Solving the distance-based critical node problem

Hosseinali Salemi
Department of Industrial and Manufacturing Systems Engineering, Iowa State University,
Ames, IA 50011, hsalemi@iastate.edu

Austin Buchanan
School of Industrial Engineering & Management, Oklahoma State University,
Stillwater, OK 74078, buchanan@okstate.edu

In critical node problems, the task is to 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.

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

History:

1. Introduction

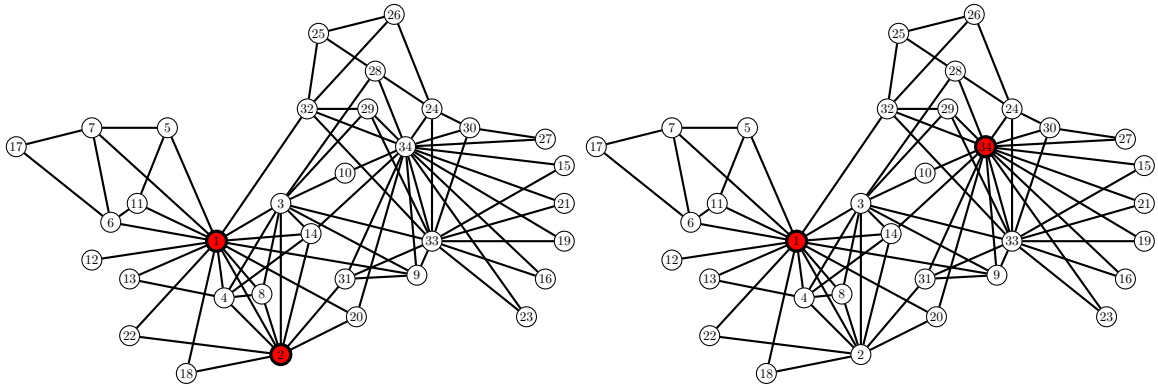
Some nodes in a network are more important than others. Examples include a hub in an airline network, a major train station in a rail network, or Paul Erdős in the mathematical collaboration network. The removal of a small subset of nodes like these can dramatically disrupt the connectivity of the network. 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 classes of 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 $G - D$ (which is the subgraph of G in which the vertices D and their incident edges have been removed). This is 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 (DCNPs) (Borgatti 2006, Veremyev et al. 2015), including the particular variant that we consider in this paper. In it, the task is to delete a subset of b critical nodes $D \subseteq V$ so as to minimize the number of node pairs that remain connected via a short path (i.e., length at most k) in the interdicted graph $G - D$. For generality, we will consider a knapsack constraint for the deletion budget (instead of a cardinality constraint) and edge-weighted distances (instead of hops). We will also allow for a more general objective function that, instead of simply counting *how many* pairs of nodes are close in $G - D$, allows for the node pairs to be weighted. The formal problem definition will be given in Section 2.3.

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, leaving a total of 286 node pairs connected via some path (of any length). 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. Removing the administrator and instructor leaves 168 node pairs connected by paths *of length at most two*.



(a) CNP solution (b) DCNP solution

Figure 1 **Solutions for the `Karate` graph when $b = 2$ and $k = 2$.**

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. Often, distances are measured in hops and k is considered as a small constant, allowing us to write the number of variables and constraints as $\mathcal{O}(|V|^2)$ and $\mathcal{O}(|V||E|)$, respectively. 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 of G) 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 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 as the leaf rule.

Outline. 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 by

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

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 by $\mathrm{dist}_G(u, v)$, 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

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

This definition applies for hop-based and edge-weighted distances. A vertex subset $S \subseteq V$ is a clique if $E[S] = \binom{S}{2}$. A vertex v is simplicial if its neighborhood $N(v)$ is a clique. A vertex is called a leaf if it has just one neighbor, in which case its neighbor is called its stem. A shortest-path tree of graph G rooted at vertex r is a spanning tree T of G where the length of the unique path from r to each vertex v in T equals $\text{dist}_G(r, v)$.

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, Di Summa et al. (2011) show that (a variant of) CNP is NP-hard in trees. Interested readers are encouraged to consult Di Summa et al. (2011), Shen et al. (2013), Addis et al. (2013) for more.

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 (version 9) 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-based constraints, together with some valid inequalities. 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 two given nodes.

The recursive formulation proposed by Veremyev et al. (2015) is generic enough to handle all of the DCNP classes mentioned above. Another noteworthy IP approach was given by Hooshmand et al. (2020) who propose a Benders decomposition algorithm for Class 4.

In this paper, we consider a generalization of Class 1. We impose a knapsack-style budget constraint in which each vertex i has a deletion cost a_i and require that the deletion set $D \subseteq V$ satisfy $\sum_{i \in D} a_i \leq b$. To capture the objective function, we need to know which pairs of vertices remain "close" in the interdicted graph $G - D$, i.e., which node pairs $\{i, j\} \in \binom{V}{2}$ have $\text{dist}_{G-D}(i, j) \leq k$ with respect to the (nonnegative) edge weights w_e. This can be expressed via the k-th power graph. Specifically, the "close" pairs in $G - D$ are precisely the edges of the k-th power graph $(G - D)^k$, which can be written as $E((G - D)^k)$. In the pure Class 1 problem, the objective is to minimize the cardinality of this set, $|E((G-D)^k)|$. However, realizing that not all node pairs $e \in \binom{V}{2}$ are equally important to be close, we allow for pair-specific connection costs c_e. The DCNP objective function is then given by

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

Recognize that the connection costs c_e (which can be negative) are defined for every pair of nodes, while the edge weights w_e (which cannot be negative) are defined only for edges of G. Our variant of DCNP can be concisely expressed as follows.

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 $\sum_{i \in D} a_i \leq b$ minimizing $\text{obj}(D)$.

Under this problem definition, the formulation of Veremyev et al. can be simplified to the following form. In it, 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_{ij}^s 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.

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

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

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

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

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

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

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

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

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

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

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

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

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. For a leaf vertex i, the variable y_i can be fixed to zero if its stem j satisfies $\deg_G(j) \geq 2$ and $a_j \leq a_i$.

With enhancement 2, the number of variables reduces from $\Theta(k|V|^2)$ to $\mathcal{O}(k|E^k|)$. When k is a fixed constant, this is $\mathcal{O}(|E^k|)$. 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. An example is the *dominant* of P, denoted by $P^\uparrow$ (Balas and Fischetti 1996, Schrijver 2003, Conforti et al. 2013).

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

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

where the inequality is a component-wise vector comparison and the symbol + denotes the Minkowski sum.

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 $P = \left\{(x,y) \in \mathbb{R}^n \times \mathbb{R}^d \;\middle|\; Ax + Gy \leq b\right\}$ with respect to x is the polyhedron

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

where the inequality is a component-wise vector comparison, 0^d is the d-dimensional vector of zeros, and the symbol + denotes the Minkowski sum.

When dealing with MIP formulations, we will refer simply to "the partial dominant of the formulation" instead of the technically correct, but more cumbersome, "partial dominant of the formulation's LP relaxation feasible region."

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

The following lemma is often useful for obtaining the partial dominant. It was suggested to us by Kelly Sullivan.

LEMMA 1. *Consider a polyhedron P defined as the set of $(x,y) \in \mathbb{R} \times \mathbb{R}^n$ that satisfy*

$$Ay \leq b \tag{2a}$$

$$l_i(y) \leq x \qquad \forall i \in I \tag{2b}$$

$$x \leq u_j(y) \qquad \forall j \in J, \tag{2c}$$

where each l_i and u_j is an affine function of y. Then, the partial dominant $P^{\uparrow_x}$ with respect to the (single) x variable can be obtained by dropping the upper constraints (2c) *and pairing each lower and upper constraint. That is, $P^{\uparrow_x}$ is the set of $(x,y) \in \mathbb{R} \times \mathbb{R}^n$ that satisfy*

$$Ay \leq b \tag{3a}$$

$$l_i(y) \leq x \qquad \forall i \in I \tag{3b}$$

$$l_i(y) \leq u_j(y) \qquad \forall i \in I,\ \forall j \in J. \tag{3c}$$

Proof. Let $\hat{P}$ be the polyhedron defined by (3a), (3b), and (3c). We show that $\hat{P} = P^{\uparrow_x}$.

($\hat{P} \subseteq P^{\uparrow_x}$) If $\hat{P} = \emptyset$, then the inclusion is trivial. So, suppose that $(\hat{x}, \hat{y}) \in \hat{P}$. To show that $(\hat{x}, \hat{y})$ belongs to $P^{\uparrow x}$, we seek $\check{x}$ with $(\check{x}, \hat{y}) \in P$ and $\hat{x} \geq \check{x}$. To wit, define

$$\check{x} \coloneqq \max\{l_i(\hat{y}) \mid i \in I\},$$

and see that $\check{x} \leq \hat{x}$ by assumption that $\hat{x}$ satisfies constraints (3b). Now, we claim that $(\check{x}, \hat{y})$ belongs to P. First, $\hat{y}$ satisfies constraint (2a) by assumption that it satisfies the identical constraint (3a). Second, $(\check{x}, \hat{y})$ satisfies constraints (2b) by construction of $\check{x}$. Finally, $(\check{x}, \hat{y})$ satisfies each constraint (2c) by

$$\check{x} = \max\{l_i(\hat{y}) \mid i \in I\} \leq u_j(\hat{y}),$$

where the inequality holds by assumption that $\hat{y}$ satisfies constraints (3c).

($\hat{P} \supseteq P^{\uparrow_x}$) If $P^{\uparrow_x} = \emptyset$, then the inclusion is trivial. So, suppose that $(\hat{x}, \hat{y}) \in P^{\uparrow_x}$. By definition of partial dominant, there exists $\check{x}$ such that $(\check{x}, \hat{y}) \in P$ and $\hat{x} \geq \check{x}$. Then, since $(\check{x}, \hat{y})$ belongs to P, this means that

$$A\hat{y} \leq b \tag{4a}$$

$$l_i(\hat{y}) \leq \check{x} \qquad \forall i \in I \tag{4b}$$

$$\check{x} \leq u_j(\hat{y}) \qquad \forall j \in J. \tag{4c}$$

We now claim that $(\hat{x}, \hat{y}) \in \hat{P}$. First, $(\hat{x}, \hat{y})$ satisfies constraint (3a) by inequality (4a). Second, $(\hat{x}, \hat{y})$ satisfies constraints (3b) by $l_i(\hat{y}) \leq \check{x}$ from (4b) and by $\check{x} \leq \hat{x}$. Finally, $(\hat{x}, \hat{y})$ satisfies constraints (3c) by inequalities (4b) and (4c). □

Lemma 2. *The partial dominant can be constructed sequentially. That is, if polyhedron P is defined over variables $(x, y) \in \mathbb{R}^d \times \mathbb{R}^n$ and $x = (x_1, x_2, \ldots, x_d)$, then*

$$P^{\uparrow_x} = (((P^{\uparrow_{x_1}})^{\uparrow_{x_2}}) \cdots)^{\uparrow_{x_d}}.$$

See Section 1 of the Online Supplement for the proof of Lemma 2.

We first identify the partial dominant of the recursive formulation, where the LP feasible region of the recursive formulation is denoted by R.

Lemma 3. *The partial dominant $\mathrm{R}^{\uparrow_x}$ of the recursive formulation can be obtained by omitting constraints $x_e \leq u_{ij}^k$ and $0 \leq x_e \leq 1$ from the definition of* R.

Proof. Consider a variable x_e from the definition of R. It appears only in the constraints $x_e = u_{ij}^k$, $x_e \geq 0$, and $x_e \leq 1$. In inequality form, we have lower constraints $x_e \geq u_{ij}^k$ and $x_e \geq 0$ and upper constraints $x_e \leq u_{ij}^k$ and $x_e \leq 1$. Since u_{ij}^k is itself bounded between zero and one, we can drop the implied bounds $x_e \geq 0$ and $x_e \leq 1$. Lemma 1 states that the partial dominant with respect to x_e is given by dropping $x_e \leq u_{ij}^k$ and pairing the lower and upper constraints to get the constraint $u_{ij}^k \leq u_{ij}^k$ which is redundant and can also be dropped. Repeatedly applying this procedure gives the result, as Lemma 2 guarantees. □

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

The path-like formulation relies on *length-bounded (vertex) connectors* (Salemi and Buchanan 2020), defined below. See Schrijver (2003, pp. 203) for information about the related and more well-studied (edge) s,t-connectors.

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

We remark that (minimal) length-k i,j-connectors are not necessarily synonymous with paths when distances are edge-weighted. Figure 2 gives an example.

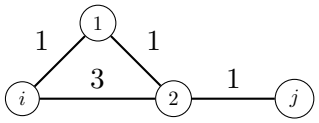


Figure 2 **When $k = 3$, $C = \{i, 1, 2, j\}$ is a (minimal) length-k i,j-connector, but C does not induce a path graph.**

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

Lemma 4. *If edge weights w_e are nonnegative, then every $C \in \mathcal{C}$ is a minimal length-k i,j-connector for a unique pair of vertices $\{i,j\}$. When some edge weights w_e are negative this is not necessarily true.*

See Section 2 of the Online Supplement for the proof of Lemma 4.

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.

$$\min \sum_{e \in E^k} c_e x_e \tag{5a}$$

$$z_C + \sum_{v \in C} y_v \geq 1 \qquad \forall C \in \mathcal{C} \tag{5b}$$

$$z_C + y_v \leq 1 \qquad \forall v \in C, \quad \forall C \in \mathcal{C} \tag{5c}$$

$$x_e \geq z_C \qquad \forall C \in C^k_{ij}, \quad \forall e = \{i, j\} \in E^k \tag{5d}$$

$$x_e \leq \sum_{C \in C^k_{ij}} z_C \qquad \forall e = \{i, j\} \in E^k \tag{5e}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{5f}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E^k \tag{5g}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{5h}$$

$$z_C \in \{0, 1\} \qquad \forall C \in \mathcal{C}. \tag{5i}$$

Constraints (5b) and (5c) 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 (5d) and (5e) 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 (5c) and (5e) can be omitted when the connection costs c_e are nonnegative.

PROPOSITION 1. *The path-like formulation is correct, even under edge-weighted distances.*

See Section 2 of the Online Supplement for the proof of Proposition 1.

Next, we bound the size of the path-like formulation with help from the following lemma. The proof is omitted since it is straightforward.

LEMMA 5. *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 of length at most k.*

Note that the hop-based assumption in Lemma 5 is necessary, as Figure 2 shows.

LEMMA 6. *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.*

See Section 2 of the Online Supplement for the proof of Lemma 6.

REMARK 1. By Lemma 6, when distances are hop-based, the number of variables, constraints, and nonzeros in the path-like formulation is $\mathcal{O}(m^{(k+1)/2})$ when k is an odd constant, and $\mathcal{O}(nm^{k/2})$ when k is an even constant. Meanwhile, when k is a constant, the (enhanced) recursive formulation has $\mathcal{O}(|E^k|)$ variables, $\mathcal{O}(nm)$ constraints, and $\mathcal{O}(nm)$ nonzeros.

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

LEMMA 7. *The partial dominant* $\text{PATH}^{\uparrow_x}$ *of the path-like formulation can be obtained by omitting constraints $x_e \le \sum_{C \in C^k_{ij}} z_C$ and $0 \le x_e \le 1$ from the definition of* PATH.

Proof. Consider a variable x_e from the definition of PATH. It appears only in the constraints $x_e \ge 0$, $x_e \le 1$, $x_e \le \sum_{C' \in C^k_{ij}} z_{C'}$, and $x_e \ge z_C$ for $C \in C^k_{ij}$. Since each z_C itself is nonnegative, we can drop the implied bound $x_e \ge 0$. By Lemma 1, the partial dominant with respect to x_e is obtained by dropping $x_e \le \sum_{C' \in C^k_{ij}} z_{C'}$ and $x_e \le 1$, and pairing the lower and upper constraints to get $z_C \le 1$ and $z_C \le \sum_{C' \in C^k_{ij}} z_{C'}$ for $C \in C^k_{ij}$, which are redundant and can be removed. Repeatedly applying this procedure for each x_e variable gives the result, as Lemma 2 guarantees. □

In the following, we use proj to denote projection. That is, if a polyhedron P is defined over variables $(x,y) \in \mathbb{R}^d \times \mathbb{R}^n$, then its projection into the x-space of variables is given by

$$\text{proj}_x P := \{x \mid (x,y) \in P\}.$$

The following lemma implies that the conflict constraints (5c), 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 (6) 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 8. *If a point* $(\hat{x}, \hat{y}, \hat{z})$ *satisfies the constraints defining* $\text{PATH}^{\uparrow_x}$ *except perhaps* (5c)*, then there is a similar point* $(\hat{x}, \hat{y}, \tilde{z})$*, where* $\tilde{z}_C = \max\left\{0, 1 - \sum_{c \in C} \hat{y}_c\right\}$ *for every* $C \in \mathcal{C}$*, that belongs to* $\text{PATH}^{\uparrow_x}$ *and that satisfies:*

$$\tilde{z}_{C\setminus\{i\}} - \hat{y}_i \leq \tilde{z}_C \leq \tilde{z}_{C\setminus\{i\}} \qquad \forall C \in C_{ij}^k \text{ with } |C| \geq 3, \forall \{i,j\} \in E^k. \tag{6}$$

See Section 2 of the Online Supplement for the proof of Lemma 8.

3.2. The Thin Formulation

The thin formulation relies on *length-bounded separators*, defined below, cf. Baier et al. (2010) and Salemi and Buchanan (2020). Schrijver (2003, pp. 33,132) has more information about the related (vertex) s,t-separators, and Carvajal et al. (2013), Buchanan et al. (2015), Fischetti et al. (2017) use them for imposing connectivity in MIPs.

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*. The collection of all minimal length-k i,j-separators is denoted by S_{ij}^k.

In Figure 3, the minimal length-5 i,j separators are $S_{ij}^5 = \{\{i\}, \{j\}, \{2\}\}$. The first two sets $\{i\}$ and $\{j\}$ satisfy the definition because they contain i or j, while the last set $\{2\}$ satisfies the definition because the distance $\text{dist}_{G-\{2\}}(i,j) = 6$ is greater than 5. Meanwhile, the minimal length-6 i,j-separators are $S_{ij}^6 = \{\{i\}, \{j\}, \{1,2\}, \{2,3\}\}$.

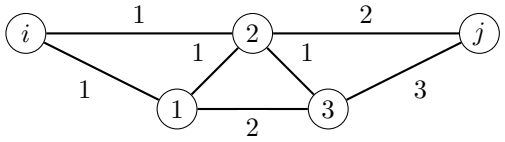


Figure 3 **An edge-weighted graph to illustrate length-k i,j-separators.**

The thin formulation, which uses the same x and y variables as before, is as follows.

$$\min \sum_{e \in E^k} c_e x_e \tag{7a}$$

$$x_e + \sum_{v \in S} y_v \leq |S| \qquad \forall S \in S_{ij}^k, \quad \forall e = \{i,j\} \in E^k \tag{7b}$$

$$x_e + \sum_{v \in C} y_v \geq 1 \qquad \forall C \in C_{ij}^k, \quad \forall e = \{i,j\} \in E^k \tag{7c}$$

$$\sum_{i \in V} a_i y_i \leq b \tag{7d}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^k \tag{7e}$$

$$y_i \in \{0,1\} \qquad \forall i \in V. \tag{7f}$$

Constraints (7b) ensure that, for any power graph edge $e = \{i,j\}$ and minimal separator $S \in S_{ij}^k$, either the endpoints of e have distance more than k in the remaining graph ($x_e = 0$) or at least one vertex of S is intact ($\sum_{v \in S} y_v \leq |S| - 1$). Constraints (7c) ensure that, for any power graph edge $e = \{i,j\}$ and minimal connector $C \in C_{ij}^k$, either the endpoints of e have distance at most k in the remaining graph ($x_e = 1$) or at least one vertex of C is deleted ($\sum_{v \in C} y_v \geq 1$). Observe that the thin formulation has $|E^k| + n$ variables and can have exponentially many constraints (7b) and (7c). Fortunately, however, we will see that constraints (7b) can be omitted when the connection costs c_e are positive, and constraints (7c) can be separated in polynomial time.

PROPOSITION 2. *The thin formulation is correct, even under edge-weighted distances.*

See Section 2 of the Online Supplement for the proof of Proposition 2.

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

LEMMA 9. *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. Consider a variable x_e from the definition of THIN. It appears in the constraints $x_e \geq 0$, $x_e \leq 1$, $x_e \geq 1 - \sum_{v \in C} y_v$ for $C \in C_{ij}^k$, and $x_e \leq |S| - \sum_{v \in S} y_v$ for $S \in S_{ij}^k$. By Lemma 1, the partial dominant with respect to x_e is given by dropping constraints of the form $x_e \leq 1$ and $x_e \leq |S| - \sum_{v \in S} y_v$ and pairing the lower and upper constraints to get constraints of the form $1 - \sum_{v \in C} y_v \leq 1$, $0 \leq 1$, $0 \leq |S| - \sum_{v \in S} y_v$, and $1 - \sum_{v \in C} y_v \leq |S| - \sum_{v \in S} y_v$. The first three are redundant and can be dropped. To show that the last is redundant, consider a minimal separator $S \in S_{ij}^k$ and a minimal connector $C \in C_{ij}^k$. They must intersect each other, so we can pick a vertex $u \in S \cap C$. Then, any $\bar{y} \in [0,1]^n$ has

$$\sum_{s \in S} \bar{y}_s - \sum_{c \in C} \bar{y}_c = \sum_{s \in S \setminus \{u\}} \bar{y}_s - \sum_{c \in C \setminus \{u\}} \bar{y}_c \leq \sum_{s \in S \setminus \{u\}} \bar{y}_s \leq |S| - 1,$$

showing that the inequality $1 - \sum_{v \in C} y_v \leq |S| - \sum_{v \in S} y_v$ is implied by the 0-1 bounds. Repeatedly applying this procedure gives the result, as Lemma 2 guarantees. □

3.3. Formulation Strength

We now 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 1. *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 10 and 11, which are given and proved in Section 2 of the Online Supplement. □

3.3.2. Strength of Partial Dominants

THEOREM 2. *For every (hop-based) instance of DCNP,*

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

See Section 2 of the Online Supplement for the Proof of Theorem 2.

4. Implementation Details

This section details our implementations of the recursive, path-like, and thin formulations:

- 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 (7c).

4.1. Variable Fixing

Recall the leaf fixing of Veremyev et al. (2015) which, informally, is that leaf vertices are not critical.

REMARK 2. Assume hop-based distances and unit connection costs $c_e = 1$. If the stem j of a leaf vertex i satisfies $\deg_G(j) \geq 2$ and $a_j \leq a_i$, there is an optimal solution with $y_i = 0$.

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

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

PROPOSITION 3. *Assume hop-based distances and $c_e \geq 0$ for $e \in E^k$. If vertices $I \subseteq V$ satisfy conditions 1 and 2 below, there is an optimal deletion set $D^* \subseteq V$ with $D^* \cap I = \emptyset$.*

(i) I is an independent set of simplicial vertices in G;

(ii) 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 $e' \in \delta_{G^k}(u)$.*

See Section 3 of the Online Supplement for the proof of Proposition 3.

Now, to use Proposition 3, 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 $S \coloneqq \{v \in V \mid v \text{ is simplicial in } G\}$;
2. find $S' \coloneqq \{v \in S \mid v \text{ satisfies condition (ii)}\}$;
3. pick one vertex v_i from each of the connected 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 (ii).

PROPOSITION 4. *The algorithm* `FindNoncriticalNodes` *finds a maximum independent set of simplicial nodes (that satisfies condition (ii) of Proposition 3) in time $\mathcal{O}(nm)$.*

See Section 3 of the Online Supplement for the proof of Proposition 4.

Let L be a maximum independent set of leaves, which can be found 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 L, as reported in Table 1. For example, simplicial vertex fixing finds 2,484 and 4,846 more non-critical nodes on `hep-th` and `cond-mat`, respectively.

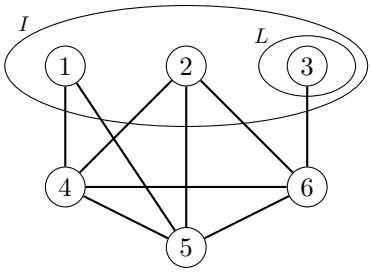


Figure 4 **An illustration of a set of leaves L and an independent set of simplicial nodes I.**

4.2. Heuristic

Our heuristic for DCNP 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 between i and j that contain vertex v. The betweenness centrality of vertex v is:

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

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

Proposition 5. *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.*

See Section 3 of the Online Supplement for the proof of Proposition 5.

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

Algorithm 1 Heuristic for DCNP

initialize D to be the $2b$ vertices with largest betweenness centrality values
for counter $= 1, 2, \ldots, b$ **do**
    let $v \in \arg\min\{\text{obj}(D \setminus \{u\}) \mid u \in D\}$
    $D \leftarrow D \setminus \{v\}$
return D

lead to faster implementations in practice, due to smaller hidden constants. For this reason, our implementation has a slightly slower worst-case 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

The thin formulation generally has exponentially many constraints (7c), so it is important to study the associated separation problem so violated inequalities can be added on-the-fly.

Problem: Separation problem for length-k i, j-connector inequalities (7c).
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 (7c) 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 be solved in polynomial-time if and only if the associated separation problem can be solved in polynomial time. Meanwhile, the ability to perform integer separation provides no such theoretical

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 ($|I|$).

				$b=5$			$b=10$			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	16,185	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

guarantees, but still can be practically useful in a branch-and-cut algorithm (Buchanan et al. 2015, Fischetti et al. 2017, Validi and Buchanan 2020, Salemi and Buchanan 2020). Table 2 summarizes the results.

Table 2 The complexity of the separation problems, under hop-based and edge-weighted distances.

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

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-path trees: time $\mathcal{O}(m+n)$ versus time $\mathcal{O}(m+n\log n)$ in line 3. Algorithm 2 identifies a violated inequality of type (7c) by seeking a power graph edge $e=\{i,j\}$ with $x_e^*=0$ (line 5) and a length-k i,j-connector in $G-D^*$ (line 6) whose vertices $V(P)$ are intact, $\sum_{v\in V(P)} y_v^*=0$.

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-path 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** $\text{dist}_{G-D^*}(i,j)\le k$ and $x_e^*=0$ **then**
6: from the shortest-path tree, find a shortest path P from i to j in $G-D^*$
7: add $x_e+\sum_{v\in V(P)} y_v\ge 1$
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 $\mathcal{O}(nm+n^2\log n)$ when distances are edge-weighted.

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

However, our implementation omits the break (line 8). We find the extra time spent in the separation procedure is justified. The intuition is as follows. Suppose 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, the 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 dynamic programming (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 3.

Algorithm 3 identifies a violated inequality of type (7c) by seeking a power graph $e = \{i, j\}$ and a k-hop path P for which x_e^* plus the path's "cost" ($\sum_{v \in V(P)} y_v^*$) is sufficiently less than one. For simplicity, it is given for 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 by predecessors. Specifically, $p(v, s)$ stores, at termination, the predecessor of vertex v in the cheapest path with at most s hops from i to v.

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

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

Lines 5-10 are the most costly steps, 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 (7c) takes time $\mathcal{O}(knm)$.

To enable Algorithm 3 to handle integer-weighted edges, one should 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 3. *Under edge-weighted distances, fractional separation for* $\text{THIN}^{\uparrow_x}$ *is NP-hard.*

Proof. We prove 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. Construct the edge-weighted graph $G = (V, E)$ shown in Figure 5. 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 $x^*_e = 1$ for all other pairs $e \in E^k \setminus \{\{i,j\}\}$.

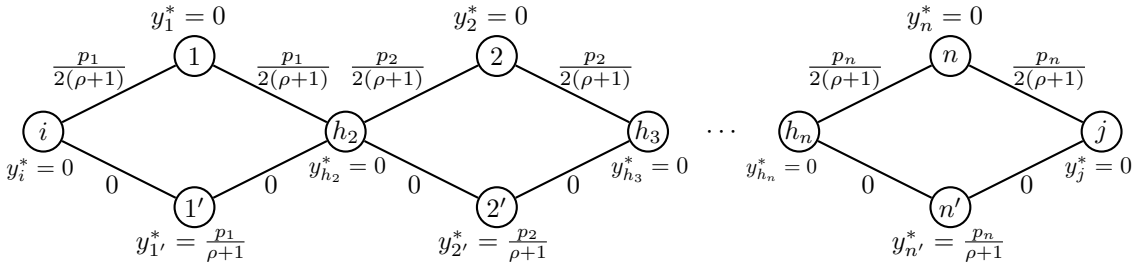


Figure 5 The graph used in the NP-hardness reduction for Theorem 3.

Observe (x^*, y^*) satisfies the budget (7d), the 0-1 bounds on x and y, and all connector constraints (7c) except perhaps for power graph edge $\{i, j\}$. We show the PARTITION instance has a solution if and only if (x^*, y^*) violates one of the inequalities defining $\text{THIN}^{\uparrow_x}$.

($\Longrightarrow$) 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 of the graph in Figure 5 whose top row counterpart v does not belong to T. We observe 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 the associated inequality is violated, $x^*_{\{i,j\}} + \sum_{v\in C} y^*_v < 1$. To wit, let $\hat{E}(G[C])$ be the edges of $E\,(G[C])$ of positive weight and see

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

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

($\Longleftarrow$) Suppose (x^*, y^*) violates at least one inequality defining $\text{THIN}^{\uparrow_x}$. By the discussion above, this 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 5 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

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

5. Computational Experiments

In this section, we provide results of experiments with different implementations of the partial dominant of DCNP formulations:

- R, direct implementation of recursive formulation by Veremyev et al. (2015);
- PATH, direct implementation of path-like formulation;
- THIN, direct implementation of thin formulation;
- THIN_F, branch-and-cut implementation of thin formulation via fractional separation;
- THIN_I, branch-and-cut implementation of thin formulation via 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 2021). 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 $k \in \{3, 4, 5\}$. Using small values like these ensures that paths are counted only when they are short, which is what distinguishes DCNP from CNP. Also, as DCNP can be brute-forced for small b, we consider $b \in \{5, 10, 15\}$.

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 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 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 number of branch-and-bound nodes is typically small, often less than 100 (for all implementations). For the THIN_I and THIN_F implementations, the number of lazy cuts is also small, even when compared to the number of x_e variables, $|E^k|$. These detailed numbers can be found on GitHub.

Table 3 Running times, or the best [lower bound, upper bound] at termination, when $(k, b) = (3, 5)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash.

Graph	n	m	$\lvert E^k\rvert$	obj	R	PATH	THIN_I	THIN_F	THIN
karate	34	78	480	41	0.16	0.09	0.11	0.07	**0.06**
dolphins	62	159	1,107	662	5.35	3.18	**0.72**	0.78	0.76
lesmis	77	254	2,500	517	0.64	0.45	0.26	0.22	**0.19**
LindenStrasse	232	303	3,251	1,810	0.78	0.45	0.38	0.36	**0.30**
polbooks	105	441	3,510	2,555	15.86	8.41	3.79	3.21	**2.61**
adjnoun	112	425	5,634	3,719	2.83	3.04	1.65	1.34	**1.05**
football	115	613	6,247	5,362	257.38	276.13	141.52	98.08	**87.39**
netscience	1,589	2,742	13,087	8,390	19.07	2.00	2.72	2.60	**1.29**
jazz	198	2,742	18,461	16,136	[15732,16185]	[16074,16136]	1851.30	1915.34	**1079.89**
SmallWorld	233	994	25,721	6,964	23.68	9.56	7.95	8.10	**5.96**
Erdos971	429	1,312	34,086	25,737	21.35	14.96	22.28	16.14	**10.55**
S.Cerevisae	1,458	1,948	39,091	25,190	25.86	10.31	9.89	9.62	**5.76**
USAir	332	2,126	46,573	29,486	55.61	52.58	40.96	40.78	**27.54**
power	4,941	6,594	53,125	50,410	195.05	64.19	59.43	77.79	**46.39**
H.Pylori	706	1,392	62,028	37,626	100.81	17.18	24.25	19.81	**9.09**
Harvard500	500	2,043	83,993	16,448	67.30	26.33	18.99	19.09	**16.09**
homer	542	1,619	91,527	45,828	412.57	369.77	62.75	67.02	**34.26**
celegansm	453	2,025	91,531	44,967	187.10	55.10	41.06	**40.96**	42.00
email	1,133	5,451	289,259	263,409	1543.05	202.01	187.57	192.16	**118.61**
hep-th	8,361	15,751	376,431	345,320	MEM	378.45	255.79	261.45	**172.41**
PGPgiant	10,680	24,316	1,145,492	857,035	MEM	2018.44	**686.43**	722.59	702.71
cond-mat	16,726	47,594	1,761,969	1,633,299	MEM	LPNS	3705.23	3701.27	**2393.82**

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

Table 4 Running times, or the best [lower bound, upper bound] at termination, when $(k,b)=(3,10)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash.

Graph	n	m	$\lvert E^k\rvert$	obj(D)	R	PATH	THIN$_{\text{I}}$	THIN$_{\text{F}}$	THIN
karate	34	78	480	6	0.19	0.13	0.14	0.15	**0.08**
dolphins	62	159	1,107	335	5.42	2.44	1.67	1.39	**1.13**
lesmis	77	254	2,500	160	0.61	0.41	0.37	0.31	**0.20**
LindenStrasse	232	303	3,251	1,151	0.84	0.47	0.47	0.47	**0.34**
polbooks	105	441	3,510	1,715	22.18	11.27	7.65	6.17	**3.67**
adjnoun	112	425	5,634	2,501	3.09	3.46	2.81	2.86	**1.18**
football	115	613	6,247	4,523	2164.34	985.71	602.82	633.83	**271.57**
netscience	1,589	2,742	13,087	6,785	20.24	2.85	2.87	2.70	**1.91**
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	1266.10	89.64	64.27	57.24	**21.62**
Erdos971	429	1,312	34,086	20,240	733.33	197.72	75.52	76.32	**55.62**
S.Cerevisae	1,458	1,948	39,091	19,861	33.20	19.71	18.11	17.73	**11.82**
USAir	332	2,126	46,573	19,157	2325.24	1082.62	318.61	324.05	**293.01**
power	4,941	6,594	53,125	48,602	294.30	175.40	155.89	150.63	**143.68**
H.Pylori	706	1,392	62,028	27,807	92.74	22.09	24.94	24.62	**11.86**
Harvard500	500	2,043	83,993	8,581	57.33	30.72	14.78	**14.65**	18.20
homer	542	1,619	91,527	24,882	45.09	34.63	36.68	33.90	**21.64**
celegansm	453	2,025	91,531	25,556	[25183,25556]	2320.80	**111.42**	120.49	134.34
email	1,133	5,451	289,259	241,128	[241060,241144]	1571.03	344.73	492.86	**203.43**
hep-th	8,361	15,751	376,431	321,486	MEM	576.06	468.22	456.14	**368.17**
PGPgiant	10,680	24,316	1,145,492	744,908	MEM	[744769,748537]	**1572.26**	1846.43	1597.51
cond-mat	16,726	47,594	1,761,969	1,541,815	MEM	LPNS	4785.19	4877.03	**3402.80**

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.

In Section 4 of the Online Supplement, we provide results of solving instances for $(k,b)=(3,15)$ and $(k,b)=(4,15)$ with implementations R and THIN. In addition, we report the results of implementations R and THIN$_{\text{I}}$ when $k=5$ and $b\in\{3,4,5\}$. Extensive computational results for all values of $k\in\{3,4,5\}$ and $b\in\{5,10,15\}$ and different implementations of DCNP formulations, including "full" versions of the recursive and path-like formulations are available in the `Results` directory of the GitHub repository (Salemi and Buchanan 2021).

In Section 5 of the Online Supplement, we compare the performance of the R and THIN models under (i) no fixing, (ii) leaf fixing, and (iii) simplicial fixing to better understand the effects of fixing.

Table 5 Running times, or the best [lower bound, upper bound] at termination, when $(k,b)=(4,5)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash. A question mark "?" indicates that the optimal objective value for an instance is not known.

Graph	n	m	$\lvert E^k\rvert$	obj(D)	R	PATH	THIN$_{\text{I}}$	THIN$_{\text{F}}$	THIN
karate	34	78	553	44	0.34	0.19	0.06	**0.05**	0.10
dolphins	62	159	1,459	764	17.19	47.90	1.37	**0.94**	1.39
lesmis	77	254	2,899	583	2.56	4.33	**0.80**	0.82	0.96
LindenStrasse	232	303	7,257	3,526	4.60	2.23	2.29	2.18	**1.42**
polbooks	105	441	4,685	3,333	66.01	158.82	**12.73**	15.26	25.12
adjnoun	112	425	6,178	4,777	322.88	1559.11	**47.18**	75.12	263.58
football	115	613	6,555	?	[5667,5987]	[5657,5994]	[5763,5984]	[5759,5986]	[5648,5987]
netscience	1,589	2,742	22,847	11,786	42.11	5.69	5.14	5.99	**3.43**
jazz	198	2,742	19,336	17,350	[16626,17799]	LPNS	[16802,17799]	[16852,17799]	[16597,17799]
SmallWorld	233	994	27,028	9,606	72.86	49.30	25.71	**21.00**	23.60
Erdos971	429	1,312	62,059	49,325	468.71	136.45	623.46	415.54	**70.97**
S.Cerevisae	1,458	1,948	117,958	66,402	64.58	51.37	54.38	54.80	**28.20**
USAir	332	2,126	53,447	33,795	[33390,33906]	[33390,33906]	788.40	684.15	**359.37**
power	4941	6,594	105,233	97,949	318.20	139.74	**80.21**	80.94	86.89
H.Pylori	706	1,392	162,758	109,368	400.49	115.16	578.75	583.24	**80.86**
Harvard500	500	2,043	119,080	37,491	863.34	231.74	327.04	308.96	**193.00**
homer	542	1,619	133,947	76,068	976.82	251.41	1068.27	938.58	**132.43**
celegansm	453	2,025	100,476	71,463	[70233,71463]	[70233,71463]	[66186,71463]	[70184,71463]	**2617.86**
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	2885.81	[329778,1216604]	[329778,1216604]	**1935.73**
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

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$_{\text{I}}$ implementation, as the other implementations would require an exponential number of initial constraints or have an NP-hard separation problem (Theorem 3). The tables also report the heuristic's objective (heur) and the time spent by the heuristic (htime).

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

Table 6 Running times, or the best [lower bound, upper bound] at termination, when $(k,b)=(4,10)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash. A question mark "?" indicates that the optimal objective value for the instance is not known.

Graph	n	m	$\lvert E^k\rvert$	obj(D)	R	PATH	THIN$_{\text{I}}$	THIN$_{\text{F}}$	THIN
karate	34	78	553	6	0.33	0.22	0.25	**0.07**	0.12
dolphins	62	159	1,459	428	38.61	46.83	5.46	**3.38**	6.44
lesmis	77	254	2,899	178	3.30	4.65	0.67	**0.50**	1.26
LindenStrasse	232	303	7,257	1,913	2.76	2.17	2.06	1.90	**1.38**
polbooks	105	441	4,685	2,118	164.64	674.46	**24.95**	29.30	87.65
adjnoun	112	425	6,178	3,694	1259.60	[3364,3708]	**214.90**	250.94	1137.01
football	115	613	6,555	?	[4707,5356]	[4664,5423]	[4807,5423]	[4774,5408]	[4708,5419]
netscience	1,589	2,742	22,847	8,778	88.36	12.47	6.94	13.54	**6.59**
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]	**93.41**	110.91	234.22
Erdos971	429	1,312	62,059	41,097	[40360,41375]	[40251,41375]	3539.78	**3233.67**	[40545,41097]
S.Cerevisae	1,458	1,948	117,958	47,324	72.44	56.65	49.26	49.46	**40.26**
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	573.16	289.76	**196.07**	250.69	199.54
H.Pylori	706	1,392	162,758	82,441	555.33	169.56	1724.67	1648.07	**108.47**
Harvard500	500	2,043	119,080	16,708	521.37	328.87	163.32	176.06	**122.37**
homer	542	1,619	133,947	46,396	967.32	264.70	3313.51	2486.69	**167.61**
celegansm	453	2,025	100,476	48,046	[47866,52157]	[47866,52157]	[38641,52157]	[47861,48046]	**863.15**
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	3086.37	[306113,1125603]	[306113,1125603]	**1810.36**
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	[1495035,6571705]	[1495035,6571705]	MEM

Table 7 Running times, or the best [lower bound, upper bound] for edge-weighted networks when $\alpha=0.05$ and $b\in\{5,10\}$ using implementation THIN$_{\text{I}}$. We also report the heuristic's objective (heur) and heuristic time (htime). A question mark "?" indicates that the optimal objective value for the instance is not known.

					$b=5$				$b=10$			
Graph	n	m	k	$\lvert E^k\rvert$	heur	opt	htime	(total) time	heur	opt	htime	(total) time
Albany	90	149	44	204	139	136	0.07	0.12	94	91	0.21	0.35
Buffalo	90	149	260	205	127	127	0.08	0.12	94	89	0.17	0.21
DC-NY-BOS	317	509	5,286	2,505	1967	1910	0.90	2.91	1636	1510	2.70	5.05
Korean	324	440	50	2,619	2227	2038	0.74	1.11	1918	1724	2.51	3.06
Anaheim	416	634	7,709	4,348	3587	3540	1.36	2.32	3055	3012	5.18	6.73
Barcelona	930	1,798	127	21,778	20640	20259	8.15	149.38	19674	18977	29.63	3576.68
Rome	3,353	4,831	2,888	281,058	259481	?	112.26	[253268,259481]	251184	?	392.30	[222668,251184]
Austin	7,388	10,591	464	1,368,735	1352766	?	602.50	[30408,1352766]	1336801	?	2271.42	[30275,1336801]
Chicago	12,979	20,627	889	4,213,117	4190256	?	1977.27	[65083,4190256]	4168480	?	7664.38	[64922,4168480]

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

Table 8 Running times, or the best [lower bound, upper bound] for edge-weighted networks when $\alpha = 0.10$ and $b \in \{5, 10\}$ using implementation THIN$_\mathsf{I}$. We also report the heuristic's objective (heur) and heuristic time (htime). A question mark "?" indicates that the optimal objective value for the instance is not known.

					$b=5$				$b=10$			
Graph	n	m	k	$\lvert E^k\rvert$	heur	opt	htime	(total) time	heur	opt	htime	(total) time
Albany	90	149	65	403	256	247	0.08	0.22	158	157	0.17	0.35
Buffalo	90	149	410	402	272	269	0.08	0.22	195	179	0.21	0.34
DC-NY-BOS	317	509	8,641	5,010	4224	3848	0.91	4.59	3394	3154	3.31	11.54
Korean	324	440	78	5,308	4521	4025	0.80	1.75	3669	3154	2.72	4.37
Anaheim	416	634	11,036	8,637	7161	7009	1.50	4.20	6251	5977	5.22	9.48
Barcelona	930	1,798	185	43,449	41644	40126	8.16	[38374,41104]	39344	?	31.77	[33164,38674]
Rome	3,353	4,831	4,189	561,987	524792	?	108.96	[462892,524792]	506983	?	397.36	[13502,506983]
Austin	7,388	10,591	716	2,729,813	2698516	?	604.51	[30793,2698516]	2678413	?	2270.52	[30660,2678413]
Chicago	12,979	20,627	1,325	8,428,119	8386922	?	2104.38	[65187,8386922]	8350868	?	8232.85	[65026,8350868]

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 6, 7, and 8 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. For example, both CNP solutions split Grand Island (the wheel-like subgraph near the upper-left whose nodes are shown by squares in Figure 8) 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.

6. Conclusion and Future Work

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

[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.

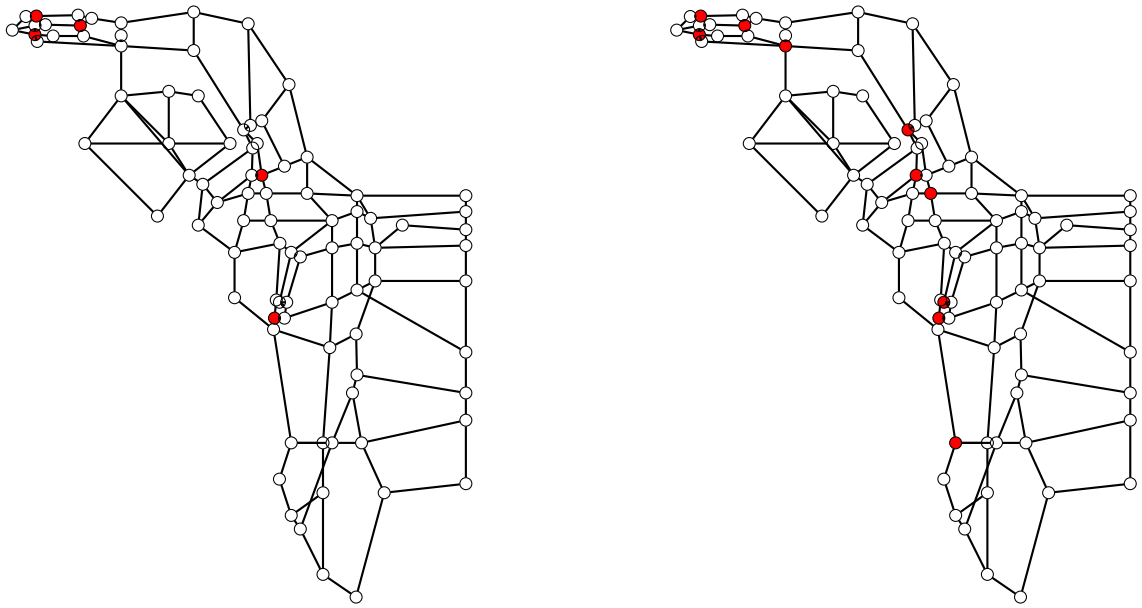


Figure 6 **DCNP solutions for Buffalo network when $\alpha = 0.05$ and the budget is $b \in \{5, 10\}$.**

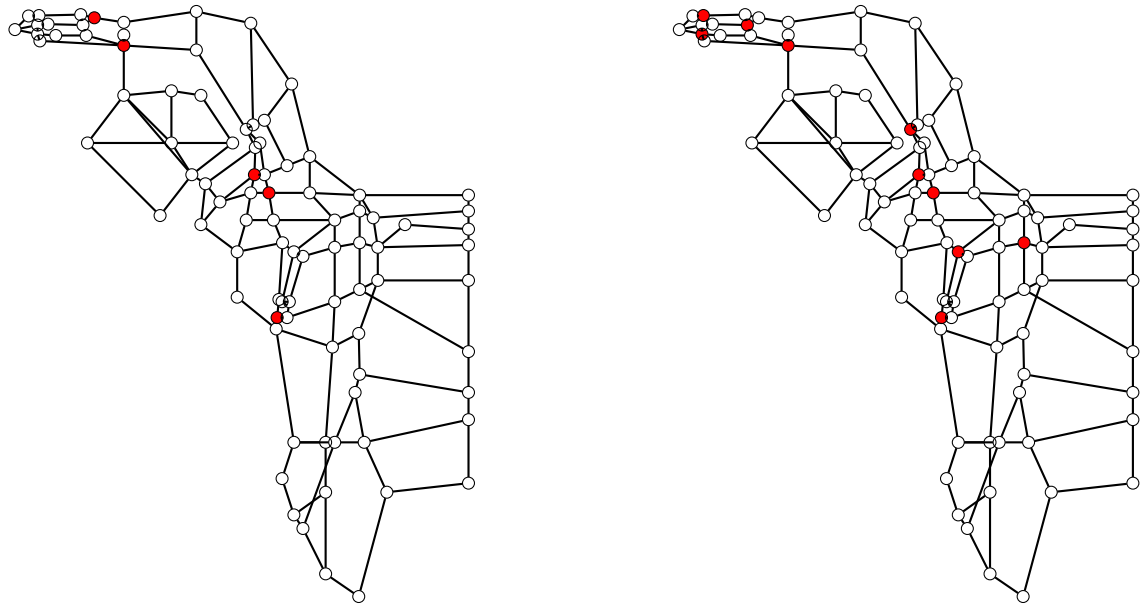


Figure 7 **DCNP solutions for Buffalo network when $\alpha = 0.10$ and the budget is $b \in \{5, 10\}$.**

formulation is the fastest formulation on real-life graphs, often taking a tenth of the time of the recursive formulation, and solving larger instances than were solvable before. A branch-and-cut implementation of the thin formulation is also able to handle instances in which distances are edge-weighted. This enables us to solve road network instances of the

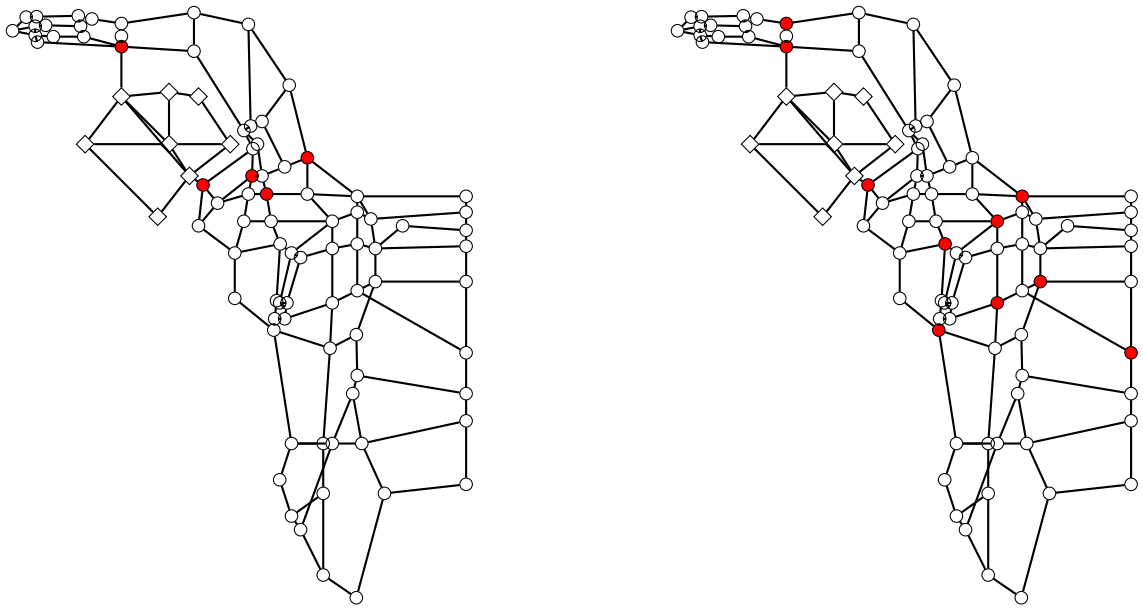


Figure 8 **CNP solutions for Buffalo network when the budget is $b \in \{5, 10\}$.**

distance-based critical node problem; such instances could not have been handled with previous formulations.

We mention a few opportunities for future work. One direction is to further speed up the computations to handle larger instances, perhaps through decomposition methods or through combinatorial bounds (instead of LP bounds). Another direction would be to extend the techniques to other problem variants considered by Veremyev et al. (2015). A last direction is to swap the objective and the constraints, i.e., to delete a minimum number of vertices $|D|$ such that the number of close pairs $|E((G-D)^k)|$ is bounded by a user-specified threshold t.

At a late stage of the review process, we became aware of a subsequent work on the distance-based critical node problem that uses some of our ideas (Alozie et al. 2021).

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1662757. We thank Kelly Sullivan for suggesting Lemma 1 and a supporting proof.

References

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

Addis B, Di Summa M, Grosso A (2013) 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.

Alozie GU, Arulselvan A, Akartunalı K, Pasiliao Jr EL (2021) Efficient methods for the distance-based critical node detection problem in complex networks. *Computers & Operations Research* 131:105254.

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

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

Baier G, Erlebach T, Hall A, Köhler E, Kolman P, Pangrác O, Schilling H, Skutella M (2010) Length-bounded cuts and flows. *ACM Transactions on Algorithms (TALG)* 7(1):1–27.

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

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

Boginski V, Commander CW (2009) Identifying critical nodes in protein-protein interaction networks. *Clustering Challenges in Biological Networks*, 153–167 (World Scientific).

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

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

Buchanan A, Sung JS, Butenko S, Pasiliao EL (2015) An integer programming approach for fault-tolerant connected dominating sets. *INFORMS Journal on Computing* 27(1):178–188.

Carvajal R, Constantino M, Goycoolea M, Vielma JP, Weintraub A (2013) Imposing connectivity constraints in forest planning models. *Operations Research* 61(4):824–836.

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

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

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

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

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

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

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

DIMACS-10 (2017) 10th DIMACS Implementation Challenge - Graph Partitioning and Graph Clustering. `http://www.cc.gatech.edu/dimacs10/downloads.shtml`, accessed: 2020-04-12.

Fischetti M, Leitner M, Ljubić I, Luipersbeck M, Monaci M, Resch M, Salvagnin D, Sinnl M (2017) Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation* 9(2):203–229.

Freeman LC (1977) A set of measures of centrality based on betweenness. *Sociometry* 35–41.

Garey MR, Johnson DS (1979) *Computers and Intractability* (W.H. Freeman and Company).

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

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

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

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

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

Salemi H, Buchanan A (2020) Parsimonious formulations for low-diameter clusters. *Mathematical Programming Computation* 12(3):493–528.

Salemi H, Buchanan A (2021) Implementation of solving the distance-based critical node problem. `https://github.com/halisalemi/DCNP`.

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

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

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

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

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

Validi H, Buchanan A (2020) The optimal design of low-latency virtual backbones. *INFORMS Journal on Computing* 32(4):952–967.

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

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

Walteros JL, Pardalos PM (2012) Selected topics in critical element detection. *Applications of Mathematics and Informatics in Military Science*, 9–26 (Springer).

Walteros JL, Veremyev A, Pardalos PM, Pasiliao EL (2019) Detecting critical node structures on graphs: A mathematical programming approach. *Networks* 73(1):48–88.

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

Online Supplement for "Solving the distance-based critical node problem"

Hosseinali Salemi
Department of Industrial and Manufacturing Systems Engineering, Iowa State University,
Ames, IA 50011, hsalemi@iastate.edu

Austin Buchanan
School of Industrial Engineering & Management, Oklahoma State University,
Stillwater, OK 74078, buchanan@okstate.edu

---

1. Proofs from Section 2

Proof of Lemma 2. The proof is by induction on d. The base case, where $d = 1$, is trivial. For the inductive step, suppose the statement is true for $d = k$; we show it holds for $k + 1$.

($\subseteq$) If $P^{\uparrow_x} = \emptyset$, then the inclusion is trivial. So, suppose that $(\overline{x}, \overline{y}) \in P^{\uparrow_x}$. By definition of $P^{\uparrow_x}$, there exists a point $(\underline{x}, \overline{y}) \in P$ where $\overline{x}_i \geq \underline{x}_i$ for every $i \in [k+1]$. Observe that $(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_k, \underline{x}_{k+1}, \overline{y})$ belongs to the partial dominant of P with respect to $(x_1, x_2, \ldots, x_k)$:

$$P^{\uparrow_{(x_1,x_2,\ldots,x_k)}} := \left\{(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k, x_{k+1}, y) \in \mathbb{R}^{k+1} \times \mathbb{R}^n \;\middle|\; \exists (x, y) \in P : \hat{x}_i \geq x_i \;\forall i \in [k]\right\},$$

because $(\underline{x}, \overline{y}) \in P$ and $\overline{x}_i \geq \underline{x}_i$ for all $i \in [k]$. By the induction assumption,

$$P^{\uparrow_{(x_1,x_2,\ldots,x_k)}} = (((P^{\uparrow_{x_1}})^{\uparrow_{x_2}}) \cdots)^{\uparrow_{x_k}}.$$

Then, $(\overline{x}, \overline{y})$ belongs to the next partial dominant $(((P^{\uparrow_{x_1}})^{\uparrow_{x_2}}) \cdots)^{\uparrow_{x_{k+1}}}$ which equals

$$\left\{(x_1, x_2, \ldots, x_k, \hat{x}_{k+1}, y) \in \mathbb{R}^{k+1} \times \mathbb{R}^n \;\middle|\; \exists (x, y) \in (((P^{\uparrow_{x_1}})^{\uparrow_{x_2}}) \cdots)^{\uparrow_{x_k}} : \hat{x}_{k+1} \geq x_{k+1}\right\},$$

because $(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_k, \underline{x}_{k+1}, \overline{y}) \in (((P^{\uparrow_{x_1}})^{\uparrow_{x_2}}) \cdots)^{\uparrow_{x_k}}$ and $\overline{x}_{k+1} \geq \underline{x}_{k+1}$.

($\supseteq$) If $(((P^{\uparrow_{x_1}})^{\uparrow_{x_2}}) \cdots)^{\uparrow_{x_{k+1}}} = \emptyset$, then the inclusion is trivial. So, suppose that $(\overline{x}, \overline{y}) \in (((P^{\uparrow_{x_1}})^{\uparrow_{x_2}}) \cdots)^{\uparrow_{x_{k+1}}}$. By the induction assumption, $(\overline{x}, \overline{y})$ belongs to the equivalent polyhedron $(P^{\uparrow_{(x_1,x_2,\ldots,x_k)}})^{\uparrow_{x_{k+1}}}$, where $P^{\uparrow_{(x_1,x_2,\ldots,x_k)}}$ is the partial dominant with respect to

$(x_1, x_2, \ldots, x_k)$. By definition of the partial dominant $(P^{\uparrow(x_1,x_2,\ldots,x_k)})^{\uparrow x_{k+1}}$, there is a point $(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_k, \underline{x}_{k+1}, \overline{y}) \in P^{\uparrow(x_1,x_2,\ldots,x_k)}$ with $\overline{x}_{k+1} \geq \underline{x}_{k+1}$. By the definition of $P^{\uparrow(x_1,x_2,\ldots,x_k)}$, this in turn implies another point $(\underline{x}, \overline{y})$ from P with $\overline{x}_i \geq \underline{x}_i$ for all $i \in [k]$. We have thus shown that $(\underline{x}, \overline{y})$ is a point from P that "lies below" $(\overline{x}, \overline{y})$ in the x-space, i.e., $\overline{x}_i \geq \underline{x}_i$ for all $i \in [k+1]$, and thus $(\overline{x}, \overline{y})$ belongs to $P^{\uparrow x}$. □

2. Proofs from Section 3

Proof of Lemma 4. 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 also a minimal length-k u,v-connector. We are to show $\{i,j\} = \{u,v\}$.

In the first case, suppose that the sets $\{i,j\}$ and $\{u,v\}$ have at least one vertex in common. Without loss of generality, suppose that $i = u$. We are to show that $j = v$. Since C is minimal, $G[C]$ is connected. Consider a shortest-path 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-path 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-path 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. □

Proof of Proposition 1. 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

$$\exists z^* \in \{0,1\}^{|C|} \text{ s.t. } (x^*, y^D, z^*) \text{ satisfies (5b)-(5e)} \iff \forall e = \{i,j\} \in E^k, x^*_e = \begin{cases} 1 & \text{if } \text{dist}_{G-D}(i,j) \leq k \\ 0 & \text{if } \text{dist}_{G-D}(i,j) > k. \end{cases}$$

($\implies$) Suppose that there exists a binary vector $z^* \in \{0,1\}^{|\mathcal{C}|}$ such that (x^*, y^D, z^*) satisfies constraints (5b)-(5e). 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 $C \subseteq V \setminus D$. Then, $x_e^* = 1$ by

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

where the first inequality holds by constraints (5d) and the second inequality holds by constraints (5b). 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 (5c). Constraints (5e) then show that $x_e^* \leq \sum_{C \in C_{ij}^k} z_C^* \leq 0$, as desired.

($\impliedby$) Suppose that for each edge $e = \{i,j\} \in E^k$ of the power graph, $x_e^* = 1$ if and only if $\text{dist}_{G-D}(i,j) \leq k$. We construct a binary vector $z^* \in \{0,1\}^{|\mathcal{C}|}$ such that (x^*, y^D, z^*) satisfies constraints (5b)-(5e). For each connector $C \in \mathcal{C}$, define

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

By this definition, (x^*, y^D, z^*) satisfies constraints (5b) and (5c).

Now, we show that each constraint (5d) 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 (5e) 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 $\sum_{C \in C_{ij}^k} z_C^* \geq 1 = x_e^*$, as desired. □

Proof of Lemma 6. If $m \leq k$ then the number of edges (and minimal connectors) is bounded by a constant in which case the lemma easily holds, so we assume that $k < m$. To prove the claim, we construct an injective map f from the set of minimal connectors $\mathcal{C} = \cup_{i,j} C_{ij}^k$ to a set F of appropriate size. By Lemma 5, 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 $q \le k$. For such a connector C, define $f(C)$ as follows

$$f(C) \coloneqq \begin{cases} \left(\emptyset, \left\{\{i,v_1\},\{v_2,v_3\},\ldots,\{v_{q-1},j\}\right\}\right) & \text{if } |C| \ge 2 \text{ is even } (q \text{ odd}) \\ \left(i, \left\{\{v_1,v_2\},\{v_3,v_4\},\ldots,\{v_{q-1},j\}\right\}\right) & \text{if } |C| \ge 3 \text{ is odd } (q \text{ even}) \text{ and } i < j \\ \left(j, \left\{\{i,v_1\},\{v_2,v_3\},\ldots,\{v_{q-2},v_{q-1}\}\right\}\right) & \text{if } |C| \ge 3 \text{ is odd } (q \text{ even}) \text{ and } i > j. \end{cases}$$

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

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

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

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

where the inequality holds because $\binom{a}{b} \le \binom{a}{c}$ holds for all positive integers $b \le c \le \lfloor a/2 \rfloor$. When $k \ge 2$ is even,

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

Proof of Lemma 8. 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 (5b), constraint (5f), and the 0-1 bounds on y and z. To show inequalities (5c) are satisfied, consider a minimal connector $C \in \mathcal{C}$ and vertex $v \in C$ and see that

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

To show each constraint (5d) is satisfied, consider a power graph edge $e = \{i, j\} \in E^k$ and minimal length-k i, j-connector $C \in C^k_{ij}$. In the first case, where $\tilde{z}_C = 0$, constraint (5d) is obviously satisfied. In the other case, where $\tilde{z}_C = 1 - \sum_{c \in C} \hat{y}_c$, constraint (5d) holds by

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

So, by Lemma 7, $(\hat{x}, \hat{y}, \tilde{z})$ belongs to $\text{PATH}^{\uparrow_x}$.

Finally, to show that inequalities (6) 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

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

Proof of Proposition 2. 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

$$(x^*, y^D) \text{ satisfies (7b) and (7c)} \iff \forall e = \{i, j\} \in E^k, x^*_e = \begin{cases} 1 & \text{if } \text{dist}_{G-D}(i, j) \leq k \\ 0 & \text{if } \text{dist}_{G-D}(i, j) > k. \end{cases}$$

($\implies$) Suppose (x^*, y^D) satisfies constraints (7b) and (7c). In the first case, suppose that the endpoints of the power graph edge $e = \{i, j\} \in E^k$ satisfy $\text{dist}_{G-D}(i, j) \leq k$. In this case, there exists a minimal length-k i, j-connector C with $C \subseteq V \setminus D$. Then,

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

where the inequality holds by constraints (7c). So, $x^*_e = 1$, as desired. In the other case, suppose that the endpoints of $e = \{i, j\} \in E^k$ satisfy $\text{dist}_{G-D}(i, j) > k$. In this case, there exists a minimal length-k i, j-separator S that is a subset of D. Then,

$$x^*_e + |S| = x^*_e + \sum_{v \in S} y^D_v \leq |S|,$$

where the inequality holds by constraints (7b). So, $x_e^* = 0$, as desired.

($\Longleftarrow$) Suppose that for each power graph edge $e = \{i,j\} \in E^k$, $x_e^* = 1$ if and only if $\text{dist}_{G-D}(i,j) \leq k$. To show that constraints (7b) are satisfied, consider an edge $e = \{i,j\} \in E^k$ and a minimal length-k i,j-separator $S \in S_{ij}^k$. In the first case, where $x_e^* = 0$, the constraint is obviously satisfied. In the other case, $x_e^* = 1$ and $\text{dist}_{G-D}(i,j) \leq k$. This implies that at least one vertex in S remains intact, i.e., $\sum_{v \in S} y_v^D \leq |S| - 1$. Then, constraint (7b) is satisfied, as

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

To show that constraints (7c) are satisfied, consider an edge $e = \{i,j\} \in E^k$ and a minimal length-k i,j-connector $C \in C_{ij}^k$. In the first case, where $x_e^* = 1$, the constraint is obviously satisfied. In the other case, $x_e^* = 0$ and $\text{dist}_{G-D}(i,j) > k$. Because $\text{dist}_{G-D}(i,j) > k$, D hits every length-k i,j-connector. In particular, D hits C, i.e., $\sum_{v \in C} y_v^D \geq 1$. Then, constraint (7c) is satisfied, as

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

LEMMA 10. *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 of Lemma 10. 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

$$z_C^* \coloneqq \min\left\{x_e^*, 1 - \max_{v \in C} y_v^*\right\},$$

where $e = \{i,j\} \in E^k$ is the unique pair of vertices for which C is a minimal length-k i,j-connector (by Lemma 4). Observe that (x^*, y^*, z^*) satisfies constraint (5f) and 0-1 bounds.

To show that constraints (5b) 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 (7c). Otherwise, $z_C^* = 1 - \max\{y_v^* \mid v \in C\}$, and

$$z_C^* + \sum_{v \in C} y_v^* \geq z_C^* + \max_{v \in C} y_v^* = 1 - \max_{v \in C} y_v^* + \max_{v \in C} y_v^* = 1.$$

To show constraints (5c) are satisfied, consider a connector $C \in \mathcal{C}$ and $v \in C$. 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 (5d) are satisfied, consider a power graph edge $e = \{i, j\} \in E^k$ and a minimal length-k i, j-connector $C \in C_{ij}^k$. Then,

$$x_e^* \geq \min\left\{x_e^*, 1 - \max_{v \in C} y_v^*\right\} = z_C^*.$$

To show constraints (5e) 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 $x_e^* \leq \sum_{C \in C_{ij}^k} z_C^*$ 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, $S \coloneqq \cup_{C \in C_{ij}^k} v_C$ is a length-k i, j-separator because it includes a vertex from each minimal length-k i, j-connector. We claim that S satisfies

$$|S| - \sum_{s \in S} y_s^* \leq |C_{ij}^k| - \sum_{C \in C_{ij}^k} y_{v_C}^*. \tag{1}$$

To show inequality (1), let $q_s = |\{C \in C_{ij}^k \mid v_C = s\}|$ for every separator vertex $s \in S$. This (positive) value q_s is the number of connectors $C \in C_{ij}^k$ for which s is selected as v_C. Then,

$$\sum_{C \in C_{ij}^k} y_{v_C}^* = \sum_{s \in S} q_s y_s^* = \sum_{s \in S} y_s^* + \sum_{s \in S} (q_s - 1) y_s^* \leq \sum_{s \in S} y_s^* + \sum_{s \in S} (q_s - 1) = \sum_{s \in S} y_s^* + |C_{ij}^k| - |S|,$$

where the first equality holds since each $y_{v_C}^*$ appears q_s times in the term $\sum_{C \in C_{ij}^k} y_{v_C}^*$, and the last equality holds since $\sum_{s \in S} q_s$ is equal to the number of minimal connectors. This proves inequality (1). Finally,

$$x_e^* \leq |S| - \sum_{s \in S} y_s^* \leq |C_{ij}^k| - \sum_{C \in C_{ij}^k} y_{v_C}^* = \sum_{C \in C_{ij}^k} (1 - y_{v_C}^*) = \sum_{C \in C_{ij}^k} z_C^*,$$

where the first inequality holds by constraint (7b), and the second inequality holds by inequality (1). So, (x^*, y^*, z^*) satisfies constraints (5e), and thus $(x^*, y^*, z^*) \in \text{PATH}$.

Figure 1 shows the inclusion can be strict for any $k \geq 2$ under hop-based distances. We construct a DCNP instance and point (x^*, y^*, z^*) belonging to PATH but $(x^*, y^*) \notin \text{THIN}$.

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

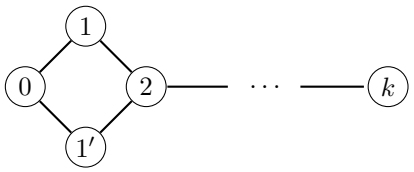


Figure 1 **The graph** $G=(V,E)$ **used to show** $\text{proj}_{x,y}\,\text{PATH} \neq \text{THIN}$.

$x^*_e = \frac{3}{8}$ for all other power graph edges. Finally, let $z^*_C = \frac{3}{8}$ for all connectors $C \in \mathcal{C}$. Observe that the minimal length-k $0,k$-connectors are $C^k_{0,k} = \{\{0,1,2,\ldots,k\},\{0,1',2,\ldots,k\}\}$. Note that $S=\{2\}$ is a length-k $0,k$-separator. While (x^*,y^*,z^*) satisfies the path-like formulation, it violates inequality $x_{e'} + y_2 \leq 1$ of type (7b) from the thin formulation. □

LEMMA 11. $\text{proj}_{x,y}\,\text{R}$ *is incomparable with* $\text{proj}_{x,y}\,\text{PATH}$ *and with* THIN.

Proof of Lemma 11. Since $\text{THIN} \subseteq \text{proj}_{x,y}\,\text{PATH}$ by Lemma 10, it suffices to show that $\text{proj}_{x,y}\,\text{R} \nsubseteq \text{proj}_{x,y}\,\text{PATH}$ and $\text{THIN} \nsubseteq \text{proj}_{x,y}\,\text{R}$.

($\text{proj}_{x,y}\,\text{R} \nsubseteq \text{proj}_{x,y}\,\text{PATH}$). We construct a DCNP instance and 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.

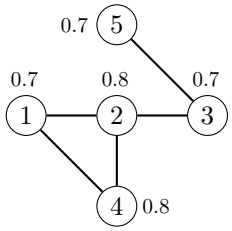


Figure 2 **The graph** $G=(V,E)$ **used to show** $\text{proj}_{x,y}\,\text{R} \nsubseteq \text{proj}_{x,y}\,\text{PATH}$.

Consider the graph in Figure 2. 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

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

with $\hat{u}^s_{ij} = \hat{u}^s_{ji}$ for all i and j. 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 (5d) force $\hat{z}_C \leq \hat{x}_e = 0.3$ and constraints (5e) 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 $(\hat{x},\hat{y},\hat{z}) \in \text{PATH}$.

($\text{THIN} \not\subseteq \text{proj}_{x,y} \text{R}$). 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 3.

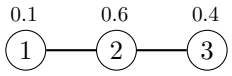


Figure 3 **The graph $G = (V, E)$ used to show** $\text{THIN} \not\subseteq \text{proj}_{x,y} \text{R}$.

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 $(\hat{x}, \hat{y}, \hat{u}) \in \text{R}$. □

Proof of Theorem 2. 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 10.

($\text{proj}_{x,y} \text{PATH}^{\uparrow_x} \subseteq \text{proj}_{x,y} \text{R}^{\uparrow_x}$) 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 8, there is a similar point $(\hat{x}, \hat{y}, \tilde{z}) \in \text{PATH}^{\uparrow_x}$ that satisfies inequalities (6). 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 3, 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 (5b) of the path-like formulation.

To show constraints (1d) hold, 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 $\tilde{z}_C$ in the definition of $\hat{u}^s_{ij}$, see that

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

where the inequality holds by inequality (5c).

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 $\tilde{z}_C$ in the definition of $\hat{u}^s_{ij}$, 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 (6), 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 $\{i, j\} \notin E$, 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 $\tilde{z}_C$ in the definition of $\hat{u}^s_{ij}$. 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 as it 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 (6).

Finally, to show 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 $\tilde{z}_C$ in the definition of $\hat{u}^s_{ij}$, observe that

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

where the inequality holds by constraint (5d) 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 9, 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 (7d) and the 0-1 bounds on $\hat{y}$.

To show constraints (7c) are satisfied, consider a power graph edge $e = \{i,j\} \in E^k$ and 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, $\hat{x}_e \ge 0$ holds for power graph edges $e = \{i,j\} \in E^k$ as $\hat{x}_e \ge \hat{u}^k_{ij} \ge 0$. □

3. Proofs from Section 4

Proof of Proposition 3. Consider a vertex subset $I \subseteq V$ satisfying conditions (i) and (ii), and let $D_0 \subseteq V$ be a feasible solution with $D_0 \cap I \ne \emptyset$. We claim there is a feasible solution $D_1 \subseteq V$ satisfying:

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 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

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

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.

$$\sum_{e \in \delta_1} c_e \leq \sum_{e \in \delta_0} c_e \tag{2}$$

$$\sum_{e \in E_1 \setminus \delta_1} c_e \leq \sum_{e \in E_0 \setminus \delta_0} c_e. \tag{3}$$

The former inequality (2) holds because if $\{i, v\} \in \delta_1$ then $\{u, v\} \in \delta_0$ since if there is a short path from i to v in $G - D_1$, then the same path—but with u substituted for i—exists in $G - D_0$; moreover, $c_{\{i,v\}} \leq c_{\{u,v\}}$ and all connection costs are nonnegative. Meanwhile, the latter inequality (3) 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 $\text{dist}_{G-D_0}(v, w) \leq k$. So, $\{v, w\} \in E_0 \setminus \delta_0$. □

Proof of Proposition 4. 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 (ii), it suffices to check that $\max\{c_e \mid e \in \delta(i)\} \leq \min\{c_{e'} \mid e' \in \delta(u)\}$ for each $u \in N(i)$. Finally, steps 3 and 4 also take linear time $\mathcal{O}(m+n)$. Thus, the total time is $\mathcal{O}(nm)$. This is not too costly given that creating the power graph already takes time $\mathcal{O}(nm)$.

By steps 1 and 2, $I \subseteq S'$ and so every vertex $i \in I$ is simplicial and satisfies condition (ii) of Proposition 3. Moreover, step 3 ensures I is independent. Finally, to prove 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., $|F| \leq p$. For this, observe that each component G' of $G[S]$ is a complete graph. This holds because if G' is not complete, then it has $q := \operatorname{diam}(G') \geq 2$, in which case a diameter-inducing path u_0-u_1-$\cdots$-u_q has a vertex u_1 that is not simplicial in G' and not simplicial in G, a contradiction. So, I is maximum. □

Proof of Proposition 5. First consider the hop-based case. Line 1 takes time $\mathcal{O}(mn)$ by using the algorithm of Brandes (2001). In line 3, we compute the objective value $\text{obj}(D \setminus \{u\})$ for at most $2b$ vertices u, and each function evaluation takes time $\mathcal{O}(mn)$. This means that line 3 takes time $\mathcal{O}(bmn)$. Since there are b iterations of the for loop, lines 2-4 take time $\mathcal{O}(b^2mn)$. When distances are edge-weighted, the analysis is similar, except that the betweenness centrality computations and function evaluations take time $\mathcal{O}(mn + n^2 \log n)$.

□

4. Omitted Tables

Tables 1 and 2 show results of the models R and THIN when $(k,b) = (3,15)$ and $(k,b) = (4,15)$, respectively. Similar to when $b \in \{5,10\}$, THIN outperforms R, by solving all instances that R can solve and 10 others. Specifically, THIN solves all of the instances when $(k,b) = (3,15)$, while R terminates with a gap on 4 instances and experiences a memory crash on 3 others. For a concrete example, observe that R cannot solve instance `SmallWorld` when $(k,b) = (3,15)$, while THIN finishes in 40.75 seconds. Also, while R fails to solve `celegansm` when $(k,b) = (4,15)$, THIN finishes in under 14 minutes.

Table 1 Running times, or the best [lower bound, upper bound] at termination, when $(k, b) = (3, 15)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash.

Graph	n	m	$\lvert E^k \rvert$	opt	R	THIN
karate	34	78	480	0	0.13	**0.08**
dolphins	62	159	1,107	130	4.40	**2.01**
lesmis	77	254	2,500	81	1.41	**0.80**
LindenStrasse	232	303	3,251	756	1.16	**0.47**
polbooks	105	441	3,510	1,028	7.57	**3.20**
adjnoun	112	425	5,634	1,681	62.78	**18.42**
football	115	613	6,247	3,706	[3605,3706]	**1134.48**
netscience	1,589	2,742	13,087	5,832	24.09	**3.12**
jazz	198	2,742	18,461	10,843	[10415,11682]	**3230.26**
SmallWorld	233	994	25,721	3,428	[3346,3428]	**40.75**
Erdos971	429	1,312	34,086	15,247	23.53	**13.28**
S.Cerevisae	1,458	1,948	39,091	16,328	48.44	**20.17**
USAir	332	2,126	46,573	11,546	56.40	**24.65**
power	4,941	6,594	53,125	47,055	515.28	**300.16**
H.Pylori	706	1,392	62,028	21,665	131.37	**16.86**
Harvard500	500	2,043	83,993	6,050	59.25	**17.48**
homer	542	1,619	91,527	15,575	1954.93	**43.05**
celegansm	453	2,025	91,531	13,531	[13476,13531]	**143.95**
email	1,133	5,451	289,259	221,776	1754.64	**244.01**
hep-th	8,361	15,751	376,431	302,530	MEM	**772.16**
PGPgiant	10,680	24,316	1,145,492	664,152	MEM	**3072.69**
cond-mat	16,726	47,594	1,761,969	1,461,085	MEM	**5691.04**

Table 2 Running times, or the best [lower bound, upper bound] at termination, when $(k, b) = (4, 15)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash. A question mark "?" indicates that the optimal objective value for the instance is not known.

Graph	n	m	$\lvert E^k \rvert$	opt	R	THIN
karate	34	78	553	0	0.20	**0.14**
dolphins	62	159	1,459	157	23.05	**8.44**
lesmis	77	254	2,899	81	4.28	**1.49**
LindenStrasse	232	303	7,257	1,069	3.49	**1.23**
polbooks	105	441	4,685	1,424	301.24	**83.65**
adjnoun	112	425	6,178	2,625	2402.56	**1597.36**
football	115	613	6,555	?	[3732,4748]	[3730,4748]
netscience	1,589	2,742	22,847	7,130	115.95	**8.44**
jazz	198	2,742	19,336	?	[11245,14545]	[11250,14524]
SmallWorld	233	994	27,028	4,737	[4426,5428]	**1015.02**
Erdos971	429	1,312	62,059	?	[32068,33530]	[32458,33530]
S.Cerevisae	1,458	1,948	117,958	37,906	84.92	**45.56**
USAir	332	2,126	53,447	14,605	204.19	**118.80**
power	4,941	6,594	105,233	88,312	757.77	**351.47**
H.Pylori	706	1,392	162,758	62,589	488.45	**121.86**
Harvard500	500	2,043	119,080	9,648	708.36	**119.03**
homer	542	1,619	133,947	?	[30917,32208]	[31135,32208]
celegansm	453	2,025	100,476	29,262	[29153,30093]	**800.97**
email	1,133	5,451	548,801	?	LPNS	LPNS
hep-th	8,361	15,751	1,340,125	1,043,763	MEM	**2343.01**
PGPgiant	10,680	24,316	4,211,853	?	MEM	MEM
cond-mat	16,726	47,594	7,586,150	?	MEM	MEM

Tables 3, 4, and 5 provide results for $k = 5$ and $b \in \{5, 10, 15\}$. We compare the R model of Veremyev et al., as well as THIN_I, which is the integer separation variant of the thin model. As one might expect, these instances are more challenging to solve than when $k \in \{3, 4\}$ due to larger number of variables and constraints. For example, observe that for `PGPgiant` and `cond-mat`, there are more than 10 and 22 million variables and unsurprisingly both models face memory issues. However, here again, THIN_I performs better than R. Over all 66 instances of the three tables, the R model solves 23 instances, terminates with gap on 15, fails to solve the root LP for 19, and crashes on 9. Meanwhile, the numbers for THIN_I are uniformly better: 33, 21, 6, and 6, respectively. For a specific example, observe that THIN_I solves `power` with 4,941 nodes and 6,594 edges for all budget values $b \in \{5, 10, 15\}$ under 8 minutes, while R fails on all of them.

Table 3 Running times, or the best [lower bound, upper bound] at termination, when $(k, b) = (5, 5)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash. A question mark "?" indicates that the optimal objective value for the instance is not known.

Graph	n	m	$\lvert E^k \rvert$	opt	R	THIN_I
karate	34	78	561	45	0.47	**0.20**
dolphins	62	159	1,717	771	5.38	**2.07**
lesmis	77	254	2,926	635	58.21	**2.01**
LindenStrasse	232	303	12,768	5,900	65.39	**11.60**
polbooks	105	441	5,316	4,230	1006.12	**407.74**
adjnoun	112	425	6,216	4,850	[4775,4949]	**404.70**
football	115	613	6,555	?	[5620,5995]	[5688,5995]
netscience	1,589	2,742	34,931	15,421	141.89	**14.39**
jazz	198	2,742	19,495	?	[16715,18121]	[16594,18121]
SmallWorld	233	994	27,028	12,689	[12172,12689]	**127.25**
Erdos971	429	1,312	79,891	?	LPNS	[60579,67444]
S.Cerevisae	1,458	1,948	270,717	148,782	574.35	**425.89**
USAir	332	2,126	54,890	34,280	[33930,34280]	**3243.82**
power	4,941	6,594	185,992	169,121	LPNS	**182.15**
H.Pylori	706	1,392	225,276	?	LPNS	[109368,173874]
Harvard500	500	2,043	124,560	62,117	[60057,65505]	**2368.56**
homer	542	1,619	143,902	?	LPNS	[86989,87446]
celegansm	453	2,025	102,196	?	LPNS	[73474,83161]
email	1,133	5,451	631,370	?	LPNS	LPNS
hep-th	8,361	15,751	3,533,314	?	MEM	LPNS
PGPgiant	10,680	24,316	10,744,511	?	MEM	MEM
cond-mat	16,726	47,594	22,860,636	?	MEM	MEM

Table 4 Running times, or the best [lower bound, upper bound] at termination, when $(k, b) = (5, 10)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash. A question mark "?" indicates that the optimal objective value for the instance is not known.

Graph	n	m	$\lvert E^k \rvert$	opt	R	THIN_I
karate	34	78	561	6	0.41	**0.17**
dolphins	62	159	1,717	459	57.23	**9.76**
lesmis	77	254	2,926	180	4.29	**1.56**
LindenStrasse	232	303	12,768	2,889	47.46	**12.61**
polbooks	105	441	5,316	2,160	1427.12	**176.12**
adjnoun	112	425	6,216	3,898	[3455,4007]	**2568.78**
football	115	613	6,555	?	[4664,5460]	[4747,5460]
netscience	1,589	2,742	34,931	10,108	94.94	**13.33**
jazz	198	2,742	19,495	?	LPNS	[13881,16414]
SmallWorld	233	994	27,028	6,363	311.71	**211.67**
Erdos971	429	1,312	79,891	?	LPNS	[40250,58611]
S.Cerevisae	1,458	1,948	270,717	101,678	**479.93**	769.82
USAir	332	2,126	54,890	?	[24031,28649]	[23485,28649]
power	4,941	6,594	185,992	157,108	LPNS	**318.85**
H.Pylori	706	1,392	225,276	?	LPNS	[82441,143178]
Harvard500	500	2,043	124,560	26,709	2396.14	**2149.89**
homer	542	1,619	143,902	?	LPNS	[46396,58639]
celegansm	453	2,025	102,196	?	LPNS	[47865,67719]
email	1,133	5,451	631,370	?	LPNS	LPNS
hep-th	8,361	15,751	3,533,314	?	MEM	LPNS
PGPgiant	10,680	24,316	10,744,511	?	MEM	MEM
cond-mat	16,726	47,594	22,860,636	?	MEM	MEM

Table 5 Running times, or the best [lower bound, upper bound] at termination, when $(k, b) = (5, 15)$. LPNS indicates that the LP relaxation was not solved within one hour, and MEM denotes a memory crash. A question mark "?" indicates that the optimal objective value for the instance is not known.

Graph	n	m	$\lvert E^k \rvert$	opt	R	THIN_I
karate	34	78	561	0	0.23	**0.11**
dolphins	62	159	1,717	166	39.20	**9.40**
lesmis	77	254	2,926	81	7.71	**1.64**
LindenStrasse	232	303	12,768	1,354	6.27	**5.45**
polbooks	105	441	5,316	1,467	1386.05	**122.11**
adjnoun	112	425	6,216	?	[2153,3199]	[2665,2999]
football	115	613	6,555	?	[3540,4939]	[3758,4754]
netscience	1,589	2,742	34,931	7,758	702.98	**13.44**
jazz	198	2,742	19,495	?	[11284,15106]	[11249,15106]
SmallWorld	233	994	27,028	4,974	[4746,5430]	**3222.91**
Erdos971	429	1,312	79,891	?	LPNS	[32045,51062]
S.Cerevisae	1,458	1,948	270,717	78,746	[78489,79097]	**1216.69**
USAir	332	2,126	54,890	16,115	**571.54**	[16013,16115]
power	4,941	6,594	185,992	147,565	LPNS	**454.44**
H.Pylori	706	1,392	225,276	?	LPNS	[62589,113272]
Harvard500	500	2,043	124,560	15,165	[15131,19294]	**3421.44**
homer	542	1,619	143,902	?	LPNS	[30836,44906]
celegansm	453	2,025	102,196	?	LPNS	[29153,46764]
email	1,133	5,451	631,370	?	LPNS	LPNS
hep-th	8,361	15,751	3,533,314	?	MEM	LPNS
PGPgiant	10,680	24,316	10,744,511	?	MEM	MEM
cond-mat	16,726	47,594	22,860,636	?	MEM	MEM

5. Effects of Variable Fixing

To understand the effects of variable fixing, we compare the performance of the R and THIN models under three settings: (i) with no variable fixing, (ii) with leaf fixing, and (iii) with simplicial fixing. Since the fixing procedures do not depend on k nor b, we report results for one set of parameter values, $(k, b) = (3, 5)$.

The results for the THIN model are reported in Table 6. We observe that leaf fixing generally has a negligible impact over no fixing. In most cases, the running times differ by at most three seconds, with a few exceptions: `email` by -10 seconds, `hep-th` by $+22$ seconds, and `PGPgiant` by $+16$ seconds. Thus, the overall performance appears degraded by leaf fixing. We have no explanation for this phenomenon except for solver erraticism (Fischetti and Monaci 2014). The effects of simplicial fixing over no fixing are more pronounced and more uniformly positive, including: `jazz` by -61 seconds, `celegansm` by -36 seconds, `email` by -22 seconds, `hep-th` by -239 seconds, `PGPgiant` by -77 seconds. Lastly, `cond-mat` is solved in 2386 seconds with simplicial fixing, but its LP relaxation is left unsolved after one hour if no fixing is done.

The results for the R model are reported in Table 7. With this model, the effects of fixing on performance appear more erratic. Under leaf fixing, the notable changes in runtime include: `S.Cerevisae` by $+5$ seconds, `power` by $+8$ seconds, `H.pylori` by $+24$ seconds, `Harvard` by -12 seconds, `homer` by -217 seconds, `celegansm` by $+33$ seconds, and `email` by -457 seconds. Overall, there seems to be an (expected) improvement. Under simplicial fixing, the erraticism continues: `power` by $+8$ seconds, `H.pylori` by $+53$ seconds, `Harvard500` by -12 seconds, `homer` by -321 seconds, `celegansm` by -14 seconds, and `email` by -84 seconds. If anything, these results say more about the MIP solver's sensitivity to small changes in the R model rather than about the different fixing procedures.

For our final set of remarks, let us compare the two models under these fixing procedures. Recall that the previous state-of-the-art approach was the R model with leaf fixing. We see that the THIN model (even without fixing) noticeably outperforms it, including speedups of: 234 vs. 87 seconds for `football`, 190 seconds vs. 46 seconds for `power`, 512 seconds vs. 35 seconds for `homer`, 229 seconds vs. 80 seconds for `celegansm`, and 1173 seconds vs. 136 seconds for `email`. There are also three instances where the previous state-of-the-art approach was unable to solve within one hour, but the THIN model (no fixing) could solve: `jazz`, `hep-th`, and `PGPgiant`. So, while the new simplicial fixing procedure is helpful, the THIN model appears to be the biggest source of improvement.

Table 6 Running times of THIN when $(k, b) = (3, 5)$ with no fixing, leaf fixing, and simplicial fixing.

Graph	n	m	$\lvert E^k \rvert$	$\lvert L \rvert$	$\lvert I \rvert$	none	leaf	simplicial
karate	34	78	480	1	12	0.05	0.05	**0.04**
dolphins	62	159	1,107	9	9	0.79	**0.73**	0.74
lesmis	77	254	2,500	17	32	0.25	**0.21**	**0.21**
LindenStrasse	232	303	3,251	88	92	0.24	0.22	**0.21**
polbooks	105	441	3,510	0	4	**2.51**	2.54	2.56
adjnoun	112	425	5,634	10	12	1.04	1.01	**0.97**
football	115	613	6,247	0	0	86.72	86.93	**86.48**
netscience	1,589	2,742	13,087	205	680	1.41	1.48	**1.19**
jazz	198	2,742	18,461	5	14	1142.72	1139.48	**1081.24**
SmallWorld	233	994	25,721	20	64	5.62	5.53	**5.40**
Erdos971	429	1,312	34,086	79	116	11.12	10.92	**9.93**
S.Cerevisae	1,458	1,948	39,091	722	770	6.17	6.04	**5.68**
USAir	332	2,126	46,573	55	122	29.01	28.24	**26.09**
power	4,941	6,594	53,125	1,226	1,414	46.48	46.61	**46.13**
H.pylori	706	1,392	62,028	263	268	9.42	9.14	**8.63**
Harvard500	500	2,043	83,993	79	134	17.62	16.28	**14.54**
homer	542	1,619	91,527	198	289	35.49	34.38	**32.51**
celegansm	453	2,025	91,531	6	95	79.62	76.64	**43.40**
email	1,133	5,451	289,259	151	197	135.99	126.21	**114.46**
hep-th	8,361	15,751	376,431	1,481	3,965	410.79	432.41	**171.94**
PGPgiant	10,680	24,316	1,145,492	4,229	5,299	781.78	798.01	**704.55**
cond-mat	16,726	47,594	1,761,969	1,849	6,695	LPNS	LPNS	**2385.66**

Table 7 Running times of R when $(k, b) = (3, 5)$ with no fixing, leaf fixing, and simplicial fixing.

Graph	n	m	$\lvert E^k \rvert$	$\lvert L \rvert$	$\lvert I \rvert$	none	leaf	simplicial
karate	34	78	480	1	12	0.15	0.16	**0.14**
dolphins	62	159	1,107	9	9	5.31	5.19	**5.22**
lesmis	77	254	2,500	17	32	0.66	0.60	**0.55**
LindenStrasse	232	303	3,251	88	92	0.85	0.76	**0.70**
polbooks	105	441	3,510	0	4	**11.78**	11.82	15.10
adjnoun	112	425	5,634	10	12	2.90	2.76	**2.68**
football	115	613	6,247	0	0	234.83	**234.07**	235.46
netscience	1,589	2,742	13,087	205	680	15.89	15.64	**15.56**
jazz	198	2,742	18,461	5	14	[15716,16185]	[15717,16185]	[15732,16185]
SmallWorld	233	994	25,721	20	64	22.18	**21.79**	21.87
Erdos971	429	1,312	34,086	79	116	19.30	**18.54**	20.39
S.Cerevisae	1,458	1,948	39,091	722	770	**18.73**	23.31	22.19
USAir	332	2,126	46,573	55	122	57.66	57.09	**54.13**
power	4,941	6,594	53,125	1,226	1,414	**180.53**	189.85	188.23
H.pylori	706	1,392	62,028	263	268	**44.40**	68.75	97.60
Harvard500	500	2,043	83,993	79	134	76.05	64.08	**63.97**
homer	542	1,619	91,527	198	289	729.04	511.75	**408.50**
celegansm	453	2,025	91,531	6	95	196.21	229.47	**182.39**
email	1,133	5,451	289,259	151	197	1629.49	**1172.96**	1545.02
hep-th	8,361	15,751	376,431	1,481	3,965	MEM	MEM	MEM
PGPgiant	10,680	24,316	1,145,492	4,229	5,299	MEM	MEM	MEM
cond-mat	16,726	47,594	1,761,969	1,849	6,695	MEM	MEM	MEM

References

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

Fischetti M, Monaci M (2014) Exploiting erraticism in search. *Operations Research* 62(1):114–122.