Solving the distance-based critical node problem

Hosseinali Salemi and Austin Buchanan

School of Industrial Engineering & Management
Oklahoma State University
{hosseinali.salemi,buchanan}@okstate.edu

April 13, 2020

Abstract

In critical node problems, the task is identify a small subset of so-called critical nodes whose deletion maximally degrades a network's "connectivity" (however that is measured). Problems of this type have been widely studied, e.g., for limiting the spread of infectious diseases. However, existing approaches for solving them have typically been limited to networks having fewer than 1,000 nodes. In this paper, we consider a variant of this problem in which the task is to delete b nodes so as to minimize the number of node pairs that remain connected by a path of length at most k. With the techniques developed in this paper, instances with up to 17,000 nodes can be solved exactly. We introduce two integer programming formulations for this problem (thin and path-like) and compare them with an existing recursive formulation. While the thin formulation generally has an exponential number of constraints, it admits an efficient separation routine. Also helpful is a new, more general preprocessing procedure that, on average, fixes three times as many variables than before.

Keywords: critical node; distance constraint; integer program; branch-and-cut; network interdiction; dominant; partial dominant;

1 Introduction

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 paper. For generality, we consider a knapsack constraint for the deletion budget (instead of a cardinality constraint) and edge-weighted distances (instead of hops).

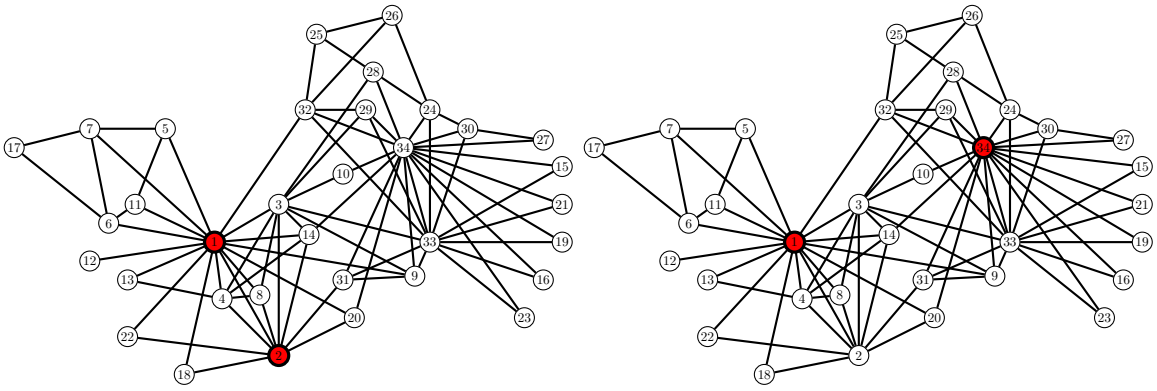
Problem: Distance-Based Critical Node Problem (DCNP).
Input: Simple graph $G = (V, E)$, edge weights $w_e \geq 0$ for $e \in E$, deletion budget b, deletion cost a_i for each vertex $i \in V$, connection cost c_e for each pair of nodes $e \in \binom{V}{2}$, distance threshold k.
Output: A subset of vertices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

Here, the objective function $\mathrm{obj}(D)$ 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,

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

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 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 1: Solutions for the `Karate` graph 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.

In this paper, 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.

1.1 Our Contributions

Section 2 gives notation and a brief overview of the literature, including the formulation of Veremyev et al. (2015) which we call the *recursive* formulation. Section 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 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 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 6.

2 Background and Related Work

This section gives the notation and terminology used throughout the paper, as well as a brief overview of the CNP and DCNP literature.

2.1 Notation and Terminology

Consider a simple graph $G = (V, E)$ with vertex set V and edge set $E \subseteq \binom{V}{2}$. We typically let $n \coloneqq |V|$ and $m \coloneqq |E|$. Denote by $N_G(v) \coloneqq \{u \in V \mid \{u, v\} \in E\}$ the neighbors of v in G; its cardinality is the degree, denoted $\deg_G(v) \coloneqq |N_G(v)|$. The subset of edges incident to v is denoted

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

We assume a nonnegative weight w_e is given for each edge $e \in E$. The length of a path is the sum of its edges' weights. The distance from vertex u to vertex v in graph G, denoted $\mathrm{dist}_G(u, v)$, is the length of a shortest path from u to v. When distances are *hop-based*, each edge weight w_e is

one, and the length of a path is equal to its number of edges. When $S \subseteq V$ is a subset of vertices, $E(S) \coloneqq E \cap \binom{S}{2}$ denotes the subset of edges with both endpoints in S. The subgraph of G induced by S is denoted by $G[S] \coloneqq (S, E(S))$. When $D \subseteq V$ is a subset of vertices, $G - D \coloneqq G[V \setminus D]$ denotes the subgraph of G obtained by removing D and any incident edges. If H is a graph, then $E(H)$ denotes its edge set. The k-th power $G^k = (V, E^k)$ of graph $G = (V, E)$ has the same vertex set as G and edge set

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

This definition applies 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.

2.2 The Critical Node Problem (CNP)

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).

2.3 The Distance-Based Critical Node Problem (DCNP)

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. (2020) 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 paper, 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.

$$\min \sum_{e \in E^k} c_e x_e \tag{1a}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{1b}$$

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

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

$$u^s_{ij} = u^1_{ij} \qquad \forall \{i,j\} \in E, \quad \forall s \in \{2,3,\ldots,k\} \tag{1e}$$

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

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

$$u^s_{ij} = u^s_{ji} \qquad \forall i,j \in V, i \neq j, \quad \forall s \in \{1,2,\ldots,k\} \tag{1h}$$

$$u^k_{ij} = x_e \qquad \forall e = \{i,j\} \in E^k \tag{1i}$$

$$u^s_{ij} \in \{0,1\} \qquad \forall i,j \in V, i \neq j, \quad \forall s \in \{1,2,\ldots,k\} \tag{1j}$$

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

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

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 this formulation, including the interpretation of its constraints and how to extend it 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 $u^s_{ij} = u^s_{ji}$ it suffices to define just one of these variables, e.g., those with $i < j$.
2. The variables u^s_{ij} 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 $y_i = 0$, provided that 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 $O(k|E^k|)$. Next we will see that binary restrictions $x_e \in \{0, 1\}$ can be omitted and the constraints $u^k_{ij} = x_e$ can be relaxed to $u^k_{ij} \leq x_e$ when the connection costs c_e are nonnegative.

2.4 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 paper, we refer to the *dominant* of P, typically denoted by $P^\uparrow$ (Balas and Fischetti, 1996; Schrijver, 2003; Conforti et al., 2013).

Definition 1. *The dominant $P^\uparrow$ of a polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is the polyhedron*

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

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, 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 previous literature, to our knowledge.

Definition 2. *The partial dominant $P^{\uparrow_x}$ of a polyhedron $P = \left\{(x, y) \in \mathbb{R}^n \times \mathbb{R}^d \;\middle|\; Ax + Gy \leq b\right\}$ with respect to x is the polyhedron*

$$P^{\uparrow_x} \coloneqq \left\{(\hat{x}, y) \in \mathbb{R}^n \times \mathbb{R}^d \;\middle|\; \exists (x, y) \in P : \hat{x} \geq x\right\} = P + (\mathbb{R}^n_+, 0^d),$$

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 $\min\left\{c^T x \;\middle|\; (x, y) \in P\right\} = \min\left\{c^T x \;\middle|\; (x, y) \in P^{\uparrow_x}\right\}$.

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 1. *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 $\tilde{x}_e \coloneqq \hat{u}^k_{ij}$. 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 New Formulations

This section proposes 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.

3.1 The Path-Like Formulation

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

Definition 3 (length-k i, j-connector)**.** *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* $\mathrm{dist}_{G[C]}(i, j) \leq k$*. If no proper subset of C is a length-k i, j-connector, then C is said to be* minimal.

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

Lemma 2. *If edge weights 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 $C \in C^k_{ij}$ 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 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.

$$\min \sum_{e \in E^k} c_e x_e \tag{2a}$$

$$z_C + \sum_{v \in C} y_v \geq 1 \qquad \forall C \in \mathcal{C} \tag{2b}$$

$$z_C + y_v \leq 1 \qquad \forall v \in C, \quad \forall C \in \mathcal{C} \tag{2c}$$

$$x_e \geq z_C \qquad \forall C \in C^k_{ij}, \quad \forall e = \{i, j\} \in E^k \tag{2d}$$

$$x_e \leq \sum_{C \in C^k_{ij}} z_C \qquad \forall e = \{i, j\} \in E^k \tag{2e}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{2f}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E^k \tag{2g}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{2h}$$

$$z_C \in \{0, 1\} \qquad \forall C \in \mathcal{C}. \tag{2i}$$

Constraints (2b) and (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 (2d) and (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 ($\sum_{C \in C^k_{ij}} z_C \geq 1$). Later we will see that constraints (2c) and (2e) can be omitted when the connection costs c_e are nonnegative.

Theorem 1. *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

$$\exists z^* \in \{0, 1\}^{|\mathcal{C}|} \text{ s.t. } (x^*, y^D, z^*) \text{ satisfies (2b)-(2e)} \iff \forall e = \{i, j\} \in E^k, x^*_e = \begin{cases} 1 & \text{if } \text{dist}_{G-D}(i, j) \leq k \\ 0 & \text{if } \text{dist}_{G-D}(i, j) > k \end{cases}.$$

($\implies$) Suppose that there exists a binary vector $z^* \in \{0, 1\}^{|\mathcal{C}|}$ such that (x^*, y^D, z^*) satisfies constraints (2b)-(2e). In the first case, suppose the endpoints of power graph edge $e = \{i, j\} \in E^k$ satisfy $\text{dist}_{G-D}(i, j) \leq k$. 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 (2d) and the second inequality holds by constraints (2b). In the other case, $\text{dist}_{G-D}(i, j) > k$. In this case, D hits every minimal length-k i, j-connector C,

i.e., for every $C \in C_{ij}^k$ there exists a vertex from C that also belongs to D. Then $z_C^* = 0$ by $z_C^* + 1 = z_C^* + y_v^D \leq 1$, where the inequality holds by constraints (2c). Constraints (2e) then show that $x_e^* \leq \sum_{C \in C_{ij}^k} z_C^* \leq 0$, as desired.

($\Longleftarrow$) Suppose that for each edge $e = \{i, j\} \in E^k$ of the power graph, $x_e^* = 1$ if and only if $\text{dist}_{G-D}(i, j) \leq k$. We construct a binary vector $z^* \in \{0, 1\}^{|\mathcal{C}|}$ such that (x^*, y^D, z^*) satisfies constraints (2b)-(2e). For each connector $C \in \mathcal{C}$, define

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

By this definition, (x^*, y^D, z^*) satisfies constraints (2b) and (2c).

Now, we show that each constraint (2d) is satisfied. Consider an edge $e = \{i, j\} \in E^k$ of the power graph and a minimal length-k i, j-connector $C \in C_{ij}^k$. 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 (2e) is satisfied. Consider $e = \{i, j\} \in E^k$. 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 $\sum_{C \in C_{ij}^k} z_C^* \geq 1 = x_e^*$, and the constraint is satisfied. □

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

Lemma 3. *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 4. *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 $\mathcal{C} = \cup_{i,j} C_{ij}^k$ to a set F of appropriate size. By Lemma 3, every minimal length-k i, j-connector $C \in C_{ij}^k$ induces an i, j-path graph, say $i = v_0$-v_1-v_2-$\cdots$-$v_q = j$, where $q \leq k$. For such a connector C, define $f(C)$ as follows

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

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 $n = O(m)$, and since k is a constant, $nk = O(m)$. 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,

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

and when $k \geq 2$ is even,

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

□

Remark 1. *By Lemma 4, the number of variables, constraints, and nonzeros in the path-like formulation is $O(m^{(k+1)/2})$ when k is an odd constant, and $O(nm^{k/2})$ when k is an even constant.*

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

Lemma 5. *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 $\tilde{x}_e \coloneqq \max\left\{\hat{z}_C \;\middle|\; C \in C_{ij}^k\right\}$. 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 $(\hat{x}, \hat{y}, \hat{z}) \in P$. □

The following lemma implies that the conflict constraints (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) 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 6. *If a point $(\hat{x}, \hat{y}, \hat{z})$ satisfies the constraints defining $\text{PATH}^{\uparrow_x}$ except perhaps* (2c)*, then there is a similar point $(\hat{x}, \hat{y}, \tilde{z})$ that belongs to $\text{PATH}^{\uparrow_x}$ and that satisfies the following inequalities.*

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

Proof. For every $C \in \mathcal{C}$, let $\tilde{z}_C = \max\left\{0, 1 - \sum_{c \in C} \hat{y}_c\right\}$. Clearly, $(\hat{x}, \hat{y}, \tilde{z})$ satisfies constraints (2b), constraint (2f), and the 0-1 bounds on the y and z variables. To show that inequalities (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 (2d) is satisfied, consider a power graph edge $e = \{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C^k_{ij}$. Now, in the first case, where $\tilde{z}_C = 0$, constraint (2d) is obviously satisfied. In the other case, where $\tilde{z}_C = 1 - \sum_{c \in C} \hat{y}_c$, constraint (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 5, $(\hat{x}, \hat{y}, \tilde{z})$ belongs to $\text{PATH}^{\uparrow_x}$.

Finally, to show that inequalities (3) are satisfied, consider a power graph edge $\{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C^k_{ij}$ with at least three vertices, and see that

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

□

3.2 The Thin Formulation

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

Definition 4 (length-k i, j-separator)**.** *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.

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.

$$\min \sum_{e \in E^k} c_e x_e \tag{4a}$$

$$x_e + \sum_{v \in S} y_v \leq |S| \qquad \forall S \in S_{ij}^k, \quad \forall e = \{i,j\} \in E^k \tag{4b}$$

$$x_e + \sum_{v \in C} y_v \geq 1 \qquad \forall C \in C_{ij}^k, \quad \forall e = \{i,j\} \in E^k \tag{4c}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{4d}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^k \tag{4e}$$

$$y_i \in \{0,1\} \qquad \forall i \in V. \tag{4f}$$

Observe that there can be exponentially many constraints (4b) and (4c). Fortunately, however, we will see that constraints (4b) can be omitted when the connection costs c_e are positive, and constraints (4c) can be separated in polynomial time.

Theorem 2. *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

$$(x^*, y^D) \text{ satisfies (4b) and (4c)} \iff \forall e = \{i,j\} \in E^k, x_e^* = \begin{cases} 1 & \text{if } \mathrm{dist}_{G-D}(i,j) \leq k \\ 0 & \text{if } \mathrm{dist}_{G-D}(i,j) > k. \end{cases}$$

($\implies$) Suppose (x^*, y^D) satisfies constraints (4b) and (4c). In the first case, suppose that the endpoints of the power graph edge $e = \{i,j\} \in E^k$ satisfy $\mathrm{dist}_{G-D}(i,j) \leq k$. 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 (4c). So, $x_e^* = 1$, as desired. In the other case, suppose that the endpoints of $e = \{i,j\} \in E^k$ satisfy $\mathrm{dist}_{G-D}(i,j) > k$. 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 (4b). So, $x_e^* = 0$, as desired.

($\impliedby$) Suppose that for each power graph edge $e = \{i,j\} \in E^k$, $x_e^* = 1$ if and only if $\mathrm{dist}_{G-D}(i,j) \leq k$. To show that constraints (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., $\sum_{v\in S} y^D_v \leq |S| - 1$. Then, constraint (4b) is satisfied, as

$$x^*_e + \sum_{v\in S} y^D_v = 1 + \sum_{v\in S} y^D_v \leq 1 + (|S| - 1) = |S|.$$

To show that constraints (4c) are satisfied, consider an edge $e = \{i,j\} \in E^k$ and a minimal length-k i,j-connector $C \in C^k_{ij}$. 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., $\sum_{v\in C} y^D_v \geq 1$. Then, constraint (4c) is satisfied, as

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

□

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

Lemma 7. *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

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

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 $C \in C^k_{ij}$, see that

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

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 $\tilde{C} \in C^k_{ij}$,

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

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 $(\tilde{x}, \hat{y}) \in \text{THIN}$. 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 $(\tilde{x}, \hat{y}) \in \text{THIN}$. 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 $C \in C^k_{ij}$, implying that $(\hat{x}, \hat{y}) \in P$, as desired. □

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 $\text{proj}_{x,y} \text{PATH}$, and that $\text{proj}_{x,y} \text{R}$ is incomparable with THIN and with $\text{proj}_{x,y} \text{PATH}$. 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.

3.3.1 Strength of Full Formulations

Theorem 3. *For every instance of DCNP, the inclusion* $\text{THIN} \subseteq \text{proj}_{x,y} \text{PATH}$ *holds. Meanwhile,* $\text{proj}_{x,y} \text{R}$ *is incomparable with* $\text{proj}_{x,y} \text{PATH}$ *and with* THIN.

Proof. This follows by Lemmata 8 and 9, which are shown below. □

Lemma 8. *For every instance of DCNP, the inclusion* $\text{THIN} \subseteq \text{proj}_{x,y} \text{PATH}$ *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 2). Observe that (x^*, y^*, z^*) satisfies constraint (2f) and the 0-1 bounds.

To show that constraints (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 (4c). In the other case, where $z^*_C = 1 - \max\{y^*_v \mid v \in C\}$,

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

To show that constraints (2c) are satisfied, consider a connector $C \in \mathcal{C}$ and vertex $v \in C$. Then,

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

To show that constraints (2d) are satisfied, consider a power graph edge $e = \{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C^k_{ij}$. Then,

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

To show that constraints (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 $x^*_e \leq \sum_{C \in C^k_{ij}} z^*_C$ 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, $S \coloneqq \cup_{C \in C^k_{ij}} v_C$ is a length-k i, j-separator, which we claim satisfies

$$|S| - \sum_{s \in S} y^*_s \leq |C^k_{ij}| - \sum_{C \in C^k_{ij}} y^*_{v_C}. \tag{5}$$

To show inequality (5), let $q_s = |\{C \in C^k_{ij} \mid v_C = s\}|$ for every separator vertex $s \in S$. This (positive) value q_s is the number of connectors $C \in C^k_{ij}$ for which s is selected as v_C. Then,

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

thus proving inequality (5). Finally,

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

where the first inequality holds by constraint (4b) of the thin formulation, and the second inequality holds by inequality (5). So, (x^*, y^*, z^*) satisfies constraints (2e), and thus $(x^*, y^*, z^*) \in \text{PATH}$.

Figure 2 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 $(x^*, y^*) \notin \text{THIN}$.

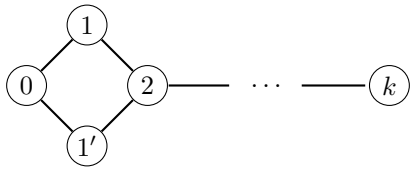


Figure 2: The graph $G = (V, E)$ used to show $\text{proj}_{x,y} \text{PATH} \neq \text{THIN}$.

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 (4b) from the thin formulation is violated. □

Lemma 9. $\text{proj}_{x,y} \text{R}$ *is incomparable with* $\text{proj}_{x,y} \text{PATH}$ *and with* THIN.

Proof. Since THIN $\subseteq \text{proj}_{x,y}\,\text{PATH}$ by Lemma 8, it suffices to show that $\text{proj}_{x,y}\,\text{R} \nsubseteq \text{proj}_{x,y}\,\text{PATH}$ and THIN $\nsubseteq \text{proj}_{x,y}\,\text{R}$.

($\text{proj}_{x,y}\,\text{R} \nsubseteq \text{proj}_{x,y}\,\text{PATH}$). 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.

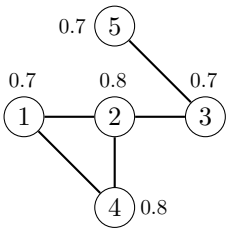


Figure 3: The graph $G = (V, E)$ used to show $\text{proj}_{x,y}\,\text{R} \nsubseteq \text{proj}_{x,y}\,\text{PATH}$.

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 $(\hat{x}, \hat{y}, \hat{z}) \in \text{PATH}$. 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 (2d) force $\hat{z}_C \leq \hat{x}_e = 0.3$ and constraints (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 $(\hat{x}, \hat{y}, \hat{z}) \in \text{PATH}$.

(THIN $\nsubseteq \text{proj}_{x,y}\,\text{R}$). We construct a DCNP instance and a point $(\hat{x}, \hat{y}) \in \text{THIN}$ for which no $\hat{u}$ has $(\hat{x}, \hat{y}, \hat{u}) \in \text{R}$. Consider the graph in Figure 4.

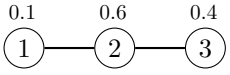


Figure 4: The graph $G = (V, E)$ used to show THIN $\nsubseteq \text{proj}_{x,y}\,\text{R}$.

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 $(\hat{x}, \hat{y}, \hat{u}) \in \text{R}$. Constraints (1e) and (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 (1g). Thus, no $\hat{u}$ satisfies $(\hat{x}, \hat{y}, \hat{u}) \in \text{R}$. □

3.3.2 Strength of Partial Dominants

Theorem 4. *For every (hop-based) instance of DCNP,*

$$\text{THIN}^{\uparrow_x} = \text{proj}_{x,y}\,\text{PATH}^{\uparrow_x} = \text{proj}_{x,y}\,\text{R}^{\uparrow_x}\,.$$

Proof. We show that the following three inclusions hold.

$$\text{THIN}^{\uparrow_x} \subseteq \text{proj}_{x,y}\,\text{PATH}^{\uparrow_x} \subseteq \text{proj}_{x,y}\,\text{R}^{\uparrow_x} \subseteq \text{THIN}^{\uparrow_x}\,.$$

($\text{THIN}^{\uparrow x} \subseteq \text{proj}_{x,y} \text{PATH}^{\uparrow x}$) 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 8.

($\text{proj}_{x,y} \text{PATH}^{\uparrow x} \subseteq \text{proj}_{x,y} \text{R}^{\uparrow x}$) 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 $\text{PATH}^{\uparrow x}$. By Lemma 6, there is a similar point $(\hat{x}, \hat{y}, \tilde{z}) \in \text{PATH}^{\uparrow x}$ that satisfies inequalities (3). We construct a $\hat{u}$ such that $(\hat{x}, \hat{y}, \hat{u})$ belongs to $\text{R}^{\uparrow x}$. Specifically, for distinct vertices $i, j \in V$ and $s \in \{1, 2, \ldots, k\}$, let

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

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

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

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

where the inequality holds by constraint (2b) of the path-like formulation.

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

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

where the inequality holds by inequality (2c).

To show that constraints (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 $s \in \{1, 2, \ldots, k\}$, so

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

To show that constraints (1f) are satisfied, consider a "missing" edge $\{i, j\} \notin E$ and $s \in \{2, 3, \ldots, k\}$. Let $C' \in C^s_{ij}$ 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,

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

Here, the first inequality holds by inequality (3), and the second holds because $C' \setminus \{i\}$ is a minimal length-$(s-1)$ q, j-connector and by definition of $\hat{u}^{s-1}_{qj}$.

To show that constraints (1g) are satisfied, consider a "missing" edge $\{i, j\} \in E$, a neighbor $t \in N(i)$, and $s \in \{2, 3, \ldots, k\}$. Let $\bar{C} \in C^{s-1}_{tj}$ 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 $\text{dist}_{G[C^*]}(i, j) = \text{dist}_{G[\bar{C} \cup \{i\}]}(i, j) \leq s$. Further, $\hat{C} = C^* \setminus \{i\}$ belongs to $\mathcal{C}$. Observe that

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

where the first and second inequalities hold by inequality (3).

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

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

where the inequality holds by constraint (2d) of the path-like formulation.

($\text{proj}_{x,y} \text{R}^{\uparrow_x} \subseteq \text{THIN}^{\uparrow_x}$) 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 $\text{R}^{\uparrow_x}$. We show $(\hat{x}, \hat{y}) \in \text{THIN}^{\uparrow_x}$. By Lemma 7, it suffices to show that $(\hat{x}, \hat{y})$ satisfies all constraints defining THIN except perhaps for constraints of the form $x_e + \sum_{v \in S} y_v \le |S|$ and $x_e \le 1$. First, see that $(\hat{x}, \hat{y})$ satisfies constraint (4d) and the 0-1 bounds on $\hat{y}$.

To show that constraints (4c) are satisfied, consider a power graph edge $e = \{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C^k_{ij}$ that, say, induces the path $i = c_0\text{-}c_1\text{-}\cdots\text{-}c_s = j$, where $s \le k$. Then,

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

where the first inequality holds by $x_e \ge u^k_{ij}$, the middle inequalities hold by constraints (1g), the equality holds by constraints (1e), and the last inequality holds by constraints (1c).

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

4 Implementation Details

This section details our 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 (4c).

4.1 Variable Fixing

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

Remark 2. *Assume hop-based distances and unit connection costs $c_e = 1$. If i is a leaf vertex whose stem j satisfies $\deg_G(j) \ge 2$ and $a_j \le a_i$, 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 $O(mn)$.

Proposition 1. *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

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

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.

$$\sum_{e \in \delta_1} c_e \leq \sum_{e \in \delta_0} c_e \tag{6}$$

$$\sum_{e \in E_1 \setminus \delta_1} c_e \leq \sum_{e \in E_0 \setminus \delta_0} c_e. \tag{7}$$

The former inequality (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 (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 1, 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 $S \coloneqq \{v \in V \mid \text{is simplicial in } G\}$;
2. find $S' \coloneqq \{v \in S \mid v \text{ satisfies condition 2}\}$;
3. pick one vertex v_i from each of the components G_1, G_2, …, G_p of $G[S']$;
4. return $I \coloneqq \{v_1, v_2, \ldots, v_p\}$.

In our computational experiments, instances have unit deletion costs $a_v = 1$ and unit connection costs $c_e = 1$. Under these settings $S' = S$, allowing us to skip step 2 in our implementation.

Proposition 2. *The algorithm* `FindNoncriticalNodes` *finds a maximum independent set of simplicial nodes (that satisfies condition 2 of Proposition 1) 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 them $\{i, j\} \in E$ 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 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 1. 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 $q := \text{diam}(G') \geq 2$, 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, 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 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.

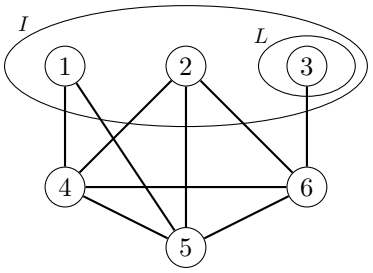


Figure 5: An illustration of the sets L and I.

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 5 (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 bewteen i and j that contain vertex v. The betweenness centrality of vertex v is defined as*

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

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 $= 1, 2, \ldots, b$ **do**
3: let $v \in \text{argmin}\{\text{obj}(D \setminus \{u\}) \mid u \in D\}$
4: $D \leftarrow D \setminus \{v\}$
5: **return** D

Proposition 3. *Algorithm 1 takes time $\mathcal{O}(b^2 mn)$ 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 $\mathcal{O}(b^2 mn)$. 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 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 paper is on exact approaches, and instances like `cond-mat` lie at the boundary of tractability, so this simple heuristic suffices for our purposes.

4.3 The Separation Problem

Since the thin formulation generally has exponentially many constraints (4c), it is important to 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 (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 (4c) of the type $x_e + \sum_{v \in C} y_v \geq 1$ that (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 thin formulation can solved in polynomial-time if and only if the associated separation problem can be solved in polynomial time. Meanwhile, the ability to perform integer

				$b = 5$			$b = 10$			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 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 ($|I|$).

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., 2017; Validi and Buchanan, 2019; Salemi and Buchanan, 2020a). Table 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 2: The complexity of the integer and fractional separation problems, under hop-based and edge-weighted distances.

4.3.1 Integer Separation

Algorithm 2 solves the separation problem when given an integer point $(x^*, y^*) \in \{0, 1\}^{|E^k|+|V|}$. It 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** $\text{dist}_{G-D^*}(i, j) \leq k$ 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 3. *Algorithm 2 runs in time $\mathcal{O}(nm)$ when distances are hop-based and in time $O(nm + n^2 \log n)$ 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 $O(knm)$. Meanwhile, when distances are edge-weighted, the paths P may cross roughly n vertices, perhaps requiring us to increase the time bound to $O(n^3)$.

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.

4.3.2 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 5.

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.

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 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 4. *Under hop-based distances, fractional separation of inequalities* (4c) *takes time $\mathcal{O}(knm)$.*

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

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

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 5. *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 $p_1, p_2, \ldots, p_n$ 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 6. 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 $e \in E^k \setminus \{\{i, j\}\}$.

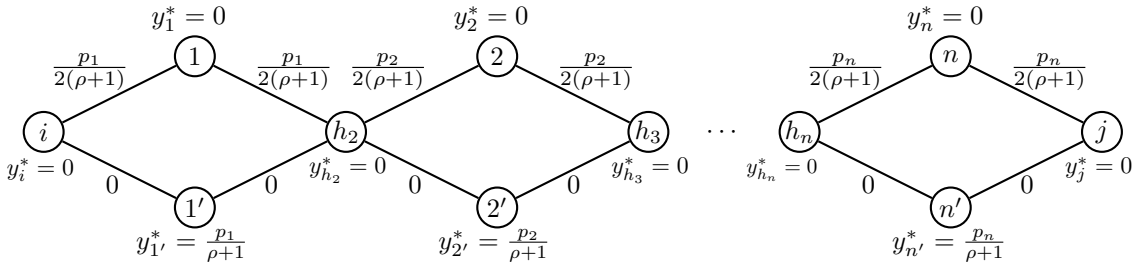


Figure 6: The graph used in the NP-hardness reduction.

Observe that (x^*, y^*) satisfies the budget constraint (4d), the 0-1 bounds on x and y, and all connector constraints (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 $\text{THIN}^{\uparrow_x}$.

($\implies$) 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 $E\,(G[C])$ with positive weight and see

$$\text{dist}_{G[C]}(i, j) = \sum_{e \in E(G[C])} w_e = \sum_{e \in \hat{E}(G[C])} w_e = \frac{\sum_{v \in T} p_v}{\rho + 1} = \frac{\rho}{\rho + 1} = k$$

$$x^*_{\{i,j\}} + \sum_{v \in C} y^*_v = x^*_{\{i,j\}} + \sum_{v \in B} y^*_v = 0 + \frac{\sum_{v \in \bar{T}} p_v}{\rho + 1} = \frac{\rho}{\rho + 1} < 1.$$

($\Longleftarrow$) Suppose that (x^*, y^*) violates at least one of the inequalities defining $\text{THIN}^{\uparrow_x}$. 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 6 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

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

□

5 Computational Experiments

In this section, 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_F, a branch-and-cut implementation of the thin formulation using fractional separation;
- THIN_I, a branch-and-cut implementation of the thin formulation using integer separation.

Each implementation uses the heuristic and variable fixing procedure given in Section 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.

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 and 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 and 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.

The results given in Tables 5 and 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	$\lvert E^k\rvert$	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: 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 4: Running times, or bounds at termination, when $(k, b) = (3, 10)$.

Graph	n	m	$\lvert E^k\rvert$	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 5: Running times, or bounds at termination, when $(k, b) = (4, 5)$.

Graph	n	m	$\lvert E^k\rvert$	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 6: Running times, or bounds at termination, when $(k, b) = (4, 10)$.

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 7 and 8 provide results when $\alpha \in \{0.05, 0.10\}$. We only provide results for the THIN_I implementation, as the other implementations would require an exponential number of initial constraints or have an NP-hard separation problem (Theorem 5). The tables also report the heusistic's objective (heur) and the time spent by the heuristic (htime).

					$b = 5$				$b = 10$			
Graph	n	m	k	$\lvert E^k\rvert$	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 7: Results for $\alpha = 0.05$.

					$b = 5$				$b = 10$			
Graph	n	m	k	$\lvert E^k\rvert$	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 8: Results for $\alpha = 0.10$.

We see that the THIN_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 $\{x_e\}_{e \in E^k}$.

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[1]. Figures 7, 8, and 9 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 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.

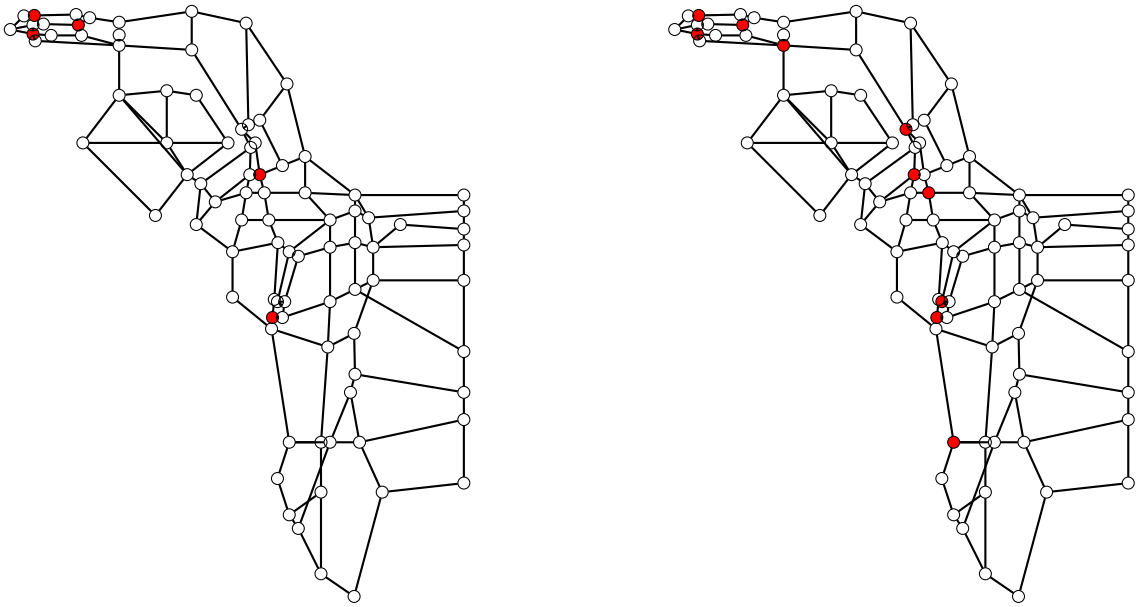


Figure 7: DCNP solutions for Buffalo network when $\alpha = 0.05$ and $b \in \{5, 10\}$.

6 Conclusion

In this paper, 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

[1]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.

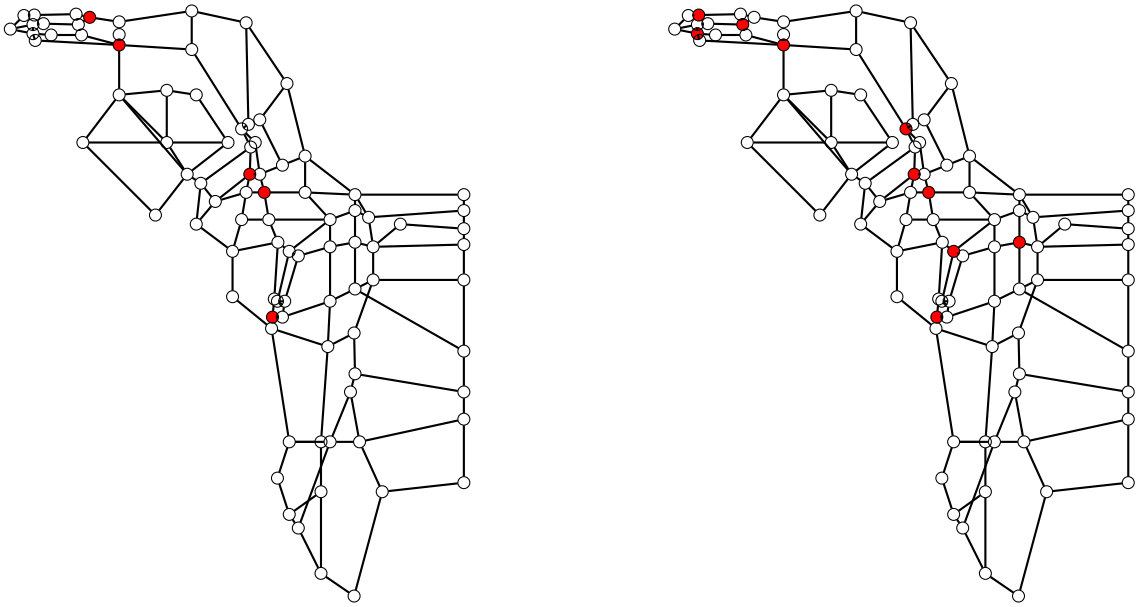


Figure 8: DCNP solutions for Buffalo network when $\alpha = 0.10$ and $b \in \{5, 10\}$.

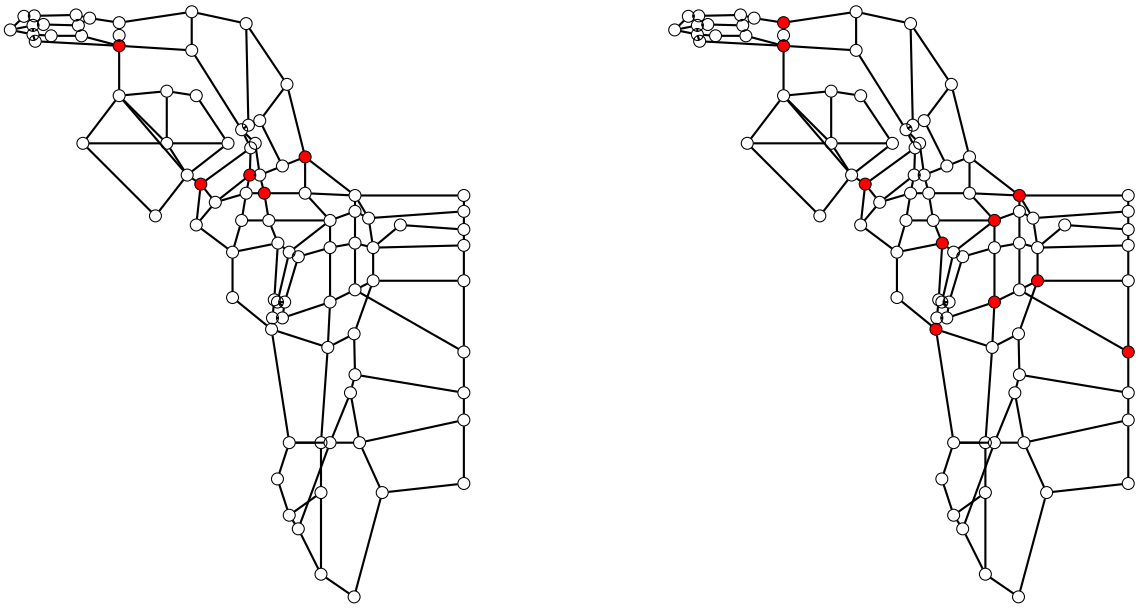


Figure 9: CNP solutions for Buffalo network when $b \in \{5, 10\}$.

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.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1662757.

References

Bernardetta Addis, Marco Di Summa, and Andrea 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.

Bernardetta Addis, Roberto Aringhieri, Andrea Grosso, and Pierre Hosteins. Hybrid constructive heuristics for the critical node problem. *Annals of Operations Research*, 238(1-2):637–649, 2016.

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

Ashwin Arulselvan, Clayton W Commander, Lily Elefteriadou, and Panos M Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

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

V. Batagelj and A. Mrvar. Pajek datasets. `http://vlado.fmf.uni-lj.si/pub/networks/data/`, 2006. Accessed: 2020-04-12.

Vladimir Boginski and Clayton W Commander. Identifying critical nodes in protein-protein interaction networks. In *Clustering Challenges in Biological Networks*, pages 153–167. World Scientific, 2009.

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

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

Austin Buchanan, Je Sang Sung, Sergiy Butenko, and Eduardo L Pasiliao. An integer programming approach for fault-tolerant connected dominating sets. *INFORMS Journal on Computing*, 27(1): 178–188, 2015.

Hadi Charkhgard, Vignesh Subramanian, Walter Silva, and Tapas 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.

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

Clayton W Commander, Panos M Pardalos, Valeriy Ryabchenko, Stan Uryasev, and Grigoriy Zrazhevsky. The wireless network jamming problem. *Journal of Combinatorial Optimization*, 14 (4):481–498, 2007.

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

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

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

Marco Di Summa, Andrea Grosso, and Marco 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: 2020-04-12.

Matteo Fischetti, Markus Leitner, Ivana Ljubić, Martin Luipersbeck, Michele Monaci, Max Resch, Domenico Salvagnin, and Markus Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017.

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

Michael R Garey and David S Johnson. *Computers and Intractability*. W.H. Freeman and Company, 1979.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition, 1993.

F Hooshmand, F Mirarabrazi, and SA MirHassani. Efficient Benders decomposition for distance-based critical node detection problem. *Omega*, 93:102037, 2020.

Myer Kutz. *Handbook of Transportation Engineering*, volume 768. McGraw-Hill New York, NY, USA:, 2004.

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

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

Hosseinali Salemi and Austin Buchanan. Parsimonious formulations for low-diameter clusters. *Mathematical Programming Computation*, 2020a. To appear.

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

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

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

Ben Stabler, Hillel Bar-Gera, and Elizabeth 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.

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

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

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

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

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

Jose L Walteros, Alexander Veremyev, Panos M Pardalos, and Eduardo L Pasiliao. Detecting critical node structures on graphs: A mathematical programming approach. *Networks*, 73(1): 48–88, 2019.

Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.