A Decomposition Branch-and-Cut Algorithm for the Maximum Cross-Graph k-Club Problem

Hao Pan
hao.pan@okstate.edu
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, Oklahoma, USA

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

Juan S. Borrero
juan.s.borrero@okstate.edu
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, Oklahoma, USA

ABSTRACT

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/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. We introduce algorithmic ideas to solve this problem and evaluate their performance on some benchmark instances.

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 subset is a group whose member nodes 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 [12].

© 2022 Copyright held by the owner/authors(s).
Published in Proceedings of the 10th International Network Optimization Conference (INOC), March 1-4, 2022, Aachen, Germany.
Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.

The clique and its graph-theoretic relaxations have been extensively studied and used as cluster models in diverse fields [19]. Major categories include distance based relaxations k-clique and k-club [6], and edge count, degree, and density based relaxations k-defective clique [25], k-plex [5], and quasi-clique [15], 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, a single graph is typically a snapshot that reflects node relationships at the point in time it is recorded.

Alternatively, relationships between a group of nodes may be different under different conditions. Jointly mining node relationships under different conditions might uncover novel clusters that cannot be found by individually analyzing 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 [21]. 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 [20].

In this paper we consider a *cross-graph k-club* model to represent 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.

2 CROSS-GRAPH k-CLUBS

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 $S \subseteq V(G)$, $G[S]$ denotes the subgraph induced by S, obtained by deleting nodes outside S and their incident edges. We denote by $\text{dist}_G(i, j)$ the minimum number of edges on a path connecting nodes i and j in graph G and its diameter as $\text{diam}(G) := \max\{\text{dist}_G(i, j) : i, j \in V(G)\}$.

Definition 2.1 ([14]). Given a graph G and a positive integer k, a subset of nodes $S \subseteq V(G)$ is called a *k-clique* if $\text{dist}_G(i, j) \leq k$ for every pair of nodes $i, j \in S$.

A k-clique S allows two vertices u and v to be in S even if every path between u and v of length at most k in G includes vertices outside S (see Figure 1). By contrast, in a k-club, at least one of those paths should be contained in S as the definition below states.

Definition 2.2 ([16]). Given a graph G and a positive integer k, a subset of nodes $S \subseteq V(G)$ is called a *k-club* if $\text{diam}(G[S]) \leq k$.

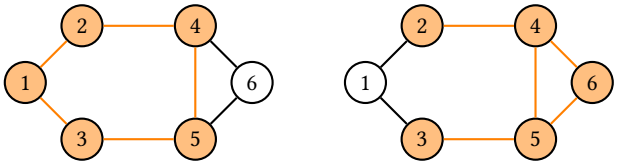

Figure 1: The set $\{1, 2, 3, 4, 5\}$ forms a 2-club; the set $\{2, 3, 4, 5, 6\}$ forms a 2-clique, but does not induce a 2-club [2].

For low values of parameter k, typically no more than four, the k-club can be an appropriate choice for modeling cohesive social subgroups or 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 [21], which also appears to be the earliest formal study of a cross-graph cluster 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 2.3. A subset of nodes $S \subseteq V(\mathcal{G})$ is called a *p-graph k-club* if S is a k-club in each graph in the collection $\mathcal{G}$.

This paper focuses on *the maximum p-graph k-club problem*, which seeks to find a p-graph k-club of maximum cardinality in $\mathcal{G}$. We use the prefix "p-graph" when we know or 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 [21].

3 LITERATURE REVIEW

The (1-graph) maximum k-club problem is NP-hard for every value of parameter k fixed in the problem [8]. Consequently, the maximum p-graph k-club problem is NP-hard for every fixed positive integer k as it includes the maximum k-club problem as special case. Shahinpour and Butenko [23] provide a comprehensive survey on the complexity results and algorithmic approaches for the maximum k-club problem. Given the focus of this paper, we limit our review to integer programming (IP) formulations of the maximum k-club problem and prevailing works on cross-graph models.

The first IP formulation for the maximum k-club problem seen in the literature was the chain formulation given by Bourjolly et al [8]. The chain formulation introduces a binary variable for each path of length at most k connecting a nonadjacent pair of nodes i and j, in addition to binary variables indicating membership in the k-club. For the special case of $k = 2$, the binary variables for the paths are unnecessary as each path of length 2 between i and j is uniquely identified by the common neighbor internal to that path. Thus, we obtain the so-called common neighbor formulation for the maximum 2-club problem. Path enumeration gets increasingly challenging as k takes values larger than two, and for arbitrary k it can take up to $O(n^{k+1})$ binary variables and constraints to fully describe the chain formulation on a graph with n nodes.

For the maximum 3-club problem, Almeida and Carvalho [4] introduced a compact neighborhood formulation, as well as a node cut set formulation with exponentially many constraints in the worst case. Veremyev and Boginski [26] introduced two polynomial-sized IP formulations for the maximum k-club problem, one using binary variables and the other using integer variables, obtained by linearizing a polynomial formulation. Both are described by $O(kn^2)$ variables and constraints on an n-node graph, and are known to be the first compact IP formulations for the maximum k-club problem for general k.

Moradi and Balasundaram [13, 17] proposed a branch-and-cut algorithm that is based on a delayed application of canonical hypercube cuts to eliminate integral solutions to the initial relaxation (a maximum k-clique formulation) that do not correspond to a k-club. They also introduced an iterative graph decomposition framework based on variable fixing to solve the problem on potentially small subgraphs of the original graph and to take advantage of intermediate solutions in preprocessing (vertex deletion) between iterations.

Salemi and Buchanan [22] introduced a cut-like formulation and a path-like formulation using length-bounded separators and connectors. Their study generalizes for arbitrary k, some of the aforementioned formulations for the special cases, when $k = 2, 3$. Generally, the cut-like formulation could use exponentially many constraints, but only $|V|$ binary variables. This formulation and the associated decomposition branch-and-cut algorithm demonstrated effective computational performance making it the current state-of-the-art mathematical programming approach to solve the maximum k-club problem for arbitrary k.

Although previous works on cross-graph models in the literature are limited, we mention some examples that are related and motivated our study. To extract reliable patterns across multiple pieces of data, Pei et al [21] mined cross-graph quasi-cliques and developed algorithms to enumerate them across multiple graphs. Jiang and Pei [11] extended this work to the problem of finding frequent cross-graph quasi-cliques. They seek to enumerate maximal node subsets that form quasi-cliques in at least a minimum number of graphs in the collection. We must clarify that the terminology has been reused as these are "degree-based" quasi-cliques [18], and not "density-based" quasi-cliques [15].

Sim et al [24] introduced 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. 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.

4 IP FORMULATIONS

We begin with an IP formulation that is a direct extension of the cut-like formulation for the maximum k-club problem to the cross-graph setting. Then we propose a new formulation based on what we refer to as 'pairwise peeling.'

4.1 A conjunctive cut-like formulation

We use the following additional notations in the formulation. Let $\overline{G}$ denote the complement graph of G. Given a graph G and a pair of nonadjacent nodes u and v, a subset of nodes S is called a length-k

u, v-separator if $\text{dist}_{G\setminus S}(u, v) > k$, where $G \setminus S$ denotes the graph obtained from G by deleting the nodes in S along with its incident edges. In other words, every path of length at most k in G between u and v uses nodes from S. By $\mathcal{S}_G(u, v)$, we 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 set of common neighbors, *i.e.*, nodes adjacent to both u and v in G. For a subset of nodes C, we use the short form $x(C)$ to denote $\sum_{u\in C} x_u$. Consider the following optimization problem:

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

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

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

Formulation (1) is a conjunction of the cut-like formulation of the maximum k-club problem introduced by Salemi and Buchanan [22], across all the graphs in the collection. Henceforth, we refer to formulation (1) as the conjunctive cut-like formulation (CCF). It is readily verified that the CCF is a correct formulation in the sense that x is an incidence vector of a cross-graph k-club if and only if it is feasible to the CCF.

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

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

where,

$$D_{uv} \coloneqq \big\{w \in V(\mathcal{G}) \setminus \{u, v\} : \text{dist}_G(u, w) \le k \text{ and } \text{dist}_G(v, w) \le k\ \forall G \in \mathcal{G}\big\} \tag{3}$$

is the set of nodes that are at a distance of at most k from u and v in all the graphs in $\mathcal{G}$. The validity of constraints (2) follows from the observation that if $x_u = x_v = 1$, then $x(S \setminus D_{uv}) = 0$ as 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 minimalizing the separator S by removing nodes that are not on 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 $x(S \cap D_{uv}) < x(S)$ for at least one separator $S \in \mathcal{S}_G(u, v)$, as illustrated next.

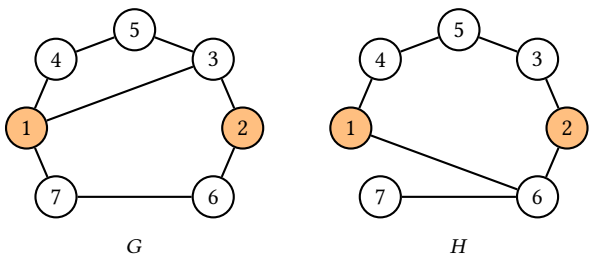

Figure 2: Inequality $x_1 + x_2 \le 1$ is valid for the problem on $\mathcal{G} = \{G, H\}$ when $k = 2$.

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

4.2 A conjunctive formulation based on pairwise peeling

Based on the idea of intersecting the length-k u, v-separator S in constraint (1b) with D_{uv} to obtain a tighter constraint (2), we can 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 graph in the collection. As the deletion of nodes tends to 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.

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 $\mathcal{S}_H(1, 6) = \big\{\{3\}, \{4\}\big\}$.

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

However, the inequality $x_1 + x_6 \le 1$ 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, $x_1 + x_6 \le 1$ is valid.

Algorithm 1 formalizes the idea illustrated by the foregoing example to generate tighter constraints, and we refer to it as the *pairwise peeling algorithm*. We denote by $\mathcal{J}$ the node pairs that are nonadjacent in some graph in the collection $\mathcal{G}$, *i.e.*,

$$\mathcal{J} \coloneqq \Big\{\{u, v\} \subset V(\mathcal{G}) : u \ne v, uv \in E(\overline{G}) \text{ for some } G \in \mathcal{G}\Big\}.$$

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

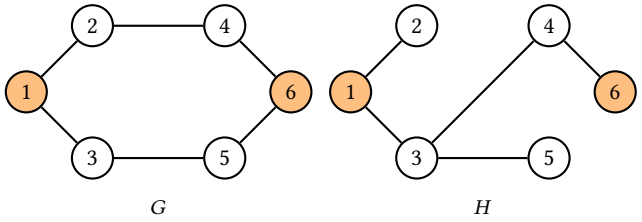

Figure 3: Inequality $x_1 + x_6 \leq 1$ is valid for the problem on $\{G, H\}$ when $k = 3$.

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:

$$x_u + x_v - x(S) \leq 1 \quad \forall S \in \mathcal{S}_G(u, v) \text{ and } \forall G \in \mathcal{G}_{uv} \text{ such that } uv \in E(\overline{G}), uv \in \mathcal{J}. \quad (4)$$

In Algorithm 1, the do-while loop executes at most $|V(\mathcal{G})| + 1$ times, as we delete at least one node in each iteration. In one execution of the do-while loop, the nested for-loops execute $O(p|V(\mathcal{G})|)$ times. Computing distances from a single node w and deleting w if necessary, can be completed in $O(|V(G)| + |E(G)|)$. Although the worst-case complexity is unattractive, we did not find this algorithm to be time-consuming when compared to the impact it has on overall running time in our computational experiments.

Algorithm 1: Pairwise Peeling

Input: $\mathcal{G}$, k, $uv \in \mathcal{J}$
Output: $\mathcal{G}_{uv}$

```
do
    W ← ∅
    for G ∈ 𝒢 do
        for w ∈ V(𝒢) \ (W ∪ {u, v}) do
            if dist_G(u, w) > k or dist_G(v, w) > k then
                W ← W ∪ {w}
                delete w from every graph in 𝒢
while W ≠ ∅;
return 𝒢_uv ← 𝒢
```

Proposition 4.1. *If constraints* (1b) *in formulation* (1) *are replaced by constraints* (4)*, the resulting formulation is correct for the maximum p-graph k-club problem.*

The claim follows from the observation that the incidence vector of a p-graph k-club satisfies constraints (4) and every binary vector satisfying these constraints also satisfies constraints (1b).

Proposition 4.2. *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 the new formulation as the pairwise peeled cut-like formulation (PPCF). For each $uv \in \mathcal{J}$, constraint (4) is at least as strong as constraint (2) (which in turn dominates constraint (1b)). Through our computational experiments reported in the next section, we assess the gains made by using Algorithm 1 to generate potentially stronger constraints.

5 COMPUTATIONAL EXPERIMENTS

The goal of our computational study is to compare the performance of a general purpose IP solver when using CCF and PPCF to solve the maximum p-graph k-club problem. As either formulation uses exponentially many constraints in the worst case, we generate them in a delayed fashion and implement two decomposition branch-and-cut (BC) algorithms that use the same master problem based on cross-graph k-cliques defined below.

Definition 5.1. A subset of nodes $S \subseteq V(\mathcal{G})$ is called a *p-graph k-clique* if S is a k-clique in each graph in the collection $\mathcal{G}$.

5.1 Overview of the solvers

Like the single-graph counterparts, a cross-graph k-clique is a graph-theoretic relaxation of a cross-graph k-club. The maximum p-graph k-clique problem is equivalent to the classical maximum clique problem on the *power-intersection graph* of $\mathcal{G}$, *i.e.*, the graph with node set $V(\mathcal{G})$ and edge set containing every pair of distinct nodes that are at distance at most k in every graph in the collection. We can then define the complementary edge set $\hat{E}$ containing all the conflicting pairs of nodes as:

$$\hat{E} := \left\{ \{u, v\} \subseteq V(\mathcal{G}) : \text{dist}_G(u, v) > k \text{ in some graph } G \in \mathcal{G} \right\},$$

and we use it in our master IP formulation (5):

$$\max x(V(\mathcal{G})) \quad (5a)$$

$$x_u + x_v \leq 1 \qquad \forall uv \in \hat{E} \quad (5b)$$

$$x_u \in \{0, 1\} \qquad \forall u \in V(\mathcal{G}) \quad (5c)$$

The two decomposition BC algorithms, referred to as CCF and PPCF henceforth, would add constraints (1b) and (4), respectively, whenever a feasible solution to the master problem (5) that is not a cross-graph k-club is encountered anywhere in the BC tree. Our implementations are also enhanced by preprocessing (see Algorithm 2) based on a feasible solution S to the problem obtained using the "DROP heuristic" for k-clubs [7, 22] on the *intersection graph* J with node set $V(\mathcal{G})$ and edge set $\bigcap\{E(G) : G \in \mathcal{G}\}$.

Note that a k-club in J is a cross-graph k-club of $\mathcal{G}$. Then, in the power-intersection graph of $\mathcal{G}$ denoted by J^k, if a node u has fewer than $|S|$ neighbors, 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

Algorithm 2: Preprocessing

Input: $\mathcal{G}, k$
Output: a preprocessed graph collection $\mathcal{G}$
1 obtain the intersection graph J of all graphs in $\mathcal{G}$
2 compute a k-club S of J using DROP heuristic
3 obtain the power-intersection graph J^k of $\mathcal{G}$
4 CorePeel($\mathcal{G}, J^k, |S|$)
5 CommunityPeel($\mathcal{G}, J^k, |S|$)
6 recursively delete edge uv from *every* graph $G \in \mathcal{G}$ *and* J^k in which it is present, if u and v are in distinct connected components of some graph $G \in \mathcal{G}$ or J^k
7 **return** $\mathcal{G}$

J^k). Hence, *core peeling* [1] recursively deletes nodes with degree less than $|S|$ in J^k, and correspondingly from every graph in $\mathcal{G}$.

After core-peeling, as long as $V(J^k)$ is not empty, J^k will be an $|S|$-core as every node that survives will have at least $|S|$ neighbors. Now, a pair of nodes u and v that are adjacent in J^k can belong to a cross-graph k-club larger than S only if they have at least $|S| - 1$ common neighbors in J^k. If not, the edge uv is deleted from J^k and from every graph in the collection where u and v are adjacent in the *community peeling* [27] step.

Graph J^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^k. These edges 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^k; then, we can delete edge ab from J^k. Hence, we can recursively delete edges from every graph in the expanded collection $\mathcal{G} \cup \{J^k\}$ until all of them have identical connected components (in terms of the node subsets inducing the components). The preprocessing steps can be implemented to run in time that is linear or quadratic in the size of the input [22, 27], and their running time is negligible in practice compared to the BC algorithm.

To avoid unnecessary constraints (5b) added for pairs of nodes which reside in different components of J^k, we extend formulation (5) by introducing a binary variable for each of the connected components of J^k and enforce that nodes selected must belong to the same component. In the master problem, only conflict constraints related to a pair of nonadjacent nodes in a same connected component would be enumerated. Similarly, any node pair for which a lazy constraint is generated must also belong to the same connected component of J^k.

5.2 Preliminary results

We report computational experiments conducted on 64-bit Linux® compute nodes with dual Intel® "Skylake" 6130 CPUs with 96 GB RAM. Optimization models are solved using Gurobi™ Optimizer v9.0.1 [10] and the algorithms are implemented in C++. We consider the following parameter values in our experiments: $k \in \{2, 3, 4\}$ and $p \in \{2, 3, 4, 5\}$.

To generate test instances for this preliminary computational study, we generate graphs using the algorithm described in [8] (BG graphs). These graphs are known to be challenging for the (single-graph) maximum k-club problem. We first generate a set of 10 graphs $\{G_1, G_2, \ldots, G_{10}\}$, each with 200 nodes, by specifying an edge density and running BG algorithm 10 times. From our preliminary results, BG_1 (generated with edge density of 0.15%) and BG_4 (generated with edge density of 0.1%) are challenging instances when $k = 2$. When $k = 3$ or 4, BG_2 (generated with edge density of 0.05%) and BG_3 (generated with edge density of 0.025%) are challenging instances. For each p, k pair, we report results averaged over instances obtained from the collection of 10 graphs with the edge density that is most challenging for the chosen value of k.

Each row of Table 1 reports results averaged over $11 - p$ runs with collections $\{G_1, \ldots, G_p\}, \{G_2, \ldots, G_{p+1}\}, \ldots, \{G_{11-p}, \ldots, G_{10}\}$. Except for BG_1 instance when $p = 2$ and $k = 2$, where both approaches did not reach optimality, PPCF based BC takes a significantly shorter running times than CCF for most of the instances. For example, on BG_2 instance when $p = 4$ and $k = 3$, CCF took 1079.11 seconds while PPCF only took 47.11 seconds (over 20 times faster than CCF). Overall, PPCF is over 6 times faster than CCF in terms of average running time on challenging instances.

Table 1: Comparison of CCF and PPCF on BG instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
2	2	BG_4	9.44	88.01	12116.22	2.26	9.44	54.77	8085.56	6887.00	0.53
2	2	BG_1	≥16.33[d]	7200.23	20353.00	4.32	≥16.22	6973.67	17958.22	1310.33	4.00
2	3	BG_4	4.63	55.22	12564.38	2.27	4.63	38.41	9807.00	9802.25	0.00
2	3	BG_1	7.88	3074.61	24496.13	4.39	7.88	760.30	19702.88	3214.00	3.48
2	4	BG_4	2.43	49.74	11712.43	2.27	2.43	44.59	10442.00	10441.86	0.00
2	4	BG_1	4.43	1982.95	25226.43	4.39	4.43	338.92	20934.71	5719.86	2.94
2	5	BG_4	2.00	40.70	10166.33	2.27	2.00	36.36	9253.67	9253.67	0.00
2	5	BG_1	2.50	1533.07	25121.67	4.41	2.50	201.43	21074.00	8851.50	2.36
3	2	BG_2	12.00	3008.56	47557.00	4.54	12.00	576.44	26125.44	6079.22	3.46
3	3	BG_2	3.13	1748.18	45065.50	4.52	3.13	105.67	21304.25	12592.63	1.91
3	4	BG_2	3.00	1079.11	40636.29	4.55	3.00	47.11	17284.29	15254.00	0.61
3	5	BG_2	3.00	818.45	36544.33	4.47	3.00	40.02	15971.67	15687.17	0.10
4	2	BG_3	8.78	192.51	23791.11	3.60	8.78	40.72	12169.78	10365.56	0.79
4	3	BG_3	4.00	99.02	18502.13	3.37	4.00	31.06	11116.88	10921.00	0.01
4	4	BG_3	4.00	59.55	13925.57	3.21	4.00	30.74	9216.86	8992.71	0.00
4	5	BG_3	4.00	46.07	11001.67	3.16	4.00	29.12	7622.17	7406.50	0.00

[a] Average number of lazy constraints added.
[b] Average number of negative terms in a lazy constraint.
[c] Average number of strengthened lazy constraints added.
[d] On instances not solved to optimality we report the lower-bound provided by the best solution found.

Table 1 also shows the number of lazy constraints added by each algorithm on average. For each instance, we observe fewer lazy constraints used in PPCF than CCF. Together with the fact that PPCF took much less time than CCF for each instance, this observation is indicative of the strength of using constraints (4) over constraints (1b). On average, CCF added over 1.5 times more lazy constraints than PPCF. Column #SLC under PPCF reports the number of strengthened lazy constraints added, *i.e.*, these are strictly constraints (4). On average, over 70% of lazy constraints added in PPCF are of this type. For BG_4 instance when $p = 5$ and $k = 2$, all lazy constraints added in PPCF are of this type.

The columns labeled #NCT in Table 1 report the average number of terms with coefficient -1 on the left hand side of the added lazy constraints. This is another indirect indicator of the strength of the lazy constraints—generally, the smaller this number, the stronger the constraint. For most of the instances, we observed a significantly

smaller value under PPCF than CCF. Note that the value of #NCT is zero for five instances under PPCF. The lazy constraints of this type are actually conflict constraints with no negative terms on the left hand side. The foregoing observations strongly suggest that PPCF approach based on pairwise peeling constraints significantly improves our ability to find maximum p-graph k-clubs.

Figure 4 shows the performance profiles [9] based on running times of CCF and PPCF algorithms for solving the maximum p-graph k-club problem on the challenging instances. The red (dash-dotted) and blue (dotted) lines plot the fraction $f_i(\tau)$ of instances solved within a factor τ of the shortest running time (over all values of parameters p, k, and all challenging instances) by solvers i = CCF and PPCF. The profiles clearly suggest that the computational benefits of using lazy constraints strengthened by pairwise peeling is quite significant.

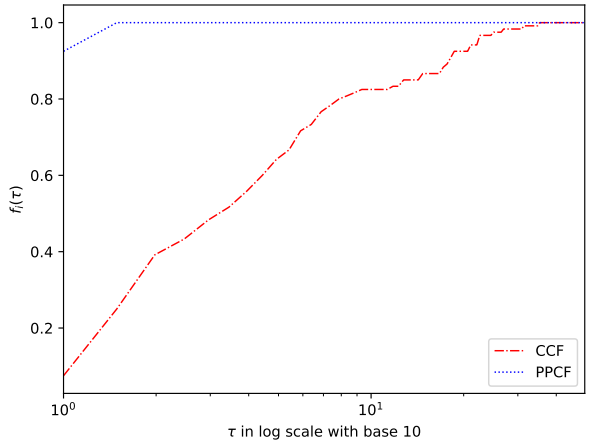

Figure 4: Performance profiles of CCF and PPCF algorithms.

6 CONCLUDING REMARKS

In this paper, we consider cross-graph k-clubs for modeling low-diameter clusters that are conserved in a collection of graphs. We introduce a pairwise peeling approach designed to strengthen constraints in a straightforward conjunction of a known formulation. This approach helps with scale-reduction and integrates information across graphs in the collection to produce tighter constraints. These claims are supported by the empirical evidence obtained from our preliminary experiments. The decomposition branch-and-cut algorithm based on this approach seems to be promising, and capable of handling moderately large instances and graph collections.

ACKNOWLEDGEMENT

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 OAC-1531128. The research of the third author was partially supported by the Office of Naval Research under Grant N00014-19-1-2329.

REFERENCES

[1] J. Abello, P. M. Pardalos, and M. G. C. Resende. 1999. On maximum clique problems in very large graphs. In *External memory algorithms and visualization*, J. Abello and J. Vitter (Eds.). DIMACS Series on Discrete Mathematics and Theoretical Computer Science, Vol. 50. American Mathematical Society, Boston, MA, USA, 119–130.
[2] R. D. Alba. 1973. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology* 3, 1 (1973), 113–126.
[3] R. Alhajj and J. Rokne (Eds.). 2018. *Encyclopedia of Social Network Analysis and Mining*. Springer, New York.
[4] M. T. Almeida and F. D. Carvalho. 2012. Integer models and upper bounds for the 3-club problem. *Networks* 60 (2012), 155–166. Issue 3.
[5] B. Balasundaram, S. Butenko, and I. V. Hicks. 2011. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research* 59, 1 (January-February 2011), 133–142.
[6] B. Balasundaram, S. Butenko, and S. Trukhanov. 2005. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization* 10, 1 (August 2005), 23–39.
[7] J.-M. Bourjolly, G. Laporte, and G. Pesant. 2000. Heuristics for finding k-clubs in an undirected graph. *Computers & Operations Research* 27 (2000), 559–569.
[8] J.-M. Bourjolly, G. Laporte, and G. Pesant. 2002. An exact algorithm for the maximum k-club problem in an undirected graph. *European Journal of Operational Research* 138 (2002), 21–28.
[9] E. D. Dolan and J. J. Moré. 2002. Benchmarking optimization software with performance profiles. *Mathematical programming* 91, 2 (2002), 201–213.
[10] Gurobi Optimization, LLC. 2020. Gurobi Optimizer Reference Manual. http://www.gurobi.com
[11] D. Jiang and J. Pei. 2009. Mining Frequent Cross-Graph Quasi-Cliques. *ACM Transactions on Knowledge Discovery in Data* 2, 4 (2009), 16:1–42.
[12] B. H. Junker and F. Schreiber (Eds.). 2008. *Analysis of Biological Networks*. Wiley, New York.
[13] Y. Lu, E. Moradi, and B. Balasundaram. 2018. Correction to: Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters* 12, 8 (November 2018), 1959–1969.
[14] R. D. Luce. 1950. Connectivity and generalized cliques in sociometric group structure. *Psychometrika* 15, 2 (1950), 169–190.
[15] Z. Miao and B. Balasundaram. 2020. An Ellipsoidal Bounding Scheme for the Quasi-Clique Number of a Graph. *INFORMS Journal on Computing* 32, 3 (2020), 763–778. Codes/instances online at: https://github.com/baski363/Quasi-clique.
[16] R. J. Mokken. 1979. Cliques, Clubs and Clans. *Quality and Quantity* 13, 2 (1979), 161–173.
[17] E. Moradi and B. Balasundaram. 2018. Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters* 12, 8 (November 2018), 1947–1957.
[18] G Pastukhov, A. Veremyev, V. Boginski, and O. A. Prokopyev. 2018. On maximum degree-based-quasi-clique problem: Complexity and exact approaches. *Networks* 71, 2 (2018), 136–152.
[19] J. Pattillo, N. Youssef, and S. Butenko. 2013. On clique relaxation models in network analysis. *European Journal of Operational Research* 226, 1 (2013), 9–18.
[20] J. Pei, D. Jiang, and A. Zhang. 2005. Mining cross-graph quasi-cliques in gene expression and protein interaction data. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. IEEE, 353 – 356.
[21] J. Pei, D. Jiang, and A. Zhang. 2005. On mining cross-graph quasi-cliques. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD '05)*. ACM, New York, NY, USA, 228–238.
[22] H. Salemi and A. Buchanan. 2020. Parsimonious formulations for low-diameter clusters. *Mathematical Programming Computation* 12, 3 (2020), 493–528.
[23] S. Shahinpour and S. Butenko. 2013. Distance-based clique relaxations in networks: s-clique and s-club. In *Models, Algorithms, and Technologies for Network Analysis*, B. I. Goldengorin, V. A. Kalyagin, and P. M. Pardalos (Eds.). Vol. 59. Springer, New York, 149–174.
[24] K. Sim, G. Liu, V. Gopalkrishnan, and J. Li. 2011. A case study on financial ratios via cross-graph quasi-bicliques. *Information Sciences* 181, 1 (2011), 201–216.
[25] S. Trukhanov, C. Balasubramaniam, B. Balasundaram, and S. Butenko. 2013. Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations. *Computational Optimization and Applications* 56, 1 (September 2013), 113–130. https://doi.org/10.1007/s10589-013-9548-5
[26] A. Veremyev and V. Boginski. 2012. Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European Journal of Operational Research* 218, 2 (2012), 316–326.
[27] A. Verma, A. Buchanan, and S. Butenko. 2015. Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing* 27, 1 (2015), 164–177.