Parsimonious formulations for low-diameter clusters

Hosseinali Salemi and Austin Buchanan

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

May 20, 2020

Abstract

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

Keywords: k-club; low-diameter; cluster; integer programming; hop constraint; distance constraint; bounded diameter; length-bounded cut;

1 Introduction

Cluster detection is a common problem encountered in network analysis, and an oft-required property of a "good" cluster is that it have a small diameter. Such constraints frequently appear in social network analysis, where a cluster represents a group of people who can quickly communicate with each other. Intuitively, the shorter the communication paths, the more tightly knit the cluster is. In the ideal case, all paths are of length 1, i.e., each member of the cluster can communicate directly with everyone else in the cluster. This results in the concept of a clique in a graph, which is a subset $S \subseteq V$ of vertices that induces a subgraph of diameter at most 1, i.e., $\text{diam}(G[S]) \leq 1$. Relaxing this definition to "induces a subgraph of diameter at most k" yields a k-club.

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

This notion of a k-club was originally introduced in sociology (Alba, 1973; Mokken, 1979), and has found applications in the analysis of biological networks (Balasundaram et al., 2005), as well as in text mining, terrorist networks, and network security (Shahinpour and Butenko, 2013b).

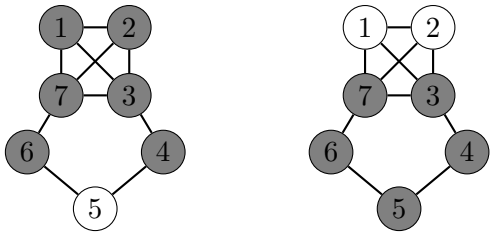

Figure 1: The gray vertices on the right form a 2-club. The gray vertices on the left do not, as vertex 4 and vertex 6 are distance 3 from each other in the induced subgraph.

Essentially the same type of diameter constraint appears in political districting and wildlife reserve design applications where compactness is key.

Effectively enforcing these diameter-at-most-k (i.e., k-club) constraints in an integer program (IP) has proven difficult for researchers. Notable techniques include introducing a binary variable for (the interior of) each path of length at most k (Bourjolly et al., 2002; Wotzlaw, 2014) and linearizing multilinear 0-1 formulations (Veremyev and Boginski, 2012; Veremyev et al., 2015). A naïve implementation of the path-based formulation quickly becomes impractical as the value of k increases, since its number of variables is $\Theta(n^{k-1})$, where n is the number of nodes, under hop-based distances (Wotzlaw, 2014)—or size $\Theta(n^{k+1})$ if the path variables include the path's endpoints (Bourjolly et al., 2002). Any practical approach based on it would require a complicated branch-and-price implementation even for moderate values of k. The linearized multilinear 0-1 formulations of Veremyev et al. (Veremyev and Boginski, 2012; Veremyev et al., 2015) use $\Theta(kn^2)$ variables and typically solve real-life 200-node instances of the maximum k-club problem with $k = 4$ in ten minutes, but routinely fail to solve 300-node instances with $k = 5$ in under one hour (Moradi and Balasundaram, 2018). One limitation of these formulations is that they cannot be used to solve very large instances (with thousands of nodes), as the variables will number in the millions. Also, they only apply when distances are measured in terms of hops (cf. the pseudopolynomial formulation of Veremyev et al. (2015)).

In this paper, we propose new *path-like* and *cut-like* formulations that generalize or improve upon several previously existing formulations for k-clubs. We include "-like" in their names to emphasize that they are not based on the usual notions of paths and cuts. Instead, they are based on length-bounded connectors and length-bounded separators, which are defined below. Examples to illustrate the definitions are given in Figure 2.

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

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

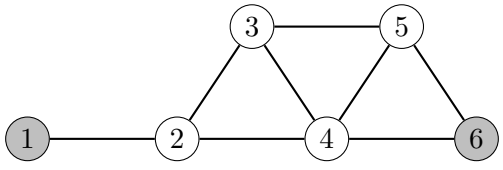

Figure 2: Under hop-based distances, the vertex subset $\{2, 4\}$ is a length-3 $1, 6$-connector, and the vertex subset $\{4\}$ is a length-3 $1, 6$-separator.

The path-like formulation that we propose generalizes the folklore $k = 2$ common neighbor formulation (see constraints (19)), simplifies and dominates the chain formulations of Bourjolly et al. (2002) and Wotzlaw (2014), and generalizes the $k = 3$ neighborhood F_N formulation of Almeida and Carvalho (2012). We observe that, when distances are hop-based and m refers to the number of edges, the path-like formulation has $O(m^{(k-1)/2})$ variables when k is an odd constant and $O(nm^{(k-2)/2})$ variables when k is an even constant. In particular, the formulation has size $O(n+m)$ when $k = 3$ and size $O(nm)$ when $k = 4$. This makes the path-like formulation a reasonable option when $k \leq 4$, distances are hop-based, and the graph is sparse.

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

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

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

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

1.1 Preliminaries

Consider a simple, edge-weighted graph $G = (V, E)$ with vertex set V and edge set $E \subseteq \binom{V}{2}$, where $\binom{V}{2} := \{\{u, v\} \mid u, v \in V,\ u \neq v\}$. We often let $n := |V|$ and $m := |E|$. The weight w_e of each edge $e \in E$ is assumed to be nonnegative. We suppose, without loss of generality, that each edge weight w_e is at most k. (Otherwise, the edge cannot belong to a path of length at most k, thus making it irrelevant for purposes of k-clubs and safe to delete.) The length of a path is the sum of its edges' weights. The distance from node a to node b in graph G is the length of a shortest path from a to b and is denoted $\text{dist}_G(a, b)$. In some cases, distances are "hop-based," meaning that each edge has weight one. The diameter of G is the maximum of these pairwise distances and is denoted

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

The subgraph of G induced by the vertex subset $S \subseteq V$ is denoted by

$$G[S] := \left(S, E \cap \binom{S}{2} \right).$$

For the vertex subset $S \subseteq V$, let $G - S := G[V \setminus S]$ represent the subgraph obtained by removing the vertices of S (and any incident edges). Similarly, for the edge subset $F \subseteq E$, let $G - F := (V, E \setminus F)$. The neighborhood of a vertex v in graph $G = (V, E)$ is denoted $N_G(v) := \{w \in V \mid \{v, w\} \in E\}$.

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

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

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

1.2 Related Work

Previously, we mentioned the integer programming formulations for k-clubs given by Bourjolly et al. (2002) (cf. Wotzlaw (2014)) and Veremyev et al. (Veremyev and Boginski, 2012; Veremyev et al., 2015). Another recent approach by Moradi and Balasundaram (2018), cf. Lu et al. (2018), is to:

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

$$\sum_{i \in V:\ x_i^*=0} x_i + \sum_{i \in V:\ x_i^*=1} (1 - x_i) \geq 1. \tag{2}$$

This is similar in spirit to what we propose, except that our length-k a,b-separator inequalities (1) exploit the problem's structure and are naturally stronger than the canonical hypercube cuts (2) which only cut off x^*.

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

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

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

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

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

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

1.3 Our Contributions

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

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

In Section 3.3, we examine some complexity issues relating to the cut-like formulation. Namely, we observe that the associated separation problem is polynomial-time solvable when $k \in \{2, 3, 4\}$ (if distances are measured in terms of hops) and prove NP-hardness for every $k \geq 5$. Note that Almeida and Carvalho (2012) never addressed the complexity of separation for their F_S formulation (i.e., the $k = 3$ case of our cut-like formulation) and resorted to heuristic separation in their implementation[1]. The polynomiality of separation for the case $k = 4$ is intimately linked with observations of Lovász et al. (1978) on length-bounded cuts. We also remark that these flow-based separation routines immediately lead to polynomial-size (but impractical) extended formulations when $k \in \{3, 4\}$.

In Section 4, we perform computational experiments. They demonstrate the superiority of the cut-like formulation over all other formulations, including the path-like formulation and the compact formulations of Veremyev et al. (Veremyev and Boginski, 2012; Veremyev et al., 2015). In many cases, the differences in running time are dramatic, with the cut-like formulation taking a second or two and the other approaches taking hours. This happens on both real-life and synthetic testbeds that have been considered in the previous literature on the maximum k-club problem. Taking inspiration from Moradi and Balasundaram (2018), we also propose a decomposition procedure to handle the exorbitant number of conflict constraints generated for some of the larger instances. Our code is publicly available (Salemi and Buchanan, 2019).

Finally, we conclude in Section 5.

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

2 The Path-Like Formulation

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

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

$$\max \sum_{i \in V} x_i \tag{3}$$

$$x_a + x_b \leq 1 + \sum_{P \in P^k_{ab}} y_P \qquad \forall \{a, b\} \in \binom{V}{2} \setminus E \tag{4}$$

$$y_P \leq x_i \qquad \forall i \in V(P),\ \forall P \tag{5}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{6}$$

$$y_P \in \{0, 1\} \qquad \forall P. \tag{7}$$

Here, P^k_{ab} is the collection of paths of length at most k between nodes a and b, and $\forall P$ is shorthand for all paths of length at most k. Constraints (4) ensure that, for every pair of nonadjacent vertices a and b in the chosen k-club, there is at least one path between them of length at most k. Constraints (5) ensure that these paths can be crossed only when all of their nodes are selected in the k-club.

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

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

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

$$\max \sum_{i \in V} x_i \tag{8}$$

$$x_a + x_b \leq 1 + \sum_{C \in C^k_{ab}} y_C \qquad \forall \{a, b\} \in \binom{V}{2} \setminus E \tag{9}$$

$$y_C \leq x_i \qquad \forall i \in C,\ \forall C \tag{10}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{11}$$

$$y_C \in \{0, 1\} \qquad \forall C. \tag{12}$$

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

The advantages of defining variables for length-bounded connectors (instead of for paths) include:

- there is no confusion regarding whether the "endpoints" are part of the variable's definition;
- the order in which the vertices are visited in a path is stricken from the variable definition, resulting in fewer variables;
- it allows us to define variables only for *minimal* connectors, which again reduces the formulation's size.

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

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

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

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

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

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

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

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

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

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

and when $k \geq 2$ is even,

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

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

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

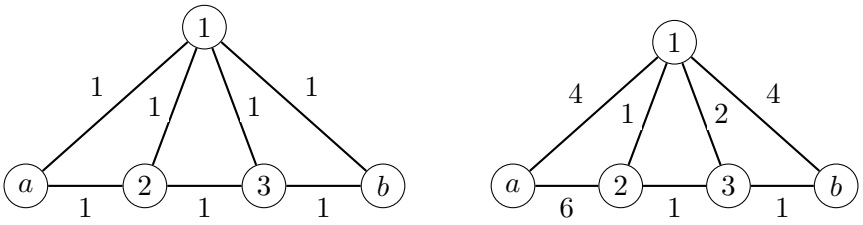

Figure 3: Examples of the collection C^k_{ab} of minimal length-k a, b-connectors. On the left, $C^3_{ab} = \{\{1\}, \{2, 3\}\}$. On the right, $C^7_{ab} = \{\{1, 3\}\}$. The edge weights are given next to the edges.

Remark 1. *Figure 3 shows that if distances are* not *hop-based, a minimal length-k a, b-connector might not induce a path graph.*

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

2.1 The hop-based case $k = 3$

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

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

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

we can write the constraints (9) as

$$x_a + x_b \leq 1 + \sum_{v \in N(a) \cap N(b)} y_{\{v\}} + \sum_{e \in E_{ab}} y_e. \tag{13}$$

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

$$\sum_{i \in C} x_i \leq (|C| - 1) + y_C. \tag{14}$$

Observing that constraints (14) and (10) will force $x_v = y_{\{v\}}$ for each $v \in N(a) \cap N(b)$, the path-like formulation with constraints (14) reduces to[2]:

$$\begin{aligned}
&\max \sum_{i \in V} x_i \\
&x_a + x_b \leq 1 + \sum_{v \in N(a) \cap N(b)} x_v + \sum_{e \in E_{ab}} y_e && \forall \{a, b\} \in \binom{V}{2} \setminus E \\
&x_u + x_v \leq 1 + y_e && \forall e = \{u, v\} \in E \\
&y_e \leq x_v && \forall v \in e,\ \forall e \in E \\
&x_i \in \{0, 1\} && \forall i \in V \\
&y_e \in \{0, 1\} && \forall e \in E.
\end{aligned}$$

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

2.2 The hop-based case $k = 4$

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

$$V_{ij}(a, b) := \{v \in V \mid \operatorname{dist}_G(a, v) = i,\ \operatorname{dist}_G(v, b) = j\}. \tag{15}$$

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

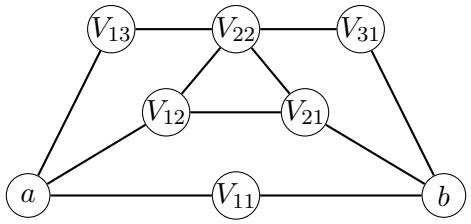

Figure 4: Vertices (and edges) within minimal length-4 a, b-connectors.

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

1. $\{i\}$ where $i \in V_{11}$;
2. $\{p, q\}$ where $p \in V_{12}$, $q \in V_{21}$, $pq \in E$;
3. $\{p, v, q\}$ where $p \in V_{12}$, $v \in V_{22}$, $q \in V_{21}$, $pv \in E$, $vq \in E$, $pq \notin E$;

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

4. $\{p, v, w\}$ where $p \in V_{12}$, $v \in V_{22}$, $w \in V_{31}$, $pv \in E$, $vw \in E$;
5. $\{u, v, q\}$ where $u \in V_{13}$, $v \in V_{22}$, $q \in V_{21}$, $uv \in E$, $vq \in E$;
6. $\{u, v, w\}$ where $u \in V_{13}$, $v \in V_{22}$, $w \in V_{31}$, $uv \in E$, $vw \in E$.

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

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

3 The Cut-Like Formulation

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

Recall that a subset $S \subseteq V \setminus \{a, b\}$ of vertices in a graph $G = (V, E)$ is called a length-k a, b-separator if the distance from node a to node b in the graph $G - S$ is greater than k. In other words, each a, b-path of length $\leq k$ crosses at least one vertex of S. See Figure 2 for an example.

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

$$\max \sum_{i \in V} x_i \tag{16}$$

$$x_a + x_b \leq 1 + x(S) \qquad \forall (a, b, S) \tag{17}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V. \tag{18}$$

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

In the hop-based case $k = 2$, observe that the only minimal length-k a, b-separator is $N(a) \cap N(b)$ so constraints (17) reduce to the folklore $k = 2$ constraints:

$$x_a + x_b \leq 1 + x(N(a) \cap N(b)) \qquad \forall \{a, b\} \in \binom{V}{2} \setminus E. \tag{19}$$

Theorem 2. *The cut-like formulation is correct, even when distances are edge-weighted.*

Proof. We show that the vertex subset $K \subseteq V$ is a k-club if and only if its characteristic vector $x^K \in \{0, 1\}^n$ satisfies all constraints (17).

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

($\Longleftarrow$) By the contrapositive. Suppose that $K \subseteq V$ is not a k-club. That is, there exist vertices $a, b \in K$ such that $\text{dist}_{G[K]}(a, b) > k$. Then, $S := V \setminus K$ is a length-k a, b-separator in G. So, x^K violates the length-k a, b-separator inequality (17) since $x_a^K + x_b^K = 2 > 1 + 0 = 1 + \sum_{i \in S} x_i^K$. $\square$

3.1 Facet Characterization

In this section, we provide the exact conditions under which the length-k a, b-separator inequality (17) induces a facet of the k-club polytope $\mathrm{CLUB}_k(G) = \mathrm{CLUB}_k(G, w)$ of graph G. To our knowledge, this is the first known nontrivial facet of the *edge-weighted* k-club polytope, besides the folklore inequalities mentioned in Proposition 2.

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

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

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

1. $\mathrm{dist}_{G-S}(a, d) > k$ *and* $\mathrm{dist}_{G-S}(b, d) > k$*;*
2. *for every $s \in S$, at least one of the following holds:*
   - *(a)* $\mathrm{dist}_{G_s}(d, s) + \mathrm{dist}_{G_s}(s, a) > k$*;*
   - *(b)* $\mathrm{dist}_{G_s}(d, s) + \mathrm{dist}_{G_s}(s, b) > k$*;*

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

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

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

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

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

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

$$\mathrm{dist}_{G_s}(d, a) \leq \mathrm{dist}_{G_s}(d, s) + \mathrm{dist}_{G_s}(s, a) \tag{20}$$
$$\mathrm{dist}_{G_s}(d, b) \leq \mathrm{dist}_{G_s}(d, s) + \mathrm{dist}_{G_s}(s, b). \tag{21}$$

We claim that these inequalities (20) and (21) hold at equality. Suppose not. Then at least one of them holds strictly; without loss, let it be the former, i.e., $\mathrm{dist}_{G_s}(d, a) < \mathrm{dist}_{G_s}(d, s) + \mathrm{dist}_{G_s}(s, a)$. This implies that $\mathrm{dist}_{G_s}(d, a) = \mathrm{dist}_{G-S}(d, a)$, in which case we arrive at the contradiction

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

Here, the inequality near the middle holds because G_s is a supergraph of $G[K]$, and the last inequality holds because K is a k-club that contains a and d. This shows that inequalities (20) and (21) hold at equality, and so the following holds.

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

This shows that property 2 fails. □

Theorem 3. *The length-k a,b-separator inequality $x_a + x_b \leq 1 + x(S)$ induces a facet of the k-club polytope* $\text{CLUB}_k(G)$ *of G if and only if:*

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

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

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

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

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

We construct n different k-clubs as follows. See Figure 5 for an illustration.

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

  (i) $\text{length}(P_{v_1}) \leq \text{length}(P_{v_2}) \leq \cdots \leq \text{length}(P_{v_q})$, and

  (ii) if $i < j$ and $\text{length}(P_{v_i}) = \text{length}(P_{v_j})$, then $\text{hops}(P_{v_i}) \leq \text{hops}(P_{v_j})$.

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

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

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

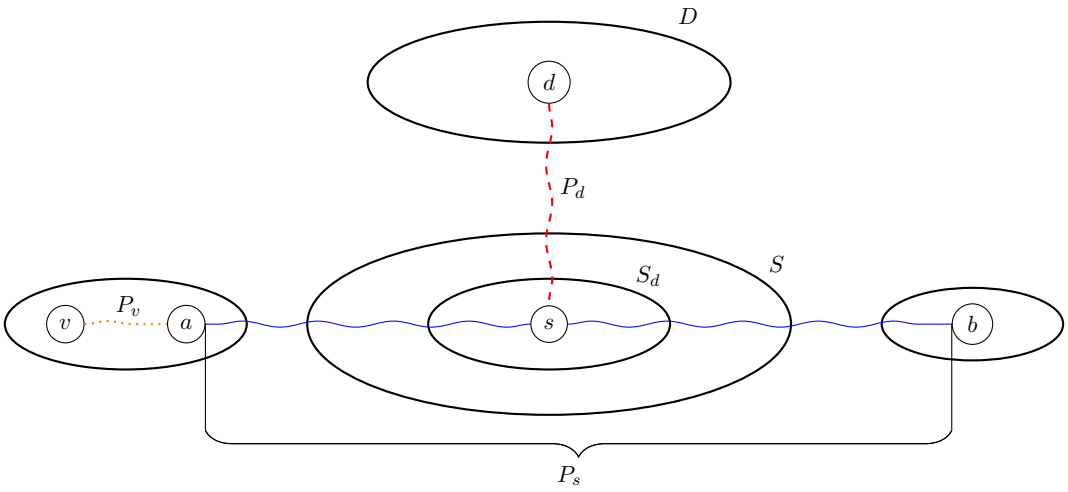

Figure 5: Construction of n different k-clubs in the proof of Theorem 3.

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

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

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

We claim that the characteristic vectors of the aforementioned k-clubs are linearly independent—and thus affinely independent. To see this, it is enough to show that these (column) vectors, arranged from left to right in the order in which they were described, form an upper triangular matrix with ones on the main diagonal, as depicted in Figure 6. (The entries of each column are to be arranged so that the first q entries correspond to $(v_1, v_2, \ldots, v_q)$, the next $|S|$ entries correspond to the vertices of S, and the last $|D|$ entries correspond to $(d_1, d_2, \ldots, d_t)$.) By our construction of the k-clubs, it can be seen that the center submatrix is the identity matrix and that the three submatrices near the lower-left corner are zero matrices. So, all that is left to show is that the upper-left and lower-right submatrices are themselves upper triangular.

To prove that the upper-left submatrix is upper triangular, we are to show that each k-club K_i is a subset of $\{v_1, v_2, \ldots, v_i\}$. Suppose that this is not true for some k-club K_i. Then, K_i contains a vertex $v_j \in V \setminus (S \cup D)$ with $i < j$. By construction of K_i, this implies that vertex v_j is on a shortest v_i-R path in $G - S$, and by the construction of the ordering $(v_1, v_2, \ldots, v_q)$, this implies that $\text{length}(P_{v_i}) \leq \text{length}(P_{v_j})$. Let P'_{v_j} be the subpath of P_{v_i} that starts at v_j and ends at (a vertex of) R. Then,

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

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

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

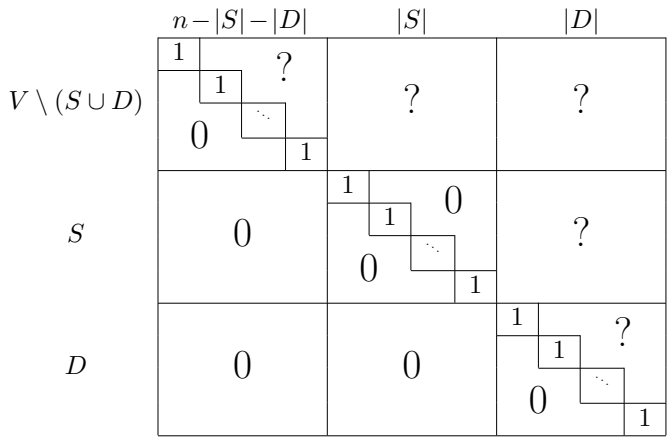

Figure 6: A matrix whose columns represent the k-clubs in the proof of Theorem 3.

which in turn implies that

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

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

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

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

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

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

which in turn implies that

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

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

3.2 Formulation Strength

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

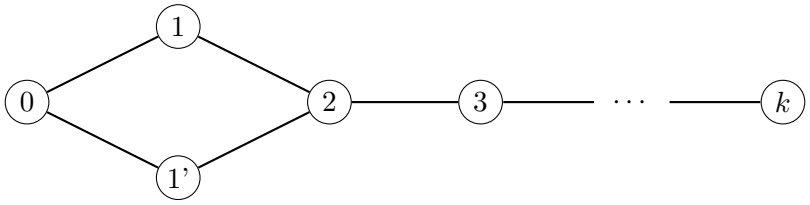

Figure 7: When $k \geq 3$, setting $x_0^* = x_k^* = 1$ and $x_i^* = \frac{1}{2}$ for all other nodes is feasible for the path-like formulation but not for the cut-like formulation. Indeed, the length-k a, b-separator inequality $x_0 + x_k \leq 1 + x_2$ is violated.

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

Proof. First we show that the cut-like formulation is always at least as strong. Suppose that x^* satisfies the LP relaxation of the cut-like formulation. For each minimal length-k a, b-connector $C \in C_{ab}^k$, let i_C be a minimum-weight vertex of C, i.e., $i_C \in C$ and $x_{i_C}^* = \min\{x_i^* \mid i \in C\}$, and set $y_C^* = x_{i_C}^*$. We show that (x^*, y^*) satisfies the LP relaxation of the path-like formulation. Obviously, (x^*, y^*) satisfies the constraints (10) and the 0-1 bounds. So, consider constraint (9) for some nonadjacent vertices a and b. Observe that $S := \cup_{C \in C_{ab}^k} \{i_C\}$ is a length-k a, b-separator, so $x_a^* + x_b^* \leq 1 + \sum_{s \in S} x_s^*$. So, constraint (9) is satisfied as

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

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

The equality of the path-like and cut-like LP relaxations is easy to see in the hop-based case $k = 2$, since the IP formulations themselves are equivalent. Figure 7 shows that the inclusion can be strict when $k \geq 3$. □

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

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

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

($\implies$) By the contrapositive. Suppose there is an independent set I with three vertices. Then, the inequality $x(I) \leq 1$ induces a facet of $\text{CLUB}_k(G[I])$. Lifting this seed inequality shows

that $\text{CLUB}_k(G)$ has a facet-defining inequality of the form $x(I) + \sum_{i \in V \setminus I} \pi_i x_i \leq 1$ with at least three positive coefficients. But, this inequality is not part of the definition of $Q_k(G)$, so $Q_k(G) \neq \text{CLUB}_k(G)$.

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

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

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

3.3 Separation Problem and Extended Formulations

In this section, we discuss the separation problem for the length-k a, b-separator inequalities (17), particularly for the case that distances are measured in hops. Since there are only $O(n^2)$ minimal inequalities when $k = 2$, we will ignore that case. We observe that separation is polynomial-time solvable for $k \in \{3, 4\}$ by reduction to min-cut, and show that separation is hard for each $k \geq 5$. By the equivalence of separation and optimization (Grötschel et al., 1993), we can optimize over the LP relaxations of the cut-like formulation in polynomial time when $k \in \{3, 4\}$, but this is hard when $k \geq 5$.

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

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

Problem: Separation for length-k a, b-separator inequalities (17).
Input: A graph $G = (V, E)$, vertex weights $x^* \in [0, 1]^n$, positive integer k.
Output: (if any exist) nonadjacent vertices $a, b \in V$ and a length-k a, b-separator $S \subseteq V \setminus \{a, b\}$ such that $x^*_a + x^*_b - \sum_{i \in S} x^*_i > 1$.

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

Problem: Length-Bounded Node-Cut.
Input: A graph $G = (V, E)$, nonadjacent nodes p and q, a weight $w_v \geq 0$ for each $v \in V \setminus \{p, q\}$, and a positive integer L.
Output: A vertex subset $S \subseteq V \setminus \{p, q\}$ of minimum weight $\sum_{i \in S} w_i$ such that $\text{dist}_{G-S}(p, q) > L$ (with respect to hop-based distances).

3.3.1 Separation is hard for $k \geq 5$

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

Theorem 4. *For each $k \geq 5$, it is coNP-complete to determine whether a given $x^* \in \mathbb{R}^n$ satisfies all length-k a, b-separator inequalities* (17)*.*

Proof. Membership in coNP is clear, as a suitable witness for a "no" instance is given by nonadjacent nodes a and b and a length-k a, b-separator $S \subseteq V \setminus \{a, b\}$. Consider an instance of Length-Bounded Node-Cut given by graph $G = (V, E)$, nodes p and q, and distance threshold $L \geq 5$ and a target weight W for the cut S. We suppose that the node weights w_v are all equal to one. This variant of Length-Bounded Node-Cut is NP-hard (Baier et al., 2010). Let $k = L$ and $n = |V|$. We construct an $x^* \in [0, 1]^n$ that violates a length-k a, b-separator inequality (17) for G if and only if G has a length-L-bounded node-cut of size W. Namely, let $x^*_p = x^*_q = \frac{2n+1}{4n}$ and $x^*_i = \frac{1}{2n(W+1)}$ for all other nodes i of G.

($\Longleftarrow$) Suppose G has a length-L-bounded node-cut $S' \subseteq V$ of size W. Then x^* violates the inequality (17) for $a = p$, $b = q$, and $S = S'$, since

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

($\Longrightarrow$) Suppose that x^* violates a length-k a, b-separator inequality (17) with length-k a, b-separator $S \subseteq V \setminus \{a, b\}$. Then no vertex of $V \setminus \{p, q\}$ can belong to $\{a, b\}$ since otherwise

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

and the inequality (17) is satisfied. Thus, $\{a, b\} = \{p, q\}$. Then,

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

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

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

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

Thus, to solve the separation problem for the length-k a, b-separator inequalities when given $x^* \in [0, 1]^n$, one can solve, for each pair $\{a, b\}$ of nonadjacent vertices, a Length-Bounded Node-Cut

problem to find a minimum-weight S and then check whether $x^*_a + x^*_b - \sum_{i \in S} x^*_i > 1$. Since there are $\binom{n}{2} - m$ a, b-pairs, the total running time would be $O(mn^3)$.

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

4 Computational Experiments

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

All of our experiments are conducted on a Dell Precision Tower 7000 Series (7810) machine running Windows 10 enterprise, x64, with Intel® Xeon® Processor E52630 v4 (10 cores, 2.2GHz, 3.1GHz Turbo, 2133MHz, 25MB, 85W) – that is 20 logical processors – and 32 GB memory. The IP formulations are implemented in Microsoft Visual Studio 2015 in C++ for Gurobi version 7.5.1. We impose a time limit of 3600 seconds on each instance and set the `method` parameter to concurrent.

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

4.1 Heuristic and Preprocessing

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

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

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

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

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

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

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

Our heuristic:

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

Creating the k-th power graph G^k takes time $O(nm)$ by running BFS from each node. Step 2, which finds a maximal clique in G^k, can be implemented to run in time linear in the size of G^k (Matula and Beck, 1983) and finds large cliques in practice (Walteros and Buchanan, 2019). Step 3, which performs DROP, can take time $O(nm)$ in each of the $O(n)$ iterations of the while loop. This gives a time bound of $O(mn^2)$, which is rather pessimistic given that the number of iterations of the while loop in step 3 is often zero in practice. Moreover, the time bound $O(nm)$ within each iteration is also pessimistic given that when this loop is entered, S is much smaller than n. Indeed, as the results in Table 1 show, the times for our heuristic are very reasonable in practice. The longest time of 22.73 seconds is for the graph `cs4` with $k = 4$. (This is the sum of the heuristic time and the preprocessing time.) For context, our implementation takes 23 seconds to compute the diameter of this graph using BFS..

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

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

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

Table 1: Heuristics and preprocessing on DIMACS-10 graphs. For each k, we report the heuristic's objective (heur), the number of remaining vertices after preprocessing (n'), and the total time in seconds for the heuristic and preprocessing (time).

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

As before, step 1 takes time $O(nm)$ when distances are hop-based. Step 2 takes linear time $O(|V'|+|E'|) = O(n^2)$ with respect to G' by the algorithm of Matula and Beck (1983)[3]. Thus, the total time $O(nm)$ is dominated by step 1.

Table 1 shows that the time spent on this preprocessing is reasonable and the reduction in graph size can be substantial. On these 66 instances, our heuristic finds (what turn out to be) 36 optimal solutions, and the preprocessing is able to prove optimality for 26 of them.

4.2 Implementing the Cut-Like Formulation

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

Initialize the formulation with the conflict constraints $x_a + x_b \leq 1$ for vertices a and b that are far apart in G, i.e., $\text{dist}_G(a, b) > k$. We could instead add stronger cuts of the form $x(I) \leq 1$, where $I \subseteq V$ is an independent set in the k-th power graph G^k. However, MIP solvers detect these stronger cuts automatically and effectively through *clique merging* (Achterberg et al., 2019).

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

Minimalize:

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

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

We experimented with a few alternative implementations but did not pursue them due to relatively poor initial performance. For example, worse performance was observed when adding a single violated inequality (instead of adding one violated inequality for every far pair $\{a, b\}$). Attempts at fractional separation for $k = 3$ were also unsuccessful, although the implementation was rather primitive. It is possible that more advanced implementations would improve the results,

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

e.g., using heuristic or approximate separation (Baier et al., 2010), but we were already content with the performance of lazy integer separation. Experiments with lifting (using Lemma 1 or using another quicker approach) did not improve the performance. These unsuccessful attempts with lifting can be found in the commented portions of the callback code.

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

4.3 Results for Real-Life Instances

In this section, we compare running times on the instances considered by Shahinpour and Butenko (2013a) and Moradi and Balasundaram (2018) that were drawn from the 10th DIMACS Implementation Challenge on Graph Partitioning and Graph Clustering (DIMACS-10, 2017). We exclude all instances that were solved by the heuristic and preprocessing from Section 4.1 as they would provide little insight. To make the comparisons as fair as possible, all experiments are conducted using the same computer, MIP solver, heuristic solution, and preprocessing. We employ the connected component variables z_j in each formulation.

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

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

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

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

Table 3 and Table 4 demonstrate that CUT performs the best among the four formulations. It solves all of the instances that R, CHC, and PATH can solve, plus 7, 5, and 4 others, respectively. CUT's best performance (relative to the other formulations) is on `email` with $k = 4$, where it finishes in 1.82 seconds, while the others struggle. Moreover, formulations R and CHC fail on some small instances such as on the 115-node graph `football` when $k = 3$. Some of the larger instances like `cs4` cause each of the formulations to crash. Investigating further, we find that CUT performs poorly on those instances that require a large number of conflict constraints (more than 2 million), while performing quite well on all other instances (never taking longer than 7 minutes). The number of conflict constraints for each instance is reported in Table 5. These observations motivate the decomposition procedure given next.

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

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

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

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

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

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

Table 5: For each $k \in \{2, 3, 4\}$, we report the number of conflict constraints for each instance after preprocessing.

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

4.3.1 Dealing with Too Many Conflicts

As discussed earlier, we initialize the cut-like formulation with the conflict constraints $x_a + x_b \leq 1$ for vertices a and b that are far apart in G. Sometimes this results in too many conflicts for our computer to handle, as reported in Table 5. For example, `whitaker3`, `crack`, and `cs4` have more than 47 million, 52 million, and 252 million conflicts, respectively, when $k = 3$. This leads to memory crashes (e.g., `cs4`) or difficulties solving the IP (`uk`). In our experiments, instances with more than 2 million conflicts overburden the MIP solver. To handle such instances, we developed the decomposition method given below, which we call ICUT.

ICUT:

1. initialize K to be the k-club found by the heuristic and preprocessing proposed in Section 4.1;
2. compute a minimum-degree ordering $(v_1, v_2, \ldots, v_n)$ of G^k;
3. $T \leftarrow \emptyset$;
4. for $i = n$ down to 1 do
   - $T \leftarrow T \cup \{v_i\}$;
   - $S_i \leftarrow N^k_{G[T]}(v_i)$;
   - if $|S_i| \leq |K|$ continue;
   - $K_i \leftarrow$ a maximum k-club in $G[S_i]$ that contains v_i;
   - if $|K_i| > |K|$ then $K \leftarrow K_i$;
5. return K.

This method iteratively solves small subproblems, much like the Iterative Trim Decomposition Branch-and-Cut (ITDBC) algorithm of Moradi and Balasundaram (2018). However, ICUT uses the cut-like formulation instead of the CHC formulation and defines its subproblems differently. This was motivated by the observation that ITDBC takes time $\Theta(n^2m)$ just to identify the subproblems, which ended up being a bottleneck in our initial experiments. To address this, we borrowed ideas from the maximum clique literature to decompose the maximum k-club problem into n subproblems in time $\Theta(nm)$. In particular, we use a minimum degree ordering, which, as defined by Nagamochi (2010), is a vertex ordering $(v_1, v_2, \ldots, v_n)$ in which vertex v_i has minimum degree in the subgraph induced by $\{v_i, v_{i+1}, \ldots, v_n\}$ for all $i \in [n]$. This vertex ordering is closely related to k-cores and degeneracy orderings, and can be found in linear time by an adaptation of the algorithms of Matula and Beck (1983); Batagelj and Zaversnik (2003) as described, for example, by Walteros and Buchanan (2019). Similar to Moradi and Balasundaram (2018), we terminate a subproblem early (with the Gurobi parameter `Cutoff`) if it has been determined that no solution better than $|K|$ exists.

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

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

Table 6 compares the time to solve the maximum k-club problem for $k \in \{2, 3, 4\}$ using CUT and ICUT. The results show that ICUT outperforms CUT when the instance has a large number

of conflict constraints (more than 2 million). While ICUT solves all instances in under 9 minutes, CUT fails to solve 13 of them. As an example, when $k = 3$, ICUT solves `data` in 5.01 seconds, while CUT cannot solve it to optimality. In a more extreme example, ICUT solves each `cs4` instance in under 2 minutes, while CUT cannot handle the large number of conflicts, crashing on each of them.

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

Table 6: We report the total time in seconds (including preprocessing, heuristic, and model build time), or the best lower and upper bounds [LB,UB] within a 3600 second time limit. Cases where using a formulation leads to memory crashes are reported as MEM. Instances solved to optimality by the heuristic and preprocessing proposed in Section 4.1 are indicated by blank cells.

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

4.4 Results for Synthetic Instances

In Tables 7 and 8, we compare running times on the synthetic instances considered by Veremyev and Boginski (2012) and by Moradi and Balasundaram (2018). These instances were randomly generated by Veremyev and Boginski with 10 graphs at each parameter setting (n, ρ) where n is the number of nodes and ρ is the edge density. We consider the same diameter bounds $k \in \{3, 4, 5, 6, 7\}$ considered by Veremyev and Boginski and by Moradi and Balasundaram. However, for formulation PATH, we only give results for $k \in \{3, 4\}$ due to its prohibitively large size. The heuristic and

preprocessing proposed in Section 4.1 are employed in the following experiments.

Table 7: Results for synthetic graphs with $k \in \{3, 4\}$. For each (n, ρ) we give the average time over the 10 instances. If not all 10 were solved within the 1 hour time limit, we only give the number solved (in parenthesis).

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

Table 8: Results for synthetic graphs with $k \in \{5, 6, 7\}$.

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

As Tables 7 and 8 show, formulation CUT is also the clear winner on synthetic graphs. For example, when $(n, k, \rho) = (300, 5, 1.5\%)$, CUT solves all 10 instances in an average of 1.41 seconds, while the formulations R and CHC solve none of them (within the 1 hour time limit). Even worse, formulations R and CHC fail on instances with as few as 200 and 100 nodes, respectively. In fact, they solve none of the instances with $(n, k, \rho) = (200, 4, 2\%)$.

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

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

5 Conclusion

In this paper, we propose new path-like and cut-like formulations for detecting low-diameter clusters in graphs. They simplify, generalize, and outperform several previous formulations. Indeed, on the testbed of synthetic graphs developed by Veremyev and Boginski (2012), our cut-like formulation for the maximum k-club problem never takes longer than 35 minutes, while previously existing formulations fail to solve 84 instances (or more) in a 1 hour time limit. Similar performance is observed on real-life instances that were considered by Shahinpour and Butenko (2013a) and Moradi and Balasundaram (2018).

We suspect that our cut-like formulation will work well for problems in wildlife reserve design, political districting, and others where compactness is key in a "good" solution. Our implementation is relatively simple and publicly available, allowing these extensions to be done in future work. Our implementation does not use any procedures (e.g., fractional separation routines) that require tuning.

In ongoing work, we study fault-tolerant variants of k-clubs, like the r-robust k-clubs of Veremyev and Boginski (2012) and the h-hereditary k-clubs defined by Pattillo et al. (2013). Indeed, the latter can be formulated by modifying the length-k a, b-separator inequalities to $hx_a + hx_b \leq h + x(S)$. Based on experience here and in the past (Buchanan et al., 2015; Validi and Buchanan, 2019), we expect this to be a practical approach for small values of h.

Another topic for future work is to identify even stronger formulations. Observe that (before clique merging) the LP relaxation of the cut-like formulation (our strongest formulation) has the all-half vector as a feasible solution, regardless of the graph's topology. Can we do better? One possible technique is to start with inequalities of the form $x(I) \leq 1$ where I is an independent set and lift the other vertices. When $k = 2$, this gives the I2DS inequalities of Mahdavi Pajouh et al. (2016), which give all nontrivial facets when the input graph is a forest. Is there a similar, nice set of inequalities for general k? If so, can they be used effectively? We note that Mahdavi Pajouh et al. show empirically that highly-violated I2DS inequalities often exist, but they are NP-hard to separate and have yet to be successfully employed.

Acknowledgements

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

References

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

Tobias Achterberg, Robert E Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 2019.

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

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

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

Yuichi Asahiro, Yuya Doi, Eiji Miyano, Kazuaki Samizo, and Hirotaka Shimizu. Optimal approximation algorithms for maximum distance-bounded subgraph problems. *Algorithmica*, 80(6): 1834–1856, 2018.

Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondřej Pangrác, Heiko Schilling, and Martin Skutella. Length-bounded cuts and flows. *ACM Transactions on Algorithms (TALG)*, 7(1):4, 2010.

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

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

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

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

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

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

Rodolfo Carvajal, Miguel Constantino, Marcos Goycoolea, Juan Pablo Vielma, and Andrés Weintraub. Imposing connectivity constraints in forest planning models. *Operations Research*, 61(4): 824–836, 2013.

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

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

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

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512–530, 2001.

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

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

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

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

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

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

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

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

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

James B Orlin. Max flows in $O(nm)$ time, or better. In *Proceedings of the forty-fifth annual ACM Symposium on Theory of Computing*, pages 765–774. ACM, 2013.

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

Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM Symposium on Theory of Computing*, pages 515–524. ACM, 2013.

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

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

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

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

Mikkel Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, 3(69):330–353, 2004.

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

Alexander Veremyev and Vladimir Boginski. Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European Journal of Operational Research*, 218(2):316–326, 2012.

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

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

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

Yiming Wang, Austin Buchanan, and Sergiy Butenko. On imposing connectivity constraints in integer programs. *Mathematical Programming*, 166(1-2):241–271, 2017.

Andreas Wotzlaw. On solving the maximum k-club problem. *arXiv preprint arXiv:1403.5111*, 2014.

Junming Xu. *Topological structure and analysis of interconnection networks*. Kluwer, 2001.

Appendix – Derivation of Recursive Formulation

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

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

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

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

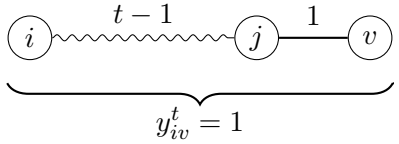

Figure 8: An illustration to explain the variable y^t_{ij}.

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

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

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

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

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

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

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

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

Observe that this constraint can easily be modified to impose different distance requirements de-

pending on i and j. In summary, the formulation is as follows, where $T_{\geq 2} := \{2, \ldots, k\}$.

$$
\begin{aligned}
& x_i + x_j \leq y_{ij}^1 + 1 && (i,j) \in A \\
& y_{ij}^1 \leq x_i && (i,j) \in A \\
& y_{ij}^1 \leq x_j && (i,j) \in A \\
& y_{ij}^{t-1} + x_v \leq y_{iv}^t + 1 && i \in V \setminus \{j, v\},\ (j,v) \in A,\ t \in T_{\geq 2} \\
& y_{iv}^t \leq x_v && i \in V \setminus \{v\},\ v \in V,\ t \in T_{\geq 2} \\
& y_{iv}^t \leq \sum_{j \in N^-(v)} y_{ij}^{t-1} && i \in V \setminus \{v\},\ v \in V,\ t \in T_{\geq 2} \\
& x_i + x_j \leq 1 + \sum_{t=1}^{k} y_{ij}^t && i \in V \setminus \{j\},\ j \in V \\
& x_i \in \{0,1\} && i \in V \\
& y_{ij}^t \in \{0,1\} && i \in V \setminus \{j\},\ j \in V,\ t \in \{1, \ldots, k\}.
\end{aligned}
$$

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

Theorem 5. *The above is a correct formulation for k-clubs in digraphs (under hop-based distances) and has $O(kn^2)$ variables, $O(knm)$ constraints, and $O(knm)$ nonzeros.*

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

This formulation is essentially the k-club formulation introduced by Veremyev and Boginski (2012). However, we feel that it is important to explicitly provide it here–for a number of reasons. One reason is for completeness. A second reason is that there are multiple "recursive" formulations for k-club appearing in the literature and we wanted to point out exactly with which one we compared. For example, in later work, Veremyev et al. (2015) made a small change to the variables' definitions, defining them for paths of length *at most* k, instead of for paths of length *exactly* k. They also strengthened the formulation by disaggregating some big-M constraints, which was found to perform better computationally despite the increase in the number of constraints. (This is perhaps unsurprising given that this improvement in strength came at essentially no cost to the formulation's size measured with respect to the number of nonzeros.) A third reason is that we perform some variable fixing in our implementation and need to describe the formulation in order to clearly explain which variables we fix and why it is safe to do so. Fourth, our explanation of the formulation is perhaps simpler than the original explanation given by Veremyev and Boginski (2012), in part, because we give the direct interpretation of the constraints, instead of arriving at the formulation through a linearization procedure. A fifth reason for including this formulation in the appendix is to have a written companion to our implementation.