k-clique.

Our heuristic:

1. initialize $S \leftarrow V$ and create G^k;
2. while S is not a clique in G^k do
   - pick $v \in S$ of minimum degree in $G^k[S]$;
   - $S \leftarrow S \setminus \{v\}$;
3. while S is not a k-club in G do
   - pick $v \in S$ with the fewest nearby nodes $|N^k_{G[S]}(v) \cap S|$ in $G[S]$;
   - $S \leftarrow S \setminus \{v\}$;
4. return S.

Creating the k-th power graph G^k takes time $\mathcal{O}(nm)$ by running BFS from each node. Step 2, which finds a maximal clique in G^k, can be implemented to run in time linear in the size of G^k (Matula and Beck, 1983) and finds large cliques in practice (Walteros and Buchanan, 2019). Step 3, which performs DROP, can take time $\mathcal{O}(nm)$ in each of the $\mathcal{O}(n)$ iterations of the while loop. This gives a time bound of $\mathcal{O}(mn^2)$, which is rather pessimistic given that the number of iterations of the while loop in step 3 is often zero in practice.

Moreover, the time bound $\mathcal{O}(nm)$ within each iteration is also pessimistic given that when this loop is entered, S is much smaller than n. Indeed, as the results in Table 2.1 show, the times for our heuristic are very reasonable in practice. The longest time of 22.73 seconds is for the graph `cs4` with $k = 4$. (This is the sum of the heuristic time and the preprocessing time.) For context, our implementation takes 23 seconds to compute the diameter of this graph using BFS.

			$k = 2$			$k = 3$			$k = 4$		
Graph	n	m	heur	n'	time	heur	n'	time	heur	n'	time
karate	34	78	17	31	0.00	25	25	0.00	33	33	0.00
dolphins	62	159	10	55	0.00	29	35	0.00	39	47	0.00
lesmis	77	254	37	37	0.00	58	58	0.00	75	75	0.00
polbooks	105	441	28	83	0.00	53	54	0.00	62	104	0.00
adjnoun	112	425	50	50	0.00	82	104	0.01	107	107	0.00
football	115	613	12	115	0.00	24	115	0.01	115	115	0.01
jazz	198	2742	102	157	0.01	174	181	0.01	192	192	0.01
celegansn	297	2148	135	135	0.01	243	274	0.03	295	295	0.02
celegansm	453	2025	238	238	0.03	371	389	0.04	432	433	0.03
email	1133	5451	47	747	0.08	192	991	0.51	642	1053	0.46
polblogs	1490	16715	288	928	0.28	767	1115	0.29	1126	1187	0.35
netscience	1589	2742	35	35	0.03	54	54	0.03	85	85	0.02
add20	2395	7462	124	124	0.22	671	671	0.29	1454	1454	0.35
data	2851	15093	14	2080	0.33	28	2171	0.41	46	2255	0.34
uk	4824	6837	4	4792	0.69	7	4709	0.70	9	4728	0.82
power	4941	6594	20	20	1.00	30	30	0.94	61	61	0.88
add32	4960	9462	32	64	0.74	99	99	0.69	268	268	0.77
hep-th	8361	15751	51	51	1.76	114	1330	1.90	318	2143	2.07
whitaker3	9800	28989	8	9800	3.95	10	9800	3.91	19	9800	4.05
crack	10240	30380	8	10240	4.94	15	10237	5.05	25	10237	5.52
PGPgiantc	10680	24316	206	206	5.27	422	536	5.27	1161	1161	5.57
cs4	22499	43858	5	22499	22.28	7	22499	22.42	9	22499	22.73

Table 2.1: Heuristics and preprocessing on DIMACS-10 graphs. For each k, we report the heuristic's objective (heur), the number of remaining vertices after preprocessing (n'), and the total time in seconds for the heuristic and preprocessing (time).

Once the heuristic terminates, we remove many vertices from the graph in a preprocessing step (that is, prior to invoking the MIP solver). Specifically, if the heuristic gives us a k-club of size p, we can remove all vertices v that have fewer than p nodes in their k-hop neighborhood $N_G^k(v)$. This can be done iteratively, since a deleted vertex may impact the size of the remaining vertices' k-hop neighborhoods. This is essentially the k-core peeling used for the maximum clique problem (Abello et al., 1999; Verma et al., 2015) which is also used for k-club (Veremyev and Boginski, 2012; Moradi and Balasundaram, 2018). We implement it as follows, where the $(p-1)$-core of a graph G is the (unique) inclusion-maximal subgraph of G that has minimum degree at least $p-1$ (Seidman, 1983).

Preprocessing when given a lower bound p on the k-club number:

1. create the k-th power graph G^k;
2. find the $(p-1)$-core $G' = (V', E')$ of G^k;
3. return $G[V']$.

As before, step 1 takes time $\mathcal{O}(nm)$ when distances are hop-based. Step 2 takes linear time $\mathcal{O}(|V'| + |E'|) = \mathcal{O}(n^2)$ with respect to G' by the algorithm of Matula and Beck (1983)[5]. Thus, the total time $\mathcal{O}(nm)$ is dominated by step 1.

Table 2.1 shows that the time spent on this preprocessing is reasonable and the reduction in graph size can be substantial. On these 66 instances, our heuristic finds (what turn out to be) 36 optimal solutions, and the preprocessing is able to prove optimality for 26 of them.

2.6.2 Implementing the Cut-Like Formulation

Since the cut-like formulation can have exponentially many constraints when $k \geq 3$, we add them on-the-fly only as needed. For this functionality, we invoke the Gurobi parameter `LazyConstraints`. Our implementation proceeds roughly as follows.

Initialize the formulation with the conflict constraints $x_a + x_b \leq 1$ for vertices a and b that are far apart in G, i.e., $\text{dist}_G(a, b) > k$. We could instead add stronger cuts of the form

[5] See also the later implementation of Batagelj and Zaversnik (2003) which uses 3 arrays instead of linked lists. Our implementation is the same as Walteros and Buchanan (2019), which uses 3 arrays and *always* pulls off a vertex of minimum degree in the remaining graph. This second property is satisfied in the implementation of Matula and Beck, but not of Batagelj and Zaversnik.

$x(I) \leq 1$, where $I \subseteq V$ is an independent set in the k-th power graph G^k. However, MIP solvers detect these stronger cuts automatically and effectively through *clique merging* (Achterberg et al., 2019).

Then, when the MIP solver encounters a possible solution $x^* \in \{0,1\}^n$ that satisfies the initial constraints, we check if the selected vertices $K = \{i \in V \mid x_i^* = 1\}$ form a k-club. If not, then for every pair of "far" vertices a and b in $G[K]$ (i.e., $\text{dist}_{G[K]}(a,b) > k$), we detect and add a violated minimal length-k a,b-separator inequality. In particular, observe that in this case $S' = V \setminus K$ will be a length-k a,b-separator and we could add the violated inequality $x_a + x_b \leq 1 + x(S')$. But, this inequality can and should be strengthened to $x_a + x_b \leq 1 + x(S)$, where S is a minimal subset of S' that is also a length-k a,b-separator. For this, we use Minimalize as follows.

Minimalize:

1. initialize $S \leftarrow S'$;
2. for $s \in S$ do
   - compute $\text{dist}_{G_s}(a,s)$ and $\text{dist}_{G_s}(s,b)$, where $G_s = G - (S \setminus \{s\})$;
   - if $\text{dist}_{G_s}(a,s) + \text{dist}_{G_s}(s,b) > k$, then update $S \leftarrow S \setminus \{s\}$;
3. return S.

Observe that Minimalize returns a minimal length-k a,b-separator and that its running time is $\mathcal{O}(|S'|m) = \mathcal{O}(nm)$ when distances are hop-based. In practice, one can speed up Minimalize by first removing vertices v from S' that are not on a length-k path from a to b in G, i.e., $\text{dist}_G(a,v) + \text{dist}_G(v,b) > k$; this takes linear time when distances are hop-based. Also, a vertex v from S that neighbors both a and b will belong to every minimal length-k a,b-separator and can be fixed in S, i.e., skipped in step 2.

We experimented with a few alternative implementations but did not pursue them due to relatively poor initial performance. For example, worse performance was observed when adding a single violated inequality (instead of adding one violated inequality for every far pair $\{a, b\}$). Attempts at fractional separation for $k = 3$ were also unsuccessful, although the implementation was rather primitive. It is possible that more advanced implementations would improve the results, e.g., using heuristic or approximate separation (Baier et al., 2010), but we were already content with the performance of lazy integer separation. Experiments with lifting (using Lemma 1 or using another quicker approach) did not improve the performance. These unsuccessful attempts with lifting can be found in the commented portions of the callback code.

In the case that the graph is disconnected after preprocessing, we still solve a single MIP. Others, like Moradi and Balasundaram (2018), solve each component separately. This leads to implementation questions such as: Which components should be solved first? Instead, like Carvajal et al. (2013), we create a binary variable z_j for each component G_j of G representing the decision to pick the k-club from within component G_j. We impose that one component is chosen ($\sum_j z_j = 1$) and that a vertex i cannot be chosen if its component G_j is not selected ($x_i \leq z_j$). This adds only $\mathcal{O}(n)$ variables and constraints to the formulation and avoids many implementation questions.

2.6.3 Results for Real-Life Instances

Here we compare running times on the instances considered by Shahinpour and Butenko (2013a) and Moradi and Balasundaram (2018) that were drawn from the 10th DIMACS Implementation Challenge on Graph Partitioning and Graph Clustering (DIMACS-10, 2017). We exclude all instances that were solved by the heuristic and preprocessing from Section 2.6.1 as they would provide little insight. To make the comparisons as fair as possible, all experiments are conducted using the same computer, MIP solver, heuristic solution, and preprocessing. We employ the connected component variables z_j in each formulation.

We explore two different implementations of the $k = 2$ common-neighbor formulation. In the first implementation, which we refer to as CN, we add all cut-like constraints initially. Meanwhile, implementation CUT only adds the conflict constraints $x_a + x_b \leq 1$ upfront; other cut-like inequalities are added on-the-fly.

Table 2.2 gives the time to solve the $k = 2$ common-neighbor formulation (using CN and CUT implementations), the maximum 2-club sizes, and the number of constraints added for each implementation. As the table shows, implementation CUT avoids adding many of the constraints defining the CN formulation and is faster. For example, for `polblogs`, it suffices to add 171,237 out of 621,498 cut-like constraints, reducing the solve time from 19.49 seconds to 4.93 seconds.

				# constraints		solve time, $k = 2$	
Graph	n	m	$\bar{\omega}_2$	CN	CUT	CN	CUT
karate	34	78	18	456	144+0	0.02	0.01
dolphins	62	159	13	1,670	927+16	0.06	0.03
polbooks	105	441	28	4,346	1,847+0	0.10	0.05
football	115	613	16	5,942	3,636+92	2.65	1.49
jazz	198	2742	103	12,652	1,326+1	0.33	0.06
email	1133	5451	72	519,948	227,841+0	11.41	3.94
polblogs	1490	16715	352	621,498	171,237+0	19.49	4.93
data	2851	15093	18	3,692,869	2,122,988+0	1947.13	1224.76
uk	4824	6837	5	11,576,836	11,459,801+0	3332.52	3266.55
add32	4960	9462	32	223,111	1,017+0	1.77	0.81
whitaker3	9800	28989	9	47,986,111	47,928,432+0	[8,18]	[8,18]
crack	10240	30380	10	52,393,300	52,315,672+0	[9,18]	[9,18]
cs4	22499	43858	6	253,047,393	252,937,499+0	MEM	MEM

Table 2.2: Maximum 2-club sizes on DIMACS-10 graphs. We report the total number of cut-like constraints to solve the $k = 2$ common neighbor formulation using CN and CUT implementations. For each implementation, we also report the total time in seconds (including preprocessing, heuristic, and model build time), or the best lower and upper bounds [LB,UB] within a 3600 second time limit. Cases where using an implementation leads to memory crashes are reported as MEM.

In Table 2.3 and Table 2.4, we compare the time to solve the maximum k-club problem for $k \in \{3, 4\}$ using the following formulations:

- recursive (R), resembling (Veremyev and Boginski, 2012; Veremyev et al., 2015),
- canonical hypercube cut (CHC), by Moradi and Balasundaram (2018),
- path-like (PATH), and
- cut-like (CUT).

				solve time, $k = 3$			
Graph	n	m	$\bar{\omega}_3$	R	CHC	PATH	CUT
dolphins	34	78	29	0.54	0.01	0.04	0.01
polbooks	105	441	53	1.01	0.01	0.07	0.01
adjnoun	112	425	82	6.48	0.03	0.39	0.01
football	115	613	58	[24,63]	[24,67]	8.11	0.29
jazz	198	2742	174	69.43	0.03	3.14	0.03
celegansn	297	2148	243	96.85	5.03	4.41	0.07
celegansm	453	2025	371	337.12	0.08	6.44	0.07
email	1133	5451	212	LPNS	[192,233]	[208,239]	241.42
polblogs	1490	16715	776	LPNS	3.37	1132.06	1.67
data	2851	15093	32	LPNS	[32,36]	[28,47]	[31,36]
uk	4824	6837	8	MEM	[7,16]	[7,17]	[7,16]
hep-th	8361	15751	120	[114,124]	[114,123]	2676.83	171.82
whitaker3	9800	28989	15	MEM	[13,32]	[13,32]	[13,32]
crack	10240	30380	17	MEM	[15,31]	[15,31]	[15,31]
PGPgiantc	10680	24316	422	633.09	5.84	27.63	5.82
cs4	22499	43858	12	MEM	MEM	MEM	MEM

Table 2.3: Results for DIMACS-10 graphs. For each $k = 3$ and each formulation, we report the total time in seconds (including preprocessing, heuristic, and model build time), or the best lower and upper bounds [LB,UB] within a 3600 second time limit. Cases where the LP relaxation were not solved within the time limit (due either to build time or solve time) are reported as LPNS and cases where using a formulation leads to memory crashes are reported as MEM.

Table 2.3 and Table 2.4 demonstrate that CUT performs the best among the four formulations. It solves all of the instances that R, CHC, and PATH can solve, plus 7, 5, and 4 others, respectively. CUT's best performance (relative to the other formulations) is on `email` with $k = 4$, where it finishes in 1.82 seconds, while the others struggle. Moreover, formulations R and CHC fail on some small instances such as on the 115-node graph `football` when $k = 3$. Some of the larger instances like `cs4` cause each of the formulations to crash. Investigating

				solve time, $k = 4$			
Graph	n	m	$\bar{\omega}_4$	R	CHC	PATH	CUT
dolphins	34	78	40	1.14	0.01	0.16	0.01
polbooks	105	441	68	13.99	0.03	1.09	0.03
celegansm	453	2025	432	207.38	0.09	42.78	0.06
email	1133	5451	651	LPNS	[642,654]	LPNS	1.82
polblogs	1490	16715	1127	LPNS	0.86	LPNS	0.74
data	2851	15093	52	MEM	[49,58]	[46,75]	[49,58]
uk	4824	6837	14	MEM	[11,26]	[9,26]	[11,26]
hep-th	8361	15751	344	LPNS	[344,347]	[336,347]	404.38
whitaker3	9800	28989	23	MEM	[19,49]	LPNS	[19,49]
crack	10240	30380	31	MEM	[28,61]	LPNS	[28,61]
cs4	22499	43858	18	MEM	MEM	MEM	MEM

Table 2.4: Results for DIMACS-10 graphs. For $k = 4$ and each formulation, we report the total time in seconds (including preprocessing, heuristic, and model build time), or the best lower and upper bounds [LB,UB] within a 3600 second time limit. Cases where the LP relaxation were not solved within the time limit (due either to build time or solve time) are reported as LPNS and cases where using a formulation leads to memory crashes are reported as MEM.

further, we find that CUT performs poorly on those instances that require a large number of conflict constraints (more than 2 million), while performing quite well on all other instances (never taking longer than 7 minutes). The number of conflict constraints for each instance is reported in Table 2.5. These observations motivate the decomposition procedure given next.

2.6.4 Dealing with Too Many Conflicts

As discussed earlier, we initialize the cut-like formulation with the conflict constraints $x_a + x_b \leq 1$ for vertices a and b that are far apart in G. Sometimes this results in too many conflicts for our computer to handle, as reported in Table 2.5. For example, `whitaker3`, `crack`, and `cs4` have more than 47 million, 52 million, and 252 million conflicts, respectively, when $k = 3$. This leads to memory crashes (e.g., `cs4`) or difficulties solving the IP (`uk`). In our experiments, instances with more than 2 million conflicts overburden the MIP solver. To handle such instances, we developed the decomposition method given below, which we refer to as ICUT.

			# conflicts		
Graph	n	m	$k = 2$	$k = 3$	$k = 4$
karate	34	78	144	0	0
dolphins	62	159	927	19	33
lesmis	77	254	0	0	0
polbooks	105	441	1,847	4	728
adjnoun	112	425	0	179	0
football	115	613	3,636	308	0
jazz	198	2742	1,326	26	0
celegansn	297	2148	0	411	0
celegansm	453	2025	0	193	1
email	1133	5451	227,841	216,781	42,340
polblogs	1490	16715	171,237	39,696	1,509
netscience	1589	2742	0	0	0
add20	2395	7462	0	0	0
data	2851	15093	2,122,988	2,276,728	2,410,756
uk	4824	6837	11,459,801	11,047,360	11,112,993
power	4941	6594	0	0	0
add32	4960	9462	1,017	0	0
hep-th	8361	15751	0	700,013	1,432,395
whitaker3	9800	28989	47,928,432	47,842,511	47,728,908
crack	10240	30380	52,315,672	52,167,356	52,009,002
PGPgiantc	10680	24316	0	8,919	0
cs4	22499	43858	252,937,499	252,715,995	252,330,477

Table 2.5: For each $k \in \{2, 3, 4\}$, we report the number of conflict constraints for each instance after preprocessing.

ICUT:

```
1. initialize $K$ to be the $k$-club found by the heuristic and preprocessing of Section 2.6.1;
2. compute a minimum-degree ordering $(v_1, v_2, \ldots, v_n)$ of $G^k$;
3. $T \leftarrow \emptyset$;
4. for $i = n$ down to 1 do
    • $T \leftarrow T \cup \{v_i\}$;
    • $S_i \leftarrow N^k_{G[T]}(v_i)$;
    • if $|S_i| \leq |K|$ continue;
    • $K_i \leftarrow$ a maximum $k$-club in $G[S_i]$ that contains $v_i$;
    • if $|K_i| > |K|$ then $K \leftarrow K_i$;
5. return $K$.
```

This method iteratively solves small subproblems, much like the Iterative Trim Decomposition Branch-and-Cut (ITDBC) algorithm of Moradi and Balasundaram (2018). However, ICUT uses the cut-like formulation instead of the CHC formulation and defines its subproblems differently. This was motivated by the observation that ITDBC takes time $\Theta(n^2m)$ just to identify the subproblems, which ended up being a bottleneck in our initial experiments. To address this, we borrowed ideas from the maximum clique literature to decompose the maximum k-club problem into n subproblems in time $\Theta(nm)$. In particular, we use a minimum degree ordering, which, as defined by Nagamochi (2010), is a vertex ordering $(v_1, v_2, \ldots, v_n)$ in which vertex v_i has minimum degree in the subgraph induced by $\{v_i, v_{i+1}, \ldots, v_n\}$ for all $i \in [n]$. This vertex ordering is closely related to k-cores and degeneracy orderings, and can be found in linear time by an adaptation of the algorithms of Matula and Beck (1983); Batagelj and Zaversnik (2003) as described, for example, by Walteros and Buchanan (2019).

Similar to Moradi and Balasundaram (2018), we terminate a subproblem early (with the Gurobi parameter `Cutoff`) if it has been determined that no solution better than $|K|$ exists.

Proposition 5. *Algorithm ICUT returns a maximum k-club in G.*

Proof. Let K^* be a maximum k-club in G and let K be the k-club returned by ICUT. Clearly, $|K| \leq |K^*|$ since K is a k-club and K^* is a *maximum* k-club. For the reverse inequality, let v_{i^*} be the earliest vertex of K^* in the vertex-ordering $(v_1, v_2, \dots, v_n)$. Observe that K^* is feasible for the subproblem that gives K_{i^*}. Thus, $|K| \geq |K_{i^*}| \geq |K^*|$. Therefore $|K| = |K^*|$, i.e., K is maximum. □

Table 2.6 compares the time to solve the maximum k-club problem for $k \in \{2, 3, 4\}$ using CUT and ICUT. The results show that ICUT outperforms CUT when the instance has a large number of conflict constraints (more than 2 million). While ICUT solves all instances in under 9 minutes, CUT fails to solve 13 of them. As an example, when $k = 3$, ICUT solves `data` in 5.01 seconds, while CUT cannot solve it to optimality. In a more extreme example, ICUT solves each `cs4` instance in under 2 minutes, while CUT cannot handle the large number of conflicts, crashing on each of them.

However, for 13 instances that have a more reasonable number of conflicts, CUT performs better. For example, when $k = 4$, CUT solves `email` in 1.82 seconds, while ICUT takes 28.90 seconds. Another example is `polblogs` with $k = 3$ where CUT finishes in 1.67 seconds, while ICUT takes 37.84 seconds. Thus, CUT and ICUT both have their advantages, and either one could be preferable depending on the instance and the number of conflicts.

2.6.5 Results for Synthetic Instances

In Tables 2.7 and 2.8, we compare running times on the synthetic instances considered by Veremyev and Boginski (2012) and by Moradi and Balasundaram (2018). These instances were randomly generated by Veremyev and Boginski with 10 graphs at each parameter setting

				$k=2$			$k=3$			$k=4$	
Graph	n	m	$\bar{\omega}_2$	CUT	ICUT	$\bar{\omega}_3$	CUT	ICUT	$\bar{\omega}_4$	CUT	ICUT
karate	34	78	18	0.01	0.01						
dolphins	62	159	13	0.03	0.03	29	0.01	0.01	40	0.01	0.01
lesmis	77	254									
polbooks	105	441	28	0.05	0.03	53	0.01	0.01	68	0.03	0.08
adjnoun	112	425				82	0.01	0.08			
football	115	613	16	1.48	0.32	58	0.29	1.14			
jazz	198	2742	103	0.06	0.23	174	0.03	0.05			
celegansn	297	2148				243	0.07	0.24			
celegansm	453	2025				371	0.07	0.11	432	0.06	0.06
email	1133	5451	72	3.94	5.25	212	241.42	121.74	651	1.82	28.90
polblogs	1490	16715	352	4.93	37.06	776	1.67	37.84	1127	0.74	7.82
netscience	1589	2742									
add20	2395	7462									
data	2851	15093	18	1224.76	3.48	32	[31,36]	5.01	52	[49,58]	8.62
uk	4824	6837	5	3266.55	3.54	8	[7,16]	4.06	14	[11,26]	4.90
power	4941	6594									
add32	4960	9462	32	0.81	0.80						
hep-th	8361	15751				120	171.82	47.34	344	404.38	534.82
whitaker3	9800	28989	9	[8,18]	20.95	15	[13,32]	24.31	23	[19,49]	29.44
crack	10240	30380	10	[9,18]	25.84	17	[15,31]	28.93	31	[28,61]	37.34
PGPgiantc	10680	24316				422	5.82	5.76			
cs4	22499	43858	6	MEM	83.84	12	MEM	89.99	18	MEM	113.10

Table 2.6: We report the total time in seconds (including preprocessing, heuristic, and model build time), or the best lower and upper bounds [LB,UB] within a 3600 second time limit. Cases where using a formulation leads to memory crashes are reported as MEM. Instances solved to optimality by the heuristic and preprocessing proposed in Section 2.6.1 are indicated by blank cells.

(n, ρ) where n is the number of nodes and ρ is the edge density. We consider the same diameter bounds $k \in \{3, 4, 5, 6, 7\}$ considered by Veremyev and Boginski and by Moradi and Balasundaram. However, for formulation PATH, we only give results for $k \in \{3, 4\}$ due to its prohibitively large size. The heuristic and preprocessing proposed in Section 2.6.1 are employed in the following experiments.

				$k = 3$					$k = 4$		
n	ρ (%)	$\bar{\omega}_3$	R	CHC	PATH	CUT	$\bar{\omega}_4$	R	CHC	PATH	CUT
100	2	12.2	0.56	0.02	0.05	0.02	21.1	2.56	0.02	0.15	0.02
	3	16.2	6.18	0.14	0.25	0.09	32.3	52.42	(7)	1.29	0.13
	4	21.1	41.34	(8)	1.33	0.29	52.4	550.42	(1)	1.88	0.11
200	1	12.6	2.54	0.12	0.27	0.12	23.8	6.29	0.14	0.72	0.16
	1.5	16.6	16.42	0.70	1.35	0.75	34.8	285.07	34.95	6.04	0.70
	2	20.4	201.70	2.60	8.00	2.09	48.6	(0)	(0)	551.84	21.07
300	0.5	10.0	0.13	0.01	0.02	0.01	15.8	1.17	0.04	0.10	0.04
	1	17.4	33.94	1.88	4.96	1.95	37.8	235.45	1.85	10.77	2.03
	1.5	12.8	444.22	7.62	23.31	6.60	62.0	(0)	(1)	(7)	452.34

Table 2.7: Results for synthetic graphs with $k \in \{3, 4\}$. For each (n, ρ) we give the average time over the 10 instances. If not all 10 were solved within the 1 hour time limit, we only give the number solved (in parenthesis).

			$k = 5$				$k = 6$				$k = 7$		
n	ρ (%)	$\bar{\omega}_5$	R	CHC	CUT	$\bar{\omega}_6$	R	CHC	CUT	$\bar{\omega}_7$	R	CHC	CUT
100	2	31.4	5.29	(9)	0.02	44.1	19.37	0.51	0.02	55.7	4.69	0.03	0.01
	3	57.3	22.15	(7)	0.03	78.1	10.24	0.11	0.01	88.3	9.54	0.01	0.01
	4	85.5	8.77	0.71	0.01	95.5	9.53	0.01	0.01	97.9	10.76	0.01	0.01
200	1	35.6	86.08	1.19	0.28	55.4	(9)	(5)	0.23	80.5	(8)	(5)	0.12
	1.5	62.0	(2)	(0)	1.02	117.0	(4)	(1)	0.12	158.9	153.08	(8)	0.04
	2	133.4	(4)	(0)	0.17	183.2	46.08	0.03	0.02	193.3	59.00	0.02	0.02
300	0.5	21.3	1.46	0.04	0.04	29.2	7.68	0.04	0.05	36.5	9.67	0.05	0.05
	1	60.8	(0)	(0)	6.72	123.4	(0)	(0)	2.23	207.4	(6)	(2)	0.14
	1.5	187.4	(0)	(0)	1.41	275.4	(7)	(9)	0.04	292.5	(6)	0.04	0.03

Table 2.8: Results for synthetic graphs with $k \in \{5, 6, 7\}$.

As Tables 2.7 and 2.8 show, formulation CUT is also the clear winner on synthetic graphs. For example, when $(n, k, \rho) = (300, 5, 1.5\%)$, CUT solves all 10 instances in an average of 1.41 seconds, while the formulations R and CHC solve none of them (within the 1 hour time limit). Even worse, formulations R and CHC fail on instances with as few as 200 and 100 nodes, respectively. In fact, they solve none of the instances with $(n, k, \rho) = (200, 4, 2\%)$.

Recall that we test each formulation on 10*3*5*3 = 450 instances (180 for PATH). Formulation CUT solves each of them in under 1 hour (actually in under 35 minutes). Meanwhile, R and CHC cannot solve 84 and 117 of the synthetic instances, respectively, within the 1 hour time limit.

The formulation CUT is at its worst when $(n, k, \rho) = (300, 4, 1.5\%)$, where it averages 452.34 seconds. However, this is still considerably better than the other formulations. The 10 times (in seconds) are: 2045.94, 161.31, 1.66, 25.33, 1731.26, 5.51, 278.74, 22.82, 206.40, 44.41. These times are longer than the 1 or 2 seconds taken by CUT on most synthetic instances. It may be interesting to see what properties these instances have (that others do not) that make them so challenging.

2.7 Conclusion

In this chapter, we propose new path-like and cut-like formulations for detecting low-diameter clusters in graphs. They simplify, generalize, and outperform several previous formulations. Indeed, on the testbed of synthetic graphs developed by Veremyev and Boginski (2012), our cut-like formulation for the maximum k-club problem never takes longer than 35 minutes, while previously existing formulations fail to solve 84 instances (or more) in a 1 hour time limit. Similar performance is observed on real-life instances that were considered by Shahinpour and Butenko (2013a) and Moradi and Balasundaram (2018). We suspect that our cut-like formulation will work well for problems in wildlife reserve design, political districting, and others where compactness is key in a "good" solution. Our implementation is relatively simple and publicly available, allowing these extensions to be done in future work.

Chapter III

Solving the Distance-based Critical Node Problem

This chapter is based on work with Austin Buchanan (Salemi and Buchanan, 2020c). Some nodes in a network are more important than others. Examples include a hub in an airline network, a major train station in a rail network, or Paul Erdős in the mathematical collaboration network. The removal of a small subset of nodes like these can dramatically disrupt the connectivity of the network. By identifying those *critical nodes* whose deletion causes the most disruption, one can better understand a network and its vulnerabilities. The associated critical node problems (CNPs) have been studied across a variety of domains, from preventing the spread of disease and computer viruses (Cohen et al., 2003; Tao et al., 2006; Charkhgard et al., 2018) to inhibiting enemy wireless communication by locating jamming devices (Commander et al., 2007). Other applications have been proposed in transportation (Kutz, 2004), evacuation (Matisziw and Murray, 2009), and biology (Boginski and Commander, 2009). Depending on the application, the way in which the "connectivity" of the network is defined will differ, leading to different CNPs (Lalou et al., 2018).

A network's connectivity is sometimes measured by the number of node pairs that are connected via some path. In the associated CNP, the task is to delete from an undirected graph $G = (V, E)$ a subset of b critical nodes $D \subseteq V$ so as to minimize the number of node pairs that remain connected via a path in the interdicted graph $G - D$. This is perhaps the most well-studied CNP variant (Arulselvan et al., 2009; Di Summa et al., 2012; Addis et al., 2013; Veremyev et al., 2014).

However, it has been observed that the existence of a path between two nodes may be insufficient for them to be practically "connected"; the path should also be *short.* This is especially true for social networks, collaboration networks, and airline networks. This motivates the study of distance-based critical node problems (Borgatti, 2006; Veremyev et al., 2015), including the particular variant that we consider in this chapter. For generality, we consider a knapsack constraint for the deletion budget (instead of a cardinality constraint) and edge-weighted distances (instead of hops).

Problem: Distance-Based Critical Node Problem (DCNP).
Input: Simple graph $G = (V, E)$, edge weights $w_e \geq 0$ for each edge $e \in E$, deletion budget b, deletion cost a_i for each vertex $i \in V$, connection cost c_e for each pair of nodes $e \in \binom{V}{2}$, distance threshold k.
Output: A subset $D \subseteq V$ satisfying the budget $\sum_{i \in D} a_i \leq b$ that minimizes $\text{obj}(D)$.

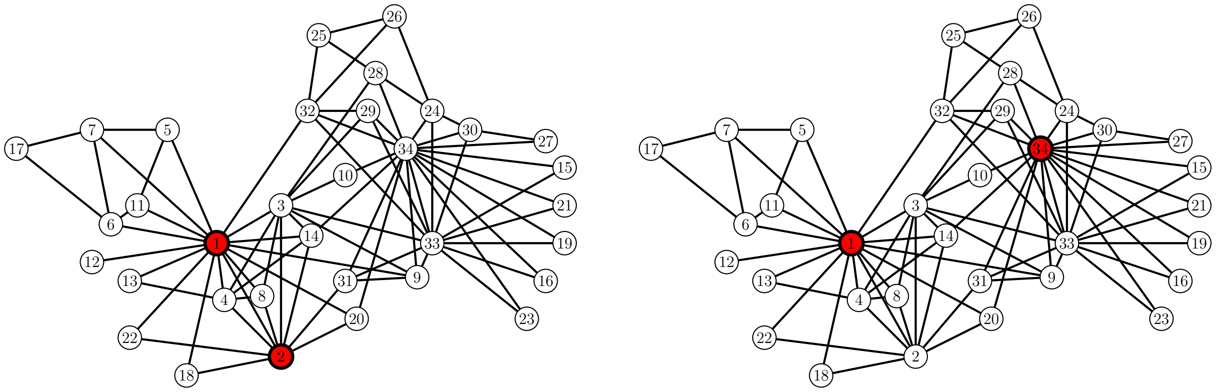
Here, the objective function $\text{obj}(D)$ to be minimized sums the connection costs c_e over all node pairs $e \in \binom{V}{2}$ that are near to each other (distance $\leq k$) in the interdicted graph $G - D$. That is,

$$\text{(DCNP objective function)} \qquad \text{obj}(D) := \sum_{e \in E((G-D)^k)} c_e$$

where $E((G-D)^k)$ denotes the edge set of the k-th power of $G - D$.

To illustrate the difference between CNP and DCNP, let us consider the graph depicted in Figure 3.1, which is the well-known karate club with 34 nodes and 78 edges (Zachary, 1977). Each node represents a member of the club, and if two members communicate outside of the club, then there is an edge between them in the graph. Due to a conflict between the administrator and instructor of the club over the price of karate lessons, the club split in two. As the figure illustrates, the CNP identifies nodes 1 and 2 as critical, and the removal of these nodes splits the graph into one large piece and several small pieces. Meanwhile, the

DCNP identifies nodes 1 and 34 as critical, one node from each side of the split; these nodes are in fact the administrator and the instructor.



(a) CNP solution

(b) DCNP solution

Figure 3.1: Solutions for the `Karate` when $b = 2$ and $k = 2$ with unit values w_e, c_e, and a_i.

The most notable exact approach for DCNP is an integer programming (IP) formulation due to Veremyev et al. (2015). It uses $\mathcal{O}(k|V|^2)$ variables and $\mathcal{O}(k|V||E|)$ constraints when distances are hop-based. Veremyev et al. also propose to extend their model to handle (integer) edge lengths, at the cost of a pseudopolynomial number of variables and constraints. Direct applications of these IP formulations are limited to instances with about 500 nodes; however, variable fixing rules (e.g., based on leaf nodes) can sometimes stretch their ability to sparse 1,500-node instances. Existing exact approaches for CNP are similarly limited.

3.1 Our Contributions

In this chapter, we propose new techniques that allow us to solve instances with up to 17,000 nodes. Key to the approach are two new IP formulations (thin and path-like), which we compare with the formulation of Veremyev et al. To our surprise, we find that the three formulations are equal in strength when the objective coefficients c_e are nonnegative, but the thin formulation is the strongest generally. Moreover, the thin formulation can handle edge-weighted distances at no extra cost in terms of the number of variables. While the thin formulation generally has an exponential number of constraints, it admits an efficient separation routine which we employ in a branch-and-cut algorithm. Also helpful for our approach is a new variable fixing procedure based on simplicial nodes that, on average, fixes three times as many variables than the leaf rule.

Section 3.2 gives a brief overview of the literature, including the formulation of Veremyev et al. (2015) which we call the *recursive* formulation. Section 3.3 introduces the new path-like and thin formulations for DCNP. We prove their correctness (under hop-based and edge-weighted distances) and establish that the three DCNP formulations are equal in strength when the objective coefficients are nonnegative. To show this, we introduce the notion of the *partial dominant* of a polyhedron, which generalizes a previously studied notion called the dominant. Section 3.4 details our implementation, including the separation routines for the thin formulation, a simple heuristic, and the improved variable fixing procedure based on simplicial nodes. Section 3.5 reports on the computational experiments and shows that the thin formulation outperforms the path-like and recursive formulations, handling instances with up to 17,000 nodes. Finally, we conclude in Section 3.6.

3.2 Background and Related Work

As previously mentioned, one can measure the "connectivity" of a network in many different ways and each one will lead to a different CNP problem definition (Lalou et al., 2018; Walteros and Pardalos, 2012; Walteros et al., 2019). Practically any variant of CNP will be NP-hard, which is typically straightforward to show. For example, a natural reduction comes from Vertex Cover in which the task is to determine whether a graph $G = (V, E)$ has a subset D of b vertices such that $G - D$ has no edges. For the variant of CNP discussed in the introduction, hardness follows by observing that G has a vertex cover of size b if and only if CNP has a solution with objective value zero, cf. Arulselvan et al. (2009). Consequently, if Vertex Cover is NP-hard for a particular graph class, then CNP is also hard for that graph class. So, for example, CNP remains NP-hard for planar graphs and for unit disk graphs (Garey and Johnson, 1979).

Interestingly, however, is that, while Vertex Cover is easy in trees, CNP is hard in trees. Indeed, Di Summa et al. (2011) show that CNP is NP-hard in trees even with 0-1 connection costs $c_e \in \{0, 1\}$ and unit deletion costs $a_i = 1$. Interested readers are encouraged to consult Di Summa et al. (2011); Shen et al. (2013); Addis et al. (2013) for more results about the complexity of CNP.

Many approaches have been proposed to solve the CNP. We will focus on exact approaches; readers can refer to Lalou et al. (2018) for discussion about heuristics and approximation algorithms. Arulselvan et al. (2009) propose an IP formulation with $\Theta(n^2)$ variables and $\Theta(n^3)$ constraints. However, they observe that the commercial MIP solver CPLEX sometimes took longer than 5,000 seconds to solve relatively small instances ($n \leq 150$), while their heuristic found optimal solutions in one second. Veremyev et al. (2014) propose formulations with $\Theta(n^2)$ variables and constraints and observe speedups over the model of Arulselvan et al. (2009). With the help of variable fixing (e.g., based on the idea that a leaf should not be critical), they are able to solve select instances with up to 1,612 vertices and 2,106 edges. Di Summa et al. (2012) introduce a formulation with $\Theta(n^2)$ variables and an exponential number of path constraints. Other valid inequalities are also proposed. The resulting branch-and-cut algorithm is tested on instances with up to 100 nodes. The base formulation of Di Summa et al. (2012) is an important precursor to—and can be seen as a special case of—our thin formulation (by choosing k sufficiently large).

Most variants of DCNP are NP-hard. Indeed, for the particular variant of DCNP that we consider, the Vertex Cover reduction from above shows NP-hardness. That is, G has a vertex cover of size b if and only if DCNP over G has a solution with objective value zero, cf. Veremyev et al. (2015). However, some special cases of DCNP admit polynomial-time algorithms (Aringhieri et al., 2019).

Veremyev et al. (2015) identify several different important classes of DCNP in which the objectives are to:

1. minimize the number of node pairs connected by a path of length at most k;
2. minimize the Harary index;
3. minimize the sum of power functions of distances;
4. maximize the generalized Wiener index;
5. maximize the distance between nodes s and t.

Hooshmand et al. (2019) propose a Benders approach for Class 4, while Veremyev et al. (2015) propose the recursive formulation which is generic enough to handle all of the DCNP classes mentioned above. In this chapter, we are particularly interested in Class 1, and in this case the formulation of Veremyev et al. can be written more simply as follows. In their formulation, y_i is a binary variable that equals one if vertex $i \in V$ is chosen to be in the deletion set D, and u^s_{ij} is a binary variable that equals one if vertices i and j are not deleted and the distance between them in $G - D$ is at most s. To simplify future analysis, we also introduce a binary variable x_e for each edge e in the k-th power graph G^k, indicating whether the distance between its endpoints in $G - D$ is at most k.

$$\min \sum_{e \in E^k} c_e x_e \tag{3.1a}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{3.1b}$$

$$u_{ij}^1 + y_i + y_j \geq 1 \qquad \forall \{i,j\} \in E \tag{3.1c}$$

$$u_{ij}^s + y_i \leq 1 \qquad \forall i,j \in V, i \neq j, \quad \forall s \in \{1,2,\dots,k\} \tag{3.1d}$$

$$u_{ij}^s = u_{ij}^1 \qquad \forall \{i,j\} \in E, \quad \forall s \in \{2,3,\dots,k\} \tag{3.1e}$$

$$u_{ij}^s \leq \sum_{t \in N(i)} u_{tj}^{s-1} \qquad \forall \{i,j\} \notin E, \quad \forall s \in \{2,3,\dots,k\} \tag{3.1f}$$

$$u_{tj}^{s-1} \leq u_{ij}^s + y_i \qquad \forall t \in N(i), \quad \forall \{i,j\} \notin E, \quad \forall s \in \{2,3,\dots,k\} \tag{3.1g}$$

$$u_{ij}^s = u_{ji}^s \qquad \forall i,j \in V, i \neq j, \quad \forall s \in \{1,2,\dots,k\} \tag{3.1h}$$

$$u_{ij}^k = x_e \qquad \forall e = \{i,j\} \in E^k \tag{3.1i}$$

$$u_{ij}^s \in \{0,1\} \qquad \forall i,j \in V, i \neq j, \quad \forall s \in \{1,2,\dots,k\} \tag{3.1j}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{3.1k}$$

$$x_e \in \{0,1\} \qquad \forall e = \{i,j\} \in E^k. \tag{3.1l}$$

In this *recursive* formulation, it suffices to write the objective only over the variables whose edges belong to the k-th power graph; other node pairs will always be far apart regardless of the choice of D, so their x variables would equal zero and contribute nothing to the objective. The reader is referred to Veremyev et al. (2015) for more information about the interpretation of the constraints and how to extend this formulation to handle integer edge-weighted distances at the cost of a pseudopolynomial number of variables and constraints.

Veremyev et al. (2015) suggested three enhancements to this formulation:

1. Since $u_{ij}^s = u_{ji}^s$ it suffices to define just one of these variables, e.g., those with $i < j$.
2. The variables u_{ij}^s can be fixed to zero (or omitted from the model) when $s < \text{dist}_G(i,j)$.

3. A leaf vertex i can be fixed $y_i = 0$ when its stem j satisfies $\deg_G(j) \geq 2$ and $a_j \leq a_i$.

With enhancement 2, the number of variables reduces from $\Theta(k|V|^2)$ to $\mathcal{O}(k|E^k|)$. Next we will see that binary restrictions $x_e \in \{0,1\}$ can be omitted and the constraints $u_{ij}^k = x_e$ can be relaxed to $u_{ij}^k \leq x_e$ when the connection costs c_e are nonnegative.

3.2.1 Partial Dominant of a Polyhedron

Sometimes it is difficult to study the facial structure of a polyhedron P. This has motivated some to study, not P, but a simpler polyhedron related to P. In this chapter, we refer to the *dominant* of polyhedron P, typically denoted by $P^{\uparrow}$ (Balas and Fischetti, 1996; Schrijver, 2003; Conforti et al., 2013).

Definition 5. *The dominant $P^{\uparrow}$ of a polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is the polyhedron*

$$P^{\uparrow} := \{\hat{x} \in \mathbb{R}^n \mid \exists x \in P : \hat{x} \geq x\} = P + \mathbb{R}^n_+.$$

Importantly, minimizing a linear objective over P is equivalent to minimizing over its dominant $P^{\uparrow}$ provided that the objective coefficients are nonnegative. That is, if $c \in \mathbb{R}^n_+$, then $\min\{c^T x \mid x \in P\} = \min\{c^T x \mid x \in P^{\uparrow}\}$.

We will see a similar phenomenon with the recursive, path-like, and thin formulations. Specifically, some of their constraints can safely be omitted when the connection costs c_e are nonnegative. The resulting formulations, which are smaller and more convenient to use, are used in our computational experiments. For this reason, when we compare the strength of the different DCNP formulations, we do not always compare the "full" formulations. Sometimes we compare the strength of their *partial dominants*. The reason for studying their partial dominants, as opposed to their dominants, is that the objective coefficients apply only to the x variables, and so we will take the dominant only with respect to x. While the idea of a partial dominant is straightforward, it has not appeared in the literature, to our knowledge.

Definition 6. *The partial dominant $P^{\uparrow x}$ of a polyhedron $P = \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^d \mid Ax + Gy \leq b\}$ with respect to x is the polyhedron*

$$P^{\uparrow x} := \{(\hat{x}, y) \in \mathbb{R}^n \times \mathbb{R}^d \mid \exists (x,y) \in P : \hat{x} \geq x\} = P + (\mathbb{R}^n_+, 0^d),$$

where 0^d is the d-dimensional vector of zeros.

Similar to before, minimizing a linear objective $c^T x + 0^T y$ over P is equivalent to minimizing over its partial dominant $P^{\uparrow x}$ provided that c is nonnegative. That is, if $c \in \mathbb{R}^n_+$, then $\min \{c^T x \mid (x,y) \in P\} = \min \{c^T x \mid (x,y) \in P^{\uparrow x}\}$.

As a warm up, we first identify the partial dominant of the recursive formulation, where the LP feasible region of the recursive formulation is denoted by R.

Lemma 2. *The partial dominant $\mathrm{R}^{\uparrow x}$ of the recursive formulation can be obtained by omitting constraints $x_e \leq u^k_{ij}$ and $0 \leq x_e \leq 1$ from the definition of* R.

Proof. Let P be the polyhedron obtained by omitting constraints $x_e \leq u^k_{ij}$ and $0 \leq x_e \leq 1$ from the definition of R. We show that $P = \mathrm{R}^{\uparrow x}$.

($P \subseteq \mathrm{R}^{\uparrow x}$). If P is empty then the inclusion is trivial, so suppose $(\hat{x}, \hat{y}, \hat{u}) \in P$. For $e = \{i,j\} \in E^k$, let $\tilde{x}_e := \hat{u}^k_{ij}$. See that $(\tilde{x}, \hat{y}, \hat{u}) \in \mathrm{R}$ and $\hat{x} \geq \tilde{x}$, implying $(\hat{x}, \hat{y}, \hat{u}) \in \mathrm{R}^{\uparrow x}$.

($P \supseteq \mathrm{R}^{\uparrow x}$). If $\mathrm{R}^{\uparrow x}$ is empty then the inclusion is trivial, so suppose $(\hat{x}, \hat{y}, \hat{u}) \in \mathrm{R}^{\uparrow x}$. By definition of partial dominant, there exists $\tilde{x} \leq \hat{x}$ for which $(\tilde{x}, \hat{y}, \hat{u}) \in \mathrm{R}$. Recognize that $\tilde{x}_e = \hat{u}^k_{ij}$ holds for $e = \{i,j\} \in E^k$. Then, $(\hat{x}, \hat{y}, \hat{u}) \in P$, because each $e = \{i,j\} \in E^k$ satisfies $\hat{u}^k_{ij} = \tilde{x}_e \leq \hat{x}_e$. □

3.3 New Formulations

Here we propose two new formulations for DCNP which we call the path-like and thin formulations. We prove that they correctly model the DCNP and that their partial dominants are equivalent in strength to that of the recursive formulation.

3.3.1 The Path-Like Formulation

This section introduces the path-like formulation. It relies on the notion of a length-bounded connector, defined below. Figure 3.2 illustrates the definition.

Definition 7 (length-k i, j-connector[1])**.** *A subset $C \subseteq V$ of vertices that contains i and j is called a length-k i, j-connector in an edge-weighted graph $G = (V, E)$ if $\text{dist}_{G[C]}(i, j) \leq k$. If no proper subset of C is a length-k i, j-connector, then C is said to be* minimal.

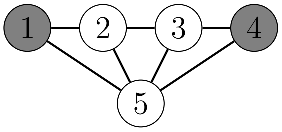


Figure 3.2: Under hop-based distances, the vertex subset $\{1, 5, 4\}$ is a minimal length-2 $1, 4$-connector and the vertex subset $\{1, 2, 3, 4\}$ is a minimal length-3 $1, 4$-connector. The vertex subset $\{1, 2, 5, 4\}$ is a length-3 $1, 4$-connector, but is not minimal.

The collection of all minimal length-k i, j-connectors is denoted C_{ij}^k, and the union of these sets over all $\{i, j\}$ pairs is denoted $\mathcal{C} := \cup C_{ij}^k$. In the lemma below, we prove that these sets C_{ij}^k are disjoint, and so $\mathcal{C}$ can be defined as a disjoint union, which will be helpful later.

Lemma 3. *If edge weights w_e are nonnegative, then every $C \in \mathcal{C}$ is a minimal length-k i, j-connector for a unique pair of vertices $\{i, j\}$. This is not true when some edge weights are negative.*

Proof. Let $i, j \in V$ be distinct vertices and let $C \in C_{ij}^k$ be a minimal length-k i, j-connector. Suppose that C is a minimal length-k u, v-connector. We are to show that $\{i, j\} = \{u, v\}$.

In the first case, suppose that the sets $\{i, j\}$ and $\{u, v\}$ have at least one vertex in common. Without loss, suppose that $i = u$. We are to show that $j = v$. Consider a shortest paths tree

[1] This definition is slightly different from the one given in Chapter II. Here, we include the "endpoints" i and j in any length-k i, j-connector.

of $G[C]$ that is rooted at i. Such a tree exists when the edge weights w_e are nonnegative. Then, j must be a leaf of this tree; if otherwise, let L be the set of leaves of the shortest paths tree that are not i and see that $C \setminus L$ would be a smaller length-k i, j-connector. Further, j must be the *only* leaf; if otherwise, $C \setminus (L \setminus \{j\})$ would be a smaller length-k i, j-connector. By the same arguments, v must be the only leaf, and so $j = v$.

In the second case, suppose that the sets $\{i, j\}$ and $\{u, v\}$ have no vertices in common. Again, consider a shortest paths tree of $G[C]$ that is rooted at i. As before, j must be the only leaf of this tree, and so the tree is in fact an i, j-path. Moreover, u and v, which are distinct from i and j, must belong to this tree and thus are in the path's interior. The vertices belonging to its u, v-subpath form a length-k u, v-connector that is smaller than C, contradicting the minimality of C. Thus, this case cannot happen.

For the last claim, consider the triangle on vertices $\{i, j, j'\}$ where edges $\{i, j\}$ and $\{i, j'\}$ have weight 2 and $\{j, j'\}$ has weight -1. In this case, $C = \{i, j, j'\}$ is a minimal length-1 i, j-connector and a minimal length-1 i, j'-connector. □

The path-like formulation uses the following variables. As before, let y_i be a binary variable representing the decision to include i in the deletion set $D \subseteq V$, and let x_e be a binary variable representing whether the distance between the endpoints of $e \in E^k$ is at most k in $G - D$. Lastly, z_C is a binary variable representing whether all vertices of C are intact in $G - D$, i.e., whether $D \cap C = \emptyset$. Here, z_C is defined for all $\{i, j\}$ pairs and for all minimal length-k i, j-connectors. The path-like formulation is then as follows.

$$\min \sum_{e \in E^k} c_e x_e \tag{3.2a}$$

$$z_C + \sum_{v \in C} y_v \geq 1 \qquad \forall C \in \mathcal{C} \tag{3.2b}$$

$$z_C + y_v \leq 1 \qquad \forall v \in C, \quad \forall C \in \mathcal{C} \tag{3.2c}$$

$$x_e \geq z_C \qquad \forall C \in C_{ij}^k, \quad \forall e = \{i, j\} \in E^k \tag{3.2d}$$

$$x_e \leq \sum_{C \in C_{ij}^k} z_C \qquad \forall e = \{i, j\} \in E^k \tag{3.2e}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{3.2f}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E^k \tag{3.2g}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{3.2h}$$

$$z_C \in \{0, 1\} \qquad \forall C \in \mathcal{C}. \tag{3.2i}$$

Constraints (3.2b) and (3.2c) ensure that a connector C is intact ($z_C = 1$) if and only if none of its vertices $v \in C$ were chosen to be in the deletion set D ($y_v = 0$ for all $v \in C$). Constraints (3.2d) and (3.2e) ensure that the endpoints of $e = \{i, j\}$ should be deemed close to each other ($x_e = 1$) if and only if there is an intact connector between them ($\sum_{C \in C_{ij}^k} z_C \geq 1$). Later we will see that constraints (3.2c) and (3.2e) can be omitted when the connection costs c_e are nonnegative.

Theorem 6. *The path-like formulation is correct, even under edge-weighted distances.*

Proof. Let $x^* \in \{0, 1\}^{|E^k|}$ be a binary vector, $D \subseteq V$ be a deletion set, and $y^D \in \{0, 1\}^n$ be its characteristic vector. To prove the claim, it suffices to show that there exists $z^* \in \{0, 1\}^{|\mathcal{C}|}$

such that (x^*, y^D, z^*) satisfies (3.2b)-(3.2e) if and only if for all $e - \{i, j\} \in E^k$,

$$x^*_e = \begin{cases} 1 & \text{if } \mathrm{dist}_{G-D}(i,j) \leq k \\ 0 & \text{if } \mathrm{dist}_{G-D}(i,j) > k \ . \end{cases}$$

($\implies$) Suppose that there exists a binary vector $z^* \in \{0,1\}^{|\mathcal{C}|}$ such that (x^*, y^D, z^*) satisfies constraints (3.2b)-(3.2e). In the first case, suppose the endpoints of power graph edge $e = \{i, j\} \in E^k$ satisfy $\mathrm{dist}_{G-D}(i,j) \leq k$. Then, there exists a minimal length-k i, j-connector C with $C \subseteq V \setminus D$, e.g., coming from a shortest i, j-path in $G - D$. Then, $x^*_e = 1$ by

$$x^*_e + 0 \geq z^*_C + 0 = z^*_C + \sum_{v \in C} y^D_v \geq 1,$$

where the first inequality holds by constraints (3.2d) and the second inequality holds by constraints (3.2b). In the other case, $\mathrm{dist}_{G-D}(i,j) > k$. In this case, D hits every minimal length-k i, j-connector C, i.e., for every $C \in C^k_{ij}$ there exists a vertex from C that also belongs to D. Then $z^*_C = 0$ by $z^*_C + 1 = z^*_C + y^D_v \leq 1$, where the inequality holds by constraints (3.2c). Constraints (3.2e) then show that $x^*_e \leq \sum_{C \in C^k_{ij}} z^*_C \leq 0$, as desired.

($\impliedby$) Suppose that for each edge $e = \{i, j\} \in E^k$ of the power graph, $x^*_e = 1$ if and only if $\mathrm{dist}_{G-D}(i,j) \leq k$. We construct a binary vector $z^* \in \{0,1\}^{|\mathcal{C}|}$ such that (x^*, y^D, z^*) satisfies constraints (3.2b)-(3.2e). For each connector $C \in \mathcal{C}$, define

$$z^*_C := \begin{cases} 1 & \text{if } D \cap C = \emptyset \\ 0 & \text{if } D \cap C \neq \emptyset. \end{cases}$$

By this definition, (x^*, y^D, z^*) satisfies constraints (3.2b) and (3.2c).

Now, we show that each constraint (3.2d) is satisfied. Consider an edge $e = \{i, j\} \in E^k$ of the power graph and a minimal length-k i, j-connector $C \in C^k_{ij}$. In the first case, where $x^*_e = 1$, the constraint is satisfied as $x^*_e = 1 \geq z^*_C$. In the other case, where $x^*_e = 0$, $\text{dist}_{G-D}(i,j) > k$ holds by the assumption. This implies that every length-k i,j-connector is hit by D. In particular, this is true for C, so $z^*_C = 0$. Thus, the constraint is satisfied as $x^*_e = 0 \geq 0 = z^*_C$.

Lastly, we show that each constraint (3.2e) is satisfied. Consider power graph edge $e = \{i,j\} \in E^k$. In the first case, where $x^*_e = 0$, the constraint is obviously satisfied. In the other case, $x^*_e = 1$ and $\text{dist}_{G-D}(i,j) \leq k$. This implies that at least one minimal length-k i,j-connector is *not* hit by D, and the associated z^* value equals one, so $\sum_{C \in C^k_{ij}} z^*_C \geq 1 = x^*_e$, and the constraint is satisfied. □

Next, we bound the size of the path-like formulation in Lemma 5 with help from the following (easy) lemma.

Lemma 4. *Suppose that distances are hop-based. A subset of vertices C is a minimal length-k i,j-connector if and only if C induces an i,j-path graph.*

Lemma 5. *Suppose that distances are hop-based, k is constant, and G is connected. Then, the number of minimal connectors $|\mathcal{C}|$ is*

- $\mathcal{O}(m^{(k+1)/2}) = \mathcal{O}(n^{k+1})$ *when k is odd, and*
- $\mathcal{O}(nm^{k/2}) = \mathcal{O}(n^{k+1})$ *when k is even.*

Proof. To prove the claim, we construct an injective map f from the set of minmial connectors $\mathcal{C} = \cup_{i,j} C^k_{ij}$ to a set F of appropriate size. By Lemma 4, every minimal length-k i,j-connector $C \in C^k_{ij}$ induces an i,j-path graph, say $i = v_0\text{-}v_1\text{-}v_2\text{-}\cdots\text{-}v_q = j$, where $q \leq k$. For such a connector C, define $f(C)$ as follows.

$$f(C) \coloneqq \begin{cases} \left(\emptyset, \Big\{\{i, v_1\}, \dots, \{v_{q-1}, j\}\Big\}\right) & \text{if } |C| \geq 2 \text{ is even } (q \text{ odd}) \\ \\ \left(i, \Big\{\{v_1, v_2\}, \dots, \{v_{q-1}, j\}\Big\}\right) & \text{if } |C| \geq 3 \text{ is odd } (q \text{ even}) \text{ and } i < j \\ \\ \left(j, \Big\{\{i, v_1\}, \dots, \{v_{q-2}, v_{q-1}\}\Big\}\right) & \text{if } |C| \geq 3 \text{ is odd } (q \text{ even}) \text{ and } i > j. \end{cases}$$

Observe that f maps a connector C to an ordered pair (v_C, E_C) where $v_C \in V \cup \{\emptyset\}$ and $E_C \subseteq E$. See that, when $|C|$ is even, f maps C to $(q+1)/2$ edges; when $|C|$ is odd, f maps C to a vertex and $q/2$ edges. Define $F \coloneqq \{f(C) \mid C \in \mathcal{C}\}$. By assumption that G is connected $n = \mathcal{O}(m)$, and since k is a constant, $nk = \mathcal{O}(m)$. Then,

$$|\mathcal{C}| = |F| \leq \sum_{\substack{q=1 \\ (q \text{ odd})}}^{k} \binom{m}{(q+1)/2} + \sum_{\substack{q=2 \\ (q \text{ even})}}^{k} n\binom{m}{q/2}.$$

So, when $k \geq 1$ is odd,

$$|\mathcal{C}| \leq \frac{k+1}{2}\binom{m}{(k+1)/2} + \frac{k-1}{2}n\binom{m}{(k-1)/2} = \mathcal{O}\left(k\binom{m}{(k+1)/2}\right) = \mathcal{O}(m^{(k+1)/2}),$$

and when $k \geq 2$ is even,

$$|\mathcal{C}| \leq \frac{k}{2}\binom{m}{k/2} + \frac{k}{2}n\binom{m}{k/2} = \mathcal{O}\left(nk\binom{m}{k/2}\right) = \mathcal{O}(nm^{k/2}).$$

□

Remark 3. *The number of variables, constraints, and nonzeros in the path-like formulation is $\mathcal{O}(m^{(k+1)/2})$ when k is an odd constant, and $\mathcal{O}(nm^{k/2})$ when k is an even constant.*

Let PATH denote the LP feasible region of the path-like formulation.

Lemma 6. *The partial dominant* $\text{PATH}^{\uparrow x}$ *of the path-like formulation can be obtained by omitting constraints* $x_e \leq \sum_{C \in C_{ij}^k} z_C$ *and* $0 \leq x_e \leq 1$ *from the definition of* PATH.

Proof. Let P be the polyhedron obtained by omitting constraints $x_e \leq \sum_{C \in C_{ij}^k} z_C$ and $0 \leq x_e \leq 1$ from the definition of PATH. We show that $P = \text{PATH}^{\uparrow x}$.

($P \subseteq \text{PATH}^{\uparrow x}$). If P is empty then the inclusion is trivial, so suppose $(\hat{x}, \hat{y}, \hat{z}) \in P$. For $e = \{i, j\} \in E^k$, let $\tilde{x}_e := \max \left\{ \hat{z}_C \;\middle|\; C \in C_{ij}^k \right\}$. See that $\tilde{x} \leq \hat{x}$ and $(\tilde{x}, \hat{y}, \hat{z}) \in \text{PATH}$, implying $(\hat{x}, \hat{y}, \hat{z}) \in \text{PATH}^{\uparrow x}$.

($P \supseteq \text{PATH}^{\uparrow x}$). If $\text{PATH}^{\uparrow x}$ is empty then the inclusion is trivial, so suppose $(\hat{x}, \hat{y}, \hat{z}) \in \text{PATH}^{\uparrow x}$. By definition of partial dominant, there exists $\tilde{x}$ such that $\tilde{x} \leq \hat{x}$ and $(\tilde{x}, \hat{y}, \hat{z}) \in \text{PATH}$. See that $\hat{x}_e \geq \tilde{x}_e \geq \hat{z}_C$ holds for all $e = \{i, j\} \in E^k$ and $C \in C_{ij}^k$, implying $(\hat{x}, \hat{y}, \hat{z})$ belongs to P. □

The following lemma implies that the conflict constraints (3.2c), while needed for $\text{PATH}^{\uparrow x}$, are not needed for $\text{proj}_{x,y} \text{PATH}^{\uparrow x}$. Thus, they can be omitted in implementation when the objective coefficients c_e are nonnegative. Additionally, we will later use the inequalities (3.3) to compare the strength of $\text{proj}_{x,y} \text{PATH}^{\uparrow x}$ with that of $\text{proj}_{x,y} \text{R}^{\uparrow x}$.

Lemma 7. *If a point* $(\hat{x}, \hat{y}, \hat{z})$ *satisfies the constraints defining* $\text{PATH}^{\uparrow x}$ *except perhaps* (3.2c), *then there is a similar point* $(\hat{x}, \hat{y}, \tilde{z}) \in \text{PATH}^{\uparrow x}$ *that satisfies the following inequalities.*

$$\tilde{z}_{C \setminus \{i\}} - \hat{y}_i \leq \tilde{z}_C \leq \tilde{z}_{C \setminus \{i\}} \qquad \forall C \in C_{ij}^k \textit{ with } |C| \geq 3, \forall \{i, j\} \in E^k. \tag{3.3}$$

Proof. For every minimal connector $C \in \mathcal{C}$, let $\tilde{z}_C = \max \left\{ 0, 1 - \sum_{c \in C} \hat{y}_c \right\}$. Clearly, the point $(\hat{x}, \hat{y}, \tilde{z})$ satisfies constraints (3.2b), constraint (3.2f), and the 0-1 bounds on the y and z variables. To show that inequalities (3.2c) are satisfied, consider a minimal connector $C \in \mathcal{C}$

and a vertex $v \in C$ and see that

$$\tilde{z}_C + \hat{y}_v = \max\left\{0, 1 - \sum_{c \in C} \hat{y}_c\right\} + \hat{y}_v = \max\left\{\hat{y}_v, 1 - \sum_{c \in C \setminus \{v\}} \hat{y}_c\right\} \leq 1.$$

To show that each constraint (3.2d) is satisfied, consider a power graph edge $e = \{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C_{ij}^k$. Now, in the first case, where $\tilde{z}_C = 0$, constraint (3.2d) is obviously satisfied. In the other case, where $\tilde{z}_C = 1 - \sum_{c \in C} \hat{y}_c$, constraint (3.2d) is satisfied by

$$\hat{x}_e \geq \hat{z}_C \geq 1 - \sum_{c \in C} \hat{y}_c = \tilde{z}_C.$$

So, by Lemma 6, $(\hat{x}, \hat{y}, \tilde{z})$ belongs to PATH$^{\uparrow x}$.

Finally, to show that inequalities (3.3) are satisfied, consider a power graph edge $\{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C_{ij}^k$ with at least three vertices, and see that

$$\begin{aligned}
\tilde{z}_{C \setminus \{i\}} - \hat{y}_i &= \max\left\{0, 1 - \sum_{c \in C \setminus \{i\}} \hat{y}_c\right\} - \hat{y}_i \\
&= \max\left\{-\hat{y}_i, 1 - \sum_{c \in C} \hat{y}_c\right\} \\
&\leq \max\left\{0, 1 - \sum_{c \in C} \hat{y}_c\right\} \quad (= \tilde{z}_C) \\
&\leq \max\left\{0, 1 - \sum_{c \in C \setminus \{i\}} \hat{y}_c\right\} = \tilde{z}_{C \setminus \{i\}}.
\end{aligned}$$

□

3.3.2 The Thin Formulation

Before stating the thin formulation, let us first define the notion of a length-bounded separator. Figure 3.3 illustrates the definition.

Definition 8 (length-k i, j-separator[2]). *A subset $S \subseteq V$ of vertices is called a length-k i, j-separator in an edge-weighted graph $G = (V, E)$ if either*

- *S contains i or j (or both), or*
- *S contains neither i nor j and $\text{dist}_{G-S}(i, j) > k$.*

If no proper subset of S is a length-k i, j-separator, then S is said to be minimal.

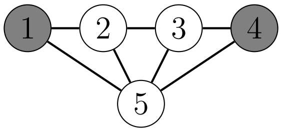


Figure 3.3: Under hop-based distances, the vertex subset $\{5\}$ is a length-2 $1, 4$-separator and vertex subset $\{1\}$ is a length-k $1, 4$-separator for any $k \in \{1, 2, \ldots\}$.

We denote by S_{ij}^k as the collection of all minimal length-k i, j-separators. Then, the thin formulation, which uses the same x and y variables as before, is as follows.

[2] This definition is slightly different from the one given in Chapter II. Note that if one removes a vertex i from a graph, then the distance from i to all other vertices is more than any given k. Therefore, vertex i or j (or both) can form a length-k i, j-separator.

$$\min \sum_{e \in E^k} c_e x_e \tag{3.4a}$$

$$x_e + \sum_{v \in S} y_v \leq |S| \qquad \forall S \in S_{ij}^k, \quad \forall e = \{i,j\} \in E^k \tag{3.4b}$$

$$x_e + \sum_{v \in C} y_v \geq 1 \qquad \forall C \in C_{ij}^k, \quad \forall e = \{i,j\} \in E^k \tag{3.4c}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{3.4d}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^k \tag{3.4e}$$

$$y_i \in \{0,1\} \qquad \forall i \in V. \tag{3.4f}$$

Observe that there can be exponentially many constraints (3.4b) and (3.4c). Fortunately, however, we will see that constraints (3.4b) can be omitted when the connection costs c_e are positive, and constraints (3.4c) can be separated in polynomial time.

Theorem 7. *The thin formulation is correct, even under edge-weighted distances.*

Proof. Let $x^* \in \{0,1\}^{|E^k|}$ be a binary vector, $D \subseteq V$ be a deletion set, and $y^D \in \{0,1\}^n$ be its characteristic vector. It suffices to show that

$$(x^*, y^D) \text{ satisfies (3.4b) and (3.4c)} \iff \forall e = \{i,j\} \in E^k, x_e^* = \begin{cases} 1 & \text{if } \operatorname{dist}_{G-D}(i,j) \leq k \\ 0 & \text{if } \operatorname{dist}_{G-D}(i,j) > k. \end{cases}$$

($\implies$) Suppose (x^*, y^D) satisfies constraints (3.4b) and (3.4c). In the first case, suppose that the endpoints of the power graph edge $e = \{i,j\} \in E^k$ satisfy $\operatorname{dist}_{G-D}(i,j) \leq k$. In this case, there exists a minimal length-k i,j-connector C with $C \subseteq V \setminus D$, e.g., coming from a shortest i, j-path in $G - D$. Then,

$$x_e^* + 0 = x_e^* + \sum_{v \in C} y_v^D \geq 1,$$

where the inequality holds by constraints (3.4c). So, $x_e^* = 1$, as desired. In the other case, suppose that the endpoints of $e = \{i, j\} \in E^k$ satisfy $\text{dist}_{G-D}(i, j) > k$. In this case, there exists a minimal length-k i, j-separator S that is a subset of D. Then,

$$x_e^* + |S| = x_e^* + \sum_{v \in S} y_v^D \leq |S|,$$

where the inequality holds by constraints (3.4b). So, $x_e^* = 0$, as desired.

($\Longleftarrow$) Suppose that for each power graph edge $e = \{i, j\} \in E^k$, $x_e^* = 1$ if and only if $\text{dist}_{G-D}(i, j) \leq k$. To show that constraints (3.4b) are satisfied, consider an edge $e = \{i, j\} \in E^k$ and a minimal length-k i, j-separator $S \in S_{ij}^k$. In the first case, where $x_e^* = 0$, the constraint is obviously satisfied. In the other case, $x_e^* = 1$ and $\text{dist}_{G-D}(i, j) \leq k$. This implies that at least one vertex in S remains intact, i.e., $\sum_{v \in S} y_v^D \leq |S| - 1$. Then, constraint (3.4b) is satisfied, as

$$x_e^* + \sum_{v \in S} y_v^D = 1 + \sum_{v \in S} y_v^D \leq 1 + (|S| - 1) = |S|.$$

To show that constraints (3.4c) are satisfied, consider an edge $e = \{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C_{ij}^k$. In the first case, where $x_e^* = 1$, the constraint is obviously satisfied. In the other case, $x_e^* = 0$ and $\text{dist}_{G-D}(i, j) > k$. Because $\text{dist}_{G-D}(i, j) > k$, D hits every length-k i, j-connector. In particular, D hits C, i.e., $\sum_{v \in C} y_v^D \geq 1$. Then, constraint (3.4c) is satisfied, as

$$x_e^* + \sum_{v \in C} y_v^D = 0 + \sum_{v \in C} y_v^D \geq 1.$$

□

Let THIN denote the LP feasible region of the thin formulation.

Lemma 8. *The partial dominant* $\text{THIN}^{\uparrow x}$ *of the thin formulation can be obtained by omitting constraints* $x_e + \sum_{v \in S} y_v \leq |S|$ *and* $x_e \leq 1$ *from the definition of* THIN*.*

Proof. Let P be the polyhedron obtained by omitting constraints $x_e + \sum_{v \in S} y_v \leq |S|$ and $x_e \leq 1$ from the definition of THIN. We show that $P = \text{THIN}^{\uparrow x}$.

($P \subseteq \text{THIN}^{\uparrow x}$). If P is empty then the inclusion is trivial, so suppose that $(\hat{x}, \hat{y}) \in P$. Let

$$\tilde{x}_e := \max \left\{ 0, \max \left\{ 1 - \sum_{v \in C} \hat{y}_v \;\middle|\; C \in C_{ij}^k \right\} \right\}.$$

Recalling that $\hat{x}_e \geq 0$ for all $e = \{i, j\} \in E^k$ and $\hat{x}_e + \sum_{v \in C} \hat{y}_v \geq 1$ for all $C \in C_{ij}^k$, see that

$$\tilde{x}_e = \max \left\{ 0, \max \left\{ 1 - \sum_{v \in C} \hat{y}_v \;\middle|\; C \in C_{ij}^k \right\} \right\} \leq \hat{x}_e.$$

Thus, since $\tilde{x} \leq \hat{x}$, it now suffices to show that $(\tilde{x}, \hat{y}) \in$ THIN. The connector constraints are satisfied by $(\tilde{x}, \hat{y})$ because, for any $\tilde{C} \in C_{ij}^k$,

$$\tilde{x}_e + \sum_{v \in \tilde{C}} \hat{y}_v \geq \max \left\{ 1 - \sum_{v \in C} \hat{y}_c \;\middle|\; C \in C_{ij}^k \right\} + \sum_{v \in \tilde{C}} \hat{y}_v \geq 1 - \sum_{v \in \tilde{C}} \hat{y}_v + \sum_{v \in \tilde{C}} \hat{y}_v = 1,$$

where the first inequality holds by definition of $\tilde{x}$. Finally, we argue that the separator constraints are satisfied. For contradiction purposes, suppose not, i.e., that there is a minimal length-k i, j-separator S^* for which $\tilde{x}_e + \sum_{v \in S^*} \hat{y}_v > |S^*|$. Then $\tilde{x}_e + \hat{y}_v > 1$ for all $v \in S^*$

since otherwise we arrive at the contradiction

$$|S^*| < \tilde{x}_e + \sum_{u \in S^*} \hat{y}_u = \tilde{x}_e + \hat{y}_v + \sum_{u \in S^* \setminus \{v\}} \hat{y}_u \leq 1 + (|S^*| - 1) = |S^*|.$$

Let C^* be a minimal length-k i,j-connector C that maximizes $1 - \sum_{v \in C} \hat{y}_v$. Since $\tilde{x}_e > 0$ by $\tilde{x}_e + \sum_{v \in S^*} \hat{y}_v > |S^*|$, we have $\tilde{x}_e = 1 - \sum_{v \in C^*} \hat{y}_v$. By minimality of S^*, there is a vertex $u \in S^* \cap C^*$. This gives the contradiction $\tilde{x}_e + \hat{y}_u \leq \tilde{x}_e + \sum_{v \in C^*} y_v = 1 < \tilde{x}_e + \hat{y}_v$. So, $(\tilde{x}, \hat{y})$ satisfies the separator constraints, thus $(\tilde{x}, \hat{y}) \in$ THIN. So, $(\hat{x}, \hat{y}) \in \text{THIN}^{\uparrow x}$, as desired.

($P \supseteq \text{THIN}^{\uparrow x}$). If $\text{THIN}^{\uparrow x}$ is empty then the inclusion is trivial, so suppose that $(\hat{x}, \hat{y}) \in \text{THIN}^{\uparrow x}$. By definition of partial dominant, there exists $\tilde{x}$ such that $\tilde{x} \leq \hat{x}$ and $(\tilde{x}, \hat{y}) \in$ THIN. See that $\hat{x}_e + \sum_{v \in C} \hat{y}_v \geq \tilde{x}_e + \sum_{v \in C} \hat{y}_v \geq 1$ for all $e = \{i,j\} \in E^k$ and $C \in \mathcal{C}^k_{ij}$, implying that $(\hat{x}, \hat{y}) \in P$, as desired. □

3.3.3 Formulation Strength

In this subsection, we compare the strength of the LP relaxations associated with the three DCNP formulations: THIN, PATH, and R. First, we observe that THIN is stronger than $\text{proj}_{x,y}$ PATH, and that $\text{proj}_{x,y}$ R is incomparable with THIN and with $\text{proj}_{x,y}$ PATH. However, when the objective coefficients are nonnegative, the appropriate objects to compare are their partial dominants $\text{THIN}^{\uparrow x}$, $\text{proj}_{x,y} \text{PATH}^{\uparrow x}$, and $\text{proj}_{x,y} \text{R}^{\uparrow x}$ which we find to have equal strength.

Strength of Full Formulations

Theorem 8. *For every instance of DCNP, the inclusion* THIN $\subseteq$ $\text{proj}_{x,y}$ PATH *holds. Meanwhile,* $\text{proj}_{x,y}$ R *is incomparable with* $\text{proj}_{x,y}$ PATH *and with* THIN*.*

Proof. This follows by Lemmata 9 and 10, which are shown below. □

Lemma 9. *For every instance of DCNP, the inclusion* THIN $\subseteq \text{proj}_{x,y}$ PATH *holds. Moreover, the inclusion can be strict for any $k \geq 2$ under hop-based distances.*

Proof. Suppose that (x^*, y^*) belongs to THIN. We construct a z^* such that (x^*, y^*, z^*) belongs to PATH. For each $C \in \mathcal{C}$, let

$$z^*_C := \min\left\{x^*_e, 1 - \max_{v \in C} y^*_v\right\},$$

where $e = \{i, j\} \in E^k$ is the unique pair of vertices for which C is a minimal length-k i, j-connector (guaranteed by Lemma 3). Observe that (x^*, y^*, z^*) satisfies constraint (3.2f) and the 0-1 bounds.

To show that constraints (3.2b) are satisfied, consider a connector $C \in \mathcal{C}$ and let $e = \{i, j\} \in E^k$ be the associated "endpoints" of this connector. In the first case, where $z^*_C = x^*_e$,

$$z^*_C + \sum_{v \in C} y^*_v = x^*_e + \sum_{v \in C} y^*_v \geq 1,$$

where the inequality holds by constraints (3.4c). In the other case, $z^*_C = 1 - \max\{y^*_v \mid v \in C\}$,

$$z^*_C + \sum_{v \in C} y^*_v \geq z^*_C + \max_{v \in C} y^*_v = 1 - \max_{v \in C} y^*_v + \max_{v \in C} y^*_v = 1.$$

To show that constraints (3.2c) are satisfied, consider a connector $C \in \mathcal{C}$ and vertex $v \in C$. Observe that

$$z^*_C + y^*_v = \min\left\{x^*_e, 1 - \max_{i \in C} y^*_i\right\} + y^*_v \leq 1 - \max_{i \in C} y^*_i + y^*_v \leq 1 - y^*_v + y^*_v = 1.$$

To show that constraints (3.2d) are satisfied, consider a power graph edge $e = \{i, j\} \in E^k$

and a minimal length-k i,j-connector $C \in \mathcal{C}^k_{ij}$. Then,

$$x^*_e \geq \min\left\{x^*_e, 1 - \max_{v \in C} y^*_v\right\} = z^*_C.$$

To show that constraints (3.2e) are satisfied, consider a power graph edge $e = \{i,j\} \in E^k$. In the first case, where at least one of the minimal length-k i,j-connectors C satisfies $z^*_C = x^*_e$, the inequality $x^*_e \leq \sum_{C \in \mathcal{C}^k_{ij}} z^*_C$ is clear. In the other case, all minimal length-k i,j-connectors C satisfy $z^*_C = 1 - \max\{y^*_v \mid v \in C\}$. For each such connector C, let v_C be a vertex $v \in C$ that maximizes y^*_v. Then, $S \coloneqq \cup_{C \in \mathcal{C}^k_{ij}} v_C$ is a length-k i,j-separator, which we claim satisfies

$$|S| - \sum_{s \in S} y^*_s \leq |\mathcal{C}^k_{ij}| - \sum_{C \in \mathcal{C}^k_{ij}} y^*_{v_C}. \tag{3.5}$$

To show inequality (3.5), let $q_s = |\{C \in \mathcal{C}^k_{ij} \mid v_C = s\}|$ for every separator vertex $s \in S$. This (positive) value q_s is the number of connectors $C \in \mathcal{C}^k_{ij}$ for which s is selected as v_C. Then,

$$\sum_{C \in \mathcal{C}^k_{ij}} y^*_{v_C} = \sum_{s \in S} q_s y^*_s = \sum_{s \in S} y^*_s + \sum_{s \in S}(q_s - 1)y^*_s \leq \sum_{s \in S} y^*_s + \sum_{s \in S}(q_s - 1) = \sum_{s \in S} y^*_s + |\mathcal{C}^k_{ij}| - |S|,$$

thus proving inequality (3.5). Finally,

$$x^*_e \leq |S| - \sum_{s \in S} y^*_s \leq |\mathcal{C}^k_{ij}| - \sum_{C \in \mathcal{C}^k_{ij}} y^*_{v_C} = \sum_{C \in \mathcal{C}^k_{ij}} (1 - y^*_{v_C}) = \sum_{C \in \mathcal{C}^k_{ij}} z^*_C,$$

where the first inequality holds by constraint (3.4b) of the thin formulation, and the second inequality holds by inequality (3.5). So, (x^*, y^*, z^*) satisfies constraints (3.2e), and thus $(x^*, y^*, z^*) \in$ PATH.

Figure 3.4 shows that the inclusion can be strict for any $k \geq 2$ under hop-based distances. We construct a DCNP instance and a point (x^*, y^*, z^*) that belongs to PATH but the point

(x^*, y^*) dose not belong to THIN.

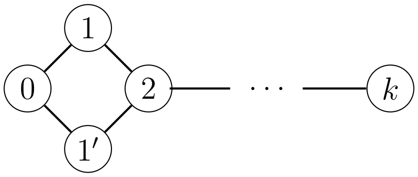


Figure 3.4: The graph $G = (V, E)$ used to show $\operatorname{proj}_{x,y} \text{PATH} \neq \text{THIN}$.

To complete the example, let $c_e = 1$ for all $e \in E^k$, $a_v = 1$ for all $v \in V$, and $b = \frac{n}{2}$. Also, let $y^*_v = \frac{1}{2}$ for all vertices v, $x^*_{e'} = \frac{3}{4}$ for the particular power graph edge $e' = \{0, k\} \in E^k$, and $x^*_e = \frac{3}{8}$ for all other power graph edges. Finally, let $z^*_C = \frac{3}{8}$ for all connectors $C \in \mathcal{C}$. Note that $S = \{2\}$ is a length-k $0, k$-separator. While (x^*, y^*, z^*) satisfies the path-like formulation, the inequality $x_{e'} + y_2 \leq 1$ of type (3.4b) from the thin formulation is violated. □

Lemma 10. $\operatorname{proj}_{x,y} \text{R}$ *is incomparable with* $\operatorname{proj}_{x,y} \text{PATH}$ *and with* THIN*.*

Proof. Since $\text{THIN} \subseteq \operatorname{proj}_{x,y} \text{PATH}$ by Lemma 9, it suffices to show that $\operatorname{proj}_{x,y} \text{R} \nsubseteq \operatorname{proj}_{x,y} \text{PATH}$ and $\text{THIN} \nsubseteq \operatorname{proj}_{x,y} \text{R}$.

($\operatorname{proj}_{x,y} \text{R} \nsubseteq \operatorname{proj}_{x,y} \text{PATH}$). We construct a DCNP instance and a point $(\hat{x}, \hat{y}, \hat{u})$ that belongs to R and show there is no $\hat{z}$ for which $(\hat{x}, \hat{y}, \hat{z})$ belongs to PATH. Consider the graph in Figure 3.5.

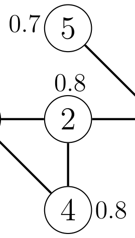


Figure 3.5: The graph $G = (V, E)$ used to show $\operatorname{proj}_{x,y} \text{R} \nsubseteq \operatorname{proj}_{x,y} \text{PATH}$.

The $\hat{y}$ values are given next to the nodes. Let $k = 3$, $b = 5$, and $a_v = 1$ for all $v \in V$.

Also, for all $s \in \{1,2,3\}$, let

$$\hat{u}^s_{1,2} = 0.2 \quad \hat{u}^s_{1,3} = 0.3 \quad \hat{u}^s_{1,4} = 0.0 \quad \hat{u}^s_{1,5} = 0.1 \quad \hat{u}^s_{2,3} = 0.1$$

$$\hat{u}^s_{2,4} = 0.1 \quad \hat{u}^s_{2,5} = 0.0 \quad \hat{u}^s_{3,4} = 0.2 \quad \hat{u}^s_{3,5} = 0.0 \quad \hat{u}^s_{4,5} = 0.1$$

Finally, let $\hat{x}_e = \hat{u}^3_{ij}$ for all $e = \{i,j\} \in E^3$. Observe that $(\hat{x}, \hat{y}, \hat{u})$ belongs to R. We claim that there is no $\hat{z}$ for which $(\hat{x}, \hat{y}, \hat{z})$ belongs to PATH. For contradiction purposes, suppose $(\hat{x}, \hat{y}, \hat{z}) \in$ PATH. Consider the power graph edge $e = \{1,3\}$ and see that the only minimal length-k $1,3$-connector is $C = \{1,2,3\}$. Constraints (3.2d) force $\hat{z}_C \leq \hat{x}_e = 0.3$ and constraints (3.2e) force $0.3 = \hat{x}_e \leq \hat{z}_C$. This implies that $\hat{z}_C = 0.3$. This contradicts $\hat{z}_C + \hat{y}_2 \leq 1$. Thus, no $\hat{z}$ satisfies $(\hat{x}, \hat{y}, \hat{z}) \in$ PATH.

(THIN $\not\subseteq \operatorname{proj}_{x,y}$ R). We construct a DCNP instance and a point $(\hat{x}, \hat{y}) \in$ THIN for which no $\hat{u}$ has $(\hat{x}, \hat{y}, \hat{u}) \in$ R. Consider the graph in Figure 3.6.

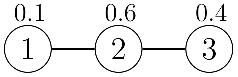


Figure 3.6: The graph $G = (V, E)$ used to show THIN $\not\subseteq \operatorname{proj}_{x,y}$ R.

The $\hat{y}$ values are given next to the nodes. Let $k = 3$, $b = 3$, and $a_v = 1$ for all $v \in V$. Also, let $\hat{x}_{\{1,2\}} = 0.3$, $\hat{x}_{\{1,3\}} = 0.2$, and $\hat{x}_{\{2,3\}} = 0.4$. Observe that $(\hat{x}, \hat{y})$ belongs to THIN. We claim that there is no $\hat{u}$ for which $(\hat{x}, \hat{y}, \hat{u})$ belongs to R. For contradiction purposes, suppose $(\hat{x}, \hat{y}, \hat{u}) \in$ R. Constraints (3.1e) and (3.1i) force $\hat{x}_{\{2,3\}} = \hat{u}^3_{2,3} = \hat{u}^2_{2,3} = 0.4$ and $\hat{x}_{\{1,3\}} = \hat{u}^3_{1,3} = 0.2$. This contradicts $\hat{u}^2_{2,3} \leq \hat{u}^3_{1,3} + \hat{y}_1$ of type (3.1g). Thus, no $\hat{u}$ satisfies $(\hat{x}, \hat{y}, \hat{u}) \in$ R. □

Strength of Partial Dominants

Theorem 9. *For every (hop-based) instance of DCNP,*

$$\mathrm{THIN}^{\uparrow x} = \operatorname{proj}_{x,y} \mathrm{PATH}^{\uparrow x} = \operatorname{proj}_{x,y} \mathrm{R}^{\uparrow x}.$$

Proof. We show that the following three inclusions hold.

$$\mathrm{THIN}^{\uparrow x} \subseteq \operatorname{proj}_{x,y} \mathrm{PATH}^{\uparrow x} \subseteq \operatorname{proj}_{x,y} \mathrm{R}^{\uparrow x} \subseteq \mathrm{THIN}^{\uparrow x}.$$

($\mathrm{THIN}^{\uparrow x} \subseteq \operatorname{proj}_{x,y} \mathrm{PATH}^{\uparrow x}$) If arbitrary polyhedra P and Q satisfy $P \subseteq Q$, then $P^{\uparrow x} \subseteq Q^{\uparrow x}$. So, the stated inclusion holds by Lemma 9.

($\operatorname{proj}_{x,y} \mathrm{PATH}^{\uparrow x} \subseteq \operatorname{proj}_{x,y} \mathrm{R}^{\uparrow x}$) We prove the statement for hop-based distances, as the recursive formulation only applies to this case. Suppose $(\hat{x}, \hat{y}, \hat{z})$ belongs to $\mathrm{PATH}^{\uparrow x}$. By Lemma 7, there is a similar point $(\hat{x}, \hat{y}, \tilde{z}) \in \mathrm{PATH}^{\uparrow x}$ that satisfies inequalities (3.3). We construct a $\hat{u}$ such that $(\hat{x}, \hat{y}, \hat{u})$ belongs to $\mathrm{R}^{\uparrow x}$. Specifically, for distinct vertices $i, j \in V$ and $s \in \{1, 2, \dots, k\}$, let

$$\hat{u}_{ij}^s \coloneqq \max\{\tilde{z}_C \mid C \in C_{ij}^s\}.$$

By Lemma 2, to show that $(\hat{x}, \hat{y}, \hat{u})$ belongs to $\mathrm{R}^{\uparrow x}$, it suffices to show that $(\hat{x}, \hat{y}, \hat{u})$ satisfies all constraints defining R except for the constraints $x_e \leq u_{ij}^k$ and the 0-1 bounds on x. First see that $(\hat{x}, \hat{y}, \hat{u})$ satisfies constraints (3.1b) and (3.1h), as well as the 0-1 bounds on y and u.

To show that constraints (3.1c) are satisfied, consider an edge $\{i, j\} \in E$. Since $C' \coloneqq \{i, j\}$ is the only minimal length-1 i, j-connector,

$$\hat{u}_{ij}^1 + \hat{y}_i + \hat{y}_j = \max\{\tilde{z}_C \mid C \in C_{ij}^1\} + \hat{y}_i + \hat{y}_j = \tilde{z}_{C'} + \hat{y}_i + \hat{y}_j \geq 1,$$

where the inequality holds by constraint (3.2b) of the path-like formulation.

To show that constraints (3.1d) are satisfied, consider distinct vertices $i, j \in V$ and $s \in \{1, 2, \dots, k\}$. Letting $C' \in \mathcal{C}_{ij}^s$ be a minimal length-s i,j-connector C that maximizes $\tilde{z}_C$, observe that

$$\hat{u}_{ij}^s + \hat{y}_i = \tilde{z}_{C'} + \hat{y}_i \leq 1,$$

where the inequality holds by inequality (3.2c).

To show that constraints (3.1e) are satisfied, consider an edge $\{i, j\} \in E$. The only minimal length-s i,j-connector is $C' = \{i, j\}$, and this is true for all $s \in \{1, 2, \dots, k\}$, so

$$\hat{u}_{ij}^s = \max\{\tilde{z}_C \mid C \in \mathcal{C}_{ij}^s\} = \tilde{z}_{C'} = \max\{\tilde{z}_C \mid C \in \mathcal{C}_{ij}^1\} = \hat{u}_{ij}^1.$$

To show that constraints (3.1f) are satisfied, consider a "missing" edge $\{i, j\} \notin E$ and $s \in \{2, 3, \dots, k\}$. Let $C' \in \mathcal{C}_{ij}^s$ be a minimal length-s i,j-connector C that maximizes $\tilde{z}_C$, and let q be the vertex of C' that neighbors i. Then,

$$\hat{u}_{ij}^s = \tilde{z}_{C'} \leq \tilde{z}_{C' \setminus \{i\}} \leq \hat{u}_{qj}^{s-1} \leq \sum_{t \in N(i)} \hat{u}_{tj}^{s-1}.$$

Here, the first inequality holds by inequality (3.3), and the second holds because $C' \setminus \{i\}$ is a minimal length-$(s-1)$ q,j-connector and by definition of $\hat{u}_{qj}^{s-1}$.

To show that constraints (3.1g) are satisfied, consider a "missing" edge $\{i, j\} \in E$, a neighbor $t \in N(i)$, and $s \in \{2, 3, \dots, k\}$. Let $\bar{C} \in \mathcal{C}_{tj}^{s-1}$ be a minimal length-$(s-1)$ t,j-connector C that maximizes $\tilde{z}_C$. Observe that $\bar{C} \cup \{i\}$ is a length-s i,j-connector because

$$\text{dist}_{G[\bar{C} \cup \{i\}]}(i, j) \leq \text{dist}_{G[\bar{C}]}(t, j) + 1 = (s-1) + 1 = s.$$

Let C^* be the vertices on a shortest i,j-path in $G\left[\bar{C} \cup \{i\}\right]$. See that C^* is a minimal length-s i,j-connector because C^* induces an i,j-path graph and $\operatorname{dist}_{G[C^*]}(i,j) = \operatorname{dist}_{G[\bar{C}\cup\{i\}]}(i,j) \leq s$. Further, $\hat{C} = C^* \setminus \{i\}$ belongs to $\mathcal{C}$. Observe that

$$\hat{u}_{tj}^{s-1} - \hat{y}_i = \tilde{z}_{\bar{C}} - \hat{y}_i \leq \tilde{z}_{\hat{C}} - \hat{y}_i \leq \tilde{z}_{C^*} \leq \hat{u}_{ij}^{s},$$

where the first and second inequalities hold by inequality (3.3).

Finally, to show that the constraints $u_{ij}^k \leq x_e$ are satisfied, consider an edge $e = \{i,j\} \in E^k$. Letting $C' \in C_{ij}^k$ be a minimal length-k i,j-connector C that maximizes $\tilde{z}_C$, observe that

$$\hat{u}_{ij}^k = \tilde{z}_{C'} \leq \hat{x}_e,$$

where the inequality holds by constraint (3.2d) of the path-like formulation.

($\operatorname{proj}_{x,y} \mathrm{R}^{\uparrow_x} \subseteq \mathrm{THIN}^{\uparrow_x}$) We prove the statement for hop-based distances, as the recursive formulation only applies to this case. Suppose that $(\hat{x}, \hat{y}, \hat{u})$ belongs to $\mathrm{R}^{\uparrow_x}$. We show $(\hat{x}, \hat{y}) \in \mathrm{THIN}^{\uparrow_x}$. By Lemma 8, it suffices to show that $(\hat{x}, \hat{y})$ satisfies all constraints defining THIN except perhaps for constraints of the form $x_e + \sum_{v \in S} y_v \leq |S|$ and $x_e \leq 1$. First, see that $(\hat{x}, \hat{y})$ satisfies constraint (3.4d) and the 0-1 bounds on $\hat{y}$.

To show that constraints (3.4c) are satisfied, consider a power graph edge $e = \{i,j\} \in E^k$ and a minimal length-k i,j-connector $C \in C_{ij}^k$ that, say, induces the path $i = c_0\text{-}c_1\text{-}\cdots\text{-}c_s = j$, where $s \leq k$. Then,

$$
\begin{aligned}
\hat{x}_e + \sum_{v \in C} \hat{y}_v &\geq \hat{u}_{ij}^k + \sum_{v \in C} \hat{y}_v \\
&\geq \left(\hat{u}_{c_1 j}^{k-1} - \hat{y}_i\right) + \sum_{v \in C} \hat{y}_v \\
&\geq \left(\hat{u}_{c_2 j}^{k-2} - \hat{y}_{c_1} - \hat{y}_i\right) + \sum_{v \in C} \hat{y}_v \\
&\geq \dots \\
&\geq \left(\hat{u}_{c_{s-1} j}^{k-(s-1)} - \sum_{t=0}^{s-2} \hat{y}_{c_t}\right) + \sum_{v \in C} \hat{y}_v \\
&= \hat{u}_{c_{s-1} j}^1 + \hat{y}_{c_{s-1}} + \hat{y}_j \geq 1,
\end{aligned}
$$

where the first inequality holds by $x_e \geq u_{ij}^k$, the middle inequalities hold by constraints (3.1g), the equality holds by constraints (3.1e), and the last inequality holds by constraints (3.1c).

Finally, to show $\hat{x}_e \geq 0$ holds for power graph edges $e = \{i, j\} \in E^k$, observe that $\hat{x}_e \geq \hat{u}_{ij}^k \geq 0$. □

3.4 Implementation Details

Here, we detail implementations of the recursive, path-like, and thin formulations, including:

- a procedure that identifies "non-critical" nodes i for which we can fix $y_i = 0$,
- a heuristic used to provide warm start solutions,
- separation routines for the length-k i,j-connector inequalities (3.4c).

3.4.1 Variable Fixing

Recall the leaf fixing of Veremyev et al. (2015).

Remark 4. *Assume hop-based distances and unit connection costs* $c_e = 1$*. If* i *is a leaf vertex whose stem* j *satisfies* $\deg_G(j) \geq 2$ *and* $a_j \leq a_i$*, then there is an optimal solution in which* $y_i = 0$*.*

This remark follows by a swap argument: if a feasible solution D contains vertex i, then the objective will be no worse when leaf i is swapped with its stem j. All such variables y_i can be fixed to zero when no two leaves are adjacent.

We generalize this remark by showing that y_i can be fixed to zero when i is *simplicial*, i.e., its neighborhood $N(i)$ is a clique. A set of such variables $\{y_i \mid i \in I\}$ can simultaneously be fixed to zero when I is independent. Further, we observe that a *maximum cardinality* independent set of simplicial vertices can be found in time $\mathcal{O}(mn)$.

Proposition 6. *Assume hop-based distances and* $c_e \geq 0$ *for* $e \in E^k$*. If a subset of vertices* $I \subseteq V$ *satisfies conditions 1 and 2 below, then there is an optimal deletion set* $D^* \subseteq V$ *with* $D^* \cap I = \emptyset$*.*

1. *I is an independent set of simplicial vertices in G;*

2. *for every vertex $i \in I$ and every one of its neighbors $u \in N(i)$:*

    - $a_i \geq a_u$;

    - $c_e \leq c_{e'}$ *for every* $e \in \delta_{G^k}(i)$ *and* $e' \in \delta_{G^k}(u)$.

Proof. Let be a vertex subset $I \subseteq V$ satisfying conditions 1 and 2, and let $D_0 \subseteq V$ be a feasible solution with $D_0 \cap I \neq \emptyset$. We claim that there is a feasible solution $D_1 \subseteq V$, with the properties:

1. D_1 has one fewer vertex of I than D_0 does, i.e., $|D_1 \cap I| = |D_0 \cap I| - 1$; and

2. the objective value of D_1 is at least as good as that of D_0, i.e., $\text{obj}(D_1) \leq \text{obj}(D_0)$.

The proposition would then follow by repeated application of this claim.

To prove the claim, let i be a vertex from $D_0 \cap I$. In the first case, all neighbors of i belong to D_0 in which case $D_1 \coloneqq D_0 \setminus \{i\}$ proves the claim. In the other case, suppose that a neighbor $u \in N(i)$ of i does not belong to D_0. Observe that u cannot belong to I, because I is independent and contains i (a neighbor of u). So,

$$D_1 \coloneqq (D_0 \setminus \{i\}) \cup \{u\}$$

satisfies the first property $|D_1 \cap I| = |D_0 \cap I| - 1$. Also, D_1 satisfies the budget constraint by

$$\sum_{v \in D_1} a_v = \sum_{v \in D_0} a_v + a_u - a_i \leq \sum_{v \in D_0} a_v \leq b.$$

So, all that remains is to show that $\text{obj}(D_1) \leq \text{obj}(D_0)$. Using the shorthand

$$\begin{aligned} \delta_0 &= \delta_{(G-D_0)^k}(u) & \qquad \delta_1 &= \delta_{(G-D_1)^k}(i) \\ E_0 &= E((G-D_0)^k) & \qquad E_1 &= E((G-D_1)^k), \end{aligned}$$

we argue that

$$\text{obj}(D_1) = \sum_{e \in E_1} c_e = \sum_{e \in E_1 \setminus \delta_1} c_e + \sum_{e \in \delta_1} c_e \leq \sum_{e \in E_0 \setminus \delta_0} c_e + \sum_{e \in \delta_0} c_e = \sum_{e \in E_0} c_e = \text{obj}(D_0).$$

To prove the middle inequality, we show that the following inequalities hold.

$$\sum_{e \in \delta_1} c_e \leq \sum_{e \in \delta_0} c_e \tag{3.6}$$

$$\sum_{e \in E_1 \setminus \delta_1} c_e \leq \sum_{e \in E_0 \setminus \delta_0} c_e. \tag{3.7}$$

The former inequality (3.6) holds because if $\{i, v\} \in \delta_1$ then $\{u, v\} \in \delta_0$ which holds because if there is a short path from i to v in $G - D_1$, then the same path—but with u substituted for i—exists in $G - D_0$; moreover, $c_{\{i,v\}} \leq c_{\{u,v\}}$ and all connection costs are nonnegative. Meanwhile, the latter inequality (3.7) holds because $E_1 \setminus \delta_1 \subseteq E_0 \setminus \delta_0$ and by the nonnegativity of the connection costs $c_e \geq 0$. To see the inclusion $E_1 \setminus \delta_1 \subseteq E_0 \setminus \delta_0$, consider $\{v, w\} \in E_1 \setminus \delta_1$. So, $\text{dist}_{G-D_1}(v, w) \leq k$ and $i, u \notin \{v, w\}$. Let P_{vw} be a shortest v, w-path in $G - D_1$. If this path does not cross i, then the same path exists in $G - D_0$, and thus $\text{dist}_{G-D_0}(v, w) \leq k$. Meanwhile, if P_{vw} crosses i, then a similar path P'_{vw} (of the same length) can be obtained in $G - D_0$ by replacing i with u, in which case $\text{dist}_{G-D_0}(v, w) \leq k$. Thus, $\{v, w\} \in E_0 \setminus \delta_0$. □

Now, to use Proposition 6, we need a procedure that finds an independent set I of simplicial nodes. For this task, we use the algorithm below. It finds a set I of maximum size, thus maximizing the number of y_i variables that can be fixed.

```
FindNoncriticalNodes()

  1. find $S := \{v \in V \mid \text{is simplicial in } G\}$;

  2. find $S' := \{v \in S \mid v \text{ satisfies condition 2}\}$;

  3. pick one vertex $v_i$ from each of the components $G_1$, $G_2$, ..., $G_p$ of $G[S']$;

  4. return $I := \{v_1, v_2, \ldots, v_p\}$.
```

In our computational experiments, instances have unit deletion costs $a_v = 1$ for all $v \in V$ and unit connection costs $c_e = 1$ for all $e \in E^k$. Under these settings $S' = S$, allowing us to skip step 2 in our implementation.

Proposition 7. *The algorithm* `FindNoncriticalNodes` *finds a maximum independent set of simplicial nodes (that satisfies condition 2 of Proposition 6) in time* $\mathcal{O}(nm)$.

Proof. First consider the running time. Checking whether vertex v is simplicial takes time $\mathcal{O}(n+m)$. For example, store $N(v)$ as a boolean n-vector and make one pass through the edges, counting the number of the $\{i,j\} \in E$ for which i and j both belong to $N(v)$. This number will be $\binom{\deg(v)}{2}$ if and only if $N(v)$ is a clique. In this way, step 1 takes time $\mathcal{O}(nm)$. Step 2 takes linear time $\mathcal{O}(m+n)$ by precomputing and storing the values $\max\{c_e \mid e \in \delta(i)\}$ and $\min\{c_e \mid e \in \delta(i)\}$ for each vertex i. Then, to check condition 2, it suffices to check that $\max\{c_e \mid e \in \delta(i)\} \leq \min\{c_{e'} \mid e' \in \delta(u)\}$ for each $u \in N(i)$. Finally, steps 3 and 4 also take linear time $\mathcal{O}(m+n)$. Thus, the total time is $\mathcal{O}(nm)$. This is not too costly given that creating the power graph already takes time $\mathcal{O}(nm)$.

By steps 1 and 2, $I \subseteq S'$ and so every vertex $i \in I$ is simplicial and satisfies condition 2 of Proposition 6. Moreover, step 3 ensures that I is independent. Finally, to prove that I is maximum, see that any independent set of simplicial nodes F must be a subset of S'. Further, we claim that each component G_i of $G[S']$ is a complete graph, in which case no more than one vertex can be selected from each in F, i.e., $|F| \leq p$. To see this, observe that each component G' of $G[S]$ is a complete graph. This holds because if G' is not complete, then it has $q := \operatorname{diam}(G') \geq 2$, in which case a diameter-inducing path u_0-u_1-$\cdots$-u_q has a vertex u_1 that is not simplicial in G' and thus not simplicial in G, which is a contradiction. So, I is maximum. □

Meanwhile, we take L to be a maximum independent set of leaves, which can be bound in linear time $\mathcal{O}(m+n)$. Note that there is always a choice for L and I for which $L \subseteq I$, so the simplicial vertex fixing is always at least as powerful as the leaf fixing. On average, I is three times the size of I, as reported in Table 3.1. For example, simplicial vertex fixing finds 2,484 and 4,846 more non-critical nodes on the instances `hep-th` and `cond-mat`, respectively.

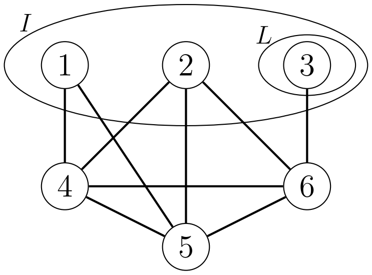


Figure 3.7: An illustration of the sets L and I.

3.4.2 Heuristic

Our heuristic is based on a CNP heuristic of Addis et al. (2016). Their heuristic, called `Greedy3`, begins with a vertex cover D of the graph. Then, vertices are removed from D in a greedy fashion until its size is *sufficiently smaller* than the deletion budget, at which point vertices are added back to D in a greedy fashion until the budget is tight. We make two changes. First, instead of beginning with a vertex cover, we begin with a set D of size $2b$ containing the vertices of largest betweenness centrality. Second, we terminate the heuristic as soon as the size of D reaches b. These changes are motivated in part by the expensive DCNP objective function evaluations, which take time $\mathcal{O}(mn)$, as opposed to $\mathcal{O}(m+n)$ for CNP. In our experience, these changes have a negligible impact on DCNP solution quality, but a noticeable impact on running time.

Definition 9 (Betweenness Centrality, Freeman (1977))**.** *Let σ_{ij} be the number of shortest paths between vertices i and j, and let $\sigma_{ij}(v)$ be the number of shortest paths between i and j that contain vertex v. The betweenness centrality of vertex v is defined as*

$$C_B(v) := \sum_{\{i,j\} \in \binom{V \setminus \{v\}}{2}} \sigma_{ij}(v)/\sigma_{ij}.$$

Brandes (2001) shows that all betweenness centrality values can be computed in time $\mathcal{O}(mn)$ when distances are hop-based, and in time $\mathcal{O}(mn + n^2 \log n)$ when distances are edge-weighted.

Algorithm 1 Heuristic for DCNP

```
1: initialize $D$ to be the $2b$ vertices with largest betweenness centrality values
2: for counter $= 1, 2, \dots, b$ do
3:     let $v \in \operatorname{argmin}\{\text{obj}(D \setminus \{u\}) \mid u \in D\}$
4:     $D \leftarrow D \setminus \{v\}$
5: return $D$
```

Proposition 8. *Algorithm 1 takes time $\mathcal{O}(b^2mn)$ when distances are hop-based, and time $\mathcal{O}(b^2(mn + n^2 \log n))$ when distances are edge-weighted.*

Proof. First consider the hop-based case. Line 1 takes time $\mathcal{O}(mn)$ by using the algorithm of Brandes (2001). In line 3, we compute the objective value $\text{obj}(D \setminus \{u\})$ for at most $2b$ vertices u, and each function evaluation takes time $\mathcal{O}(mn)$. This means that line 3 takes time $\mathcal{O}(bmn)$. Since there are b iterations of the for loop, lines 2-4 take time $\mathcal{O}(b^2mn)$. When distances are edge-weighted, the analysis is similar, except that the betweenness centrality computations and function evaluations take time $\mathcal{O}(mn + n^2 \log n)$. □

To obtain the running time $\mathcal{O}(mn + n^2 \log n)$ for the betweenness centrality computations and function evaluations, Fibonacci heaps should be used. However, binary heaps typically lead to faster implementations in practice, due to smaller hidden constants. For this reason, our implementation uses binary heaps and has a slightly slower worst-case running time of $\mathcal{O}(b^2(mn \log n + n^2 \log n))$.

Table 3.1 provides results for this heuristic on unweighted instances when $k = 3$ and $b \in \{5, 10\}$. The heuristic solutions are often very good, and are in fact optimal for 12 of the 22 different instances where $b = 5$ and 5 of the instances where $b = 10$. Further, for most instances, the running time is a few seconds; however, larger instances take minutes. The largest instance `cond-mat` which has 16,726 vertices and 47,594 edges takes the most time: 496 seconds when $b = 5$, and 1709 seconds when $b = 10$. We expect that larger instances with, say, 100,000 vertices would require different techniques. However, our focus in this chapter is on exact approaches, and instances like `cond-mat` lie at the boundary of tractability, so this simple heuristic suffices for our purposes.

					$b = 5$			$b = 10$		Preprocessing	
Graph	n	m	$\lvert E^3\rvert$	heur	time	opt	heur	time	opt	$\lvert L\rvert$	$\lvert I\rvert$
karate	34	78	480	41	0.00	41	8	0.00	6	1	12
dolphins	62	159	1,107	678	0.01	662	340	0.02	335	9	9
lesmis	77	254	2,500	535	0.01	517	160	0.01	160	17	32
LindenStrasse	232	303	3,251	1,815	0.09	1,810	1,151	0.14	1,151	88	92
polbooks	105	441	3,510	2,673	0.04	2,555	1,867	0.11	1,715	0	4
adjnoun	112	425	5,634	3,719	0.06	3,719	2,501	0.15	2,501	10	12
football	115	613	6,247	5,362	0.06	5,362	4,590	0.15	4,523	0	0
netscience	1,589	2,742	13,087	8,898	0.31	8,390	7,026	0.84	6,785	205	680
jazz	198	2,742	18,461	18,461	0.22	16,136	14,306	0.75	14,216	5	14
SmallWorld	233	994	25,721	6,964	0.14	6,964	5,011	0.33	4,967	20	64
Erdos971	429	1,312	34,086	25,737	0.56	25,737	20,442	1.57	20,240	79	116
S.Cerevisae	1,458	1,948	39,091	25,190	2.84	25,190	19,861	8.57	19,861	722	770
USAir	332	2,126	46,573	29,486	0.30	29,486	19,628	0.95	19,157	55	122
power	4,941	6,594	53,125	52,456	39.36	50,410	51,782	136.21	48,602	1,226	1,414
H.pylori	706	1,392	62,028	37,626	0.93	37,626	28,204	3.06	27,807	263	268
Harvard500	500	2,043	83,993	19,241	0.41	16,448	9,951	1.12	8,581	79	134
homer	542	1,619	91,527	45,828	0.56	45,828	24,882	1.39	24,882	198	289
celegansm	453	2,025	91,531	44,967	0.63	44,967	26,830	2.10	25,556	6	95
email	1,133	5,451	289,259	263,409	3.36	263,409	241,144	11.74	241,128	151	197
hep-th	8,361	15,751	376,431	345,320	77.05	345,320	323,268	268.53	321,486	1,481	3,965
PGPgiant	10,680	24,316	1,145,492	860,319	240.60	857,035	769,350	795.27	744,908	4,229	5,299
cond-mat	16,726	47,594	1,761,969	1,637,445	496.13	1,633,299	1,561,855	1709.02	1,541,815	1,849	6,695

Table 3.1: Heuristic and preprocessing for $k = 3$ and $b \in \{5, 10\}$. We report the heuristic's objective value (heur), the time spent by the heuristic in seconds (time), the optimal objective (opt), the numbers of leaves ($|L|$), and the number of simplicial vertices ($|I|$).

3.4.3 The Separation Problem

Since the thin formulation generally has exponentially many constraints (3.4c), we study the associated separation problem so that violated inequalities can be added on-the-fly.

Problem: Separation problem for length-k i, j-connector inequalities (3.4c).

Input: An edge-weighted graph $G = (V, E)$, a point $(x^*, y^*) \in [0, 1]^{|E^k|+|V|}$, an integer k.

Output: (if any exist) An inequality (3.4c) of the type $x_e + \sum_{v \in C} y_v \geq 1$ that the point (x^*, y^*) violates.

We cover the case where distances are hop-based, and also when they are edge-weighted.

In both cases, we consider *integer* separation where the given point (x^*, y^*) is assumed to be integer, and *fractional* separation where no such assumption is made. Fractional separation is important because of the well-known "separation=optimization" theorem (Grötschel et al., 1993) which implies that the LP relaxation of the thin formulation can be solved in polynomial-time if and only if the associated separation problem can be solved in polynomial time. Meanwhile, the ability to perform integer separation provides no such theoretical guarantees, but still can be practically useful in a branch-and-cut algorithm (Buchanan et al., 2015; Fischetti et al., 2017b; Validi and Buchanan, 2019; Salemi and Buchanan, 2020a). Table 3.2 summarizes the results.

	integer separation	fractional separation
hop-based	$\mathcal{O}(nm)$	$\mathcal{O}(knm)$
edge-weighted	$\mathcal{O}(nm + n^2 \log n)$	(weakly) NP-hard

Table 3.2: The complexity of the integer and fractional separation problems, under hop-based and edge-weighted distances.

Integer Separation

Algorithm 2 solves the separation problem when given an integer point $(x^*, y^*) \in \{0,1\}^{|E^k|+|V|}$. This algorithm applies to both the hop-based and edge-weighted cases, with the only difference being the routine used for generating the shortest paths trees: time $\mathcal{O}(m+n)$ versus time $\mathcal{O}(m + n \log n)$ in line 3.

Algorithm 2 IntegerSeparation(G, w, x^*, y^*, k)

```
1: $D^* \leftarrow \{v \in V \mid y^*_v = 1\}$
2: for $i \in V \setminus D^*$ do
3:     find a shortest paths tree of $G - D^*$ rooted at $i$ with respect to $w$
4:     for each power graph edge $e = \{i, j\} \in E^k$ that is incident to $i$ do
5:         if $\mathrm{dist}_{G-D^*}(i, j) \leq k$ and $x^*_e = 0$ then
6:             from the shortest paths tree, find a shortest path $P$ from $i$ to $j$ in $G - D^*$
7:             add $x_e + \sum_{v \in V(P)} y_v \geq 1$
8:             break                                  ▷ optional, but needed for time bound
```

Remark 5. *Algorithm 2 runs in time $\mathcal{O}(nm)$ when distances are hop-based and in time $\mathcal{O}(nm + n^2 \log n)$ when distances are edge-weighted.*

The reason for adding the break in line 8 is to ensure that the algorithm does not spend too much time generating the paths and associated inequalities. If the break is omitted, the algorithm may add up to $|E^k|$ violated inequalities, and each will be written with respect to a path P. When distances are hop-based, each P will touch at most $k + 1$ vertices, and so the time to write the inequalities is $\mathcal{O}(k|E^k|)$, which might be larger than the bound $\mathcal{O}(nm)$, giving a total time that could be written $\mathcal{O}(knm)$. Meanwhile, when distances are edge-weighted, the paths P may cross roughly n vertices, perhaps requiring us to increase the time bound to $\mathcal{O}(n^3)$.

However, our implementation omits the break (line 8). We find that the extra time spent in the separation procedure is justified in practice. The intuition is as follows. Suppose that the point (x^*, y^*) has $x^*_e = 0$ even though its endpoints are close to each other in $G - D^*$. Then it is likely that if no inequality that acts on x_e is added, then the violated inequality for x_e will continue to be violated in the next separation call, eventually leading to a larger number of branch-and-bound nodes. Moreover, many of our experiments consider the hop-based case where k is a small constant, in which case $\mathcal{O}(knm) = \mathcal{O}(nm)$, and the additional time is negligible.

Fractional Separation

When distances are hop-based, fractional separation takes time $\mathcal{O}(knm)$ by a straightforward dynamic programming algorithm (Algorithm 3). With slight modifications, it can be applied to the edge-weighted case assuming that the edge weights are integer-valued. This would give a pseudo-polynomial running time. Meanwhile, fractional separation under edge-weighted distances is generally NP-hard, as shown in Theorem 10.

For simplicity, Algorithm 3 is described with respect to a fixed vertex i. To solve the "full" separation problem, it should be applied for each vertex $i \in V$. In the pseudocode, $d(v, s)$ stores, at termination, the cost of a cheapest at-most-s-hop path P from i to v (with respect to vertex weights $y{*}_v$). We follow the convention that $d(v, s) = +\infty$ when no such path exists. The paths are stored in p via the typical predecessor idea. Specifically, $p(v, s)$ stores, at termination, the predecessor of vertex v in the cheapest path with at most s hops from i to v.

Algorithm 3 FractionalSeparationHopBased(G, x^*, y^*, k, i)

```
for $s = 0, 1, \dots, k$ do                                          ▷ initialization
    $d(i, s) \leftarrow y_i^*$
    for $v \in V \setminus \{i\}$ do
        $d(v, s) \leftarrow +\infty$
for $s \in \{1, 2, \dots, k\}$ do                                    ▷ dynamic programming
    for $v \in N_{G^k}(i)$ do
        for $u \in N_G(v)$ do
            if $d(v, s) > d(u, s-1) + y_v^*$ then
                $d(v, s) \leftarrow d(u, s-1) + y_v^*$
                $p(v, s) \leftarrow u$
for each power graph edge $e = \{i, j\} \in E^k$ that is incident to $i$ do    ▷ add cuts
    if violation $:= 1 - x_e^* - d(j, k) > 0$ then
        let $P$ be the cheapest path from $i$ to $j$ of length at most $k$ that is stored in $p$
        add $x_e + \sum_{v \in V(P)} y_v \geq 1$.
```

Lines 5-10 are the most costly portion of Algorithm 3, taking time $\mathcal{O}(km)$. Thus, by running the algorithm n times (once for each vertex $i \in V$), these lines take a total time of $\mathcal{O}(knm)$. Meanwhile, lines 11-14 take a combined time of $\mathcal{O}(k|E^k|)$ over the n different calls, which is $\mathcal{O}(knm)$.

Remark 6. *When distances are hop-based, fractional separation of the inequalities* (3.4c) *takes time $\mathcal{O}(knm)$.*

To enable Algorithm 3 to handle integer-weighted edges, one would need to adjust the updates in lines 8-10. However, the resulting algorithm would be too slow for the instances coming from our experiments, so we do not bother with this pseudo-polynomial time extension.

Theorem 10. *Fractional separation for* $\text{THIN}^{\uparrow_x}$ *under edge-weighted distances is NP-hard.*

Proof. We prove that it is NP-hard to determine whether a given point (x^*, y^*) violates an inequality defining $\text{THIN}^{\uparrow_x}$. The reduction is from Partition in which positive integers $p_1, p_2, \dots, p_n$ and a target value $\rho \coloneqq \sum_{v=1}^{n} p_v/2$ are given as input. From this, construct the edge-weighted graph $G = (V, E)$ shown in Figure 3.8. The edge weights $\{w_e\}_{e \in E}$, and values $\{y_v^*\}_{v \in V}$ are given in the figure. Next, let $k = \rho/(\rho+1)$, $b = n$, and $a_v = 1$ for all $v \in V$. Finally, let $x^*_{\{i,j\}} = 0$ for the end vertices i and j, and let $x^*_e = 1$ for all other power graph edges $e \in E^k \setminus \{\{i,j\}\}$.

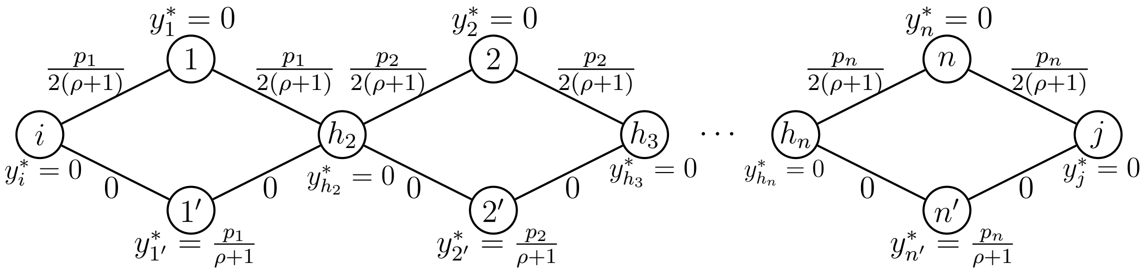


Figure 3.8: The graph used in the NP-hardness reduction.

Observe that (x^*, y^*) satisfies the budget constraint (3.4d), the 0-1 bounds on x and y, and all connector constraints (3.4c) except perhaps for power graph edge $\{i, j\}$. We show that the Partition instance has a solution if and only if (x^*, y^*) violates one of the inequalities defining $\mathrm{THIN}^{\uparrow x}$.

($\implies$) Suppose that the Partition instance admits a solution $T \subset [n]$ with $\sum_{v \in T} p_v = \rho$. Its complement $\bar{T} = [n] \setminus T$ also has weight ρ. Let $B = \{v' \mid v \in \bar{T}\}$ be the vertices v' from the bottom row whose top row counterpart v does not belong to T. We show that

$$C = \{i, j\} \cup T \cup B \cup \{h_2, h_3, \ldots, h_n\}$$

is a (minimal) length-k i, j-connector and that the associated connector inequality is violated, $x^*_{\{i,j\}} + \sum_{v \in C} y^*_v < 1$. To wit, let $\hat{E}(G[C])$ be the edges of $E\left(G[C]\right)$ with positive weight and observe that

$$\begin{aligned} \mathrm{dist}_{G[C]}(i, j) &= \sum_{e \in E(G[C])} w_e = \sum_{e \in \hat{E}(G[C])} w_e = \frac{\sum_{v \in T} p_v}{\rho + 1} = \frac{\rho}{\rho + 1} = k \\ x^*_{\{i,j\}} + \sum_{v \in C} y^*_v &= x^*_{\{i,j\}} + \sum_{v \in B} y^*_v = 0 + \frac{\sum_{v \in \bar{T}} p_v}{\rho + 1} = \frac{\rho}{\rho + 1} < 1. \end{aligned}$$

($\impliedby$) Suppose that (x^*, y^*) violates at least one of the inequalities defining $\mathrm{THIN}^{\uparrow x}$. By the discussion above, the violated inequality must be a connector inequality, say, for a minimal length-k i, j-connector C, in which case $x^*_{\{i,j\}} + \sum_{v \in C} y^*_v < 1$. Since C is a minimal i, j-connector,

$$\sum_{v \in C} y^*_v + \sum_{e \in E(G[C])} w_e = \frac{2\rho}{\rho + 1}.$$

Moreover, $\sum_{v \in C} y^*_v = \sum_{e \in E(G[C])} w_e = \frac{\rho}{\rho+1}$ because

$$\frac{\rho}{\rho + 1} = k \geq \sum_{e \in E(G[C])} w_e = \frac{2\rho}{\rho + 1} - \sum_{v \in C} y^*_v \geq \frac{\rho}{\rho + 1},$$

where the first inequality holds since C is a (minimal) length-k i, j-connector in the Figure 3.8 graph, and the last inequality holds because otherwise $x^*_{\{i,j\}} + \sum_{v \in C} y^*_v \geq 0 + \frac{\rho+1}{\rho+1} = 1$. Let T be the subset of C that belongs to the top row of vertices $\{1, 2, \ldots, n\}$, and let $\bar{T} = [n] \setminus T$. Further, let B be the subset of C that belongs to the bottom row $\{1', 2', \ldots, n'\}$. Then, $\bar{T}$ is a solution to Partition, as

$$\frac{\sum_{v \in \bar{T}} p_v}{\rho + 1} = \sum_{v \in B} y^*_v = \sum_{v \in C} y^*_v = \frac{\rho}{\rho + 1}.$$

□

3.5 Computational Experiments

Here, we experiment with different implementations of the DCNP formulations, including

- R, a direct implementation of the recursive formulation by Veremyev et al. (2015);
- PATH, direct implementation of the path-like formulation;
- THIN, a direct implementation of the thin formulation;
- THIN$_\mathrm{F}$, a branch-and-cut implementation of the thin formulation using fractional separation for inequalities (3.4c);
- THIN$_\mathrm{I}$, a branch-and-cut implementation of the thin formulation using integer separation for inequalities (3.4c).

Each implementation uses the heuristic and variable fixing procedure given in Section 3.4, and all experiments were 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 code was written in Microsoft Visual Studio 2015 in C++ for Gurobi version 7.5.1. The code and all instances are available at (Salemi and Buchanan, 2020b).

The branch-and-cut implementations invoke the parameter `LazyConstraints`; the MIP time limit is 3600 seconds; and the `method` parameter is set to concurrent.

3.5.1 Results for Hop-Based Distances

First, we report experimental results on DCNP instances in which distances are hop-based. We consider all of real-life graphs considered by Veremyev et al. (2015), as well as some other (often larger) instances. These graphs come from the Pajek dataset (Batagelj and Mrvar, 2006), the University of Florida Sparse Matrix Collection database (Davis and Hu, 2011), and the 10th DIMACS Implementation Challenge (DIMACS-10, 2017). Veremyev et al. (2015) ran experiments for $k = 3$, $b \in \{1, 2, \ldots, 10\}$, and $a_v = 1$ for all $v \in V$. Given that the performance may be sensitive to k, we consider $k \in \{3, 4\}$. Also, as DCNP can be brute-forced for small b, we consider $b \in \{5, 10\}$.

Tables 3.3 and 3.4 report the total time in seconds (including preprocessing, heuristic, and model build time), or the best lower and upper bounds [LB,UB] within a 3600 second time limit. An entry of LPNS denotes that the LP relaxation was not solved within the time limit (due to build time or solve time), while MEM denotes a memory crash.

The thin implementations perform the best and are often faster than R by an order of magnitude. Implementation THIN solves all of the instances that R and PATH can solve, plus 9 and 4 others, respectively. Some of the larger instances like `hep-th`, `PGPgiant`, and `cond-mat` cause R to crash, while the other implementations do not. Surprisingly, however, we find that the thin implementations fail to solve the 198-node instance `jazz` when $b = 10$. Examination of the logs reveals that Gurobi is spending an inordinate amount of time to solve the LP relaxations at its branch-and-bound nodes. We suspect that this behavior results from the dual simplex method encountering degeneracy. To remedy the issue, we forced Gurobi to solve the LP relaxations at every branch-and-bound node using the barrier method. This enabled Gurobi to solve this instance in 3182.93 seconds. Another observation from Tables 3.3 and 3.4 is that THIN typically outperforms THIN_I and THIN_F when $k = 3$. Indeed, THIN is quicker than THIN_I and THIN_F on 37 of these 44 instances. Tweaks to the branch-and-cut implementations (e.g., varying the cut violation threshold, varying the initial constraints, limits to the number of cuts per callback) did not change this.

Graph	n	m	$\lvert E^k \rvert$	obj	R	PATH	THIN_I	THIN_F	THIN
karate	34	78	480	41	0.14	0.08	0.06	0.04	0.04
dolphins	62	159	1,107	662	5.22	3.73	0.65	0.66	0.74
lesmis	77	254	2,500	517	0.55	0.42	0.21	0.20	0.21
LindenStrasse	232	303	3,251	1,810	0.70	0.41	0.38	0.35	0.21
polbooks	105	441	3,510	2,555	15.10	7.12	3.41	3.09	2.56
adjnoun	112	425	5,634	3,719	2.68	2.76	1.48	1.32	0.97
football	115	613	6,247	5,362	235.46	266.76	139.49	97.13	86.48
netscience	1,589	2,742	13,087	8,390	15.56	1.90	2.44	2.49	1.19
jazz	198	2,742	18,461	16,136	[15732,16185]	[16074,16136]	1840.59	1904.99	1081.24
SmallWorld	233	994	25,721	6,964	21.87	9.33	8.36	7.66	5.40
Erdos971	429	1,312	34,086	25,737	20.39	14.69	21.28	16.00	9.93
S.Cerevisae	1,458	1,948	39,091	25,190	22.19	10.09	9.41	9.62	5.68
USAir	332	2,126	46,573	29,486	54.13	50.70	39.57	40.26	26.09
power	4,941	6,594	53,125	50,410	188.23	64.64	57.42	77.24	46.13
H.Pylori	706	1,392	62,028	37,626	97.60	16.17	18.89	24.04	8.63
Harvard500	500	2,043	83,993	16,448	63.97	25.23	17.95	20.76	14.54
homer	542	1,619	91,527	45,828	408.50	365.28	63.53	63.70	32.51
celegansm	453	2,025	91,531	44,967	182.39	54.67	39.86	40.13	43.40
email	1,133	5,451	289,259	263,409	1545.02	198.05	184.24	190.21	114.46
hep-th	8,361	15,751	376,431	345,320	MEM	379.11	250.11	257.69	171.94
PGPgiant	10,680	24,316	1,145,492	857,035	MEM	2025.69	685.54	717.38	704.55
cond-mat	16,726	47,594	1,761,969	1,633,299	MEM	LPNS	3688.16	3683.96	2385.66

Table 3.3: Running times, or bounds at termination, when $(k, b) = (3, 5)$.

Graph	n	m	$\lvert E^k \rvert$	obj(D)	R	PATH	THIN_I	THIN_F	THIN
karate	34	78	480	6	0.14	0.09	0.10	0.11	0.04
dolphins	62	159	1,107	335	4.84	2.23	1.49	1.36	1.00
lesmis	77	254	2,500	160	0.51	0.37	0.28	0.30	0.17
LindenStrasse	232	303	3,251	1,151	0.72	0.47	0.44	0.44	0.31
polbooks	105	441	3,510	1,715	20.52	10.44	7.21	5.86	3.44
adjnoun	112	425	5,634	2,501	2.99	2.98	2.95	2.84	1.10
football	115	613	6,247	4,523	2158.69	987.90	605.52	644.58	265.17
netscience	1,589	2,742	13,087	6,785	16.32	2.42	2.65	2.60	1.87
jazz	198	2,742	18,461	14,216	[13074,14306]	[13020,14306]	[13620,14306]	[13500,14306]	[13579,14306]
SmallWorld	233	994	25,721	4,967	1261.81	83.68	62.39	59.83	19.48
Erdos971	429	1,312	34,086	20,240	745.14	193.83	72.79	75.72	53.68
S.Cerevisae	1,458	1,948	39,091	19,861	29.36	15.71	18.00	17.88	11.13
USAir	332	2,126	46,573	19,157	2316.72	1086.97	319.18	321.08	289.04
power	4,941	6,594	53,125	48,602	272.93	165.39	153.16	149.06	142.85
H.Pylori	706	1,392	62,028	27,807	87.17	19.08	24.28	26.10	11.45
Harvard500	500	2,043	83,993	8,581	54.83	29.59	14.00	13.67	17.17
homer	542	1,619	91,527	24,882	42.07	33.38	33.59	32.94	20.35
celegansm	453	2,025	91,531	25,556	[25183,25556]	2325.78	103.66	115.01	129.35
email	1,133	5,451	289,259	241,128	[241060,241144]	1565.70	345.82	490.56	200.83
hep-th	8,361	15,751	376,431	321,486	MEM	572.07	458.58	457.04	357.96
PGPgiant	10,680	24,316	1,145,492	744,908	MEM	[744769,748537]	1568.55	1838.73	1590.45
cond-mat	16,726	47,594	1,761,969	1,541,815	MEM	LPNS	4778.22	4884.87	3390.20

Table 3.4: Running times, or bounds at termination, when $(k, b) = (3, 10)$.

The results given in Tables 3.5 and 3.6 lead to similar conclusions for $k = 4$ and $b \in \{5, 10\}$.

That is, the thin implementations perform the best, with THIN solving all of the instances that R and PATH can solve, plus 6 and 5 others, respectively. However, the instances appear to be more challenging when $k = 4$ as opposed to $k = 3$. In fact, the largest instances `PGPgiant` and `cond-mat` cause R, PATH, and THIN to crash. This is explained by the large numbers of variables and constraints. For `PGPgiant`, there are more than 4 million variables and 25 million constraints. Meanwhile, `cond-mat` requires more than 7 million variables and 27 million constraints.

Graph	n	m	$\|E^k\|$	obj(D)	R	PATH	THIN$_\text{I}$	THIN$_\text{F}$	THIN
karate	34	78	553	44	0.31	0.18	0.05	0.06	0.05
dolphins	62	159	1,459	764	15.61	42.77	1.27	0.85	1.51
lesmis	77	254	2,899	583	2.46	3.64	0.72	0.79	1.05
LindenStrasse	232	303	7,257	3,526	4.27	2.03	1.85	1.99	1.36
polbooks	105	441	4,685	3,333	61.86	153.33	11.18	14.59	22.70
adjnoun	112	425	6,178	4,777	316.60	1556.25	46.21	72.67	259.61
football	115	613	6,555	?	[5667,5987]	[5657,5994]	[5763,5984]	[5759,5986]	[5648,5987]
netscience	1,589	2,742	22,847	11,786	35.97	5.61	4.68	5.69	3.56
jazz	198	2,742	19,336	17,350	[16626,17799]	LPNS	[16802,17799]	[16852,17799]	[16597,17799]
SmallWorld	233	994	27,028	9,606	68.80	45.82	24.01	20.38	23.94
Erdos971	429	1,312	62,059	49,325	458.64	132.55	626.12	417.16	70.57
S.Cerevisae	1,458	1,948	117,958	66,402	57.62	47.10	52.80	52.64	28.85
USAir	332	2,126	53,447	33,795	[33390,33906]	[33390,33906]	787.38	680.65	361.17
power	4941	6,594	105,233	97,949	312.64	136.98	78.32	77.88	97.41
H.Pylori	706	1,392	162,758	109,368	392.08	117.58	581.60	578.25	83.34
Harvard500	500	2,043	119,080	37,491	861.51	235.66	321.93	304.99	201.56
homer	542	1,619	133,947	76,068	968.86	246.72	1064.79	935.02	137.17
celegansm	453	2,025	100,476	71,463	[70233,71463]	[70233,71463]	[66186,71463]	[70184,71463]	2602.32
email	1,133	5,451	548,801	?	LPNS	LPNS	[258207,522824]	[258207,522824]	LPNS
hep-th	8,361	15,751	1,340,125	1,216,604	MEM	2872.79	[329778,1216604]	[329778,1216604]	1922.82
PGPgiant	10,680	24,316	4,211,853	?	MEM	MEM	[833432,3122094]	[833432,3122094]	MEM
cond-mat	16,726	47,594	7,586,150	?	MEM	MEM	[1586132,6976109]	[1586132,6976109]	MEM

Table 3.5: Running times, or bounds at termination, when $(k, b) = (4, 5)$.

Graph	n	m	$\|E^k\|$	obj(D)	R	PATH	THIN$_\text{I}$	THIN$_\text{F}$	THIN
karate	34	78	553	6	0.31	0.23	0.29	0.12	0.10
dolphins	62	159	1,459	428	36.48	44.91	5.69	3.39	5.96
lesmis	77	254	2,899	178	2.81	4.65	1.01	0.45	1.13
LindenStrasse	232	303	7,257	1,913	2.25	2.19	3.91	1.93	1.33
polbooks	105	441	4,685	2,118	166.64	676.52	25.36	29.74	84.47
adjnoun	112	425	6,178	3,694	1262.93	[3364,3708]	216.16	252.27	1135.64
football	115	613	6,555	?	[4707,5356]	[4664,5423]	[4807,5423]	[4774,5408]	[4708,5419]
netscience	1,589	2,742	22,847	8,778	78.53	11.96	6.33	12.96	6.16
jazz	198	2,742	19,336	15,259	[13894,16036]	LPNS	[14234,15615]	[14352,16036]	[13888,16036]
SmallWorld	233	994	27,028	6,046	[6014,6048]	[6024,6057]	95.76	107.22	239.14
Erdos971	429	1,312	62,059	41,097	[40360,41375]	[40251,41375]	3542.00	3249.56	[40545,41097]
S.Cerevisae	1,458	1,948	117,958	47,324	64.38	54.37	47.74	49.04	36.66
USAir	332	2,126	53,447	24,935	[22916,27343]	[22832,27343]	[24762,24935]	[24498,24958]	[23569,25281]
power	4,941	6,594	105,233	92,522	554.98	286.08	188.65	241.37	188.63
H.Pylori	706	1,392	162,758	82,441	557.10	161.63	1721.81	1637.42	111.40
Harvard500	500	2,043	119,080	16,708	526.05	331.79	172.88	181.50	116.00
homer	542	1,619	133,947	46,396	961.41	256.87	3309.51	2478.75	171.59
celegansm	453	2,025	100,476	48,046	[47866,52157]	[47866,52157]	[38641,52157]	[47861,48046]	857.29
email	1,133	5,451	548,801	?	LPNS	MEM	[236046,498364]	[236046,498364]	LPNS
hep-th	8,361	15,751	1,340,125	1,123,002	MEM	3078.12	[306113,1125603]	[306113,1125603]	1802.32
PGPgiant	10,680	24,316	4,211,853	?	MEM	MEM	[721537,2673960]	[721537,2673960]	MEM
cond-mat	16,726	47,594	7,586,150	?	MEM	MEM	[1495035,6571710]	[1495035,6571710]	MEM

Table 3.6: Running times, or bounds at termination, when $(k, b) = (4, 10)$.

3.5.2 Results for Edge-Weighted Distances

Now, we turn to DCNP instances in which distances are edge-weighted. Test instances (including edge weights w_e) were collected from the Transportation Networks Repository (Stabler et al., 2019) and the Hazmat Network Data of STOM-Group (2019). As before, we take $a_v = 1$ for all $v \in V$, and $b \in \{5, 10\}$. The distance threshold k is chosen so that the number of vertex pairs $\{i, j\}$ with $\text{dist}_G(i, j) \leq k$ is approximately $\alpha\binom{n}{2}$. Tables 3.7 and 3.8 provide results when $\alpha \in \{0.05, 0.10\}$. We only provide results for the THIN$_\text{I}$ implementation, as the other implementations would require an exponential number of initial constraints or have an NP-hard separation problem (Theorem 10). The tables also report the heusistic's objective (heur) and the time spent by the heuristic (htime).

We see that the THIN$_\text{I}$ implementation is able to solve edge-weighted instances with up to 1,000 nodes. Instances with fewer than 500 nodes are solved in a few seconds. Meanwhile, instances with more than 1,000 nodes pose quite a challenge, certainly due in part to the large number of power graph edges $|E^k|$ and associated variables $\{x_e\}_{e \in E^k}$.

					$b = 5$				$b = 10$			
Graph	n	m	k	$\|E^k\|$	heur	opt	htime	time	heur	opt	htime	time
Albany	90	149	44	204	139	136	0.08	0.19	94	91	0.19	0.27
Buffalo	90	149	260	205	127	127	0.06	0.10	94	89	0.18	0.22
DC-NY-BOS	317	509	5,286	2,505	1967	1910	0.84	2.68	1636	1510	2.96	4.99
Korean	324	440	50	2,619	2227	2038	0.80	1.16	1918	1724	2.65	3.16
Anaheim	416	634	7,709	4,348	3587	3540	1.48	2.43	3055	3012	5.34	6.80
Barcelona	930	1,798	127	21,778	20640	20259	7.08	147.83	19674	18977	28.05	[18871,19100]
Rome	3,353	4,831	2,888	281,058	259481	?	98.00	[253268,259481]	251184	?	381.45	[222668,251184]
Austin	7,388	10,591	464	1,368,735	1352766	?	594.11	[30408,1352766]	1336801	?	2289.14	[30275,1336801]
Chicago	12,979	20,627	889	4,213,117	4190256	?	1970.02	[65083,4190256]	4168480	?	7650.10	[64922,4168480]

Table 3.7: Results for $\alpha = 0.05$.

					$b = 5$				$b = 10$			
Graph	n	m	k	$\|E^k\|$	heur	opt	htime	time	heur	opt	htime	time
Albany	90	149	65	403	256	247	0.06	0.20	158	157	0.20	0.38
Buffalo	90	149	410	402	272	269	0.07	0.18	195	179	0.15	0.25
DC-NY-BOS	317	509	8,641	5,010	4224	3848	0.78	4.68	3394	3154	2.47	10.16
Korean	324	440	78	5,308	4521	4025	0.78	1.58	3669	3154	2.19	3.70
Anaheim	416	634	11,036	8,637	7161	7009	1.38	3.92	6251	5977	4.21	8.30
Barcelona	930	1,798	185	43,449	41644	40126	7.30	[38374,41104]	39344	?	25.89	[33164,38674]
Rome	3,353	4,831	4,189	561,987	524792	?	96.60	[462892,524792]	506983	?	386.29	[13502,506983]
Austin	7,388	10,591	716	2,729,813	2698516	?	598.36	[30793,2698516]	2678413	?	2249.78	[30660,2678413]
Chicago	12,979	20,627	1,325	8,428,119	8386922	?	2095.49	[65187,8386922]	8350868	?	8214.38	[65026,8350868]

Table 3.8: Results for $\alpha = 0.10$.

3.5.3 Critical Nodes of the Buffalo, NY Highway Network

To illustrate the differences between CNP and DCNP on edge-weighted instances, consider the Buffalo network. The edges of this network represent segments of the major highways near Buffalo, New York and are weighted based on their length. Roughly 5% of node pairs are within 2.6 miles of each other, and 10% of node pairs are within 4.1 miles of each other[3]. Figures 3.9, 3.10, and 3.11 provide optimal solutions for DCNP ($\alpha = 0.05$), DCNP ($\alpha = 0.10$), and CNP, respectively, when $b \in \{5, 10\}$. As expected, the CNP solutions split the network into multiple pieces with whatever nodes work. For example, both CNP solutions split Grand Island (the wheel-like subgraph near the upper-left) off from the mainland along I-190, but the DCNP solutions do not. Meanwhile, the DCNP solutions tend to favor "hubs" that have many neighbors and other nearby nodes. For example, when $b = 5$, both DCNP solutions select a node from downtown Buffalo (just below the center of the graph), while the CNP solution does not. Another observation is that the DCNP solutions appear more stable as

[3] The edge weights are originally provided rounded to the nearest hundredth mile; we multiply them by 100 so that each weight w_e is an integer, so $k = 410$ represents 4.1 miles.

the budget increases. Specifically, as the budget doubles from $b = 5$ to $b = 10$, four of the five initial DCNP nodes remain selected. For CNP, the solutions change more, with only two of the five initial nodes remaining selected.

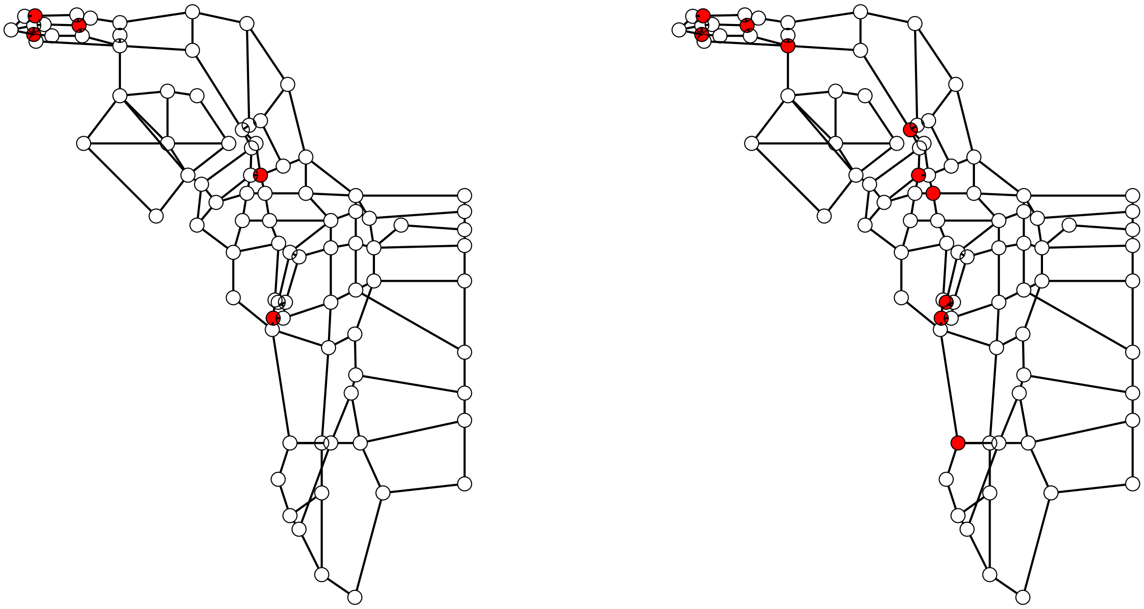


Figure 3.9: DCNP solutions for Buffalo network when $\alpha = 0.05$ and $b \in \{5, 10\}$.

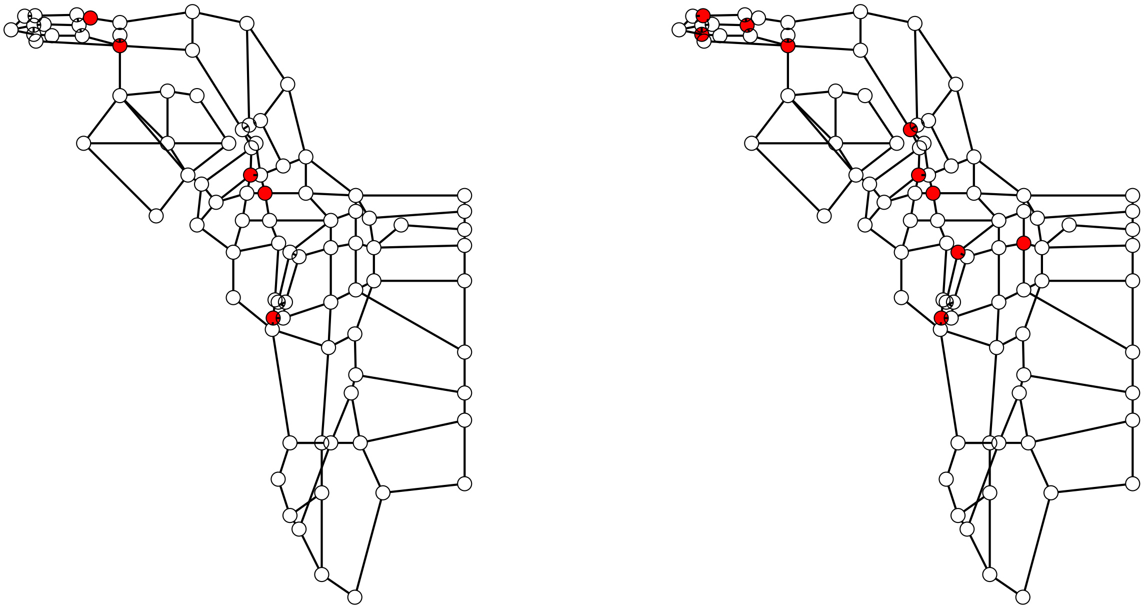


Figure 3.10: DCNP solutions for Buffalo network when $\alpha = 0.10$ and $b \in \{5, 10\}$.

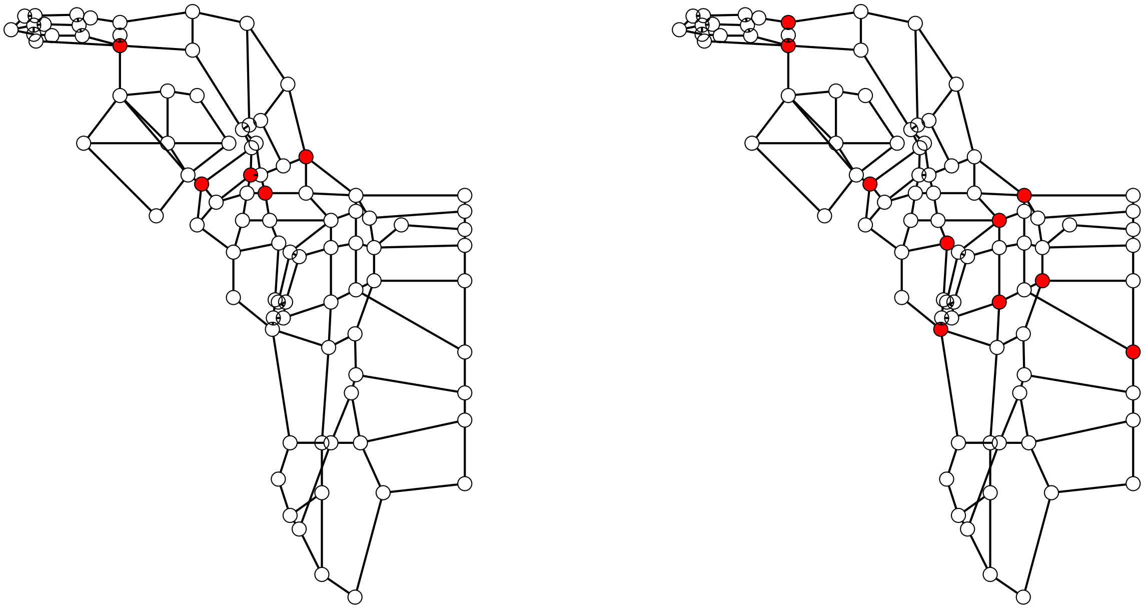


Figure 3.11: CNP solutions for Buffalo network when $b \in \{5, 10\}$.

3.6 Conclusion

In this chapter, we propose new path-like and thin integer programming formulations for the distance-based critical node problem. Under hop-based distances (and nonnegative connection costs), these new formulations are equivalent in strength to the previously existing *recursive* formulation of Veremyev et al. (2015). To prove this equivalence, we introduce the notion of the partial dominant of a polyhedron. The newly proposed thin formulation is the fastest formulation on real-life graphs, often taking a tenth of the time of the recursive formulation, and solving larger instances than were solvable before. A branch-and-cut implementation of the thin formulation is also able to handle instances in which distances are edge-weighted. This enables us to solve road network instances of the distance-based critical node problem; such instances could not have been handled with previous formulations.

Chapter IV

A Benders Decomposition Algorithm to Solve Critical Node Problem

This chapter is based on work with Austin Buchanan. Here we provide a different approach to solve the (Distance-based) Critical Node Problem. Indeed, we use the thin formulation as introduced in Chapter III and develop a Benders decomposition algorithm to solve it efficiently. Benders decomposition algorithm was initially proposed by Benders (1962) to solve mixed variables programming problems such as MIP formulations. The basic idea in the classical version of Benders algorithm is to decompose a problem by partitioning its variables.

4.1 Literature Review

Over the years, Benders decomposition methods have been used to solve optimization problems in diverse areas such as production planning (Behnamian, 2014), power management (Jenabi et al., 2015), vehicle routing (Corréa et al., 2007), and network design (Botton et al., 2013), among others. Moreover, different types of optimization problems have been solved by Benders decomposition approaches. Some examples are pure binary problems (Cordeau et al., 2001), mixed-integer nonlinear problems (De Camargo et al., 2011), robust optimization problems (Emami et al., 2017), and bilevel problems (Fontaine and Minner, 2014). The interested readers are encouraged to consult the literature review of Rahmaniani et al. (2017) for a more comprehensive list of applications and optimization problems solved by Benders decomposition approaches.

In the context of critical node problems, Hooshmand et al. (2019) propose a Benders decomposition method to maximize the generalized Wiener index. This index considers the total distances between all pairs of vertices in the remaining graph as a (dis)connectivity metric. Moreover, Naoum-Sawaya and Buchheim (2016) propose a Benders decomposition method to solve a general class of robust critical node problems (RCNP), where the connection costs c_e belong to an uncertainty set $\mathcal{U}$. They basically use a formulation that is identical to the one proposed by Di Summa et al. (2012) and decompose it into a subproblem and a master problem: The subproblem deals with realizations of the connection costs in the set $\mathcal{U}$ and contains binary variables indicating whether two nodes are connected by a path in the remaining graph. Meanwhile, the master problem includes binary variables indicating whether a node is chosen to be deleted or not. In this chapter, we present a Benders algorithm to solve a class of CNP and DCNP in which the the task is to delete a subset of b critical nodes to minimize the number of node pairs that remain connected by a path of any length and by a path of length at most k, respectively. We use thin as our original formulation.

4.2 Benders Decomposition

Recall from Chapter III that y_i is a binary variable representing the decision to include i in the deletion set $D \subseteq V$, and x_e is a binary variable representing whether the endpoints of $e \in E^k$ are distance at most k apart in $G - D$. Also, for any $e = \{i, j\} \in E^k$, let c_e denote the connection cost and b denote the deletion budget. As before, C^k_{ij} is the set of all minimal length-k-i, j connectors, as defined in Chapter III. Recall that when $c_e \geq 0$ for all $e \in E^k$, then the thin formulation for the (distance-based) critical node problem is as follows.

$$\min \sum_{e \in E^k} c_e x_e \tag{4.1a}$$

$$x_e + \sum_{v \in C} y_v \geq 1 \qquad \forall C \in C_{ij}^k,\ \forall e = \{i,j\} \in E^k \tag{4.1b}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{4.1c}$$

$$x_e \geq 0 \qquad \forall e \in E^k \tag{4.1d}$$

$$y_i \in \{0,1\} \qquad \forall i \in V. \tag{4.1e}$$

This formulation has $|E^k| + n$ variables and exponentially many constraints. Throughout this chapter, we assume that $c_e \geq 0$ for all $e \in E^k$.

By partitioning the initial continuous variables x and integer variables y of the thin formulation, we can obtain two problems: a subproblem and a master problem. Our subproblem includes the x variables and fixed values of the y variables. A solution to the subproblem provides feasibility and optimality cuts that should be added iteratively to the master problem in pursuit of finding an optimal solution. The master problem, which is a relaxed version of the original problem, only contains y variables. The objective values of the subproblem and master problem provide an upper bound and a lower bound on the optimal objective value of the original problem, respectively. The Benders algorithm terminates when these two bounds are equal. In what follows, we present our subproblem and master problem. We also explain how to generate optimality cuts.

Define $Y := \left\{ y \in [0,1]^n \;\middle|\; \sum_{i \in V} a_i y_i \leq b \right\}$. We can restate our thin formulation as follows.

$$\min_{y \in Y \cap \{0,1\}^n} \left\{ \min_{x \geq 0} \left\{ \sum_{e \in E^k} c_e x_e \;\middle|\; x_e \geq 1 - \sum_{v \in C} y_v \quad \forall C \in C_{ij}^k,\ \forall e = \{i,j\} \in E^k \right\} \right\}.$$

Define a dual variable $\pi_{C,e}$ for each constraint of the inner minimization problem (primal subproblem). Then, the above problem can be restated as

$$\min_{y \in Y \cap \{0,1\}^n} \left\{ \max_{\pi \geq 0} \left\{ \sum_{e=\{i,j\} \in E^k} \sum_{C \in C^k_{ij}} \left(1 - \sum_{v \in C} y_v \right) \pi_{C,e} \;\middle|\; \sum_{C \in C^k_{ij}} \pi_{C,e} \leq c_e \quad \forall e = \{i,j\} \in E^k \right\} \right\},$$

which is equivalent to

$$\min_{\substack{y \in Y \cap \{0,1\}^n; \\ z \in \mathbb{R}}} \left\{ z \;\middle|\; z \geq \sum_{e=\{i,j\} \in E^k} \sum_{C \in C^k_{ij}} \left(1 - \sum_{v \in C} y_v \right) \pi_{C,e} \quad \forall \pi_{C,e} \in \Pi \right\},$$

where Π is the set of all feasible solutions to the inner maximization problem (dual subproblem). Observe that the feasible region of the dual subproblem, which is always feasible and bounded, does not depend on y. This implies that no feasibility cut is needed to be added to our master problem.

Theorem 11 shows how to solve the primal and dual subproblems combinatorially when given a vector $\tilde{y} \in Y$.

Theorem 11. *Let $\tilde{y} \in Y$ and define*

$$E^k(\tilde{y}) := \left\{ e = \{i,j\} \in E^k \;\middle|\; \textit{there exists } C \in C^k_{ij} \textit{ such that } \sum_{v \in C} \tilde{y}_v < 1 \right\}.$$

For every power graph edge $e = \{i,j\} \in E^k(\tilde{y})$, let C_e be a minimal length-k i,j-connector C with minimum $\sum_{v \in C} \tilde{y}_v$. Define $\mathbf{C} := \{C_e \mid e \in E^k(\tilde{y})\}$. The points $\tilde{x}$ and $\tilde{\pi}$ are optimal solutions for the primal and dual subproblems, respectively, where

$$\tilde{x}_e \;=\; \begin{cases} 1 - \sum_{v \in C_e} \tilde{y}_v & \textit{if } e \in E^k(\tilde{y}) \\ 0 & \textit{if } e \notin E^k(\tilde{y}) \end{cases} \qquad \forall e \in E^k$$

$$\tilde{\pi}_{C,e} = \begin{cases} c_e & \textit{if } C \in \mathbf{C} \\ 0 & \textit{if } C \notin \mathbf{C} \end{cases} \qquad \forall C \in C^k_{ij},\ \forall e = \{i,j\} \in E^k.$$

Proof. Observe that $\tilde{x}$ and $\tilde{\pi}$ are feasible for the primal and dual subproblems, respectively. Then, the objective values of $\tilde{x}$ and $\tilde{\pi}$, denoted $\mathrm{obj}(\tilde{x})$ and $\mathrm{obj}(\tilde{\pi})$ satisfy

$$\mathrm{obj}(\tilde{x}) = \sum_{e \in E^k} c_e \tilde{x}_e = \sum_{e \in E^k(\tilde{y})} c_e \left(1 - \sum_{v \in C_e} \tilde{y}_v\right) = \sum_{e=\{i,j\} \in E^k(\tilde{y})} \sum_{C_e \in C^k_{ij}} \tilde{\pi}_{C,e} \left(1 - \sum_{v \in C_e} \tilde{y}_v\right) = \mathrm{obj}(\tilde{\pi}).$$

Thus, by weak duality, $\tilde{x}$ and $\tilde{\pi}$ are optimal for the primal and dual subproblems, respectively. □

Given a $\tilde{y} \in Y$ and by finding optimal dual multipliers $\tilde{\pi}$, the Benders optimality cuts can be generated (see for example (Rei et al., 2009)) as follows.

$$z \geq \sum_{e=\{i,j\} \in E^k} \sum_{C \in C^k_{ij}} \tilde{\pi}_{C,e} \left(1 - \sum_{v \in C} y_v\right) = \sum_{e \in E^k(\tilde{y})} c_e \left(1 - \sum_{v \in C_e} y_v\right).$$

Finally, the Benders master problem is as follows.

$$\min z \tag{4.2a}$$

$$z \geq \sum_{e \in E^k(\tilde{y})} c_e \left(1 - \sum_{v \in C_e} y_v\right) \qquad \forall \tilde{y} \in Y \tag{4.2b}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{4.2c}$$

$$y_i \in \{0,1\} \qquad \forall i \in V, \tag{4.2d}$$

where Y is defined as before. This formulation has $n+1$ variables and might have very large number of constraints (4.2b).

Before detailing the implementations of the master problem, we show how to generate valid inequalities that dominate constraints of the type (4.1b). To do this, we first need to define well-connected power graph edges as follows. An example to illustrate this definition is given in Figure 4.1.

Definition 10 (Well-connected power graph edge)**.** *The set of well-connected power graph edges E_w^k of a graph $G=(V,E)$ is*

$$E_w^k := \left\{ \{i,j\} \in E^k \;\middle|\; \text{dist}_{G-S}(i,j) \leq k \quad \forall S \subseteq V \setminus \{i,j\} \text{ with } \sum_{v \in S} a_v \leq b \right\}.$$

Proposition 9. *Let $e = \{i,j\} \in E_w^k$ be a well-connected power graph edge. Then,*

$$x_e + y_i + y_j \geq 1$$

is valid for the DCNP feasible region.

Proof. We show that the inequality is satisfied by every feasible point of the DCNP. Suppose not. This implies that there is a feasible point $(\hat{x}, \hat{y})$ where $\hat{x}_e + \hat{y}_i + \hat{y}_j = 0$. Since power graph edge $e = \{i, j\}$ is well-connected, there is at least one intact minimal length-k i, j-connector $C \in \mathcal{C}_{ij}^k$ (i.e., $\sum_{v \in C} \hat{y}_v = 0$) after deleting any feasible subset of vertices $S \subseteq V \setminus \{i, j\}$. So

$$\hat{x}_e + \sum_{v \in C} \hat{y}_v = 0,$$

which contradicts $\hat{x}_e + \sum_{v \in C} \hat{y}_v \geq 1$ of the type (4.1b). □

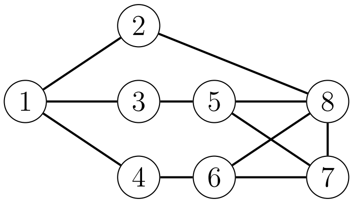


Figure 4.1: Suppose $k = 3$, $b = 2$, and $a_v = 1$ for all $v \in V$. Power graph edge $\{1, 8\}$ is well-connected because $\text{dist}_{G-S}(1, 8) \leq 3$ for any $S \subseteq V \setminus \{1, 8\}$ of size b. However, power graph edge $\{2, 3\}$ is not well-connected because $\text{dist}_{G-\{1,8\}}(2, 3) > 3$. Lastly, power graph edge $\{1, 5\}$ is well-connected for CNP, but not for DCNP since $\text{dist}_{G-\{2,3\}}(1, 5) > 3$.

Remark 7. *Let $\tilde{y} \in Y$ and define $E_w^k(\tilde{y}) := E_w^k \cap E^k(\tilde{y})$. Then, the Benders optimality cuts can be written as follows.*

$$z \geq \sum_{e=\{i,j\} \in E_w^k(\tilde{y})} c_e \left(1 - y_i - y_j\right) + \sum_{e \in E^k(\tilde{y}) \setminus E_w^k(\tilde{y})} c_e \left(1 - \sum_{v \in C_e} y_v\right).$$

4.3 Implementation

Since the master problem might have very large number of constraints (4.2b), we add them on-the-fly, as lazy constraints. Basically, we initialize the formulation with the budget constraint (4.2c) and a subset of the constraints (4.2b). Then, at a branch-and-bound node, if the MIP solver finds a possible solution $\tilde{y}$ that satisfies the initial constraints, we separate the inequalities of the type (4.2b). In what follows, we detail two possible procedures to generate high-quality initial optimality constraints. We also discuss how to fix some of the y variables to reduce the size of instances.

4.3.1 Restricted Problem

In order to obtain a pool of initial optimality cuts, we can first solve a restricted problem (RP) and then add all the optimality cuts generated while solving RP to the full problem as an initialization. Let $B \subseteq V$ of size $2b$ be a subset of vertices with the largest betweenness centrality. One possible way of restricting the problem is to fix $y_v = 0$ for all vertices $v \in V \setminus B$. Another way of generating a set of initial constraints is to restrict the full problem by adding a classical trust region or local branching constraint (Fischetti and Lodi, 2003; Santoso et al., 2005; Baena et al., 2020) as follows.

$$\sum_{i:\bar{y}_i=1} (1 - y_i) + \sum_{i:\bar{y}_i=0} y_i \leq r.$$

Here, $\bar{y}$ is a stability point and r is the radius of the trust region. Adding this constraint leads to focusing on a part of the feasible region that is a neighborhood of $\bar{y}$. Let $D \subseteq V$ be a deletion set and $\bar{y}$ be its characteristic vector. We can solve two restricted problems by adding disjunctive constraints $\sum_{i:\bar{y}_i=1} (1 - y_i) + \sum_{i:\bar{y}_i=0} y_i \leq r$ to one and $\sum_{i:\bar{y}_i=1} (1 - y_i) + \sum_{i:\bar{y}_i=0} y_i \geq r + 1$ to the other. Then, the optimal solution will be the optimal deletion set of that restricted problem with smaller objective function value.

4.3.2 Variable Fixing

Here, we discuss variable fixing procedures for the critical node problem. See Figure 4.2 for an illustration. In our discussion, we use a definition of a k-vertex connected graph as follows.

Definition 11 (k-vertex connected graph)**.** *A graph $G = (V, E)$ is k-vertex connected (or k-connected) if $|V| \geq k + 1$ and it remains connected after deleting any fewer than k vertices.*

Consider a graph $G = (V, E)$ and let H be a $(b + 1)$-connected component. Define V^{int} to be the vertices v for which $N(v) \subseteq V(H)$, $V^{\text{ext}} := V(H) \setminus V^{\text{int}}$, and $q := \max\{0, b - |V^{\text{ext}}|\}$.

Let D_0 be a deletion set with vertices from V^{int}. If there exists another $(b+1)$-connected component H' of size at least $\left|V^{\text{int}}\right| + q$, then one can safely swap all vertices in $D_0 \cap V^{\text{int}}$ with intact vertices of $V^{\text{ext}} \cup V(H')$ to obtain a feasible deletion set D_1. The objective value of D_1 is at least as good as that of D_0 because deleting vertices from V^{ext} disconnects component H from the rest of the graph, as opposed to deleting vertices from V^{int}. In addition, after deleting vertices from V^{ext}, the number of intact vertices in H' is still more than or equal to that of vertices in H. So, deleting vertices from H' causes more disconnectivity compared to deleting vertices from H. By this argument, gray vertices $\{1, 2, 4, 5\}$ are not critical in Figure 4.2. This implies that we can fix $y_v = 0$ for all $v \in \{1, 2, 4, 5\}$.

Another procedure is to pick q vertices from V^{int} and let them be S. Define $I \coloneqq V^{\text{int}} \setminus S$. Now, if there is a deletion set D_0 with $D_0 \cap I \neq \emptyset$, one can swap vertices of $D_0 \cap I$ with intact vertices of $S \cup V^{\text{ext}}$ to obtain a feasible deletion set D_1. The objective value of D_1 is at least as good as that of D_0: Swapping vertices of I with vertices of S does not change the objective value. Meanwhile, swapping vertices of I with vertices of V^{ext} might improve the objective value since this swapping might disconnect the $(b+1)$-connected component from the rest of the graph. By this argument, gray nodes $\{9, 10, 11\}$ are not critical in Figure 4.2. This implies that we can fix $y_v = 0$ for all $v \in \{9, 10, 11\}$.

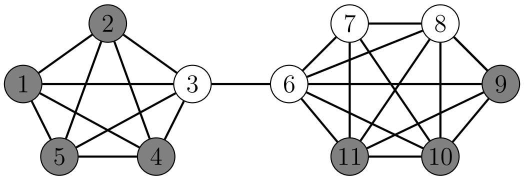


Figure 4.2: An illustration of variable fixing procedures for the critical node problem.

In the future, we intend to generalize these variable fixing procedures and incorporate them in our implementations.

4.4 Computational Experiments

In this section we provide preliminary computational results of solving hop-based instances that were given on Section 3.5.1 when $(k, b) = (3, 5)$. We use the following implementations of the master problem.

- BEN_{I}, an implementation of the master problem using integer separation.
- BEN_{F}, an implementation of the master problem using fractional separation.
- BEN_{R}, an implementation of the master problem that solves a restricted problem initially. In the restricted problem $y_v = 0$ if v is not among the $2b$ vertices with the largest betweenness centrality.
- BEN_{TR}, an implementation of the master problem that solves two restricted problems obtained by adding disjunctive trust region constraints where the heuristic solution is considered as the stability point and $r = 2$.

Each implementation uses the heuristic and variable fixing procedure given in Section 3.4, and all experiments were 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 code was written in C++ programming language for Gurobi version 9.0.0.

The preliminary results, as reported in Table 4.1, show that implementation BEN_{R} typically performs the best compared to the others. Implementations BEN_{I} and BEN_{R} fail to solve the 115-node instance `football`. Meanwhile, it takes more than 38 minutes for implementations BEN_{R} and BEN_{TR} to solve this instance. To reduce the computational time for this instance, we used a variable fixing method: we iteratively fixed $y_v = 1$ for a $v \in V$ and then solve the LP-relaxation of the thin formulation. If LP optimal was more than a known upper bound (for example the heuristic solution), then we could safely fix $y_v = 0$.

By implementing this variable-fixing procedure, we could fix 107 nodes and solve `football` in 30.83 seconds. In the future, we intend to study how we can find lower bounds on the optimal objective value without solving the LP relaxation directly. Another observation from Table 4.1 is that the direct implementation of the thin formulation (THIN), as discussed in Chapter III, typically outperforms the current implementations of the Benders master formulation. However, Benders implementations solve 5 instances faster. Specifically, they perform much better on the instances `jazz` and `celegansm`. We suspect that variable fixing procedures, stabilization techniques, and other Benders acceleration methods will help our Benders implementations to converge to the optimal solution quicker.

Graph	n	m	$\|E^k\|$	obj	BEN_I	BEN_F	BEN_R	BEN_TR	THIN
karate	34	78	480	41	0.08	0.04	0.06	0.07	0.04
dolphins	62	159	1,107	662	20.57	21.90	6.61	25.99	0.74
lesmis	77	254	2,500	517	0.19	0.30	0.30	0.44	0.21
LindenStrasse	232	303	3,251	1,810	1.22	0.85	1.01	1.27	0.21
polbooks	105	441	3,510	2,555	16.52	56.86	11.48	102.75	2.56
adjnoun	112	425	5,634	3,719	1.40	3.86	2.32	3.87	0.97
football	115	613	6,247	5,362	[5085,5362]	2307.86	[5312,5362]	2345.10	86.48
netscience	1,589	2,742	13,087	8,390	2.40	146.74	8.21	94.96	1.19
jazz	198	2,742	18,461	16,136	113.83	316.52	155.18	335.74	1081.24
SmallWorld	233	994	25,721	6,964	0.47	0.53	0.71	1.47	5.40
Erdos971	429	1,312	34,086	25,737	54.65	413.56	28.58	923.63	9.93
S.Cerevisae	1,458	1,948	39,091	25,190	8.30	49.60	10.41	15.62	5.68
USAir	332	2,126	46,573	29,486	9.40	71.92	12.80	99.77	26.09
power	4,941	6,594	53,125	50,410	2674.26	[49889,50458]	460.81	248.10	46.13
H.Pylori	706	1,392	62,028	37,626	5.25	31.28	6.04	8.88	8.63
Harvard500	500	2,043	83,993	16,448	1.22	2.25	2.43	4.34	14.54
homer	542	1,619	91,527	45,828	14.15	86.78	22.14	107.62	32.51
celegansm	453	2,025	91,531	44,967	3.17	10.03	8.76	14.09	43.40
email	1,133	5,451	289,259	263,409	[259426,263409]	[250171,263409]	2431.18	[251133,263409]	114.46
hep-th	8,361	15,751	376,431	345,320	[338982,345320]	[331215,345320]	1635.06	[342228,345320]	171.94
PGPgiant	10,680	24,316	1,145,492	857,035	710.49	[845870,857035]	892.58	3105.35	704.55
cond-mat	16,726	47,594	1,761,969	1,633,299	[1601472,1637445]	[1593891,1637445]	6646.09	[1598525,1637445]	2385.66

Table 4.1: Running times, or bounds at termination, when $(k, b) = (3, 5)$.

4.5 Conclusion

In this chapter, we propose a Benders decomposition algorithm and branch-and-cut implementations of it to solve the (distance-based) critical node problem. We show how to identify Benders optimality cuts in a combinatorial way rather than solving the dual subproblem as an LP. We also provide procedures to accelerate the convergence of the Benders decomposition.

Chapter V

Conclusion and Future Work

In this dissertation we introduce new integer programming formulations, techniques, and algorithms to solve NP-hard network optimization problems with distance constraints. We propose simpler formulations for these problems and provide algorithms to solve larger instances to optimality that could not be solved previously. Moreover, our techniques and algorithms typically improve the running times. Below, we review our contributions and provide possible future works.

In Chapter II, we introduce new integer programming formulations, which we refer to as path-like and cut-like formulations, to find low-diameter clusters. Our cut-like formulation has only n variables and imposes diameter-at-most-k constraints by using length-k a,b-separator inequalities. These constraints appear in many other distance-based network problems where compactness is key to decision-makers. We show that length-k a,b-separator inequalities perform well on different instances of k-club problem, which is a positive sign for using them for other distance-based combinatorial problems. Also, we show that these inequalities are valid even when distances are not hop-based. In an ongoing work with Yajun Lu, Balabhaskar Balasundaram, and Austin Buchanan, we study fault-tolerant variants of k-clubs, like the r-robust k-clubs of Veremyev and Boginski (2012) and the h-hereditary k-clubs defined by Pattillo et al. (2013). Indeed, the latter can be formulated by modifying the length-k a,b-separator inequalities to $hx_a + hx_b \leq h + x(S)$. Based on experience here and in the past (Buchanan et al., 2015; Validi and Buchanan, 2019), we expect this to be a practical approach for small values of h.

In Chapter III, we propose new integer programming formulations, which we call thin and path-like, to solve a class of critical node problems. In this class of problems, the task is to delete at most b nodes of a network so as to minimize the number of nodes that are remained connected via a path of length at most k. Our thin formulation uses only $|E^k| + n$ variables and enables us to solve large instances that could not be solved to optimality before. We compare our formulations with an existing recursive formulation. We show that all three formulations are equal in strength when the objective coefficients are nonnegative, but the thin formulation is the strongest generally. In addition, while our proposed formulations directly work under edge-weighted distances, the recursive model needs to be extended to a formulation with a pseudopolynomial number of variables and constraints. For the thin formulation, we devise efficient integer and fractional separation routines that we employ in a branch-and-cut algorithm. We also propose a preprocessing technique that allow us to identify nodes that are not critical and thus fix them initially in the IP formulations. A potential topic for future work is to study the applications of the distance-based critical node problem in mitigating the spread of infectious diseases; see (Charkhgard et al., 2018) for an example. Other possible topics are to study other variants of the critical node problem. For example, the variants that ask to delete at most b nodes of a network to minimize the size of the largest connected component, or to maximize the distance between two specified nodes.

In Chapter IV, we propose a Benders decomposition algorithm to solve the thin formulation that models the (distance-based) critical node problem. We show how to find Benders optimality cuts in a combinatorial manner rather than solving the dual subproblem as an LP. We also provide methods to accelerate the convergence of the Benders decomposition. Can we introduce other acceleration techniques? In the future, we intend to study variable fixing methods, stabilizing techniques, and/or preprocessing procedures to address this question.

Bibliography

J. M. Abello, P. M. Pardalos, and M. G. Resende. On maximum clique problems in very large graphs. In *External memory algorithms*, volume 50 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, 1999. ISBN 0-8218-1184-3.

T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 2019.

B. Addis, M. Di Summa, and A. Grosso. Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. *Discrete Applied Mathematics*, 161(16-17):2349–2360, 2013.

B. Addis, R. Aringhieri, A. Grosso, and P. Hosteins. Hybrid constructive heuristics for the critical node problem. *Annals of Operations Research*, 238(1-2):637–649, 2016.

R. D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3(1):113–126, 1973.

M. T. Almeida and F. D. Carvalho. Integer models and upper bounds for the 3-club problem. *Networks*, 60(3):155–166, 2012.

M. T. Almeida and F. D. Carvalho. An analytical comparison of the LP relaxations of integer models for the k-club problem. *European Journal of Operational Research*, 232(3):489–498, 2014.

R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia. Polynomial and pseudo-polynomial time algorithms for different classes of the distance critical node problem. *Discrete Applied Mathematics*, 253:103–121, 2019.

A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

Y. Asahiro, Y. Doi, E. Miyano, K. Samizo, and H. Shimizu. Optimal approximation algorithms for maximum distance-bounded subgraph problems. *Algorithmica*, 80(6):1834–1856, 2018.

D. Baena, J. Castro, and A. Frangioni. Stabilized benders methods for large-scale combinatorial optimization, with application to data privacy. *Management Science*, 2020.

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.

E. Balas and M. Fischetti. On the monotonization of polyhedra. *Mathematical Programming*, 78(1):59–84, 1996.

B. Balasundaram. *Graph theoretic generalizations of clique: Optimization and extensions.* PhD thesis, Texas A&M University, 2007.

B. Balasundaram, S. Butenko, and S. Trukhanov. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10(1):23–39, 2005.

V. Batagelj and A. Mrvar. Pajek datasets. `http://vlado.fmf.uni-lj.si/pub/networks/data/`, 2006. Accessed: 2018.

V. Batagelj and M. Zaversnik. An $O(m)$ algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

A. Bavelas. A mathematical model for group structures. *Applied Anthropology*, 7(3):16–30, 1948.

J. Behnamian. Decomposition based hybrid vns–ts algorithm for distributed parallel factories scheduling with virtual corporation. *Computers & Operations Research*, 52:181–191, 2014.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

V. Boginski and C. W. Commander. Identifying critical nodes in protein-protein interaction networks. In *Clustering challenges in biological networks*, pages 153–167. World Scientific, 2009.

S. P. Borgatti. Identifying sets of key players in a social network. *Computational & Mathematical Organization Theory*, 12(1):21–34, 2006.

Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal On Computing*, 25(1):13–26, 2013.

J.-M. Bourjolly, G. Laporte, and G. Pesant. Heuristics for finding k-clubs in an undirected graph. *Computers & Operations Research*, 27(6):559–569, 2000.

J.-M. Bourjolly, G. Laporte, and G. Pesant. An exact algorithm for the maximum k-club problem in an undirected graph. *European Journal of Operational Research*, 138(1):21–28, 2002.

U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

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.

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.

H. Charkhgard, V. Subramanian, W. Silva, and T. K. Das. An integer linear programming formulation for removing nodes in a network to minimize the spread of influenza virus infections. *Discrete Optimization*, 30:144–167, 2018.

R. Cohen, S. Havlin, and D. Ben-Avraham. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 91(24):247901, 2003.

C. W. Commander, P. M. Pardalos, V. Ryabchenko, S. Uryasev, and G. Zrazhevsky. The wireless network jamming problem. *Journal of Combinatorial Optimization*, 14(4):481–498, 2007.

M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204(1):97–143, 2013.

J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001.

A. I. Corréa, A. Langevin, and L.-M. Rousseau. Scheduling and routing of automated guided vehicles: A hybrid approach. *Computers & Operations Research*, 34(6):1688–1707, 2007.

T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

R. S. De Camargo, G. de Miranda Jr, and R. P. Ferreira. A hybrid outer-approximation/benders decomposition algorithm for the single allocation hub location problem under congestion. *Operations Research Letters*, 39(5):329–337, 2011.

M. Di Summa, A. Grosso, and M. Locatelli. Complexity of the critical node problem over trees. *Computers & Operations Research*, 38(12):1766–1774, 2011.

M. Di Summa, A. Grosso, and M. Locatelli. Branch and cut algorithms for detecting critical nodes in undirected graphs. *Computational Optimization and Applications*, 53(3):649–680, 2012.

DIMACS-10. 10th DIMACS Implementation Challenge - Graph Partitioning and Graph Clustering. `http://www.cc.gatech.edu/dimacs10/downloads.shtml`, 2017. Accessed: 2017-08-08.

S. Emami, G. Moslehi, and M. Sabbagh. A benders decomposition approach for order acceptance and scheduling problem: a robust optimization approach. *Computational and Applied Mathematics*, 36(4):1471–1515, 2017.

M. Fischetti and A. Lodi. Local branching. *Mathematical programming*, 98(1-3):23–47, 2003.

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.

P. Fontaine and S. Minner. Benders decomposition for discrete–continuous linear bilevel problems with application to traffic network design. *Transportation Research Part B: Methodological*, 70:163–172, 2014.

L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3): 215–239, 1978.

M. R. Garey and D. S. Johnson. *Computers and intractability.* W.H. Freeman and Company, 1979.

M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics.* Springer-Verlag, Berlin, second edition, 1993.

F. Hooshmand, F. Mirarabrazi, and S. MirHassani. Efficient Benders decomposition for distance-based critical node detection problem. *Omega*, 2019.

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.

M. Jenabi, S. F. Ghomi, S. A. Torabi, and S. H. Hosseinian. Acceleration strategies of benders decomposition for the security constraints power system expansion planning. *Annals of Operations Research*, 235(1):337–369, 2015.

M. Kutz. *Handbook of transportation engineering*, volume 768. McGraw-Hill New York, NY, USA:, 2004.

M. Lalou, M. A. Tahraoui, and H. Kheddouci. The critical node detection problem in networks: a survey. *Computer Science Review*, 28:92–117, 2018.

L. Lovász, V. Neumann-Lara, and M. Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9(4):269–276, 1978.

Y. Lu, E. Moradi, and B. Balasundaram. Correction to: Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters*, 12(8):1959–1969, 2018.

F. Mahdavi Pajouh and B. Balasundaram. On inclusionwise maximal and maximum cardinality k-clubs in graphs. *Discrete Optimization*, 9(2):84–97, 2012.

F. Mahdavi Pajouh, B. Balasundaram, and I. V. Hicks. On the 2-club polytope of graphs. *Operations Research*, 64(6):1466–1481, 2016.

A. R. Mahjoub and S. T. McCormick. Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. *Mathematical Programming*, 124(1-2):271–284, 2010.

R. K. Martin. Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10(3):119–128, 1991.

T. C. Matisziw and A. T. Murray. Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *Computers & Operations Research*, 36 (1):16–26, 2009.

D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983.

R. J. Mokken. Cliques, clubs and clans. *Quality & Quantity*, 13(2):161–173, 1979.

E. Moradi and B. Balasundaram. Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters*, 12(8):1947–1957, 2018.

H. Nagamochi. Minimum degree orderings. *Algorithmica*, 56(1):17, 2010.

J. Naoum-Sawaya and C. Buchheim. Robust critical node selection by benders decomposition. *INFORMS Journal on Computing*, 28(1):162–174, 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.

J. Pattillo, N. Youssef, and S. Butenko. On clique relaxation models in network analysis. *European Journal of Operational Research*, 226(1):9–18, 2013.

R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating benders decomposition by local branching. *INFORMS Journal on Computing*, 21(2):333–345, 2009.

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. Implementation of parsimonious formulations for low-diameter clusters. `https://github.com/halisalemi/ParsimoniousKClub`, 2019.

H. Salemi and A. Buchanan. Parsimonious formulations for low-diameter clusters. *Mathematical Programming Computation*, pages 1–36, 2020a.

H. Salemi and A. Buchanan. Implementation of solving the distance-based critical node problem. `https://github.com/halisalemi/DCNP`, 2020b.

H. Salemi and A. Buchanan. Solving the distance-based critical node problem. 2020c. Preprint available at `http://www.optimization-online.org/DB_HTML/2020/04/7751.html`.

T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115, 2005.

A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer Science & Business Media, 2003.

S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

S. Shahinpour and S. Butenko. Algorithms for the maximum k-club problem in graphs. *Journal of Combinatorial Optimization*, 26(3):520–554, 2013a.

S. Shahinpour and S. Butenko. Distance-based clique relaxations in networks: s-clique and s-club. In *Models, algorithms, and technologies for network analysis*, pages 149–174. Springer, 2013b.

S. Shen and J. C. Smith. Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks*, 60(2):103–119, 2012.

S. Shen, J. C. Smith, and R. Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization*, 9(3):172–188, 2012.

Y. Shen, N. P. Nguyen, Y. Xuan, and M. T. Thai. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Transactions on Networking*, 21(3): 963–973, 2013.

B. Stabler, H. Bar-Gera, and E. Sall. Transportation networks for research. `https://github.com/bstabler/TransportationNetworks`, 2019.

STOM-Group. Hazmat network data. `https://github.com/STOM-Group/Hazmat-Network-Data`, 2019.

Z. Tao, F. Zhongqian, and W. Binghong. Epidemic dynamics on complex networks. *Progress in Natural Science*, 16(5):452–457, 2006.

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.

H. Validi and A. Buchanan. The optimal design of low-latency virtual backbones. *INFORMS Journal on Computing*, 2019. To appear.

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.

A. Veremyev, V. Boginski, and E. L. Pasiliao. Exact identification of critical nodes in sparse networks via new compact formulations. *Optimization Letters*, 8(4):1245–1259, 2014.

A. Veremyev, O. A. Prokopyev, and E. L. Pasiliao. Critical nodes for distance-based connectivity and related problems in graphs. *Networks*, 66(3):170–195, 2015.

A. Verma, A. Buchanan, and S. Butenko. Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing*, 27(1):164–177, 2015.

B. Vitoriano, M. T. Ortuño, G. Tirado, and J. Montero. A multi-criteria optimization model for humanitarian aid distribution. *Journal of Global optimization*, 51(2):189–208, 2011.

J. L. Walteros and A. Buchanan. Why is maximum clique often easy in practice? *Operations Research*, 2019. To appear.

J. L. Walteros and P. M. Pardalos. Selected topics in critical element detection. In *Applications of Mathematics and Informatics in Military Science*, pages 9–26. Springer, 2012.

J. L. Walteros, A. Veremyev, P. M. Pardalos, and E. L. Pasiliao. Detecting critical node structures on graphs: A mathematical programming approach. *Networks*, 73(1):48–88, 2019.

Y. Wang, A. Buchanan, and S. Butenko. On imposing connectivity constraints in integer programs. *Mathematical Programming*, 166(1-2):241–271, 2017.

A. Wotzlaw. On solving the maximum k-club problem. *arXiv preprint arXiv:1403.5111*, 2014.

J. Xu. *Topological structure and analysis of interconnection networks.* Kluwer, 2001.

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

VITA

Hosseinali Salemi

Candidate for the Degree of

Doctor of Philosophy

Dissertation: INTEGER PROGRAMMING FORMULATIONS FOR DISTANCE-CONSTRAINED NETWORK PROBLEMS

Major Field: Industrial Engineering and Management

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Industrial Engineering & Management at Oklahoma State University, Stillwater, Oklahoma in June, 2020

Received a Master of Science in Management Engineering from Politecnico di Milano (Polytechnic University of Milan), Milan, Italy in 2015

Received a Bachelor of Science in Industrial and Systems Engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran in 2013

Experience:

Worked on project “Optimization-Based Aggregate Master Planning Tools for Bay Valley Foods, LLC,” *Bay Valley Foods, LLC,* from Fall 2017 to Summer 2020

Worked on project “FLAT: Freight Lane Assignment Tool,” *TreeHouse Foods Inc,* from Spring 2020 to Summer 2020

Instructor of “Probability and Statistics for Engineers II” at Oklahoma State University in Spring 2019.

President of INFORMS Student Chapter at Oklahoma State University from Fall 2019 to Spring 2020