ORIGINAL ARTICLE
Networks

Published in Networks
DOI: https://doi.org/10.1002/net.22246

Finding Conserved Low-Diameter Subgraphs in Social and Biological Networks

Hao Pan[1] | Yajun Lu[2] | Balabhaskar Balasundaram[1] | Juan S. Borrero[1]

[1]School of Industrial Engineering and Management, Oklahoma State University, Stillwater, Oklahoma, 74078, USA

[2]Department of Management and Marketing, Jacksonville State University, Jacksonville, Alabama, 36265, USA

Correspondence
Balabhaskar Balasundaram, School of Industrial Engineering and Management, Oklahoma State University, Stillwater, Oklahoma, 74078, USA
Email: baski@okstate.edu

Funding information
The computing for this project was performed at the High Performance Computing Center at Oklahoma State University supported in part through the National Science Foundation grant OAC1531128. The research of Juan S. Borrero has been supported by the National Science Foundation (Award ENG/CMMI # 2145553) and by the Air Force Office of Scientific Research (Award # FA9550-22-1-0236).

The analysis of social and biological networks often involves modeling clusters of interest as *cliques* or their graph-theoretic generalizations. The k-*club* model, which relaxes the requirement of pairwise adjacency in a clique to length-bounded paths inside the cluster, has been used to model cohesive subgroups in social networks and functional modules or complexes in biological networks. However, if the graphs are time-varying, or if they change under different conditions, we may be interested in clusters that preserve their property over time or under changes in conditions. To model such clusters that are conserved in a collection of graphs, we consider a *cross-graph* k-*club* model, a subset of nodes that forms a k-club in every graph in the collection. In this paper, we consider the canonical optimization problem of finding a cross-graph k-club of maximum cardinality in a graph collection. We develop integer programming approaches to solve this problem. Specifically, we introduce strengthened formulations, valid inequalities, and branch-and-cut algorithms based on delayed constraint generation. The results of our computational study indicate the significant benefits of using the approaches we introduce.

KEYWORDS
Cross-graph mining, temporal networks, k-clubs, integer programming

1 | INTRODUCTION

In graph-based data mining (or graph mining), a node models a data item with different attributes, and two nodes are joined by an edge if they are "close" to each other based on similarity measures. Graph mining in social and biological networks involves modeling clusters of interest using cliques and their graph-theoretic generalizations. In these graphs, a cohesive/tight-knit subgroup is a subset of nodes whose members are believed or verified to intimately cooperate with each other towards some specific goal. Cohesive subgroups in social networks could be identified for use in recommender systems, marketing campaigns, community detection, influence maximization, and so forth [3]. In biological networks like protein interaction networks, gene co-expression networks, and metabolic networks, clusters and network motifs are commonly used to identify functional modules that could represent protein complexes, transcriptional modules, or signaling pathways [19]. The clique and its graph-theoretic relaxations have been extensively studied and used as models of cohesive subgroups or clusters in diverse fields including social and biological network analysis [36]. Major categories include the distance based relaxations k-clique and k-club [8], and the edge count, degree, and edge density based relaxations k-defective clique [45], k-plex [7], and quasi-clique [28], respectively.

A significant body of literature on optimization methods for cluster detection seeks to find a subset of nodes satisfying a graph property while optimizing a measure of fitness like cluster size or weight. One common characteristic shared by optimization approaches to graph mining is that they identify cohesive subgraphs, critical nodes, most central actors, or other graph structures of interest in a single graph. However, in many settings the graphs are time-varying as the underlying dynamic systems they are modeling evolve over time. In this case, the single graph under consideration is typically a snapshot that reflects node relationships at the point in time it is recorded, or it aggregates information over a period of time in some manner.

Alternatively, relationships between pairs of nodes (and hence the graph model) may be different under different conditions. Jointly mining the graphs corresponding to different conditions might uncover novel clusters that cannot be found by individually analyzing the network corresponding to each condition. An example in cross-market customer segmentation is finding customers who have similar behaviors across different markets as a more robust cohesive subgroup than those found in a single market [38]. Similarly, systems biologists are interested in finding groups of co-expressing genes or interacting proteins that are conserved under different biological conditions or between different species [37]. These approaches are based on the belief that conserved modules are more likely to govern core biological functions [29, 43].

Broadly, we call the process of simultaneously mining a collection of two or more graphs for conserved structures and patterns as *cross-graph mining*. Despite its potential applications previous work on this topic is limited in the literature. Algorithms for enumerating cross-graph quasi-cliques to extract hidden patterns crossing multiple pieces of data were developed in [37, 38]. This work was extended in [18] for finding frequent cross-graph quasi-cliques, wherein the detected clusters are required to form a quasi-clique in at least a fixed number of graphs in the collection. An approach to clustering stocks that exhibit homogeneous financial ratio values by mining the complete set of cross-graph quasi-bicliques in a bipartite graph was introduced in [44] . This bipartite graph has stocks as nodes in one partition and different features of the stock data in the other partition. The cross-graph quasi-biclique model was used to handle the issue of missing values in stock data. Models and methods for mining conserved clusters in a collection of graphs without strictly imposing the cross-graph requirement can also be found in [9, 16, 17, 41, 48].

In this paper we consider a *cross-graph k-club* model to represent low-diameter clusters that are conserved in a collection of graphs. Note that the graph collection may represent temporal graphs with an implicit ordering, or may be obtained under different (experimental) conditions without any natural ordering. Although our focus is on clusters that induce low-diameter subgraphs, one may investigate any clique relaxation or another graph property in the same setting. Our main contributions in this paper[1] are integer programming (IP) approaches to find a cross-graph k-club of the largest cardinality in a given collection of graphs. Specifically, we introduce strengthened formulations, valid inequalities, and branch-and-cut algorithms based on delayed cut generation that are evaluated on a test bed of instances in our computational study (see also [33]).

The remainder of this paper is organized as follows. We formally define the problem of interest and briefly review IP formulations for the maximum k-club problem in Section 2. In Section 3, we introduce a straightforward conjunctive formulation for the maximum cross-graph k-club problem and propose techniques to strengthen the constraints. Then in Section 4, we discuss valid inequalities for the problem, including extensions of those known in the literature for the single-graph problem. In Section 6, we introduce branch-and-cut algorithms together with preprocessing techniques to solve the problem. We compare the computational results from using the algorithms based on different IP formulations, and verify the computational effectiveness of approaches we developed for the cross-graph k-club problem in solving another related problem in temporal graph mining called the k-club signature problem [6]. We conclude this paper with a summary of our contributions and identify future extensions in Section 7.

2 | BACKGROUND

For a simple graph G, we use $V(G)$ and $E(G)$ to denote its node and edge sets respectively. For simplicity we use uv to denote an edge $\{u, v\} \in E(G)$. For a subset of nodes $S \subseteq V(G)$, we use $G \setminus S$ to denote the graph obtained from G by deleting the nodes in S along with its incident edges and we let $G[S]$ denote the subgraph induced by S (obtained by deleting nodes outside S with the edges incident to these nodes). We use $\overline{G}$ and $N_G(u)$ to denote the complement of graph G and the neighborhood of node u in G, respectively. We use $N_G(u, v) = N_G(u) \cap N_G(v)$ to denote the common neighborhood of u and v in graph G. We denote by $\text{dist}_G(u, v)$ the minimum number of edges on a path connecting nodes u and v in graph G, and its diameter is given by $\text{diam}(G) := \max\{\text{dist}_G(u, v) : u, v \in V(G)\}$.

Definition 1 ([25]) Given a graph G and a positive integer k, a subset of nodes $S \subseteq V(G)$ is called a *k-clique* if $\text{dist}_G(u, v) \le k$ for every pair of nodes $u, v \in S$.

Definition 2 ([2, 30]; see also [8]) Given a graph G and a positive integer k, a subset of nodes $S \subseteq V(G)$ is called a *k-club* if $\text{diam}(G[S]) \le k$.

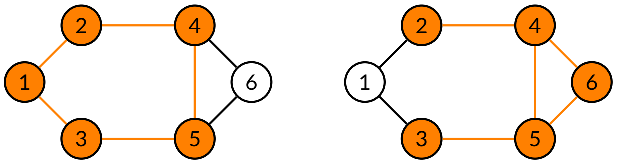

FIGURE 1 The set $\{1, 2, 3, 4, 5\}$ on the left forms a 2-club; the set $\{2, 3, 4, 5, 6\}$ on the right forms a 2-clique, but does not induce a 2-club [2].

A k-clique S allows two nodes u and v to be included even if every path between u and v of length at most k in G includes nodes outside S (see Figure 1). By contrast, in the k-club model at least one of those paths should be contained in $G[S]$. Together, k-cliques and k-clubs are well-known distance-based clique relaxations [42]. The

[1] A preliminary version of this work appeared as [33].

structural guarantees they provide typically determine their suitability for any particular graph mining application. The k-cliques, for instance, are hereditary; that is, the property is preserved under vertex deletion. In contrast, the k-club property is not preserved under node deletion. Nonetheless, the lack of heredity may be acceptable when it is more important to ensure that nodes on at least one of the length-bounded paths is completely contained within the subgraph induced by the club [2]. Since their introduction in social network analysis [49], these distance-based clique relaxations have been used in social and biological network analysis [8, 21, 35], as well as other areas. For low values of parameter k, typically no more than four, the k-club can be an appropriate choice for modeling tightly-knit clusters.

We define the cross-graph counterpart of the k-club, based on the cross-graph quasi-clique model introduced by Pei et al [38], which also appears to be the earliest formal study of a cross-graph model. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_p\}$ denote a collection of p simple, undirected graphs, all defined on a common node set denoted by $V(\mathcal{G})$.

Definition 3 A subset of nodes $S \subseteq V(\mathcal{G})$ is called a *cross-graph k-club* if S is a k-club in each graph in collection $\mathcal{G}$.

This paper focuses on *the maximum cross-graph k-club problem*, which seeks to find a cross-graph k-club of maximum cardinality in $\mathcal{G}$. We use the alternate term "p-graph" k-club if we wish to specify that there are p graphs in the collection. Otherwise, in line with past usage, we simply refer to it as a cross-graph k-club [38]. The (1-graph) maximum k-club problem is NP-hard for every fixed k [11], and remains so on graphs of diameter $k + 1$ [8]. Consequently, the maximum cross-graph k-club problem is NP-hard for every fixed positive integer k as it includes the maximum k-club problem as special case when $\mathcal{G}$ is a singleton. In our previous study on this topic we show that this problem is NP-hard even if $\mathcal{G}$ contains exactly two distinct graphs [32]. Moreover, verifying if a given cross-graph k-club can be strictly enlarged (the complementary problem to verifying maximality by inclusion) is also shown to be NP-complete for a collection containing two distinct graphs [32]. This result extends the analogous result known for (1-graph) 2-clubs to the cross-graph setting [26].

The first IP formulation in the literature for the maximum k-club problem was introduced in [11]. This so-called chain formulation introduces a binary variable for each path of length at most k connecting a nonadjacent pair of nodes. For the special case of $k = 2$, this reduces to the so-called common neighbor formulation for the maximum 2-club problem. As path enumeration gets increasingly challenging as k takes values larger than 2, it can take up to $O(n^{k+1})$ binary variables and constraints to fully describe the chain formulation. To the best of our knowledge, no systematic computational studies have been reported on the chain formulation when $k \geq 3$.

Two polynomial-sized IP formulations, one using binary variables and the other using integer variables were introduced in [46]. Fully described by $O(kn^2)$ variables and constraints, these are the first compact formulations for the maximum k-club problem for general k. A decomposition and branch-and-cut algorithm to find a maximum k-club that employs canonical hypercube cuts as delayed constraints is introduced in [31] (see also [23]). A cut-like formulation and a path-like formulation that use respectively, length-bounded separators and length-bounded connectors are introduced in [39]. The cut-like formulation could use exponentially many constraints, but only n binary variables. The computational superiority of this formulation is demonstrated by the numerical results reported in [39], which makes this the state-of-the-art mathematical programming approach to solve the maximum k-club problem for general k.

3 | INTEGER PROGRAMMING FORMULATIONS

An IP formulation for the maximum cross-graph k-club problem can be obtained by simply taking the conjunction of any IP formulation for the maximum k-club problem over all graphs in the collection. We refer to this straightforward approach as *the conjunctive formulation*. In this section, we first extend the cut-like formulation of the maximum k-club problem [39] to the cross-graph setting through conjunction. We present ideas which strengthen this formulation, and eventually arrive at a new formulation based on a preprocessing procedure that we call *pairwise peeling*. We also identify new valid inequalities for the problem and cross-graph extensions of existing valid inequalities from the literature.

Definition 4 Given a graph G and a pair of nonadjacent nodes u and v, a subset of nodes $S \subseteq V(G) \setminus \{u, v\}$ is called a *length-k u, v-separator* if $\text{dist}_{G \setminus S}(u, v) > k$.

Definition 4 implies that every path of length at most k in G between nodes u and v, uses nodes from S. Let $\mathcal{S}_G(u, v)$ denote the collection of all length-k u, v-separators that are minimal by exclusion. For the case $k = 2$, the unique minimal length-2 u, v-separator is the common neighborhood $N_G(u, v)$.

Formulation (1) that follows is the conjunctive cut-like formulation (CCF) of the maximum cross-graph k-club problem over a collection $\mathcal{G}$. For a subset of nodes $S \subseteq V(\mathcal{G})$, we use the shorthand $x(S) := \sum_{u \in S} x_u$. It is readily verified that x is an incidence vector of a cross-graph k-club if and only if it is feasible to the CCF.

$$\max x(V(\mathcal{G})) \tag{1a}$$

$$\text{s.t.} \quad x_u + x_v - x(S) \leq 1 \quad \forall S \in \mathcal{S}_G(u, v), uv \in E(\overline{G}), G \in \mathcal{G} \tag{1b}$$

$$x_u \in \{0, 1\} \quad \forall u \in V(\mathcal{G}). \tag{1c}$$

Formulation (1) can be strengthened by noting that if a node w that belongs to some minimal length-k u, v-separator of graph $G \in \mathcal{G}$ (i.e., $w \in S \in \mathcal{S}_G(u, v)$) is also at a distance strictly greater than k from either u or v in some other graph $H \in \mathcal{G}$ in the collection, then w cannot be included in a cross-graph k-club that contains both u and v. Consequently, constraints (1b) can be replaced by

$$x_u + x_v - x(S \cap D_{uv}) \leq 1, \tag{2}$$

where D_{uv} is the set of nodes that are at distance at most k from u and v in all the graphs in $\mathcal{G}$, defined as:

$$D_{uv} := \{w \in V(\mathcal{G}) \setminus \{u, v\} : \text{ dist}_G(u, w) \leq k \text{ and } \text{dist}_G(v, w) \leq k \ \forall G \in \mathcal{G}\}.$$

The validity of constraints (2) follows from the validity of (1b) and from the observation that if $x_u = x_v = 1$, then $x(S \setminus D_{uv}) = 0$, because no nodes from the set $S \setminus D_{uv}$ can be included in a cross-graph k-club containing u and v. Alternately, we can think of $S \cap D_{uv}$ as further reducing the size the separator S by removing nodes that are not in any path of length at most k between u and v, in some graph in the collection. Observe that the resulting formulation is at least as tight as the CCF. Moreover, there are instances where $S \cap D_{uv} \subset S$ for at least one separator $S \in \mathcal{S}_G(u, v)$, as illustrated in the following example, which means that there are instances where the resulting formulation is strictly tighter than (1).

Consider the maximum 2-graph 2-club problem on the graph collection in Figure 2. Formulation (1) includes the constraint $x_1 + x_2 - x_3 \leq 1$ due to node pair 1 and 2 in G and constraint $x_1 + x_2 - x_6 \leq 1$ due to the same pair of nodes in H. Note that $\text{dist}_H(1, 3) = 3$. We can therefore tighten the first constraint by intersecting the minimal separator $\{3\}$ with $D_{1,2} = \{5, 6, 7\}$ to obtain the constraint $x_1 + x_2 \leq 1$ that dominates both previous constraints.

Based on the foregoing observations, we can now envision an approach in which we further tighten the constraints with respect to each u, v pair, by *recursively* deleting nodes which are too far away from either u or v in any

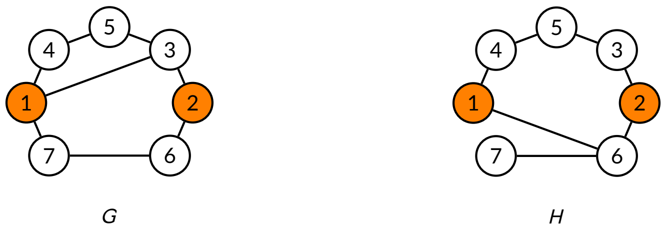

FIGURE 2 Inequality $x_1 + x_2 \leq 1$ is valid for the problem on $\mathcal{G} = \{G, H\}$ when $k = 2$.

graph in the collection. This is a recursive operation because the deletion of nodes can have a domino effect on pairwise distances in graphs, leading to more nodes meeting the condition for deletion. The resulting inequalities will be at least as strong as their counterpart in constraints (2). However, it is important to recognize that this operation is node pair specific, i.e., the graph collection obtained by deleting nodes based on a particular u, v pair is only valid for generating constraints with respect to that pair. This is because nodes deleted based on u and v might be within distance k of a different node pair.

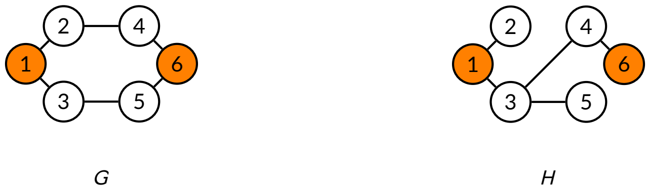

FIGURE 3 Inequality $x_1 + x_6 \leq 1$ is valid for the problem on $\{G, H\}$ when $k = 3$.

To illustrate this idea, consider the maximum 2-graph 3-club problem on the graph collection in Figure 3. Constraints (2) are listed below for the node pair 1 and 6, for graphs G and H, by noting that $D_{1,6} = \{3, 4, 5\}$, $\mathcal{S}_G(1, 6) = \big\{\{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\big\}$, and $\mathcal{S}_H(1, 6) = \big\{\{3\}, \{4\}\big\}$.

$$\begin{aligned}
&x_1 + x_6 - x_3 \leq 1\\
&x_1 + x_6 - x_5 \leq 1\\
&x_1 + x_6 - x_3 - x_4 \leq 1\\
&x_1 + x_6 - x_4 - x_5 \leq 1\\
&x_1 + x_6 - x_3 \leq 1\\
&x_1 + x_6 - x_4 \leq 1
\end{aligned}$$

However, the inequality $x_1 + x_6 \leq 1$ that can replace all of the foregoing constraints for the node pair 1 and 6 can be derived as follows: observe that $\text{dist}_H(2, 6) = 4 > 3$, thus if we want to simultaneously include nodes 1 and 6 in a 2-graph 3-club, then we cannot include node 2 and it can be deleted from G and H. Then, the $\text{dist}_{G\setminus\{2\}}(1, 4) = 4 > 3$, and consequently we cannot include node 4 either. Upon deleting nodes 2 and 4 from G and H, we find that nodes 1 and 6 are disconnected in H; so, $x_1 + x_6 \leq 1$ is valid.

Algorithm 1: Pairwise Peeling

```
Input: $\mathcal{G}, k, uv \in \mathcal{J}$
Output: $\mathcal{G}_{uv}$
1 do
2     $W \leftarrow \emptyset$
3     for $G \in \mathcal{G}$ do
4         for $w \in V(\mathcal{G}) \setminus (W \cup \{u, v\})$ do
5             if $\text{dist}_G(u, w) > k$ or $\text{dist}_G(v, w) > k$ then
6                 $W \leftarrow W \cup \{w\}$
7                 delete $w$ from every graph in $\mathcal{G}$
8 while $W \neq \emptyset$;
9 return $\mathcal{G}_{uv} \leftarrow \mathcal{G}$
```

Algorithm 1 formalizes the idea illustrated by the foregoing example to generate tighter constraints, and we refer to it as the *pairwise peeling algorithm*. Let us denote the node pairs that are nonadjacent in some graph in the collection $\mathcal{G}$ by $\mathcal{J} := \left\{\{u, v\} \subset V(\mathcal{G}) : uv \in E(\overline{G}) \text{ for some } G \in \mathcal{G}\right\}$. The algorithm takes a graph collection $\mathcal{G}$, a positive integer k, and a node pair $uv \in \mathcal{J}$ as input, and creates an auxiliary graph collection $\mathcal{G}_{uv}$ by recursively deleting from every graph in the collection, nodes that are more than distance k from either u or v in some graph in the collection. The constraints for the node pair u and v can then be generated based on the minimal separators of graphs in this auxiliary collection $\mathcal{G}_{uv}$. Thus, we can replace constraints (1b) by the following based on the pairwise peeled collection:

$$x_u + x_v - x(S) \leq 1 \quad \forall S \in \mathcal{S}_G(u, v) \text{ and } G \in \mathcal{G}_{uv} \text{ such that } uv \in E(\overline{G}), uv \in \mathcal{J}. \tag{3}$$

Proposition 1 *Replacing constraints* (1b) *in formulation* (1) *by constraints* (3) *produces a correct formulation for the maximum cross-graph k-club problem.*

The claim follows from the observation that the incidence vector of a cross-graph k-club satisfies constraints (3) and every binary vector satisfying these constraints also satisfies constraints (1b). Furthermore, constraints (2) and (3) coincide when $k = 2$, because the unique minimal length-2 u, v-separator in G is the common neighborhood $N_G(u, v)$. Its intersection with D_{uv} remains undisturbed after pairwise peeling is applied for this node pair, i.e., $N_G(u, v) \cap D_{uv} \in \mathcal{S}_G(u, v)$ for the graph $G \in \mathcal{G}_{uv}$.

Pertinently, given a graph G, a positive integer k, and a (possibly fractional) point $x^* \in [0, 1]^{|V(G)|}$, finding a length-k u, v-separator S for some node pair u, v such that $x^*_u + x^*_v - x^*(S) > 1$ is known to be NP-hard for $k \geq 5$ and is solvable in polynomial-time for $k \in \{2, 3, 4\}$ (see [5, 39]). The case $k = 2$ is straightforward, as the common neighborhood $N_G(u, v)$ is the unique minimal separator. The cases $k \in \{3, 4\}$ require solving the maximum flow problem on an auxiliary network and applying the maximum flow–minimum cut theorem to identify a "violated separator", or conclude that none exists.

Proposition 2 *The pairwise peeling algorithm will delete the same set of nodes independent of the order in which the graphs in $\mathcal{G}$ are processed by the algorithm.*

Proof Suppose for a specific $uv \in \mathcal{J}$, $(w_1, w_2, \ldots, w_q)$ is the order in which nodes were deleted using an ordering π of the graphs in $\mathcal{G}$. Then, w_1 is too far from either u or v in some graph in the original collection, and hence, must be deleted by Algorithm 1 using any other ordering of graphs in $\mathcal{G}$. If w_2 was deleted following w_1 when using π, then in any other ordering, after w_1 is deleted, we know that w_2 must be too far from either u or v, and therefore, must also be deleted. By repeating this argument, $\{w_1, w_2, \ldots, w_q\}$ must be deleted under any ordering that is different from π. As π is arbitrary, we can conclude that the final outcome of Algorithm 1 is independent of the order in which graphs in $\mathcal{G}$ are processed. ■

Henceforth, we refer to this new formulation as the *pairwise peeled cut-like formulation* (PPCF). For each $uv \in \mathcal{J}$, constraint (3) is at least as strong as constraint (2) (which in turn dominates constraint (1b)). In our computational experiments reported in Section 6, we assess the gains made by using Algorithm 1 to generate potentially stronger constraints.

4 | VALID INEQUALITIES

In this section, we introduce a family of valid inequalities for arbitrary k obtained by lifting selected zero coefficient variables in inequality (3) and another for the special case $k = 2$ that extends a result from the literature for the *2-club polytope* [27].

4.1 | Lifted cut-like constraints

We can strengthen constraint (3) by lifting the coefficients of some of the variables under certain conditions, similar to the approach taken in [39]. Consider a pair of nodes u, v for which we have produced a peeled collection $\mathcal{G}_{uv}$. For graphs $G, H \in \mathcal{G}$ (not necessarily distinct), consider a node w with $\text{dist}_G(u, w) > k$ *and* $\text{dist}_H(v, w) > k$. We know that w cannot belong to any minimal length-k u, v-separator in G or H, before the collection is peeled for the pair u, v. After peeling, w will no longer exist in any of the graphs, and therefore cannot belong to $S \in \mathcal{S}_G(u, v)$ for any $G \in \mathcal{G}_{uv}$. We are interested in finding an α_w such that inequality $x_u + x_v + \alpha_w x_w - x(S) \leq 1$ remains valid. Let $\text{XCLUB}_k(\mathcal{G})$ denote the cross-graph k-club polytope of $\mathcal{G}$, i.e., the convex hull of feasible solutions to formulation (1). We need,

$$\alpha_w \leq 1 - \max\{x_u + x_v - x(S) : x \in \text{XCLUB}_k(\mathcal{G}), x_w = 1\} = 1,$$

because $x_u = x_v = 0$ for every feasible x with $x_w = 1$ by our choice of w. We can repeat this argument by lifting another node at distance greater than k from each of the nodes u, v, and w in some graph in the collection, also with coefficient one. We can now generalize this observation to based on the following definition to yield valid inequality (4). Define a subset of nodes $I \subseteq V(\mathcal{G})$ as a *cross-graph distance-k independent set* if every pair of distinct nodes in I are at distance greater than k in some graph in $\mathcal{G}$.

Proposition 3 *Given a collection $\mathcal{G}$, a positive integer k, let $\mathcal{G}_{uv}$ denote the collection peeled for the node pair u, v. Consider a length-k u, v-separator $S \in \mathcal{S}_G(uv)$ for some $G \in \mathcal{G}_{uv}$. Suppose $I \subseteq V(\mathcal{G}) \setminus \{u, v\}$ is a maximal subset (by inclusion of nodes) such that $I \cup \{u\}$ and $I \cup \{v\}$ are cross-graph distance-k independent sets in $\mathcal{G}$. The following inequality is valid for $\text{XCLUB}_k(\mathcal{G})$:*

$$x_u + x_v + x(I) - x(S) \leq 1. \tag{4}$$

If $\text{dist}_G(u, v) > k$ for some graph $G \in \mathcal{G}_{uv}$, the empty set is the unique minimal length-k u, v-separator in G and inequality (4) includes the special case $x_u + x_v + x(I) \leq 1$, where $I \cup \{u, v\}$ forms a maximal cross-graph distance-k independent set.

4.2 | Independent set inequality for cross-graph 2-clubs

Mahdavi Pajouh et al. [27] introduced the following *independent set valid inequality* for the (single graph) 2-club polytope of graph G:

$$x(C) - \sum_{u \in V(G)\setminus C} (|N_G(u) \cap C| - 1)^+ x_u \leq 1, \tag{5}$$

where C is an independent set in graph G and the notation $(t)^+$ denotes the $\max(t, 0)$ given some real number t. Inequality (5) is valid for $\mathrm{XCLUB}_2(\mathcal{G})$ if C is an independent set in some graph $G \in \mathcal{G}$ because it is satisfied by every 2-club in G based on the result in [27], and every cross-graph 2-club of $\mathcal{G}$ is a 2-club in G. The following theorem shows that we can further strengthen this valid inequality for our setting. We use $N^2_{\mathcal{G}}(u)$ to denote the subset of nodes at distance at most two from vertex u in every graph in the collection, i.e., $N^2_{\mathcal{G}}(u) := \{v \in V(\mathcal{G}) : \mathrm{dist}_G(u, v) \leq 2\, \forall G \in \mathcal{G}\}$.

Theorem 4 Given a graph collection $\mathcal{G}$ and a set $C \subset V(\mathcal{G})$ that is independent in some graph $G \in \mathcal{G}$, inequality (6) is valid for $\mathrm{XCLUB}_2(\mathcal{G})$:

$$x(C) - \sum_{u \in V(\mathcal{G})\setminus C} (|N_G(u) \cap C \cap N^2_{\mathcal{G}}(u)| - 1)^+ x_u \leq 1. \tag{6}$$

Proof Let S be an arbitrary cross-graph 2-club of $\mathcal{G}$ and x^S be its incidence vector. It suffices to show that the following inequality holds in order to show that x^S satisfies inequality (6):

$$|C \cap S| - \sum_{u \in S\setminus C} (|N_G(u) \cap C \cap N^2_{\mathcal{G}}(u)| - 1)^+ \leq 1.$$

As $u \in S$ and S is a cross-graph 2-club, we know that $S \subseteq N^2_{\mathcal{G}}(u)$. Therefore,

$$|C \cap S| - \sum_{u \in S\setminus C} (|N_G(u) \cap C \cap N^2_{\mathcal{G}}(u)| - 1)^+ \leq |C \cap S| - \sum_{u \in S\setminus C} (|N_G(u) \cap C \cap S| - 1)^+.$$

Next, we use induction on the cardinality of $C \cap S$ to prove that:

$$|C \cap S| - \sum_{u \in S\setminus C} (|N_G(u) \cap C \cap S| - 1)^+ \leq 1.$$

If $|C \cap S| = 1$, the inequality is trivially true. For some integer $q \geq 2$, we prove the claim for $|C \cap S| = q$, by assuming the claim to hold for all C and S such that $|C \cap S| \leq q - 1$.

Arbitrarily pick a node $a \in C \cap S$ and let $C_a := C \cap S \setminus \{a\}$. Note that $C_a \subset S$ is a nonempty independent set in G. By induction hypothesis,

$$|C_a \cap S| - \sum_{u \in S\setminus C_a} (|N_G(u) \cap C_a \cap S| - 1)^+ \leq 1.$$

We can now rewrite the inequality above as:

$$|C \cap S| - 1 - (|N_G(a) \cap C_a \cap S| - 1)^+ - \sum_{u \in S \setminus C} (|N_G(u) \cap C_a \cap S| - 1)^+ \leq 1, \text{ or}$$
$$|C \cap S| - 1 - \sum_{u \in S \setminus C} (|N_G(u) \cap C_a \cap S| - 1)^+ \leq 1, \tag{7}$$

because node a belongs to the independent set C implying that $N_G(a) \cap C_a = \emptyset$.

Now, consider a node $b \in C_a$. As nodes a and b are contained in the independent set C and the cross-graph 2-club S, $\text{dist}_G(a, b) = 2$ and a common neighbor w of nodes a and b must exist in S and that node w cannot be inside the independent set C. Hence, we know that $w \in S \setminus C$ and that $|N_G(w) \cap C \cap S| = |N_G(w) \cap C_a \cap S| + 1 \geq 2$.

From inequality (7) we obtain,

$$\begin{aligned}
1 &\geq |C \cap S| - 1 - \sum_{u \in S \setminus C} (|N_G(u) \cap C_a \cap S| - 1)^+ \\
&= |C \cap S| - 1 - (|N_G(w) \cap C_a \cap S| - 1) - \sum_{u \in S \setminus (C \cup \{w\})} (|N_G(u) \cap C_a \cap S| - 1)^+ \\
&= |C \cap S| - (|N_G(w) \cap C \cap S| - 1) - \sum_{u \in S \setminus (C \cup \{w\})} (|N_G(u) \cap C_a \cap S| - 1)^+ \\
&\geq |C \cap S| - \sum_{u \in S \setminus C} (|N_G(u) \cap C \cap S| - 1)^+, \text{ establishing our claim.} \quad \blacksquare
\end{aligned}$$

Theorem 4 includes as a special case, the independent set valid inequality (5) for the single-graph 2-club polytope established in [27] by observing that if $\mathcal{G}$ is a singleton, then $N_G(u) \subseteq N^2_{\mathcal{G}}(u)$. The induction approach used offers an alternate proof of that result. Another consequence is that the separation of these more general inequalities is also NP-hard, as inequality (5) is known to be NP-hard to separate [27].

It is also worth noting that our valid inequality (6) dominates inequality (5), which is also valid for $\text{XCLUB}_2(\mathcal{G})$. Consider the two-graph collection $\mathcal{G} = \{G, H\}$ in Figure 2. For the set $C = \{1, 5, 6\}$, which is independent in G, inequality (5) yields $x_1 + x_5 + x_6 - x_3 - x_4 - x_7 \leq 1$, whereas inequality (6) yields $x_1 + x_5 + x_6 - x_4 - x_7 \leq 1$.

Both valid inequalities (4) and (6) we have considered in this section relate to inequalities established in the literature for single-graph k-clubs [27, 39]. These inequalities, under suitable conditions, are also known to induce facets of the 2-club polytope. However, we have not identified non-trivial sufficient conditions that do the same in the cross-graph setting. The primary challenge is with identifying the required number of affinely independent incidence vectors of cross-graph k-clubs that lie on the face of the convex hull induced by our valid inequalities, in order to demonstrate the dimension of that face. In contrast to the single-graph counterpart, the shortest paths that connect the same pair of nodes in a cross-graph k-club can be different in each graph in the collection, making the task of identifying affinely independent feasible solutions very challenging. Identifying facets of $\text{XCLUB}_k(\mathcal{G})$, especially when $k = 2$, is an interesting problem for future study.

5 | DELAYED CONSTRAINT GENERATION

The main goal of our computational study in Section 6 is to compare the performance of a general purpose IP solver when using CCF and PPCF to solve the maximum cross-graph k-club problem. As both formulations use exponentially many constraints in the worst case, we implement them in a delayed fashion in the two decomposition branch-and-

Algorithm 2: Preprocessing

```
   Input: A graph collection $\mathcal{G}$, a positive integer $k \geq 2$
   Output: A preprocessed graph collection $\mathcal{G}$
 1 obtain the intersection graph $J(\mathcal{G})$
 2 compute a $k$-club $S$ of $J(\mathcal{G})$ using the DROP heuristic
 3 do
 4     obtain the power intersection graph $J(\mathcal{G}^k)$
 5     CorePeel($\mathcal{G}, J(\mathcal{G}^k), |S|$)
 6     CommunityPeel($\mathcal{G}, J(\mathcal{G}^k), |S|$)
 7     CrossEdgePeel($\mathcal{G}, J(\mathcal{G}^k)$)
 8 while $\mathcal{G}$ is modified;
 9 return $\mathcal{G}$
```

cut (BC) algorithms that use the same initial root node relaxation based on cross-graph k-cliques. These delayed constraint generation approaches and preprocessing ideas are described in this section.

5.1 | Preprocessing

Before applying the decomposition BC algorithms, we apply extensions of some preprocessing techniques that are known to be effective for the single-graph counterpart to our cross-graph setting [23, 31, 39]. Algorithm 2 describes this preprocessing scheme based on a feasible solution S obtained using the "DROP heuristic" [10] for k-clubs, applied to the *intersection graph* $J(\mathcal{G})$ with node set $V(\mathcal{G})$ and edge set $\bigcap_{G \in \mathcal{G}} E(G)$. Every k-club in $J(\mathcal{G})$ is a cross-graph k-club in $\mathcal{G}$, although the converse is not true.

Peeling based on this cross-graph k-club S is designed to remove nodes and edges from graphs in the collection without affecting any feasible solution of size more than $|S|$. To this end, we first construct the *power intersection graph* of $\mathcal{G}$, denoted by $J(\mathcal{G}^k)$. The node set of $J(\mathcal{G}^k)$ is $V(\mathcal{G})$ and a pair of nodes are made adjacent in $J(\mathcal{G}^k)$ if the distance between them is at most k in every graph in the collection. Finally, we use the observation that every cross-graph k-club (and every cross-graph k-clique defined next) forms a clique of the same size in $J(\mathcal{G}^k)$, allowing us to apply peeling ideas from the maximum clique literature.

Definition 5 Given a graph collection $\mathcal{G}$, a subset of nodes $S \subseteq V(\mathcal{G})$ is called a *cross-graph k-clique* if S is a k-clique in each graph in $\mathcal{G}$.

Once a feasible solution S is available, we implement *core peeling* [1] followed by *community peeling* [47] procedures on $J(\mathcal{G}^k)$; the peeling actions are mirrored on $\mathcal{G}$. If node u has fewer than $|S|$ neighbors in $J(\mathcal{G}^k)$, it cannot belong to a cross-graph k-club larger than S (because if it did, node u would have degree at least $|S|$ in $J(\mathcal{G}^k)$). Core peeling recursively deletes nodes with degree less than $|S|$ in $J(\mathcal{G}^k)$, and also from every graph in $\mathcal{G}$. After core-peeling, $J(\mathcal{G}^k)$ will be an $|S|$-core as long as it is not null. Next, a pair of nodes u and v that are adjacent in $J(\mathcal{G}^k)$ can belong to a cross-graph k-club larger than S only if they have at least $|S| - 1$ common neighbors in $J(\mathcal{G}^k)$. If not, during community peeling step, the edge uv can be deleted from $J(\mathcal{G}^k)$ and from every graph in the collection in which u and v are adjacent.

The power intersection graph $J(\mathcal{G}^k)$ may contain more connected components after core and community peeling than before. As a result, there may exist an edge $uv \in E(G)$ for some $G \in \mathcal{G}$ whose end points u and v belong to different connected components of $J(\mathcal{G}^k)$. The edge uv can be removed from every $G \in \mathcal{G}$ containing the edge. Doing so may disconnect a graph $G \in \mathcal{G}$ so that not only u and v belong to different components, but so do some other nodes a and b that are adjacent in $J(\mathcal{G}^k)$; then, we can delete edge ab from $J(\mathcal{G}^k)$. In other words, during the "cross edge" peeling step, we recursively delete an edge uv from *every* graph in the expanded collection $\mathcal{G} \cup \{J(\mathcal{G}^k)\}$ in which it is present, if u and v are in different connected components of some graph in $\mathcal{G}$ or $J(\mathcal{G}^k)$. When this recursive procedure finishes, every graph in the expanded collection $\mathcal{G} \cup \{J(\mathcal{G}^k)\}$ will have connected components with identical node subsets inducing the components (see also [22] for a similar approach used in a different context). As this may result in changes to the graphs in $\mathcal{G}$, we iterate over these peeling steps until $\mathcal{G}$ no longer changes. Although we chose not to do so, one might also look for a new feasible solution in $J(\mathcal{G})$ before repeating the peeling steps. Next we describe our decomposition BC algorithms as applied to the collection of graphs $\mathcal{G}$ output by Algorithm 2.

5.2 | Initial root node relaxation

Denote by $\mathcal{E}$, the edge set of the complement graph of the power intersection graph of $J(\mathcal{G}^k)$, i.e., $\mathcal{E} := \Big\{\{u, v\} \subseteq V(\mathcal{G}) : \text{dist}_G(u, v) > k \text{ in some graph } G \in \mathcal{G}\Big\}$. Like the single-graph counterparts, a cross-graph k-clique is a graph-theoretic relaxation of a cross-graph k-club. The maximum cross-graph k-clique problem is equivalent to the classical maximum clique problem on $J(\mathcal{G}^k)$ formulated as:

$$\max \left\{x(V(\mathcal{G})) : x_u + x_v \leq 1 \ \forall uv \in \mathcal{E}, x_u \in \{0, 1\} \ \forall u \in V(\mathcal{G})\right\},$$

where x is the incidence vector of cross-graph k-cliques in $\mathcal{G}$. This formulation based on *conflict constraints* serves as the *initial root relaxation* that we start solving in both variants of our decomposition BC algorithms. To avoid having conflict constraints in the initial root relaxation for pairs of nodes that reside in different components of $J(\mathcal{G}^k)$, we extend the initial relaxation by using a binary variable for each connected component of $J(\mathcal{G}^k)$ and enforce that nodes selected must belong to the same component. Let C denote the set of components of $J(\mathcal{G}^k)$. The initial root relaxation problem we use is given in formulation (8).

$$\max \ x(V(\mathcal{G})) \tag{8a}$$
$$\text{s.t.} \quad x_u + x_v \leq 1 \qquad \forall uv \in E(\overline{H}) \text{ and } H \in C \tag{8b}$$
$$y(C) \leq 1 \tag{8c}$$
$$x_u \leq y_H \qquad \forall u \in V(H) \text{ and } H \in C \tag{8d}$$
$$x_u \in \{0, 1\} \qquad \forall u \in V(\mathcal{G}) \tag{8e}$$
$$y_H \in \{0, 1\} \qquad \forall H \in C \tag{8f}$$

Recall that every $G \in \mathcal{G}$ and the graph $J(\mathcal{G}^k)$ have a set of connected components that are induced by the identical node subsets of $V(\mathcal{G})$. Therefore, we could alternatively solve the maximum cross-graph k-club problem on the collection of connected components corresponding to one such identical node subset at a time. We chose to use the extended formulation (8) in order to eliminate from experimental consideration, variations that consider greedy or reverse greedy orderings based on component sizes, and those that iteratively fix a node to be included in the solution permitting us to solve the problem in the k-neighborhood of the fixed node. Although our purpose here is to demonstrate the effectiveness of using one formulation over another in a decomposition BC algorithm, we do recognize that incorporating more ideas from the literature on k-clubs and its variants [15, 20, 24, 31, 39, 40] could potentially improve the effectiveness of our methods.

The two decomposition BC algorithms, henceforth referred to by the underlying formulations CCF and PPCF, would detect a violated constraint (1b) and (3), respectively, whenever an *integral* solution is encountered in the BC tree that corresponds to a cross-graph k-clique that is not a cross-graph k-club. We chose not to separate fractional solutions based on our preliminary experiments that did not indicate noticeable performance gains for our test bed. For the special case $k = 2$, we also separate the independent set valid inequality (6). We discuss our separation procedures next.

5.3 | Separation procedures

Given a graph G, a positive integer k, and a (possibly fractional) point $x^* \in [0,1]^{|V(G)|}$, finding a length-k u, v-separator S in G, for some node pair u, v such that $x^*_u + x^*_v - x^*(S) > 1$ is known to be NP-hard for $k \geq 5$ and is solvable in polynomial-time for $k \in \{2,3,4\}$ (see [5, 39]). The case $k = 2$ is trivial as a unique minimal separator exists in the form of the common neighborhood $N_G(u, v)$. The cases $k \in \{3,4\}$ require a transformation to an auxiliary network on which we need to solve the maximum flow problem. However, we solved the separation problems using a heuristic procedure following the approach taken in [39]. Moreover, we separate constraint (1b) in CCF and constraint (3) in PPCF using Algorithm 3 and Algorithm 4, respectively, only if the BC node linear programming (LP) relaxation optimum x^* is binary. The BC root node initial relaxation (8) ensures that such an x^* corresponds to a cross-graph k-clique. All violated constraints that are detected if x^* is not a cross-graph k-club are added to the lazy-cut pool. The BC node relaxation is re-solved by applying at least some of these cuts, as determined by the solver.

Algorithm 3: CCF Separation Heuristic

```
  Input: $\mathcal{G}, k, x^* \in \{0,1\}^{|V(\mathcal{G})|}$
1 $K \leftarrow \{u \in V(\mathcal{G}) \mid x^*_u = 1\}$            ▷ $K$ is a cross-graph $k$-clique
2 for each $u, v \in K$ and $G \in \mathcal{G}$ do
3     if $\text{dist}_{G[K]}(u, v) > k$ then
4         Apply MINIMALIZE from [39] to the trivial distance-$k$ $u, v$ separator $V(G) \setminus K$ to obtain a minimal
            separator $S$ in $G$
5         add constraint $x_u + x_v - x(S) \leq 1$ violated by $x^*$ to lazy-cut pool
6 return $x^*$ corresponds to a cross-graph $k$-club
```

The separation problem for valid inequality (5) for the maximum (single graph) 2-club problem was shown to be NP-hard, and its exact and heuristic separation was computationally investigated in [27]. The separation problem for valid inequality (6) for the special case $k = 2$ can be formulated as a mixed-integer nonlinear program (MINLP) similar to the single-graph counterpart introduced in [27]. The following MINLP formulation (10) is the starting point for our approach to using them as cutting planes. However, our subsequent linearization uses fewer variables to handle the nonlinear objective compared to the approach used in [27]. Furthermore, based on the computational experience reported in [27], in our experiments, we favor the use of general-purpose mixed-integer linear programming (MILP) rounding heuristics available in the solver rather than attempting exact solution, or using simple greedy combinatorial

Algorithm 4: PPCF Separation Heuristic

```
  Input: $\mathcal{G}, k, x^* \in \{0,1\}^{|V(\mathcal{G})|}$
1 $K \leftarrow \{u \in V(\mathcal{G}) \mid x^*_u = 1\}$                ▷ $K$ is a cross-graph $k$-clique
2 for each $u, v \in K$ and $G \in \mathcal{G}$ do
3     if $\text{dist}_{G[K]}(u, v) > k$ then
4         Apply Pairwise Peeling Algorithm 1 on $\langle$a copy of $\mathcal{G}, k, uv\rangle$ to obtain $\mathcal{G}_{uv}$
5         Let $G' \in \mathcal{G}_{uv}$ correspond to $G \in \mathcal{G}$
6         Apply MINIMALIZE from [39] to the trivial distance-$k$ $u, v$ separator $V(G') \setminus K$ to obtain a minimal
            separator $S$ in $G'$
7         add constraint $x_u + x_v - x(S) \leq 1$ violated by $x^*$ to lazy-cut pool
8 return $x^*$ corresponds to a cross-graph $k$-club
```

heuristics for this separation problem.

Iterating over each graph $G \in \mathcal{G}$, we seek an independent set in G that underlies inequality (6), in order to separate the point $x^* \in [0,1]^{|V(\mathcal{G})|}$. Let the binary variable z_i indicate if node i is selected in the independent set in graph $G \in \mathcal{G}$.

$$\zeta(x^*, G) := \max \sum_{i \in V(\mathcal{G})} x^*_i z_i - \sum_{i \in V(\mathcal{G})} x^*_i (1 - z_i) \left(\sum_{j \in N_G(i) \cap N^2_{\mathcal{G}}(i)} z_j - 1 \right)^+ \tag{9}$$

A violated independent set inequality exists for graph G if and only if $\zeta(x^*, G) > 1$. We can introduce variables w_i to linearize the objective function and obtain the following separation MILP.

$$\zeta(x^*, G) := \max \sum_{i \in V(\mathcal{G})} x^*_i z_i - \sum_{i \in V(\mathcal{G})} x^*_i w_i \tag{10a}$$

$$\text{s.t.} \quad w_i \leq |N_G(i) \cap N^2_{\mathcal{G}}(i)|(1 - z_i) \qquad \forall i \in V(\mathcal{G}) \tag{10b}$$

$$w_i \geq \sum_{j \in N_G(i) \cap N^2_{\mathcal{G}}(i)} z_j - 1 - |N_G(i) \cap N^2_{\mathcal{G}}(i)| z_i \qquad \forall i \in V(\mathcal{G}) \tag{10c}$$

$$w_i \geq 0 \qquad \forall i \in V(\mathcal{G}) \tag{10d}$$

$$z_i + z_j \leq 1 \qquad \forall \{i, j\} \in E(G) \tag{10e}$$

$$z_i \in \{0, 1\} \qquad \forall i \in V(\mathcal{G}) \tag{10f}$$

Rather than attempting to solve the separation MILP (10) to optimality, we utilize it in a heuristic. Our approach, summarized in Algorithm 5, is to solve formulation (10) on each graph $G \in \mathcal{G}$ whenever the LP relaxation optimum x^* at a BC node is binary, with the aim of finding a good feasible solution or fail to find one after $|\mathcal{G}|$ attempts. Hence, we terminate the Gurobi solver early once a feasible solution of objective at least $1 + \epsilon$ is detected or the time limit $\bar{t}$ is reached.

Algorithm 5: Independent Set Inequality Separation Heuristic

```
    Input: $\mathcal{G}, k, x^* \in \{0,1\}^{|V(\mathcal{G})|}$, minimum cut violation $\epsilon$, time limit $\bar{t}$
1   for $G \in \mathcal{G}$ do
2       Find a “good” feasible solution by solving formulation (10) for input $\langle x^*, G \rangle$ with time limit $\bar{t}$ and
          minimum objective target of $1+\epsilon$ to obtain $(z^*, w^*)$          ▷ Feasibility of (10) is guaranteed
3       if objective value at $(z^*, w^*)$ is at least $1+\epsilon$ then
4           return Cutting plane (6) for independent set $C := \{i \in V(G) : z_i^* = 1\}$
```

6 | COMPUTATIONAL STUDY

We report results from our computational experiments conducted on 64-bit Linux® compute nodes with dual Intel® “Skylake" 6130 CPUs with 96 GB RAM. The algorithms are implemented in C++ and the optimization models are solved using Gurobi$^{\text{TM}}$ Optimizer v9.0.1 [14] with a solve time limit of 7200 seconds. The global cut aggressiveness parameter in Gurobi is configured to shut off all general purpose cutting planes in order to ensure that our comparisons are a better representation of the effectiveness of the user-defined cutting planes added. All unspecified Gurobi settings, including rounding heuristics and number of threads are left at their default settings.

In general, we consider the following parameter values in our experiments: $k \in \{2,3,4\}$ and the number of graphs in the collection $p \in \{2,3,4,5\}$. Our test bed is generated from the following three groups of graphs: the Tenth DIMACS Implementation Challenge benchmarks [4] (DIMACS-10 graphs), graphs used in computational studies in [46] (VB graphs), and graphs used in computational studies in [23, 27, 31] (BG graphs). These graphs are commonly used benchmarks for the maximum k-club problem. It is also known that the edge densities of these graphs have a discernible impact on whether or not the instances are challenging for particular values of parameter k [26]. We incorporate this observation by appropriately matching the BG graphs to the value of parameter k for which we solve the maximum cross-graph k-club problem. For DIMACS-10 and VB graphs, we first conduct a set of preliminary experiments to recognize challenging graph and k combinations. Graph collections for our computational experiments are generated from these graphs, and the generation procedure varies by group. In Sections 6.1 and 6.2, we discuss our experimental results by groups of graphs in our test bed, explain the generation procedures, and the selection of challenging instances in greater detail. In Section 6.3, we consider the impact of the independent set inequality on PPCF for the special case of $k=2$. In Section 6.4, we conduct a computational case study on the effectiveness of PPCF on a related problem of finding maximum k-club signature, which can be reduced to solving a series of maximum cross-graph k-club problems. Codes and instances used in our computational experiments are publicly available on GitHub [34].

For most of our experiments, we report results averaged over $11-p$ runs, i.e., on graph collections $\{G_1,\ldots,G_p\}$, $\{G_2,\ldots,G_{p+1}\},\ldots,\{G_{11-p},\ldots,G_{10}\}$. The only exception being the results reported from solving the maximum k-club signature problem in Section 6.4, where we report the largest solution from solving a series of maximum cross-graph k-club problems, following the definition of a signature. We use consistent column headings in all tables reported in this section, with any differences in the club signature results identified in corresponding table notes. We report under column headings “k” and “p” the corresponding parameter values and the graph collection is indicated under the column labeled “Collection”. Under the columns labeled “#Nodes” and “#Edges” we report the number of nodes and edges, respectively, that were removed from the graph collection in the preprocessing step using the pairwise peeling Algorithm 2. Columns labeled “obj” and “time (s)”, respectively report the average optimal objective value and average running time in seconds of the corresponding approach, unless indicated otherwise in the table notes. Columns labeled “#LC” under CCF and PPCF report the average number of lazy constraints added, namely CCF constraint (1b) and PPCF constraint (3), respectively. The average number of branch-and-cut nodes enumerated is reported under the column heading “#BCN”. The columns labeled “#NCT” report the average number of terms with coefficient -1 on the left hand side of the added lazy constraint $x_u + x_v - x(S) \leq 1$, i.e., $|S|$. This is an indirect indicator of the strength of the constraints. Generally, the smaller this number, the stronger the constraint. The extra column labeled “#SLC” (only under PPCF) reports the average number of strengthened PPCF constraints of type (3) added. In order to count the constraints of type (3) that are not of type (1b), we check if the distance-k u,v-separator S in G' obtained in step 6 of Algorithm 4 is *not* a distance-k u,v-separator in any $G \in \mathcal{G}$. That is, we only count under #SLC if for every $G \in \mathcal{G}, \text{dist}_{G \setminus S}(u,v) \leq k$. This implies that such a PPCF contraint could not have been obtained as a CCF constraint from any graph.

6.1 | BG graphs

The BG graphs we use are part of the test bed used in [31], which was generated based on the procedure outlined in [11] (see also [13]). There are four classes of 200-node BG graphs, with 10 samples each from a random generation procedure, designated by their (average) edge density: BG_15 and BG_10 have densities 15% and 10% respectively and are challenging when $k = 2$; BG_5 have densities of 5% and are challenging for $k = 3$; BG_2.5 with densities around 2.5% are challenging for $k = 4$. We report results for the challenging instances in Table 1.

PPCF based branch-and-cut takes, on average, 47.2% less time than CCF for instances solved to optimality. Note that the wall-clock time for PPCF also includes the time spent in computing the statistics reported under column #SLC. PPCF and CCF did not reach optimality for five and six BG_15 instances, respectively, for the case $p = 2$ and $k = 2$, and the statistics for these cases are very similar between the two approaches. For PPCF, over 72% of the lazy constraints added are of the stronger type (3), which could explain the noticeably better running time performance of this approach. For most of the instances, we observe a smaller value under #NCT for PPCF than CCF. Note that the value of #NCT is zero for six groups of instances under PPCF. The lazy constraints of this type are actually conflict constraints with no negative terms on the left hand side. Across all instances solved to optimality, CCF enumerated over 21% more BC nodes on average than PPCF. The foregoing observations strongly suggest that PPCF approach based on pairwise peeling constraints significantly improves our ability to find maximum cross-graph k-clubs on this group of instances.

6.2 | DIMACS-10 and VB graphs

We used 12 graphs from the DIMACS-10 benchmarks, listed in Table 2, each serving as a “seed graph” to generate a corresponding total of 10 graphs. The edge set of each graph in the collection is constructed at random; we start with an empty graph and add edges from the seed graph with a probability of 0.8. These are the same graph collections used in the computational studies reported in [6]. VB graphs are from the test bed used in [46], and all three subclasses in this group contain 10 graphs randomly generated with the same target (average) edge density using the same generation procedure as BG graphs [11, 13]. Each of these graphs has 300 nodes and the three classes are designated as VB_0.5, VB_1.0, and VB_1.5, respectively for their edge densities 0.5%, 1.0%, and 1.5%.

The results from our preliminary experiments to identify challenging collections based on DIMACS-10 and VB graphs for the cross-graph problem are reported in Tables 10–21 in [32]. We solve the collection $\{G_1, \ldots, G_p\}$ (recall that we have 10 graphs corresponding to each subclass inside each group) for each value of parameters k and p of

TABLE 1 Comparison of CCF and PPCF on BG instances.

			Reduction by Peel[g]		CCF					PPCF					
k	p	Collection	#Nodes	#Edges	obj	time (s)	#LC[a]	#NCT[b]	#BCN[c]	obj	time (s)	#LC[a]	#SLC[d]	#NCT[b]	#BCN[c]
2	2	BG_10	0	0	9.4	102.8	12,043	2.3	29,276	9.4	79.6	11,241	9,421	0.7	16,411
2	2	BG_15	0	0	16.7[e]	175.2%[e]	17,334	4.0	6,301,030	16.4[f]	172.0%[f]	17,377	1,291	3.9	6,321,129
2	3	BG_10	0	0	4.6	56.1	24,381	2.3	14,768	4.6	38.7	25,076	25,056	0.0	15,293
2	3	BG_15	0	0	7.9	948.5	25,544	4.1	337,562	7.9	977.2	25,224	4,241	3.9	343,897
2	4	BG_10	0	0	2.0	96.9	38,205	2.3	15,475	2.0	32.3	38,316	38,316	0.0	16,418
2	4	BG_15	0	0	4.4	453.1	43,709	4.3	75,002	4.4	388.0	43,698	12,654	3.9	58,895
2	5	BG_10	0	0	1.2	92.9	44,370	2.3	14,099	1.2	32.4	44,424	44,424	0.0	14,362
2	5	BG_15	0	0	2.3	352.2	64,949	4.3	48,562	2.3	296.2	64,846	29,274	3.4	36,145
3	2	BG_5	0	0	12.0	725.7	23,617	4.0	244,328	12.0	547.8	23,343	6,130	3.6	194,812
3	3	BG_5	0	0	2.4	256.0	51,806	3.9	44,706	2.4	135.5	51,537	34,877	2.0	31,791
3	4	BG_5	0	0	1.0	290.9	71,580	3.9	43,299	1.0	54.5	71,639	65,718	0.6	31,516
3	5	BG_5	0	0	1.0	310.1	88,124	4.0	41,397	1.0	33.0	88,152	87,022	0.1	24,751
4	2	BG_2.5	5	31	8.8	113.7	19,605	3.2	19,287	8.8	60.5	17,094	14,920	0.9	16,069
4	3	BG_2.5	0	85	1.1	142.9	37,352	2.8	19,743	1.1	48.2	37,175	37,150	0.0	18,930
4	4	BG_2.5	0	160	1.0	95.8	41,928	2.8	20,527	1.0	35.6	41,928	41,928	0.0	17,332
4	5	BG_2.5	0	258	1.0	88.6	43,211	2.7	12,490	1.0	20.2	43,247	43,247	0.0	12,127

[a] Average number of lazy constraints added.
[b] Average number of negative terms in a lazy constraint.
[c] Average number of branch-and-cut tree nodes.
[d] Average number of lazy constraints added that were strictly strengthened by pairwise peeling.
[e] Average MILP gap over 6 out of 9 instances that were not solved to optimality and were terminated when the time limit was reached. Here we report the average of the best solutions found for all 9 instances under the obj columns. The average running time is 6,102.8 seconds over 3 out of 9 instances that were solved to optimality.
[f] Average MILP gap over 5 out of 9 instances that were not solved to optimality and were terminated when the time limit was reached. Here we report the average of the best solutions found for all 9 instances under the obj columns. The average running time is 6,069.4 seconds over 4 out of 9 instances that were solved to optimality.
[g] Peeling was not very effective on BG instances.

interest using both algorithms. We observe that when solving most of these instances there are very few (sometimes zero) lazy constraints added by *both* CCF and PPCF. If the initial relaxation is practically sufficient to solve the problem using both approaches, we consider these instances not to be sufficiently challenging for the problem, and therefore no meaningful distinction can be made between the performance of the two algorithms. Based on our preliminary experiments, we only include those instances that required over 100 lazy constraints using *either* CCF or PPCF. We rerun the two BC algorithms and report results averaged over $11 - p$ runs (as described before) in Table 3. As a larger number of lazy constraints are needed to solve these instances, the benefits of using PPCF over CCF is also observed in the results in Table 3. Across this test bed, on average, PPCF is 12.8% faster and over 33% of the lazy constraints added by PPCF are the stronger type (3) constraints.

6.3 | PPCF with independent set inequality for cross-graph 2-clubs

In this section, we report on our experiment adding the independent set inequality (6) to our PPCF method for the special case of cross-graph 2-clubs and assess its performance on the BG instances, which are among the more challenging instances in our test bed. In our experiments, we set the minimum constraint violation parameter $\epsilon = 0.5$ with a time limit of $\bar{t} = 30$ seconds for each G. As we only separate binary points, constraint violation will always be

TABLE 2 DIMACS-10 seed graphs used in generating graph collections.

G	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$	Edge Density (%)
karate	34	78	13.90
lesmis	77	254	8.68
polbooks	105	441	8.08
adjnoun	112	425	6.84
football	115	613	9.35
celegans	453	2,025	1.98
email	1,133	5,451	0.85
polblogs	1,490	16,715	1.51
netscience	1,589	2,742	0.22
power	4,941	6,594	0.05
hep-th	8,361	15,751	0.05
PGPgiantcompo	10,680	24,316	0.04

a positive integer (within numerical tolerance). We also apply the PPCF separation Algorithm 4 to generate violated PPCF constraints following the attempt at finding a violated independent set inequality using Algorithm 5 to ensure the overall correctness of our algorithm. MILP formulation (10) is also incrementally updated before it is solved as the integral point x^* being separated only influences the objective function of this MILP. The results are reported in Table 4.

Although we expected these cuts to improve overall performance for $k = 2$ for the challenging BG instances in our test bed, we observed a deterioration in performance in terms of average running time/optimality gap and tree size. We observed similar performance losses for other values of ϵ and termination time limit. It also appears based on the numbers reported under the column labeled #ISLC that a relatively small number of violated independent set cuts were found. The average ratio of #ISLC/#LC is just 3.3%. Adding only the PPCF constraints and letting the tree enumerate appears to be the better choice in our experimental set up. However, it is possible that an entirely different approach to adding these cutting planes could lead to better performance. For instance, we could attempt aggressive fractional separation at the root, adding a round of cutting planes simultaneously by generating one for each graph, and/or adding these cutting planes only at the top levels of the tree. These are directions worth exploring for the special case of $k = 2$. Even for the single-graph counterpart, there is currently no known branch-and-cut implementation that successfully exploits independent set cuts for the maximum 2-club problem, to our best knowledge.

6.4 | Club signatures case study

In this section, we conduct a computational case study to assess the effectiveness of the approaches developed in this paper, in solving a closely related problem—*the maximum k-club signature problem*. The approach we developed to solve this problem in [6] requires solving a series of maximum cross-graph k-club problems, thus motivating the

Portions of Section 6.4 are reprinted with permission from: .

TABLE 3 Comparison of CCF and PPCF on DIMACS and VB instances.

			Reduction by Peel[e]		CCF					PPCF					
k	p	Collection	#Nodes	#Edges	obj	time (s)	#LC[a]	#NCT[b]	#BCN[c]	obj	time (s)	#LC[a]	#SLC[d]	#NCT[b]	#BCN[c]
3	2	email	259	825	127.2	664.3	4,830	3.1	48,617	127.2	660.1	5,213	308	3.0	49,894
3	2	football	0	0	30.7	0.5	1,428	2.2	438	30.7	0.6	1,468	83	2.1	452
3	3	email	247	1,132	113.0	791.8	7,896	3.4	70,188	113.0	841.2	7,674	820	3.3	86,506
3	3	football	0	0	26.1	0.6	1,512	2.4	395	26.1	0.6	1,612	204	2.1	417
3	4	email	248	1,485	104.1	950.5	8,870	3.6	51,937	104.1	750.2	9,552	1,411	3.3	32,434
3	4	football	0	0	25.1	0.5	1,616	2.5	199	25.1	0.5	1,350	255	2.2	183
3	5	email	249	1,824	97.3	1,003.9	13,026	3.8	49,815	97.3	915.9	12,322	2,219	3.4	34,256
3	5	football	0	0	24.7	0.5	1,695	2.6	220	24.7	0.5	1,612	396	2.1	164
4	2	VB_1.0	0	169	2.1	9.9	3,399	1.2	1,525	2.1	7.8	3,320	3,320	0.0	1,555
4	2	VB_1.5	6	35	4.3	336.8	27,301	1.9	22,386	4.3	155.1	27,421	27,417	0.0	21,598
4	2	email	183	462	462.9	15.6	1,310	3.7	550	462.9	16.8	1,303	29	3.6	546
4	3	VB_1.5	0	110	1.6	228.3	35,568	1.8	17,136	1.6	72.2	36,009	36,009	0.0	19,497
4	3	email	167	570	432.5	23.4	2,211	6.3	912	432.5	25.0	2,189	82	6.4	798
4	3	hep-th	7,384	29,569	175.9	209.8	2,315	3.4	1,007	175.9	216.3	3,310	240	3.2	954
4	4	VB_1.5	0	240	1.0	187.7	29,629	1.7	8,335	1.0	47.9	29,630	29,630	0.0	9,643
4	4	email	150	595	411.1	27.0	2,615	6.0	1,284	411.1	29.1	2,596	195	5.9	1,156
4	4	hep-th	7,248	38,182	154.6	339.2	4,073	3.2	2,251	154.6	336.5	3,665	490	3.1	1,600
4	5	VB_1.5	0	466	1.0	91.1	20,184	1.7	3,928	1.0	35.8	20,187	20,187	0.0	4,609
4	5	email	152	729	396.2	26.6	2,642	5.2	1,454	396.2	29.0	2,349	213	4.9	1,392
4	5	hep-th	7,016	44,855	139.5	607.8	5,489	3.8	2,298	139.5	555.9	4,049	717	3.7	1,921

[a] Average number of lazy constraints added.
[b] Average number of negative terms in a lazy constraint.
[c] Average number of branch-and-cut tree nodes.
[d] Average number of lazy constraints added that were strictly strengthened by pairwise peeling.
[e] Peeling was effective on most of these instances, with the exception of the VB graphs and the football graph.

present study. We introduce the necessary background before presenting our computational results on this problem, and use the notation $[T]$ to denote the index set $\{1, 2, \ldots, T\}$ in the following discussion. We also emphasize here that graph signatures are defined on *sequences* of graphs (e.g., temporal graphs) as opposed to an unordered collection.

Definition 6 *Given a graph sequence $\mathcal{G} = (G_t, t \in [T])$ and positive integers k and τ, we call a subset of nodes S a τ-persistent k-club signature in $\mathcal{G}$ if there exists a subsequence $\mathcal{H} = (G^t, \ldots, G^{t+\tau-1})$ of $\mathcal{G}$ such that S forms a k-club in every graph in the subsequence.*

The maximum k-club signature problem seeks to find a maximum cardinality τ-persistent k-club signature of $\mathcal{G}$. By definition, a τ-persistent k-club signature of $\mathcal{G}$ is also a τ-graph k-club on a consecutive subsequence of τ graphs; or more precisely on the graph collection obtained by ignoring the ordering. A monolithic IP formulation and a moving window (MW) method are introduced to solve the maximum 2-club signature problem in [6]. Given a graph sequence $\mathcal{G}$, the MW method involves solving $T - \tau + 1$ maximum τ-graph k-club problems (window problems) on the $T - \tau + 1$ consecutive subsequences of length τ (windows). Two versions of the MW method, MW-2CLB and MW-F2, are compared for solving the maximum 2-club signature problem in [6]. Each of the subproblems in MW-F2 is solving a conjunction of τ common neighbor formulations, while a subproblem of MW-2CLB exploits decomposition and preprocessing techniques by adding the common neighbor constraints in a delayed manner similar to CCF. The computational results in [6] show that MW-2CLB, which is essentially the same as using CCF for each window problem,

TABLE 4 Comparison of PPCF and PPCF + Independent Set Cut on BG instances for $k = 2$.

			PPCF				PPCF + Independent Set Cut				
k	p	Collection	obj	time (s)	#BCN[a]	#LC[b]	obj	time (s)	#BCN[a]	#LC[b]	#ISLC[c]
2	2	BG_10	9.4	79.6	16,411	11,241	9.4	106.3	17,987	11,350	249
2	2	BG_15	16.4[d]	172.0%[d]	6,321,129	17,377	16.4[e]	209.7%[e]	4,732,194	17,319	491
2	3	BG_10	4.6	38.7	15,293	25,076	4.6	48.4	16,371	24,102	626
2	3	BG_15	7.9	977.2	343,897	25,224	7.9	1212.0	353,595	25,217	468
2	4	BG_10	2.0	32.3	16,418	38,316	2.0	55.5	15,962	38,153	1,964
2	4	BG_15	4.4	388.0	58,895	43,698	4.4	473.6	61,862	44,109	1,087
2	5	BG_10	1.2	32.4	14,362	44,424	1.2	60.9	15,334	44,444	2,543
2	5	BG_15	2.3	296.2	36,145	64,846	2.3	382.8	37,918	66,588	2,329

[a] Average number of branch-and-cut tree nodes.

[b] Average number of PPCF lazy constraints added. This does not include the independent set cuts added.

[c] Average number of violated independent set cuts added.

[d] Average MILP gap over 5 out of 9 instances that were not solved to optimality and were terminated when the time limit was reached. Here we report the average of the best solutions found for all 9 instances. The average running time is 6,069.4 seconds over 4 out of 9 instances that were solved to optimality.

[e] Average MILP gap over 8 out of 9 instances that were not solved to optimality and were terminated when the time limit was reached. Here we report the average of the best solutions found for all 9 instances. The one instance solved to optimality took 5,804.9 seconds.

is preferable over MW-F2 and directly solving the monolithic IP formulation of the problem. In this section, we compare the performance of the moving window counterparts (MW-CCF and MW-PPCF) of the decomposition BC algorithms with preprocessing introduced in this paper in solving the maximum k-club signature problem.

Although a maximum k-club signature problem can be solved by decomposing it into a series of maximum τ-graph k-club problems, it is preferable to not treat them independent of each other. As the preprocessing/peeling procedures between two consecutive window problems may reduce the size of the subsequent window problems more effectively if given a better feasible solution found in one of the previous window problems. Therefore, we use the DROP heuristic [10] on the intersection graph of the very first window to find a feasible solution used in preprocessing, and subsequently update it with the best feasible solution found as we sequentially solve the window problems.

In the rest of this section, we report our results from two experiments comparing the performance of MW-CCF and MW-PPCF in solving the maximum k-club signature problem. Generally, we consider $k \in \{2, 3, 4\}$ and $\tau \in \{2, 3, 4, 5\}$ in our first set of experiments on only the challenging instances identified for each k from our foregoing experiments. We consider longer graph sequences (i.e., larger T) with a larger value of τ in the second set of experiments. We remark that the results reported in this section are not averaged like in the previous tables, as we are now solving the club signature version with a moving window of length τ over a T-graph sequence seeking a window containing the largest cardinality τ-graph k-club. For each window problem, we allow a Gurobi solve time limit of 3600 seconds and terminate the algorithm if two consecutive window problems are not solved to optimality.

Tables 5 and 6 summarize the results. Both MW-CCF and MW-PPCF did not solve one instance from the BG collections to optimality. Overall, MW-PPCF is over 47% faster on average than MW-CCF on BG instances that were solved to optimality, and over 23% faster on average on DIMACS-10 and VB instances. The advantage of MW-PPCF on DIMACS-10 and VB instances are not as obvious because significantly fewer lazy constraints were added on these instances.

TABLE 5 Comparison of MW-CCF and MW-PPCF on BG instances.

			MW-CCF		MW-PPCF	
k	τ	Collection	obj[b]	time (s)[c]	obj[b]	time (s)[c]
2	2	BG_10	10	927.1	10	717.3
2	3	BG_10	5	447.3	5	309.4
2	4	BG_10	3	673.9	3	226.9
2	5	BG_10	2	551.5	2	193.8
2	2	BG_15	≥ 17[a]	7202.7	≥ 17[a]	7203.1
2	3	BG_15	10	7567.5	10	7893.7
2	4	BG_15	5	3159.2	5	2708.7
2	5	BG_15	3	2121.6	3	1785.5
3	2	BG_5	14	6550.2	14	4961.0
3	3	BG_5	4	2055.4	4	1077.8
3	4	BG_5	1	2037.8	1	379.8
3	5	BG_5	1	1849.8	1	197.7
4	2	BG_2.5	11	1030.3	11	559.4
4	3	BG_2.5	2	1146.5	2	344.3
4	4	BG_2.5	1	664.8	1	247.2
4	5	BG_2.5	1	518.9	1	121.1

[a] Instance not solved to optimality; the best solution found (a valid lower bound) is reported.
[b] This column reports the optimal size of a τ-persistent k-club signature.
[c] This column reports the wall-clock running time (in seconds) of the moving window algorithm.

The aim of our second set of experiments is to explore the impact of larger values of τ on real-life graphs, especially if non-trivial (in terms of size) solutions are detected. We use the same sequence generator as before to generate 12 sequences based on the 12 DIMACS-10 graphs identified in Table 2 that includes many social and biological networks. Each instance is a sequence containing 100 graphs. For these instances, we consider $\tau = 10$ and $k = 2, 3, 4$. Results are reported in Table 7. Interestingly, we find that MW-CCF is 7% faster than MW-PPCF on average, although there is one instance `hep-th_100` that MW-CCF did not solve to optimality, but MW-PPCF did. Although not significantly behind, the additional time spent generating PPCF constraints was not worthwhile in this experiment. Nonetheless, it is interesting to see that the size of the optimal k-club signature identified is not very small despite using a larger τ.

TABLE 6 Comparison of MW-CCF and MW-PPCF on DIMACS-10 and VB instances.

			MW-CCF		MW-PPCF	
k	τ	Collection	obj[a]	time (s)[b]	obj[a]	time (s)[b]
3	2	email	135	2,441.8	135	2,893.0
4	2	email	473	133.6	473	128.8
3	3	email	123	4,528.5	123	3,589.7
4	3	email	440	129.6	440	127.1
3	4	email	114	3,428.0	114	1,392.3
4	4	email	419	128.6	419	135.8
3	5	email	107	3,122.2	107	2,979.2
4	5	email	403	112.7	403	133.9
3	2	football	35	7.5	35	4.7
3	3	football	28	5.6	28	5.0
3	4	football	26	5.4	26	3.4
3	5	football	25	6.2	25	3.1
4	3	hep-th	198	1,310.2	198	1,277.4
4	4	hep-th	171	1,465.0	171	1,365.0
4	5	hep-th	143	1,854.8	143	1,710.8
4	2	VB_1.0	3	52.7	3	40.8
4	2	VB_1.5	7	3,144.4	7	1,539.3
4	3	VB_1.5	2	1,523.0	2	752.6
4	4	VB_1.5	1	1,374.4	1	336.1
4	5	VB_1.5	1	584.1	1	214.6

[a] This column reports the optimal size of a τ-persistent k-club signature.

[b] This column reports the wall-clock running time (in seconds) of the moving window algorithm.

TABLE 7 Comparison of MW-CCF and MW-PPCF on DIMACS-10 instances with $T = 100$ and $\tau = 10$.

		MW-CCF		MW-PPCF	
k	Instance	obj[b]	time (s)[c]	obj[b]	time (s)[c]
2	adjnoun_100	14	6.0	14	5.5
3	adjnoun_100	54	8.0	54	8.1
4	adjnoun_100	89	2.5	89	2.5
2	celegans_metabolic_100	41	28.1	41	37.0
3	celegans_metabolic_100	188	174.4	188	184.4
4	celegans_metabolic_100	361	153.9	361	207.1
2	email_100	21	205.6	21	221.8
3	email_100	90	18,342.1	90	17,371.9
4	email_100	373	3,037.7	373	3,750.3
2	football_100	13	1.6	13	1.7
3	football_100	24	29.9	24	34.3
4	football_100	86	555.4	86	582.0
2	hep-th_100	24	608.3	24	632.4
3	hep-th_100	44	35,011.6	44	34,786.5
4	hep-th_100	$\geq$ 119[a]	64,143.6	119	65,629.1
2	karate_100	8	0.7	8	0.6
3	karate_100	17	0.6	17	0.6
4	karate_100	24	0.5	24	0.5
2	lesmis_100	14	1.1	14	1.2
3	lesmis_100	32	0.9	32	0.9
4	lesmis_100	51	1.0	51	1.1
2	netscience_100	21	24.0	21	25.0
3	netscience_100	27	34.0	27	35.7
4	netscience_100	40	37.4	40	38.4
2	PGPgiantcompo_100	65	11,701.3	65	11,810.1
3	PGPgiantcompo_100	196	6,283.7	196	6,536.6
4	PGPgiantcompo_100	387	38,307.6	387	43,014.0
2	polblogs_100	161	3,253.2	161	3,857.7
3	polblogs_100	581	2,334.3	581	2,576.6
4	polblogs_100	945	1,983.4	945	2,127.9
2	polbooks_100	15	1.6	15	1.9
3	polbooks_100	36	1.4	36	1.5
4	polbooks_100	52	2.1	52	2.3
2	power_100	8	208.4	8	213.7
3	power_100	16	426.8	16	443.2
4	power_100	28	571.5	28	599.5

[a] Instance not solved to optimality; the best solution found (a valid lower bound) is reported.

[b] This column reports the optimal size of a τ-persistent k-club signature.

[c] This column reports the wall-clock running time (in seconds) of the moving window algorithm.

7 | CONCLUDING REMARKS

The cross-graph k-club model is designed to mine low-diameter clusters conserved in graph collections. Such a collection may represent a time-varying graph or a graph where node relationships change under different conditions. This paper develops integer programming approaches to find a maximum cardinality cross-graph k-club from a given graph collection. Our main contribution is the strengthening of a well-known cut-like formulation for the single-graph counterpart through what we call *pairwise peeling* and assessing its computational performance in conjunction with preprocessing and delayed constraint generation. Our results strongly suggest that there is significant advantage to using the approaches we introduce in this paper.

We also identify valid inequalities for the problem for general k and if $k = 2$, essentially extending single-graph counterparts. An important by-product is an alternative proof of validity of the independent set inequality proved in [27] for the maximum 2-club problem. At this time, we are unable to establish facet-inducing conditions for these inequalities, and consider it an important next step to advance this study. Similar to what has been observed in the context of single-graph counterpart of the independent set inequalities, we have been unable to make effective use of these cuts in a branch-and-cut algorithm that outperforms our delayed constraint generation algorithm using the original constraints. We expect these inequalities to contribute to our ability to solve challenging instances of the problem as we better understand their strength and devise effective separation procedures.

Motivated by detection and deactivation of fake accounts in social media, a methodology which interdicts k-clubs of maximum cardinality in a given graph is studied in [12]. As relationships between accounts in social media are time varying, one could consider interdicting a cross-graph maximum k-club over a collection of snapshot graphs in order to identify a more robust interdiction policy. Likewise, one may consider interdiction problems in temporal graphs like the interdiction of atomic k-clubs [22] or k-club signatures [6]. The maximum cross-graph k-club problem can serve as the separation problem in these broad future directions for this study. Although our focus in this paper is on clusters that induce low-diameter subgraphs, one may investigate any clique relaxation or another graph property in the cross-graph setting depending on the domain or data underlying the graph models.

Acknowledgments

The authors are grateful to the referees for their constructive criticism of our manuscript and for the suggestions that helped us improve our content and presentation. We are also thankful to the editors of the journal and this special issue for their patience and support.

References

[1] J. Abello, P.M. Pardalos, and M.G.C. Resende, "*On maximum clique problems in very large graphs*," *External memory algorithms and visualization*, J. Abello and J. Vitter (eds.), American Mathematical Society, Boston, MA, USA, 1999, pp. 119–130.

[2] R.D. Alba, *A graph-theoretic definition of a sociometric clique*, J. Math. Sociology **3** (1973), 113–126.

[3] R. Alhajj and J. Rokne (eds.), *Encyclopedia of Social Network Analysis and Mining*, Springer, New York, 2018.

[4] D.A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner (eds.), *Graph partitioning and graph clustering: Tenth* Dimacs *Implementation Challenge Workshop* Vol. 588 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, Providence, RI, 2013.

[5] G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella, *Length-bounded cuts and flows*, ACM Trans. Algorithms (TALG) **7** (2010), 4.

[6] B. Balasundaram, J.S. Borrero, and H. Pan, *Graph signatures: Identification and optimization*, Eur. J. Oper. Res. **296** (2022), 764–775.

[7] B. Balasundaram, S. Butenko, and I.V. Hicks, *Clique relaxations in social network analysis: The maximum k-plex problem*, Oper. Res. **59** (January-February 2011), 133–142.

[8] B. Balasundaram, S. Butenko, and S. Trukhanov, *Novel approaches for analyzing biological networks*, J. Combinatorial Optim. **10** (August 2005), 23–39.

[9] M. Bentert, A.S. Himmel, H. Molter, M. Morik, R. Niedermeier, and R. Saitenmacher, *Listing all maximal k-plexes in temporal graphs*, ACM J. Experimental Algorithmics **24** (Sept. 2019), 1.13:1–1.13:27.

[10] J.M. Bourjolly, G. Laporte, and G. Pesant, *Heuristics for finding k-clubs in an undirected graph*, Comput. Oper. Res. **27** (2000), 559–569.

[11] J.M. Bourjolly, G. Laporte, and G. Pesant, *An exact algorithm for the maximum k-club problem in an undirected graph*, Eur. J. Oper. Res. **138** (2002), 21–28.

[12] N. Daemi, J.S. Borrero, and B. Balasundaram, *Interdicting low-diameter cohesive subgroups in large-scale social networks*, INFORMS J. Optim. **4** (2022), 304–325.

[13] M. Gendreau, P. Soriano, and L. Salvail, *Solving the maximum clique problem using a tabu search approach*, Ann. Oper. Res. **41** (1993), 385–403.

[14] Gurobi Optimization, LLC, 2020. *Gurobi optimizer reference manual*. `http://www.gurobi.com`, Accessed July 2024.

[15] S. Hartung, C. Komusiewicz, and A. Nichterlein, *Parameterized algorithmics and computational experiments for finding 2-clubs*, J. Graph Algorithms Appl. **19** (2015), 155–190.

[16] A.S. Himmel, H. Molter, R. Niedermeier, and M. Sorge, *Adapting the Bron–Kerbosch algorithm for enumerating maximal cliques in temporal graphs*, Social Network Anal. Mining **7** (Jul 2017), 35.

[17] C. Jiang, F. Coenen, and M. Zito, *A survey of frequent subgraph mining algorithms*, Knowledge Eng. Review **28** (2013), 75–105.

[18] D. Jiang and J. Pei, *Mining frequent cross-graph quasi-cliques*, ACM Trans. Knowledge Discovery Data **2** (2009), 16:1–42.

[19] B.H. Junker and F. Schreiber (eds.), *Analysis of Biological Networks*, Wiley, New York, 2008.

[20] C. Komusiewicz, A. Nichterlein, R. Niedermeier, and M. Picker, *Exact algorithms for finding well-connected 2-clubs in sparse real-world graphs: Theory and experiments*, Eur. J. Oper. Res. **275** (2019), 846–864.

[21] S. Laan, M. Marx, and R.J. Mokken, *Close communities in social networks: boroughs and 2-clubs*, Social Network Anal. Mining **6** (2016), 1–16.

[22] Y. Lu, Z. Miao, P. Sahraeian, and B. Balasundaram, *On atomic cliques in temporal graphs*, Optim. Lett. **17** (2023), 813–828.

[23] Y. Lu, E. Moradi, and B. Balasundaram, *Correction to: Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts*, Optim. Lett. **12** (November 2018), 1959–1969.

[24] Y. Lu, H. Salemi, B. Balasundaram, and A. Buchanan, *On fault-tolerant low-diameter clusters in graphs*, INFORMS J. Comput. **34** (2022), 3181–3199.

[25] R.D. Luce, *Connectivity and generalized cliques in sociometric group structure*, Psychometrika **15** (1950), 169–190.

[26] F. Mahdavi Pajouh and B. Balasundaram, *On inclusionwise maximal and maximum cardinality k-clubs in graphs*, Discr. Optim. **9** (May 2012), 84–97.

[27] F. Mahdavi Pajouh, B. Balasundaram, and I.V. Hicks, *On the 2-club polytope of graphs*, Oper. Res. **64** (November-December 2016), 1466–1481.

[28] Z. Miao and B. Balasundaram, *An ellipsoidal bounding scheme for the quasi-clique number of a graph*, INFORMS J. Computi **32** (2020), 763–778, Codes/instances online at: `https://github.com/baski363/Quasi-clique`. Accessed July 2024.

[29] K. Mitra, A.R. Carvunis, S.K. Ramesh, and T. Ideker, *Integrative approaches for finding modular structure in biological networks*, Nature Reviews Genetics **14** (2013), 719–732.

[30] R.J. Mokken, *Cliques, clubs and clans*, Qual. Quantity **13** (1979), 161–173.

[31] E. Moradi and B. Balasundaram, *Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts*, Optim. Lett. **12** (November 2018), 1947–1957.

[32] H. Pan, 2021. *Mining low-diameter clusters conserved in graph collections*. Available online at: `https://www.proquest.com/docview/2668445131`. Accessed July 2024.

[33] H. Pan, B. Balasundaram, and J.S. Borrero, *A decomposition branch-and-cut algorithm for the maximum cross-graph k-club problem*, Proceedings of the 10th International Network Optimization Conference (INOC), Open Proceedings, 2022, pp. 17–22, `http://www.openproceedings.org/html/pages/2022_inoc.html`. Accessed July 2024.

[34] H. Pan, B. Balasundaram, and J.S. Borrero, 2024. *Decomposition branch-and-cut algorithm implementing the pairwise peeled cut-like formulation for the maximum cross-graph k-club problem*. Codes and instances online at: `https://github.com/haonap/crossGraphKclub`. Accessed July 2024.

[35] S. Pasupuleti, *Detection of protein complexes in protein interaction networks using n-clubs*, In EvoBIO 2008: Proceedings of the 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, Springer, 2008, pp. 153–164, volume 4973 of Lecture Notes in Computer Science.

[36] J. Pattillo, N. Youssef, and S. Butenko, *On clique relaxation models in network analysis*, Eur. J. Oper. Res. **226** (2013), 9–18.

[37] J. Pei, D. Jiang, and A. Zhang, *Mining cross-graph quasi-cliques in gene expression and protein interaction data*, Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, IEEE, april 2005, pp. 353 – 356.

[38] J. Pei, D. Jiang, and A. Zhang, *On mining cross-graph quasi-cliques*, Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, ACM, New York, NY, USA, 2005, KDD '05 pp. 228–238.

[39] H. Salemi and A. Buchanan, *Parsimonious formulations for low-diameter clusters*, Math. Program. Computation **12** (2020), 493–528.

[40] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier, *Parameterized computational complexity of finding small-diameter subgraphs*, Optim. Lett. **6** (2012), 883–891.

[41] K. Semertzidis, E. Pitoura, E. Terzi, and P. Tsaparas, *Finding lasting dense subgraphs*, Data Mining Knowledge Discovery **33** (Sep 2019), 1417–1445.

[42] S. Shahinpour and S. Butenko, "*Distance-based clique relaxations in networks: s-clique and s-club*," *Models, Algorithms, and Technologies for Network Analysis*, B.I. Goldengorin, V.A. Kalyagin, and P.M. Pardalos (eds.), Springer, New York, 2013, pp. 149–174.

[43] R. Sharan and T. Ideker, *Modeling cellular machinery through biological network comparison*, Nature Biotechnology **24** (2006), 427–433.

[44] K. Sim, G. Liu, V. Gopalkrishnan, and J. Li, *A case study on financial ratios via cross-graph quasi-bicliques*, Informat. Sci. **181** (2011), 201–216.

[45] S. Trukhanov, C. Balasubramaniam, B. Balasundaram, and S. Butenko, *Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations*, Comput. Optim. Appl. **56** (September 2013), 113–130.

[46] A. Veremyev and V. Boginski, *Identifying large robust network clusters via new compact formulations of maximum k-club problems*, Eur. J. Oper. Res. **218** (2012), 316–326.

[47] A. Verma, A. Buchanan, and S. Butenko, *Solving the maximum clique and vertex coloring problems on very large sparse networks*, INFORMS J. Comput. **27** (2015), 164–177.

[48] T. Viard, M. Latapy, and C. Magnien, *Computing maximal cliques in link streams*, Theor. Comput. Sci. **609** (2016), 245–252.

[49] S. Wasserman and K. Faust, *Social Network Analysis*, Cambridge University Press, New York, 1994.