MINING LOW-DIAMETER CLUSTERS CONSERVED IN GRAPH COLLECTIONS

By

HAO PAN

Bachelor of Science in Industrial Engineering
Southwest Jiaotong University
Chengdu, Sichuan, China
2016

Bachelor of Science in Industrial Engineering
Oklahoma State University
Stillwater, Oklahoma
2016

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2021

MINING LOW-DIAMETER CLUSTERS CONSERVED IN GRAPH COLLECTIONS

Dissertation Approved:

Dr. Balabhaskar Balasundaram

Dissertation Advisor

Dr. Austin Buchanan

Dr. Juan S Borrero

Dr. Mahdi Asgari

ACKNOWLEDGMENTS

Firstly, I would like to sincerely thank my advisor Dr. Balabhaskar Balasundaram for providing me with the opportunity to pursue a doctoral degree under his supervision. I got to know Dr. Balasundaram in spring 2016, and he was my undergraduate program senior design project mentor at that time. Since I was applying for graduate school that semester, I emailed Dr. Balasundaram, asking for a recommendation letter. Soon afterwards, I learned that he was also hiring graduate students. After a meeting with him explaining what his research was about, I decided to join and work with him as a PhD student. During my graduate program, Dr. Balasundaram has been a great mentor to me. He is knowledgeable, receptive, and considerate. He put me into two industrial projects so that I could get myself prepared to pursue a career in industry. When it comes to research, he cares about my own interests and preferences. Dr. Balasundaram is also a good instructor. I took several courses with him, and he was always patient and clear when explaining difficult concepts, sophisticated algorithms, and everything. Without his guidance and support, I would not be able to finish my graduate coursework and defend my dissertation.

Secondly, I want to thank my committee members, Drs. Austin Buchanan, Juan S Borrero, and Mahdi Asgari for providing me with their invaluable insights and guidance during my PhD studies. I would also like to thank the School of Industrial Engineering and Management (IEM) at Oklahoma State University (OSU) and especially Dr. Sunderesh Heragu, department head of IEM, for the wonderful learning environment, and generous financial and administrative support. My thanks are also due to Ms. Laura Brown for her continuous support since I joined IEM. I also want to acknowledge all of my friends and

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

colleagues at OSU for their support and encouragement throughout this process.

Finally, I would like to thank my mother Meilan Ge and my father Zhenjun Pan for always being supportive, open minded, and encouraging me to pursue my dreams. Words cannot express my thankfulness and love to my parents.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: Hao Pan

Date of Degree: December 2021

Title of Study: MINING LOW-DIAMETER CLUSTERS CONSERVED IN GRAPH COLLECTIONS

Major Field: Industrial Engineering And Management

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 (experimental) 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 dissertation, we consider the canonical optimization problem of finding a cross-graph k-club of maximum cardinality. The overall goal of this dissertation is to develop integer programming approaches to solve the problem. We establish computational complexity of the problem and its related problems. We introduce a naive extension of the cut-like formulation for the maximum k-club problem and offer ideas to strengthen it. We introduce valid inequalities for the problem and extend existing inequalities valid for the single-graph problem to the cross-graph setting. We introduce algorithmic ideas to solve this problem using a decomposition branch-and-cut algorithm. For scale reduction, we explore preprocessing procedures and extended formulations. We assess computational effectiveness of the techniques we propose and evaluate their performance on benchmark instances.

We introduce and study in this dissertation, *the maximum k-club signature problem*, which aims to find a maximum cardinality cross-graph k-club in τ consecutive graphs in a sequence of graphs, where the parameter τ is specified by the user. We propose a "moving window" method that solves a sequence of several maximum cross-graph k-club problems, and assess the performance of the approaches we propose in solving the signature variant.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

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 (Alhajj and Rokne, 2018). 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 (Junker and Schreiber, 2008).

To model tightly knit clusters in graphs, the clique and its graph-theoretic relaxations have been extensively studied (Pattillo et al., 2013b). Major categories of clique relaxation models include the distance based relaxations like k-clique and k-club (Luce, 1950; Alba, 1973; Mokken, 1979), and edge count, degree, and density based relaxations like k-defective clique (Trukhanov et al., 2013), k-plex (Balasundaram et al., 2011), and quasi-clique (Pattillo

Portions of this chapter are reprinted with permission from:
Balasundaram, B., Borrero, J. S., and Pan, H. (2022). Graph signatures: Identification and optimization. *European Journal of Operational Research*, 296(3):764–775.

et al., 2013a), respectively. These clique relaxations have found applications in modeling clusters in different fields.

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.

1.1 Motivation

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. The motivating applications in social and biological network analysis as well as in other areas often yield time-series data, which in turn corresponds to a sequence of graphs over time in which the nodes and the edges change (Hu et al., 2005; Li et al., 2011; Hellmann and Staudigl, 2014; Paranjape et al., 2017). Independently analyzing each snapshot graph ignores dynamic characteristics like the persistence of a pattern in time.

Alternatively, relationships between a group of nodes may be different under different (experimental) conditions, which corresponds to a collection of graphs. In this condition, we say a collection instead of a sequence of graphs because order is of no significance. Jointly mining node relationships under different conditions, or a collection of graphs, might uncover novel clusters that cannot be found by individually analyzing each condition, or each graph. 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 (Pei et al., 2005b). 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 (Pei et al., 2005a).

In this dissertation, we consider a *cross-graph k-club model* to represent clusters that are conserved in *a collection of graphs* and induce low-diameter subgraphs (with diameter bounded above by a positive integer k). A graph collection may represent temporal graphs with an implicit ordering, or may be obtained under different conditions without any natural ordering. Although the focus of this dissertation is on clusters that induce low-diameter graphs, one may investigate any clique relaxation or other graph properties in the cross-graph setting.

1.2 Notations and Definitions

For a simple, undirected graph G, we use $V(G) \coloneqq \{1, 2, \ldots, n\}$ 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 use $\text{dist}_G(i, j)$ to denote the distance between nodes i and j in graph G and define its diameter as $\text{diam}(G) \coloneqq \max\{\text{dist}_G(i, j) : i, j \in V(G)\}$.

Definition 1 (Luce (1950))**.** 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 (Mokken (1979))**.** Given a graph G and a positive integer k, a subset of nodes $S \subseteq V(G)$ is called a *k-club* if $\text{diam}(G[S]) \leq k$.

When $k = 1$, k-clique and k-club coincide with the clique model, essentially a subset of nodes inducing a complete subgraph. Node set $\{2, 3, 4\}$ in Figure 1 forms a clique. 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

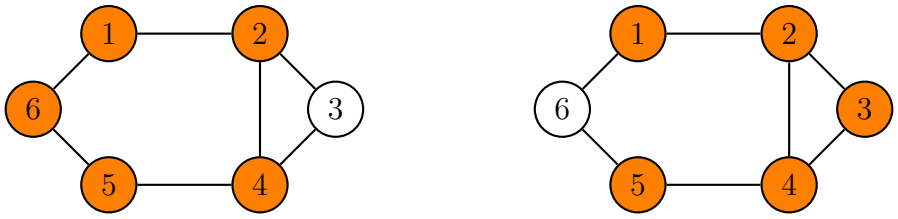


Figure 1: The set $\{1, 2, 4, 5, 6\}$ forms a 2-club; the set $\{1, 2, 3, 4, 5\}$ forms a 2-clique, but does not form a 2-club Alba (1973).

counterpart of the k-club, based on the cross-graph quasi-clique model introduced by Pei et al. (2005b), 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 $V(\mathcal{G}) \coloneqq \{1, 2, \ldots, n\}$.

Definition 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 dissertation focuses on *the maximum p-graph k-club problem*, which seeks to find a p-graph k-club of maximum cardinality given a graph collection. 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 the graph model as a cross-graph k-club (Pei et al., 2005b).

A closely related problem to the maximum p-graph k-club problem is *the maximum k-club signature problem* on a graph sequence. The special case of the problem when $k = 2$ has been introduced and preliminarily tackled by Balasundaram et al. (2022). In contrast to the p-graph k-club problem, order of graphs is of significance in the signature problem. Given a graph sequence $\mathcal{G} = (G_1, G_2, \ldots, G_T)$ and positive integers k and τ, a maximum k-club signature problem seeks to find a maximum cardinality τ-persistent k-club signature of $\mathcal{G}$, which is defined next.

Definition 4. A subset of nodes $S \subseteq V(\mathcal{G})$ is called a *τ-persistent k-club signature* if S is a k-club in at least τ consecutive graphs in $\mathcal{G}$.

The k-club signature problem and the cross-graph k-club problem are closely related because the former can be solved by solving several cross-graph k-club problems, i.e., a maximum k-club signature problem can be decomposed into a sequence of maximum cross-graph k-club problems. Due to this relationship, we verify the computational effectiveness of the techniques we introduce for solving the maximum cross-graph k-club in solving the signature counterpart in Chapter VI.

We acknowledge that part of results in Chapters I, II, VI has appeared in Balasundaram et al. (2022).

CHAPTER II

LITERATURE REVIEW

The (1-graph) maximum k-club problem is a well-known NP-hard problem (Bourjolly et al., 2002; Balasundaram et al., 2005). A comprehensive survey on the complexity results and algorithmic approaches for the maximum k-club problem can be found in Shahinpour and Butenko (2013). In Section 2.1, we briefly review integer programming (IP) approaches to the maximum k-club problem in the literature, and in Section 2.2 we review prevailing works on cross-graph models. Because we are solving the maximum k-club signature problem as a case study for the application of the techniques for the maximum cross-graph k-club problem, we also review works on mining temporal graphs in Section 2.3.

2.1 IP Approaches to the Maximum k-Club Problem

The first IP formulation in the literature for the maximum k-club problem was introduced by Bourjolly et al. (2002). It is known as the chain formulation because it introduces a binary variable for each path of length less than or equal to k connecting a nonadjacent pair of nodes i and j. For the special case of $k = 2$, the aforementioned binary variables can be replaced by those binary variables for the inclusion of the common neighbors of a nonadjacent pair of nodes. In doing so, we obtain the so-called common neighbor formulation (2.1.1) for the maximum 2-club problem. We use $\overline{G}$ and $N_G(u)$ to denote the complement of graph G and

Portions of this chapter are reprinted with permission from:
Balasundaram, B., Borrero, J. S., and Pan, H. (2022). Graph signatures: Identification and optimization. *European Journal of Operational Research*, 296(3):764–775.

the neighbors 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 use the short form $x(V(\mathcal{G}))$ to denote $\sum_{u \in V(\mathcal{G})} x_u$. The common neighbor formulation has $O(n^2)$ constraints.

$$\max\ x(V(G)) \tag{2.1.1a}$$

$$\text{s.t.} \quad x_u + x_v - x(N_G(u,v)) \leq 1 \quad \forall uv \in E(\overline{G}) \tag{2.1.1b}$$

$$x_u \in \{0,1\} \quad \forall u \in V(G). \tag{2.1.1c}$$

Note that 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 formulation as *the conjunctive formulation*. For example, if $\mathcal{G} = \{G, H\}$, the maximum 2-graph 2-club problem admits formulation (2.1.2).

$$\max\ x(V(\mathcal{G})) \tag{2.1.2a}$$

$$\text{s.t.} \quad x_u + x_v - x(N_G(u,v)) \leq 1 \quad \forall uv \in E(\overline{G}) \tag{2.1.2b}$$

$$x_u + x_v - x(N_H(u,v)) \leq 1 \quad \forall uv \in E(\overline{H}) \tag{2.1.2c}$$

$$x_u \in \{0,1\} \quad \forall u \in V(\mathcal{G}). \tag{2.1.2d}$$

Because path enumeration gets increasingly challenging as k takes values bigger 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 $k \geq 3$.

Veremyev and Boginski (2012) introduced two polynomial-sized IP formulations with one using binary variables and the other using integer variables. Both of them are fully described by $O(kn^2)$ variables and constraints, they are known to be the first compact formulations for the maximum k-club problem for general k. These two formulations are obtained by

linearizing a polynomial formulation in which each term is a monomial of up to k variables. The authors circumvent path enumeration while simultaneously enforcing that at least one path of length less than equal to k between any selected pair of nodes must also be selected.

Moradi and Balasundaram (2018) (see also Lu et al. (2018)) proposed a decomposition and branch-and-cut (BC) algorithm to find a maximum k-club. This algorithm employs canonical hypercube cuts (CHC) as delayed constraints. The algorithm begins with solving a maximum k-clique IP formulation as the master relaxation. A CHC will be added to the formulation as a delayed constraint when the algorithm encounters an integral solution that does not correspond to a k-club.

Salemi and Buchanan (2020) introduced a cut-like formulation and a path-like formulation by using length-bounded separators and connectors. Generally, the cut-like formulation could use exponentially many constraints, but only n binary variables. The computational superiority of this formulation has been demonstrated by the reported numerical results, which makes it the state-of-the-art mathematical programming approach to solve the maximum k-club problem for arbitrary k.

2.2 Cross-Graph Models in Graph Mining

Previous works on cross-graph models in the literature are limited. To extract hidden patterns crossing multiple pieces of data, Pei et al. (2005b,a) investigated mining cross-graph quasi-cliques and developed algorithms to enumerate every cross-graph quasi-clique across multiple graphs. Jiang and Pei (2009) extended this work to a more general problem of finding "frequent" cross-graph quasi-clique. In this setting, the detected clusters are required to form a quasi-clique in at least a fixed number of graphs.

Sim et al. (2011) 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. Models and methods for mining conserved clusters in a collection of graphs without strictly imposing the cross-graph requirement can also be found in the literature (Jiang et al., 2013a; Viard et al., 2016; Himmel et al., 2017; Charikar et al., 2018; Bentert et al., 2019; Semertzidis et al., 2019).

2.3 Temporal Graph Mining

As noted in Section 1.1, individually analyzing each snapshot graph in a graph sequence over time ignores dynamic characteristics of the sequence. Mining a sequence of graphs simultaneously might uncover novel clusters which could not be found by independently mining each graph. Works that recognize and aim to address these challenges have appeared in literature. A popular approach involves finding frequent subgraphs in multiple graphs (Jiang et al., 2013b), similar to the use of frequently occuring network motifs in social and biological network analysis (Paranjape et al., 2017). The sequence of graph snapshots are referred to as *relational graphs* or *transactional graphs* in this literature. In this setting, one seeks to find a subgraph, typically required to be dense or connected, that is recurrent, i.e., present in at least a minimum number of snapshots; or alternately, find a subgraph of an auxiliary graph (that summarizes the information) whose edges are present in at least a minimum number of snapshots (cf. Kuramochi and Karypis (2005); Hu et al. (2005); Yan et al. (2005)). However, counting occurrences alone may not be a satisfactory surrogate for persistency, as it is insensitive to the ordering of the snapshots.

More recently, some authors have taken the following approach to find Δ-*cliques* in temporal networks (Viard et al., 2016; Himmel et al., 2017). Consider a subset of nodes C and an interval of consecutive graph snapshots $\mathcal{I}$ that share a common node set. The subset C is called a Δ-clique in $\mathcal{I}$ if every pair of nodes in C are adjacent in at least one snapshot out of every Δ consecutive snapshots in $\mathcal{I}$. In other words, any pair of nodes in a Δ-clique over an interval of snapshots cannot be non-adjacent for more than Δ consecutive snapshots. The general approach here is to impose a minimum periodicity in the "atomic"

property (required of each vertex, edge, vertex-pair etc.) that the subgraph must satisfy in the static counterpart. Although when $\Delta > 0$, it is possible that C may not be a clique in any graph in the subsequence $\mathcal{I}$, when $\Delta = 0$, C is a clique in every snapshot in $\mathcal{I}$.

Viard et al. (2016) propose an algorithm to enumerate all maximal Δ-cliques, where maximality is by inclusion with respect to both C and $\mathcal{I}$. Viard et al. (2018) generalize their Δ-clique model for instantaneous link streams to enumerating maximal cliques that are persistent over a duration. Bentert et al. (2019) extended the Δ-clique approach to Δ-k-plexes, based on a clique relaxation model for cohesive social subgroups in social network analysis (Seidman and Foster, 1978). The recent work of Latapy et al. (2018) to generalize many basic graph theory concepts to deal with temporal networks, or what they refer to as *stream graphs*, is also of interest in this context.

Another branch of relevant literature introduces the following ideas. Jethava and Beerenwinkel (2015) considered the problem of finding a *densest common subgraph* (DCS) in a sequence of graphs with a common node set. That is, a subset of nodes that maximizes the minimum average degree[1] over the subgraphs it induces in every graph in the sequence. Semertzidis et al. (2019) extended this idea in two directions: first, they introduce four variants of the DCS problem by considering average or minimum degree of the subgraph induced by a subset of nodes, aggregated into an objective function by considering the minimum or average over all the snapshots in the sequence; second, they introduce a relaxation that requires the subset of nodes to optimize one of the measures of density over a subsequence of k snapshots. It is important to note that the k snapshots need not be consecutive in their framework. The four variants introduced by Semertzidis et al. (2019) were further investigated by Charikar et al. (2018), with both studies offering greedy algorithms, approximation and complexity results for these problems.

Balasundaram et al. (2022) introduced a graph-theoretic paradigm called a graph sig-

[1]Note that the authors define as *density*, the number edges in the induced subgraph divided by the order, i.e., half the average degree of the induced subgraph. Hence, the density metric they optimize is proportional to average degree of the induced subgraph.

nature to describe persistent patterns in temporal networks, and proposed a dynamic programming based algorithmic framework, the moving window method, for the graph signature identification problem. The authors demonstrated how this generic framework can be tailored to exploit property-specific decomposition and scale reduction techniques, and reported computation results on clique, 2-club, and core signature problems. We extend some of these results for general k-club signatures in this dissertation.

CHAPTER III

RESEARCH STATEMENT

The overall goal of this dissertation is to develop integer programming approaches to solve the maximum p-graph k-club problem in an effective manner on established testbeds. We also aim to verify the effectiveness of the developed techniques by solving the maximum k-club signature problem as a case study. We list our main research objectives below, and then describe the organization of the dissertation.

(1) Establish computational complexity of the maximum p-graph k-club and associated problems.

(2) Investigate IP formulations and valid inequalities for the maximum p-graph k-club problem.

(3) Investigate effective BC schemes for the maximum p-graph k-club problem.

(4) Verify the computational effectiveness of the developed techniques in solving the maximum k-club signature problem.

We organize the rest of the dissertation in the following manner. In Chapter IV, we establish computational complexity of three problems, the maximum p-graph k-club problem, bounded enlargement of a k-club, and bounded enlargement of a p-graph k-club. The latter two problems will be defined in the same chapter. Each of the three problems turns out to be NP-hard to solve. Interestingly, the first and third problems share a common reduction procedure devised for showing the NP-hardness of the maximum k-clique and k-club problems

in Balasundaram et al. (2005). In Chapter V, we present IP formulations and valid inequalities for the maximum p-graph k-club problem. We also extend existing valid inequalities for the maximum k-club problem to the p-graph setting. In addition, we introduce algorithmic approaches to the maximum k-club signature problem based on techniques established for the maximum cross-graph k-club problem. In Chapter VI, we conduct a computational study to assess and compare the approaches introduced, BC algorithms based on different IP formulations, for solving the maximum p-graph k-club problem. We also introduce preprocessing procedures for scale reduction. In the last section of the chapter, we verify the computational effectiveness of the established techniques for the maximum p-graph k-club problem in solving the maximum k-club signature problem. In Chapter VII, we conclude the dissertation and discuss future research directions.

CHAPTER IV

COMPUTATIONAL COMPLEXITY

The (1-graph) maximum k-club problem is NP-hard for every value of parameter k fixed in the problem (Bourjolly et al., 2002). 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. In this chapter, we present a direct proof showing that the decision version of the maximum p-graph k-club problem is NP-complete for every fixed positive integer p and k. In addition, we show that the bounded enlargement of a k-club for every fixed k is NP-hard and so is the bounded enlargement of a p-graph k-club for every fixed p and k. Bounded enlargement of a k-club is of interest because it is related to the maximality testing of a k-club, which is proven to be coNP-complete for every fixed $k \geq 2$ (Pajouh and Balasundaram, 2012).

In the proofs, we reduce the Clique problem defined below to the problems of interest, and our reduction procedure is based on reductions used by Balasundaram et al. (2005).

Clique	
Input:	A graph G, a positive integer ℓ.
Question:	Does there exist a clique of size at least ℓ in G?

4.1 Bounded Enlargement of a k-Club

Verifying maximality by inclusion of a k-club, or maximality testing a k-club defined below, is proved to be coNP-complete for every fixed $k \geq 2$ by reduction from the 3-SAT problem

by Pajouh and Balasundaram (2012). When $k = 1$, the problem reduces to maximality testing a clique (MTC), which is solvable in polynomial time by checking if any node outside the clique is adjacent to every node inside.

Maximality Testing A k-Club (MTkC)	
Input:	A graph G, a k-club S.
Question:	Is S a maximal k-club in G?

Equivalently, given a k-club S in G, one may ask if a k-club containing S with exists with size at least $|S| + 1$? The answer to this question is yes if and only if the answer to maximality testing S is no, and consequently this problem is NP-complete. Note that a k-club S' containing S with $|S'| \geq |S| + 1$ may exist in G even though no k-club exists that contains exactly one more node. This is due to the *nonhereditary* nature of k-clubs, i.e., a proper subset of a k-club is not necessarily a k-club (Mokken, 1979); see Figure 2.

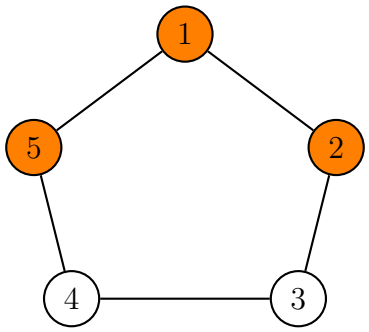


Figure 2: Set $\{1, 2, 3, 4, 5\}$ is the only 2-club strictly containing 2-club $\{1, 2, 5\}$.

The equivalent question discussed above is a special case of the bounded enlargement of a k-club (BEkC) problem defined next. Due to this observation, the BEkC problem is NP-complete for every fixed $k \geq 2$. Observe that when $k = 1$, this problem reduces to bounded enlargement of a clique (BEC). BEC, when $\ell = 1$, always assumes an opposite answer to MTC, and therefore solvable in polynomial time. Next, we offer an alternate proof showing that BEkC is NP-hard for every fixed $k \geq 2$, and we also show that BEC is NP-hard. Consequently, BEkC is NP-hard for any fixed $k \geq 1$.

Bounded Enlargement Of A k-Club (BEkC)	
Input:	A graph G, a k-club S, a positive integer ℓ.
Question:	Does there exist a k-club containing S of size at least $\lvert S\rvert + \ell$ in G?

Proposition 1. The BEkC problem is NP-complete for every fixed $k \geq 2$.

Proof. It is easy to verify that BEkC belongs to class NP. Next, we prove the NP-hardness by reduction from the Clique problem, and we borrow the instance construction procedure directly from Balasundaram et al. (2005).

Let $\langle G, \ell \rangle$ be an instance of the Clique problem. Without loss of generality, we assume G to be connected. We construct another graph G' based on the parity of k.

Case 1: k is odd. For $V(G')$, we take the union of $\lfloor k/2 \rfloor$ copies of $V(G)$, one copy of $E(G)$, and an auxiliary node 0, i.e.,

$$V(G') = \left(\bigcup_{i=1}^{\lfloor k/2 \rfloor} V^i \right) \cup E(G) \cup \{0\},$$

with $V^i = \{1^i, 2^i, \ldots, n^i\}$ denoting the i-th copy of $V(G)$ and v^i denoting the i-th copy of node $v \in V(G)$. For the edge set of G', we let

$$\begin{aligned} E(G') = & \bigcup_{i=1}^{\lfloor k/2 \rfloor - 1} \left\{ \{v^i, v^{i+1}\} : v \in V(G) \right\} \\ & \cup \left\{ \{v^{\lfloor k/2 \rfloor}, e\} : v \in e, e \in E(G) \right\} \\ & \cup \left\{ \{e, 0\} : e \in E(G) \right\}. \end{aligned}$$

As the equations above show, graph G' contains a node $\{u, v\}$ for every edge $\{u, v\} \in E(G)$, and every such node is made adjacent to a new node 0. Every such node is also attached to a pendant path $(u^1, u^2, \ldots, u^{\lfloor k/2 \rfloor})$ at node $u^{\lfloor k/2 \rfloor}$ and another path $(v^1, v^2, \ldots, v^{\lfloor k/2 \rfloor})$ at node $v^{\lfloor k/2 \rfloor}$. See an example of construction in Figures 3 and 4.

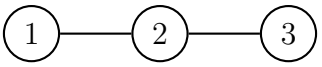


Figure 3: An instance graph G for the CLIQUE problem.

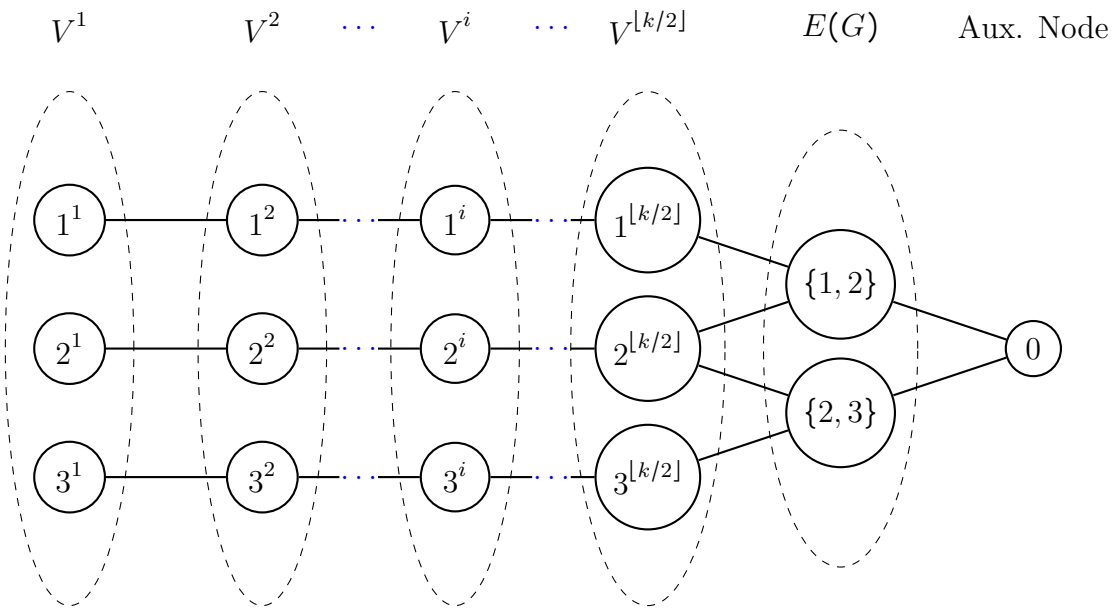


Figure 4: Illustration of graph G' for the BEkC problem when k is odd, corresponding to G in Figure 3.

Case 2: k is even. For $V(G')$, we take the union of $k/2$ copies of $V(G)$, one copy of $E(G)$, that is,

$$V(G') = \left(\bigcup_{i=1}^{k/2} V^i \right) \cup E(G).$$

For the edge set, we let

$$\begin{aligned} E(G') = & \bigcup_{i=1}^{k/2-1} \left\{ \{v^i, v^{i+1}\} : v \in V(G) \right\} \\ & \cup \left\{ \{v^{k/2}, e\} : v \in e, e \in E(G) \right\} \\ & \cup \left\{ \{e_1, e_2\} : e_1, e_2 \in E(G), e_1 \neq e_2 \right\}. \end{aligned}$$

In other words, graph G' contains a node $\{u, v\}$ for every edge $\{u, v\} \in E(G)$, and

every such node $\{u, v\}$ is also attached to a pendant path $(u^1, u^2, \ldots, u^{k/2})$ at node $u^{k/2}$ and another path $(v^1, v^2, \ldots, v^{k/2})$ at node $v^{k/2}$. In addition, there is an edge between e_1 and e_2 for any $e_1, e_2 \in E(G)$ with $e_1 \neq e_2$. See an illustration of graph G' in Figure 5 constructed based on graph G in Figure 3.

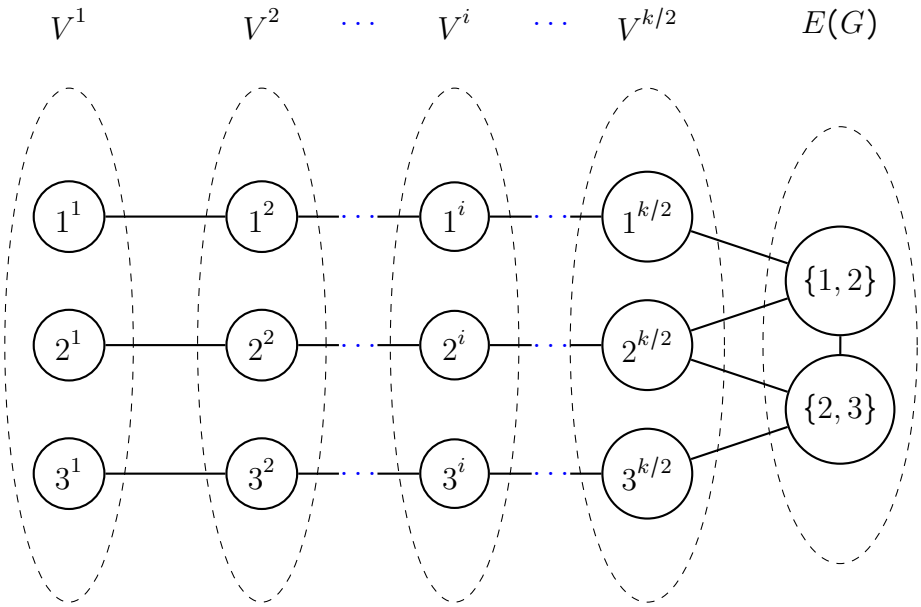


Figure 5: Illustration of graph G' for the BEkC problem when k is even, corresponding to G in Figure 3.

Observe that in both constructions, for odd and even k, $V(G') \setminus V^1$ is a k-club in G'. So given an instance $\langle G, \ell \rangle$ of Clique, we have constructed an instance $\langle G', S', \ell \rangle$ of BEkC with $S' = V(G') \setminus V^1$.

We claim that $\langle G, \ell \rangle$ is a yes-instance of Clique if and only if $\langle G', S', \ell \rangle$ is a yes-instance of BEkC.

Case 1: k is odd.

($\Leftarrow$) Suppose G' has a k-club S'' containing S' with $|S''| \geq \ell + |S'|$. Since S'' is a k-club in G', $\mathrm{dist}_{G'}(u^1, v^1) \leq k$ for any $u^1, v^1 \in S'' \cap V^1$, and so $\{u, v\}$ is a node in G' and an edge in G. Thus, $S = \left\{u \in V(G) : u^1 \in S'' \cap V^1\right\}$ forms a clique in G. Note that $|S|$

is at least ℓ, because $|S''|$ will be strictly less than $\ell + |S'|$ otherwise. Thus, $\langle G, \ell \rangle$ is a yes-instance of CLIQUE.

($\Rightarrow$) Suppose G has a clique S of size ℓ. For arbitrary u, v in S, there exists a node $\{u, v\}$ in H which is on a path of length $2\lfloor k/2 \rfloor$ connecting u^1 and v^1. Let $S'' = \left\{u^1 : u \in S\right\} \cup S'$. Note that $\mathrm{dist}_{G'[S'']}(u^1, v^1) \leq k$ and S'' forms a k-club containing S'. Thus, $\langle G', S', \ell \rangle$ is a yes-instance of BEkC.

Case 2: k is even.

($\Leftarrow$) This direction is identical to its counterpart when k is odd.

($\Rightarrow$) Suppose G has a clique S of size ℓ. For arbitrary u, v in S, there exists a node $\{u, v\}$ in H which is on a path of length k connecting u^1 and v^1. Let $S'' = \left\{u^1 : u \in S\right\} \cup S'$. Note that $\mathrm{dist}_{G'[S'']}(u^1, v^1) \leq k$ and S'' forms a k-club containing S'. Thus, $\langle G', S', \ell \rangle$ is a yes-instance of BEkC.

□

Proposition 2. The BEC problem is NP-complete.

Proof. We prove this result by reduction from the CLIQUE problem.

Let $\langle G, \ell \rangle$ be an instance of the CLIQUE problem. Again, we assume G to be connected. We construct another graph G' by letting $V(G') = V(G) \cup \{0\}$, where 0 is an auxiliary node; and $E(G') = E(G) \cup \left\{\{u, 0\} : u \in V(G)\right\}$. See illustration of G' in Figure 6 constructed from G in Figure 3.

It is easy to verify that G has a clique S of size ℓ if and only if G' has a clique S' containing clique $\{0\}$ with $|S'| = \ell + 1$. Clearly BEC belongs to class NP. □

Propositions 1 and 2 imply the following theorem.

Theorem 1. The BEkC problem is NP-complete for every fixed $k \geq 1$.

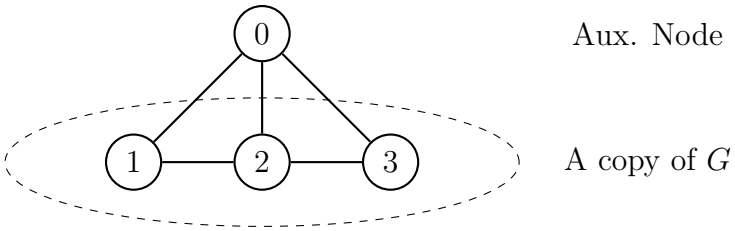


Figure 6: Illustration of graph G' for the BEC problem, corresponding to G in Figure 3.

4.2 The p-Graph Extensions

In the previous section, the BEkC problem is proved to be NP-complete for every fixed $k \geq 1$, and therefore bounded enlargement of a cross-graph k-club (BECkC) defined below is NP-complete for every fixed $k \geq 1$ because it generalizes BEkC. However, this result is trivially true for arbitrary $\mathcal{G}$, because BEkC on an instance $\langle G, S, \ell \rangle$ is equivalent to BECkC on $\langle \mathcal{G}, S, \ell \rangle$ with $\mathcal{G}$ containing p duplicates of G. As duplicate copies of graphs can be removed from the collection, we show that BECkC is NP-complete for every fixed k containing an arbitrary number of distinct graphs. Because of these two reasons, we switch to showing BECkC is NP-complete for every fixed $k \geq 1$ on graph collections containing an arbitrary number of nonidentical graphs. We prove this result by showing the special case where graph collections contain two distinct graphs. For reduction, we present different procedures for $k \geq 2$ and $k = 1$. In addition, we prove analogous results for the p-Graph k-Club (CkC) problem, the decision counterpart of the optimization problem of interest.

Bounded Enlargement Of A p-Graph k-Club (BECkC)	
Input:	A graph collection $\mathcal{G}$, a p-graph k-club S, a positive integer ℓ.
Question:	Does there exist a p-graph k-club containing S of size at least $\lvert S \rvert + \ell$ in $\mathcal{G}$?

Proposition 3. The BECkC problem is NP-complete for every fixed $k \geq 2$ on graph collections containing an arbitrary number of nonidentical graphs.

Proof. It is easy to verify that BECkC is in class NP. We prove the NP-hardness by reduction from the Clique problem.

Let $\langle G, \ell \rangle$ be an instance of the CLIQUE problem. Without loss of generality, we assume G is connected and $|E(\overline{G})| \geq 2$. We partition $E(\overline{G})$ into two subsets, Q_1, Q_2. Define a graph collection $\mathcal{G} = \{G_1, G_2\}$ by letting $V(G_i) = V(G)$ and $E(\overline{G_i}) = Q_i$ for $i = 1, 2$. Due to this construction, $E(G) = E(G_1) \cap E(G_2)$. Next, we define another graph collection $\mathcal{H} = \{H_1, H_2\}$ based on the parity of k, with H_i constructed based on G_i same way G' is constructed based on G in the proof of Proposition 1. Further, we build a third graph collection $\mathcal{J} = \{J_1, J_2\}$, again based on parity of k.

Case 1: k is odd. Let $V(J_i) = V(H_i) \cup E(\overline{G_i})$ and $E(J_i) = E(H_i) \cup \Big\{\{0, e\} : e \in E(\overline{G_i})\Big\}$, for each $i = 1, 2$.

Case 2: k is even. Let $V(J_i) = V(H_i) \cup E(\overline{G_i})$ and $E(J_i) = E(H_i) \cup \{\{e_1, e_2\} : e_1, e_2 \in E(\overline{G_i}), e_1 \neq e_2\}$, for each $i = 1, 2$.

See illustrations of $\mathcal{G}$ in Figure 8, $\mathcal{H}$ in Figures 9 and 11, and $\mathcal{J}$ in Figures 10 and 12, constructed based on G in Figure 7.

Observe in both constructions, for odd and even k, $V(\mathcal{J}) \setminus V^1$ is a 2-graph k-club in $\mathcal{J}$. So given an instance $\langle G, \ell \rangle$ of CLIQUE, we have constructed an instance $\langle \mathcal{J}, S', \ell \rangle$ of BECkC with $S' = V(\mathcal{J}) \setminus V^1$.

We claim that $\langle G, \ell \rangle$ is a yes-instance of CLIQUE if and only if $\langle \mathcal{J}, S', \ell \rangle$ is a yes-instance of BECkC.

Case 1: k is odd.

($\Leftarrow$) Suppose $\mathcal{J}$ has a 2-graph k-club S'' containing S' with $|S''| \geq \ell + |S'|$. Then $\text{dist}_{J_i}(u^1, v^1) \leq k$ for any $u^1, v^1 \in S'' \cap V^1$ and for $i = 1, 2$, and so $\{u, v\}$ is a node in J_i for $i = 1, 2$ and an edge in G. Thus, $S = \{u \in V(G) : u^1 \in S'' \cap V^1\}$ forms a clique in G. Note that $|S|$ is at least ℓ, because $|S''|$ will be strictly less than $\ell + |S'|$ otherwise. Thus, $\langle G, \ell \rangle$ is a yes-instance of CLIQUE.

($\Rightarrow$) Suppose G has a clique S of size ℓ. For arbitrary u, v in S, there exists a node $\{u, v\}$ in J_i which is on a path of length $2\lfloor k/2 \rfloor$ connecting u^1 and v^1 for $i = 1, 2$. Let

$S'' = \left\{u^1 : u \in S\right\} \cup S'$. Note that $\text{dist}_{J_i[S'']}(u^1, v^1) \leq k$ for $i = 1, 2$ and S'' forms a 2-graph k-club containing S'. Thus, $\langle \mathcal{J}, S', \ell \rangle$ is a yes-instance of BECkC.

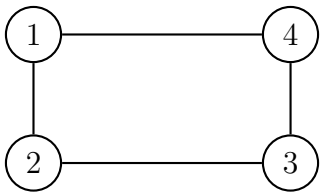
Case 2: k is even.

($\Leftarrow$) This direction is identical to its counterpart when k is odd.

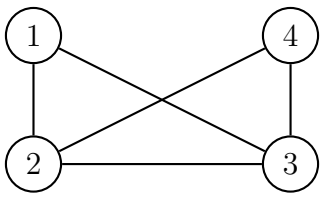
($\Rightarrow$) Suppose G has a clique S of size ℓ. For arbitrary u, v in S, there exists a node $\{u, v\}$ in J_i which is on a path of length k connecting u^1 and v^1 for $i = 1, 2$. Let $S'' = \left\{u^1 : u \in S\right\} \cup S'$. Note that $\text{dist}_{J_i[S'']}(u^1, v^1) \leq k$ for $i = 1, 2$ and S'' forms a 2-graph k-club containing S'. Thus, $\langle \mathcal{J}, S', \ell \rangle$ is a yes-instance of BECkC.

□

Figure 7: An instance graph G for the CLIQUE problem.



G_1



G_2

Figure 8: Illustration of $\mathcal{G} = \{G_1, G_2\}$ constructed based on graph G in Figure 7. Set $E(\overline{G}) = \{\{1, 3\}, \{1, 4\}, \{2, 4\}\}$ is partitioned into $E(\overline{G_1}) = \{\{1, 3\}, \{2, 4\}\}$ and $E(\overline{G_2}) = \{\{1, 4\}\}$.

When $k = 1$, BECkC reduces to bounded enlargement of a cross-graph clique (BECC).

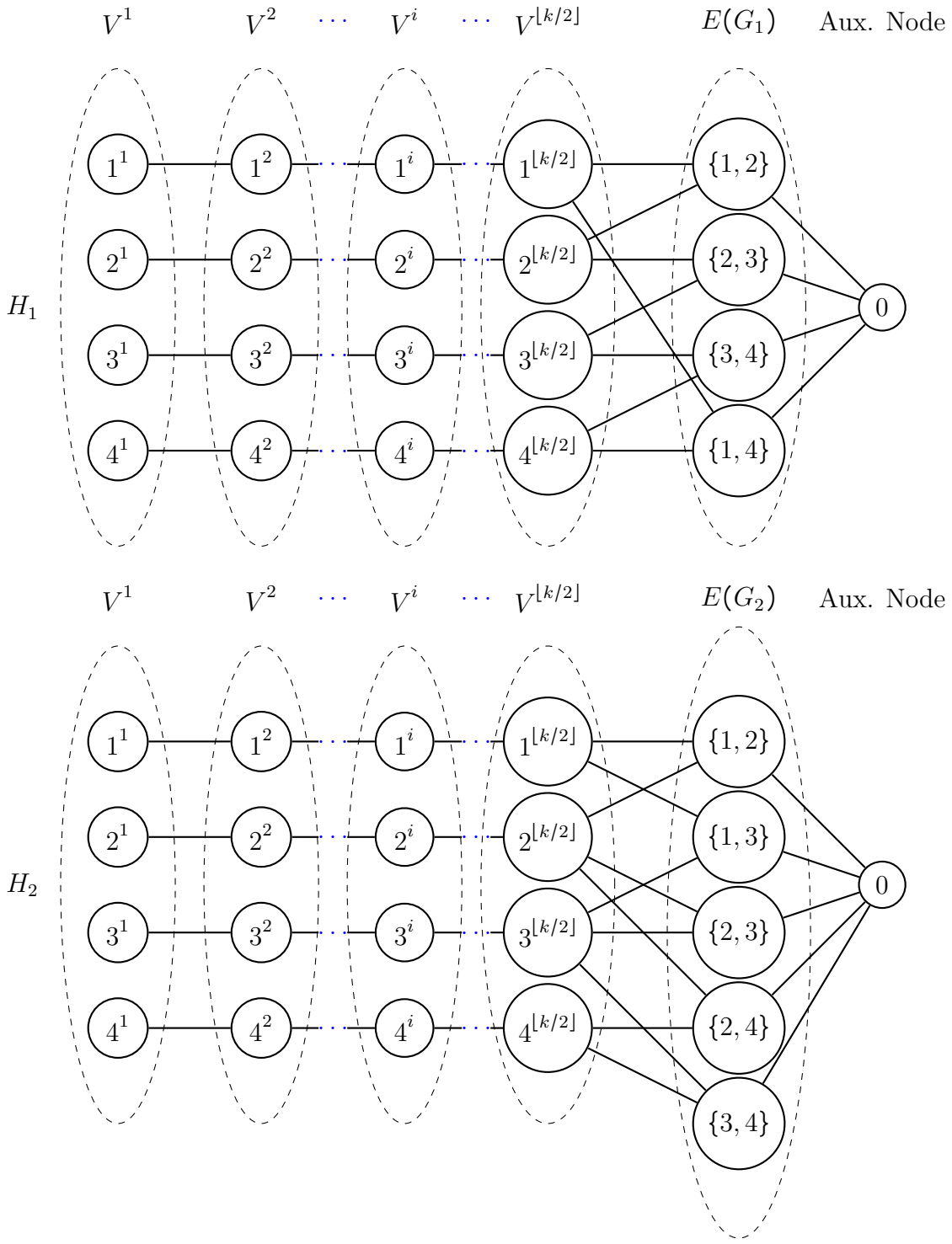


Figure 9: Illustration of $\mathcal{H} = \{H_1, H_2\}$ constructed based on $\mathcal{G}$ in Figure 8, when k is odd.

Proposition 4. The BECC problem is NP-complete on graph collections containing an arbitrary number of distinct graphs.

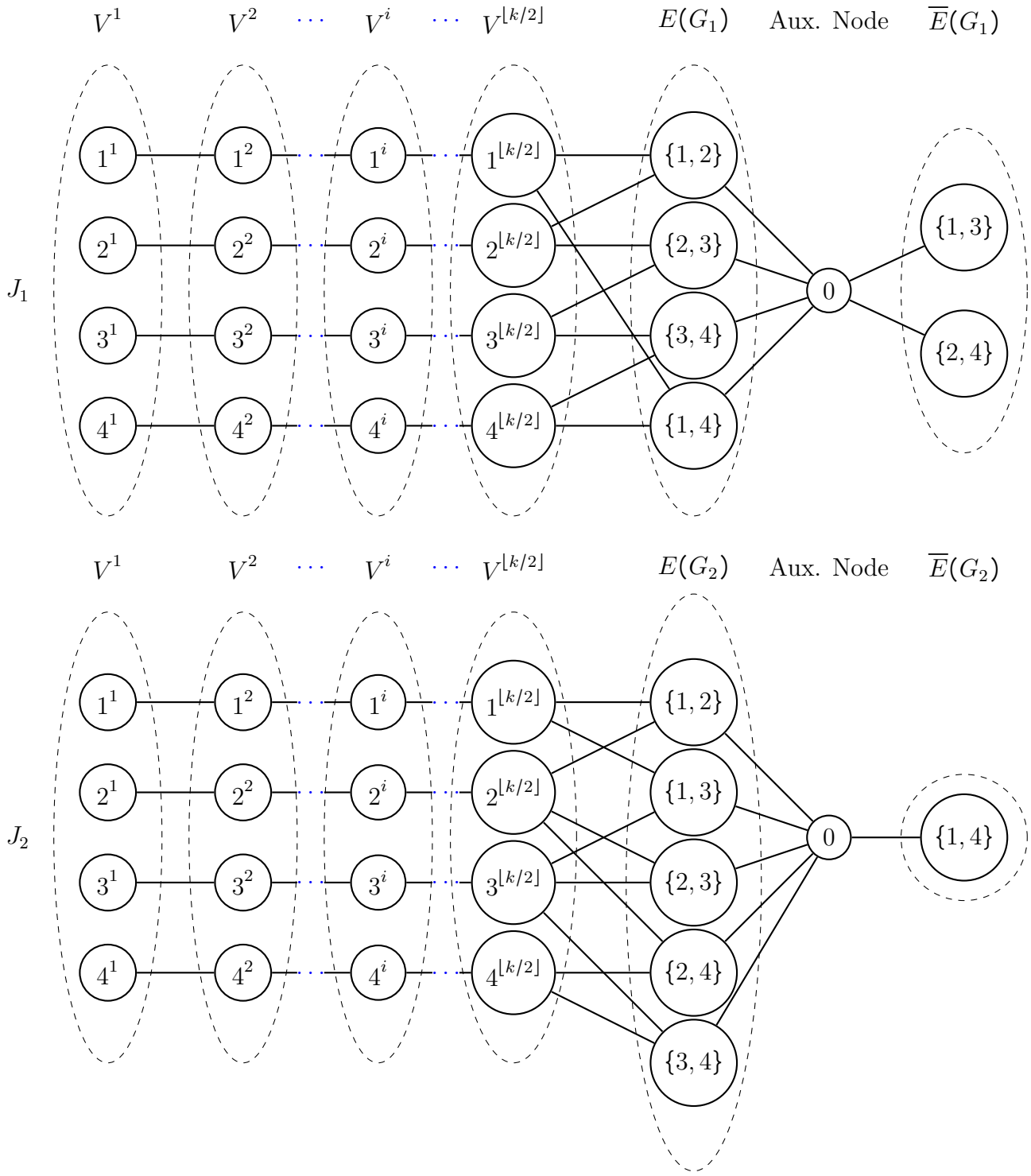


Figure 10: Illustration of $\mathcal{J} = \{J_1, J_2\}$ constructed based on $\mathcal{H}$ in Figure 9, when k is odd.

Proof. Again, we prove this result by reduction from the Clique problem.

Let $\langle G, \ell \rangle$ be an instance of the Clique problem. We assume G to be connected and

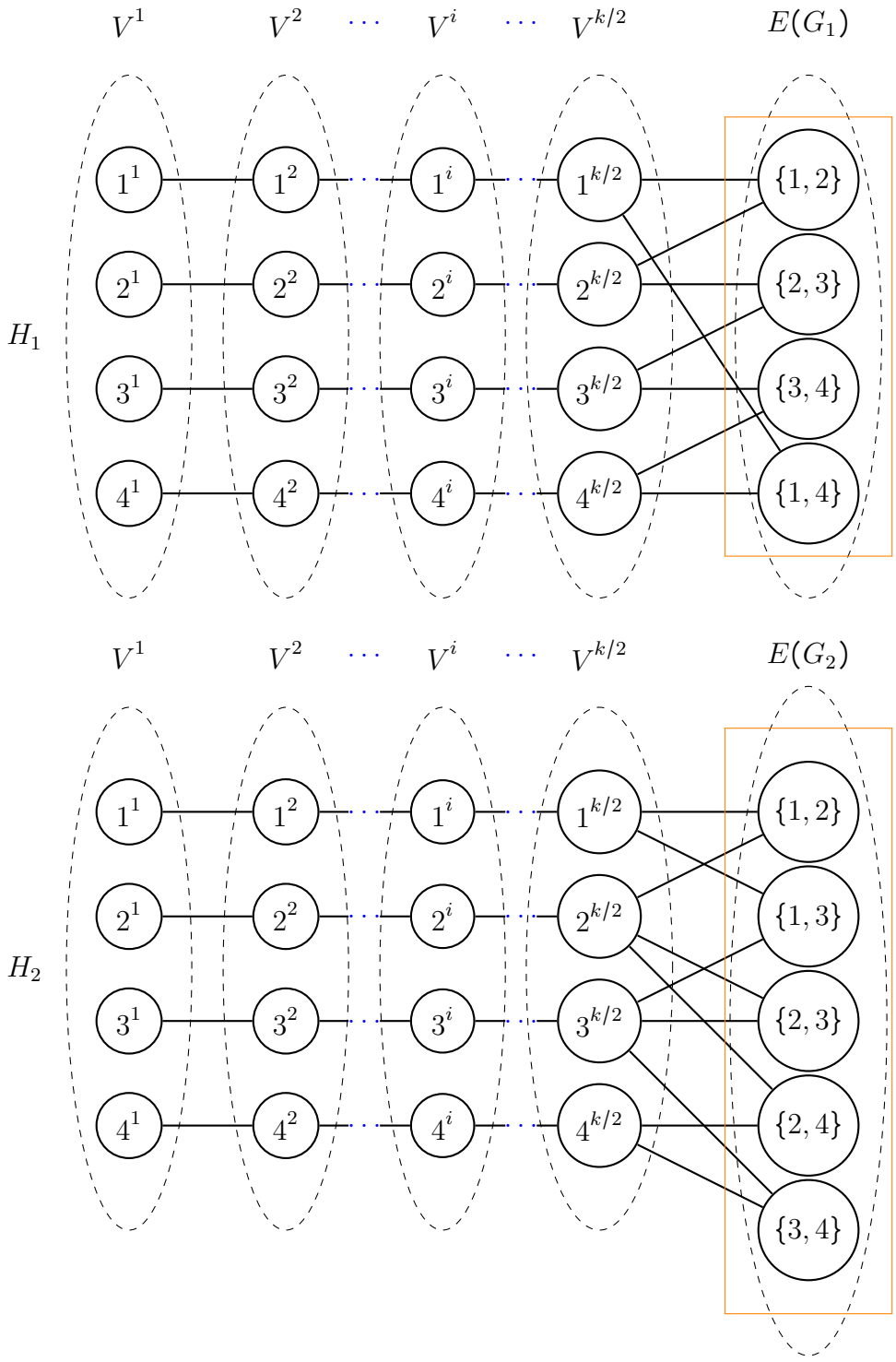


Figure 11: Illustration of $\mathcal{H} = \{H_1, H_2\}$ constructed based on $\mathcal{G}$ in Figure 8, when k is even. To avoid clutter, nodes inside an orange rectangle are pairwise adjacent, but the edges are not shown in the illustration.

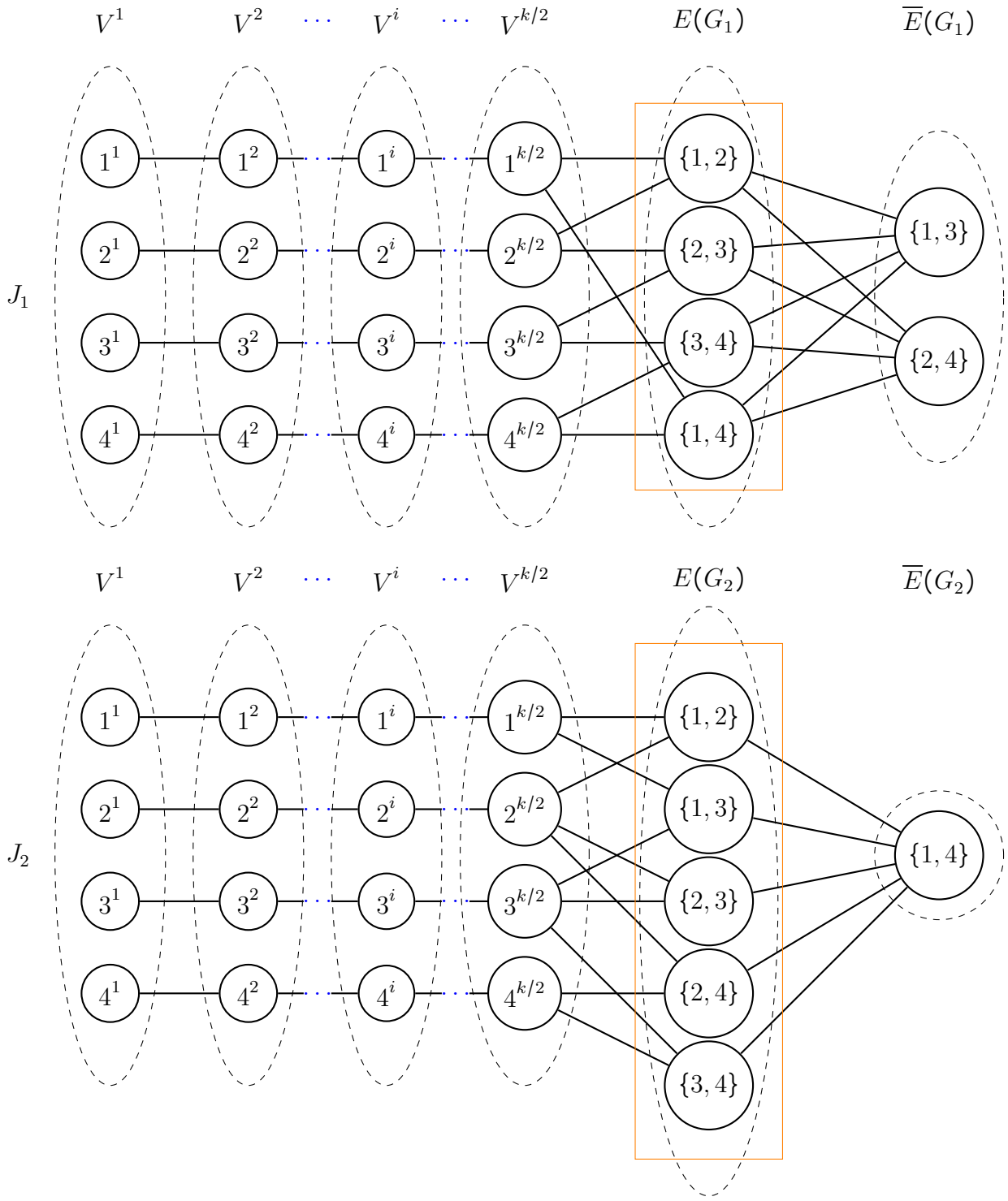


Figure 12: Illustration of $\mathcal{J} = \{J_1, J_2\}$ constructed based on $\mathcal{H}$ in Figure 11, when k is even. To avoid clutter, nodes inside an orange rectangle are pairwise adjacent, but the edges are not shown in the illustration.

$|E(\overline{G})| \geq 2$. We partition $E(\overline{G})$ into 2 subsets, Q_1, Q_2. Define a graph collection $\mathcal{G} = \{G_1, G_2\}$ by letting $V(G_i) = V(G)$, and $E(\overline{G_i}) = Q_i$ for $i = 1, 2$. We construct another graph collection $\mathcal{H}$ by letting $V(H_i) = V(G_i) \cup \{0\}$, where 0 is an anxiliary node; and $E(H_i) = E(G_i) \cup \big\{\{u, 0\} : u \in V(G_i)\big\}$ for $i = 1, 2$. In other words, H_i is constructed based on G_i same way as G' is constructed based on G in the proof of Proposition 2.

It is easy to verify that G has a clique S of size ℓ if and only if $\mathcal{H}$ has a cross-graph clique S' containing cross-graph clique $\{0\}$ with $|S'| = \ell + 1$. Clearly BEC belongs to class NP. □

Propositions 3 and 4 imply the following theorem.

Theorem 2. The BECkC problem is NP-complete for every fixed $k \geq 1$ on graph collections containing an arbitrary number of distinct graphs.

Lastly, we consider the computational complexity of the p-graph k-club problem defined below.

p-Graph k-Club (CkC)	
Input:	A graph collection $\mathcal{G}$, a positive integer ℓ.
Question:	Does there exist a p-graph k-club of size at least ℓ in $\mathcal{G}$?

Proposition 5. The CkC problem is NP-complete for every fixed $k \geq 2$ on graph collections containing an arbitrary number of distinct graphs.

Proof. Let $\langle G, \ell \rangle$ be an instance of the Clique problem and consider the $\mathcal{J}$ constructed in the proof of Proposition 3. The claim in that proof implies that $\langle G, \ell \rangle$ is a yes-instance of Clique if and only if $\langle \mathcal{J}, |S'| + \ell \rangle$ is a yes-instance of CkC with $S' = V(\mathcal{J}) \setminus V^1$, and C$k$C belongs to class NP. □

When $k = 1$, CkC reduces to the cross-graph clique (CC) problem. It is easy to verify that CC on a graph collection $\mathcal{G} = \{G_1, G_2\}$ is equivalent to Clique on the graph $(V(\mathcal{G}), E(G_1) \cap E(G_2))$.

Proposition 6. The CC problem is NP-complete on graph collections containing an arbitrary number of distinct graphs.

Proof. Consider a CLIQUE instance $\langle G, \ell \rangle$. Without loss of generality, we assume G is connected and $E(\overline{G}) \geq 2$. We construct a graph collection $\mathcal{G} = \{G_1, G_2\}$ based on G same way as in the proof of Proposition 4, and the validity of this reduction is easily verified. □

Propositions 5 and 6 imply the following theorem.

Theorem 3. The CkC problem is NP-complete for every fixed $k \geq 1$ on graph collections containing an arbitrary number of distinct graphs.

CHAPTER V

IP FORMULATIONS AND VALID INEQUALITIES

In this chapter, we consider IP approaches to solving the maximum p-graph k-club problem. This involves identifying IP formulations and valid inequalities to the maximum p-graph k-club problem. We introduce an IP formulation which is an extension of the cut-like formulation of Salemi and Buchanan (2020) for the maximum k-club problem to the cross-graph setting. We present ideas which strengthen this formulation, and eventually reach a new formulation based on what we refer to as "pairwise peeling". In addition, we identify valid inequalities for the problem and cross-graph extensions of existing valid inequalities in the literature.

5.1 IP Formulations

Given a graph G and a pair of nonadjacent nodes u and v, define length-k u, v-separator as follows. We use $G \setminus S$ to denote the graph obtained from G by deleting the nodes in S along with its incident edges.

Definition 5 (Salemi and Buchanan (2020))**.** A subset of nodes S is called a *length-k u, v-separator* if $\text{dist}_{G \setminus S}(u, v) > k$.

By this definition, 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. Given a graph collection $\mathcal{G}$,

consider the following optimization problem:

$$\max\ x(V(\mathcal{G})) \tag{5.1.1a}$$

$$\text{s.t.}\quad x_u + x_v - x(S) \leq 1 \quad \forall S \in \mathcal{S}_G(u,v), uv \in E(\overline{G}), G \in \mathcal{G} \tag{5.1.1b}$$

$$x_u \in \{0,1\} \qquad \forall u \in V(\mathcal{G}). \tag{5.1.1c}$$

Formulation (5.1.1) is a conjunction of the cut-like formulation of the maximum k-club problem introduced by Salemi and Buchanan (2020), across all the graphs in $\mathcal{G}$. Henceforth, we refer to formulation (5.1.1) 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 (5.1.1) 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 (5.1.1b) can be replaced by

$$x_u + x_v - x(S \cap D_{uv}) \leq 1, \tag{5.1.2}$$

where,

$$D_{uv} \coloneqq \left\{w \in V(\mathcal{G}) \setminus \{u,v\} : \text{dist}_G(u,w) \leq k \text{ and } \text{dist}_G(v,w) \leq k\ \forall G \in \mathcal{G}\right\} \tag{5.1.3}$$

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 (5.1.2) 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.

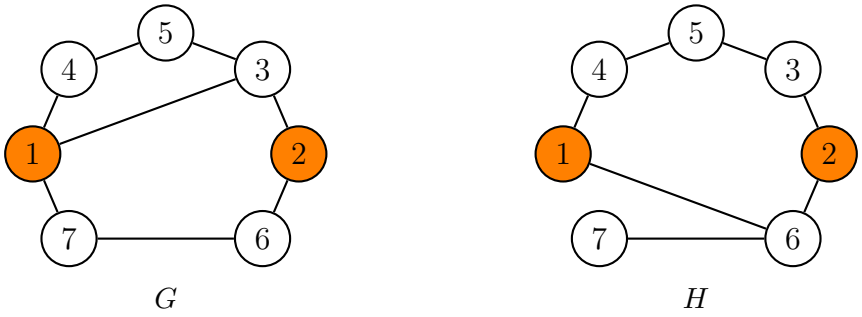


Figure 13: Inequality $x_1 + x_2 \leq 1$ is valid for the problem on $\mathcal{G} = \{G, H\}$ when $k = 2$.

Consider the maximum 2-graph 2-club problem on the graph collection in Figure 13. Formulation (5.1.1), for node pair 1 and 2, includes the constraints $x_1 + x_2 - x_3 \leq 1$ due to G and $x_1 + x_2 - x_6 \leq 1$ due to H. Note that $\text{dist}_H(1, 3) = 3$, and therefore we can tighten the first constraint by intersecting the separator $\{3\}$ with $D_{1,2} = \{5, 6, 7\}$ to obtain the constraint $x_1 + x_2 \leq 1$ that dominates both previous constraints.

The validity of constraints (5.1.2) can also be seen from a lifting point of view. Consider constraint (5.1.1b) for a particular nonadjacent node pair u, v and S. Suppose $S \setminus D_{uv}$ is not empty and $w \in S \setminus D_{uv}$. We are interested in finding the smallest α_w such that inequality $x_u + x_v - x(S \setminus \{w\}) - \alpha_w x_w \leq 1$ remains valid. Thus we need

$$\alpha_w \geq \max\{x_u + x_v - x(S \setminus \{w\}) : x \in \text{CCLUB}(\mathcal{G}), x_w = 1\} - 1,$$

with $\text{CCLUB}(\mathcal{G})$ denoting the p-graph k-club polytope of $\mathcal{G}$, i.e., the convex hull of feasible solutions to formulation (5.1.1). Because $w \in S \setminus D_{uv}$, there exists $G \in \mathcal{G}$ with either $\text{dist}_G(u, w) > k$ or $\text{dist}_G(v, w) > k$. Without loss of generality, we assume the former is true,

and it follows that,

$$\max\{x_u + x_v - x(S \setminus \{w\}) : x \in \text{CCLUB}(\mathcal{G}), x_w = 1\} \leq 1.$$

As a result, the inequality remains valid if $\alpha_w = 0$. If we repeat this procedure on each node in $S \setminus D_{uv}$, the validity of constraints (5.1.2) follows.

If we take a constraint (5.1.2) for a particular nonadjacent node pair u, v and S, and apply the same lifting procedure on a node w with $\text{dist}_{G_1}(u, w) > k$ and $\text{dist}_{G_2}(v, w) > k$ for some $G_1, G_2 \in \mathcal{G}$, then we obtain a lifted coefficient of 1 and hence,

$$x_u + x_v + x_w - x(S \cap D_{uv}) \leq 1 \tag{5.1.4}$$

is a valid inequality. By repeating this argument, we can conclude that $x_u + x_v + x(I) - x(S \cap D_{uv}) \leq 1$ is a valid inequality for $\text{CCLUB}(\mathcal{G})$ if the following conditions hold:

(1) I is a maximal cross-graph distance-k independent set of $\mathcal{G}$, i.e., $\text{dist}_G(a, b) > k$ for every distinct $a, b \in I$ for some $G \in \mathcal{G}$;

(2) for each $w \in I$, there exist $G_1, G_2 \in \mathcal{G}$ (not necessarily distinct) with $\text{dist}_{G_1}(u, w) > k$ and $\text{dist}_{G_2}(v, w) > k$.

Based on the idea of intersecting the length-k u, v-separator S in constraint (5.1.1b) with D_{uv} to obtain a tighter constraint (5.1.2), 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 (5.1.2). 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.

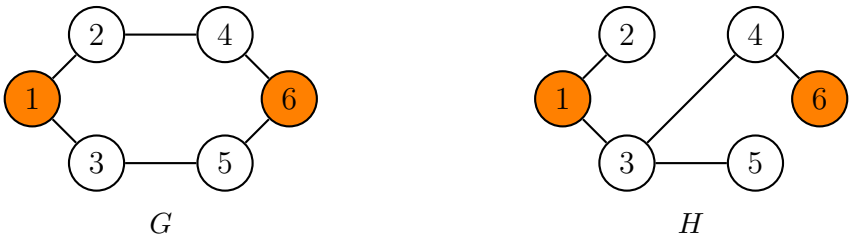


Figure 14: Inequality $x_1 + x_6 \leq 1$ is valid for the problem on $\{G, H\}$ when $k = 3$.

To illustrate this idea, consider the maximum 2-graph 3-club problem on the graph collection in Figure 14. Constraints (5.1.2) 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 $\mathcal{S}_H(1, 6) = \big\{\{3\}, \{4\}\big\}$.

$$x_1 + x_6 - x_3 \leq 1$$

$$x_1 + x_6 - x_5 \leq 1$$

$$x_1 + x_6 - x_3 - x_4 \leq 1$$

$$x_1 + x_6 - x_4 - x_5 \leq 1$$

$$x_1 + x_6 - x_3 \leq 1$$

$$x_1 + x_6 - x_4 \leq 1$$

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 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 $\mathcal{G}$, that is,

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

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
```

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 (5.1.1b) by the following based on the pairwise peeled collection:

$$x_u + x_v - x(S) \leq 1 \quad \forall S \in \mathcal{S}_G(u,v) \text{ and}$$
$$\forall G \in \mathcal{G}_{uv} \text{ such that } uv \in E(\overline{G}), uv \in \mathcal{J}. \tag{5.1.5}$$

Proposition 7. If constraints (5.1.1b) in formulation (5.1.1) are replaced by constraints (5.1.5), 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 (5.1.5) and every binary vector satisfying these constraints also satisfies constraints (5.1.1b). Observe that constraints (5.1.2) and (5.1.5) coincide when $k = 2$. This is because deleting nodes in this manner will not turn a common neighbor of u and v into a nonneighbor of u or v.

Proposition 8. 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 (5.1.5) is at least as strong as constraint (5.1.2) (which in turn dominates constraint (5.1.1b)). Through our computational experiments reported in the next chapter, we assess the gains made by using Algorithm 1 to generate potentially stronger constraints.

The lifting approach to deriving inequalities (5.1.4) also applies here. If we take constraint (5.1.5) for a particular nonadjacent node pair u, v and S, and apply the same procedure on a node w with $\text{dist}_{G_1}(u, w) > k$ and $\text{dist}_{G_2}(v, w) > k$ for some $G_1, G_2 \in \mathcal{G}$, then the

inequality below is valid:

$$x_u + x_v + x_w - x(S) \leq 1. \tag{5.1.6}$$

By repeating this argument, we can conclude that $x_u + x_v + x(I) - x(S) \leq 1$ is a valid inequality for CCLUB($\mathcal{G}$) if the following conditions hold:

(1) I is a maximal cross-graph distance-k independent set of $\mathcal{G}$, i.e., $\text{dist}_G(a, b) > k$ for every distinct $a, b \in I$ for some $G \in \mathcal{G}$;

(2) for each $w \in I$, there exist $G_1, G_2 \in \mathcal{G}$ (not necessarily distinct) with $\text{dist}_{G_1}(u, w) > k$ and $\text{dist}_{G_2}(v, w) > k$.

5.2 Additional Inequalities of Interest

In this section, we consider three classes of inequalities and prove their validity for the corresponding problems. The second and third classes are extensions of their single-graph counterparts which are already known to be valid for the maximum 2-club and k-club problems respectively. For the second class, we present an alternative proof for the validity of its single-graph counterpart so that we can easily extend it to the p-graph setting.

5.2.1 A Special Inequality for the Maximum p-Graph 2-Club Problem

For the maximum p-graph 2-club problem, consider the class of inequalities defined below,

$$x_u + x_v - x\left(\bigcap_{i=1}^{p} N_{G_i}(u, v)\right) \leq 1. \tag{5.2.1}$$

Notice that these inequalities are not valid in general to the maximum p-graph 2-club problem. Consider the example in Figure 15; inequality $x_1 + x_2 - x\left(N_G(1, 2) \cap N_H(1, 2)\right) \leq 1$ is not valid because it is violated by the incidence vector of $\{1, 2, 3, 4\}$, which is a 2-graph 2-club of $\mathcal{G}$.

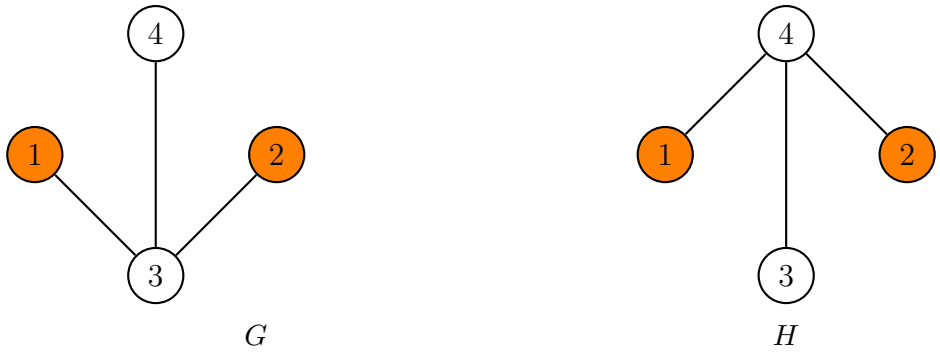


Figure 15: Inequality $x_1 + x_2 - x\left(N_G(1,2) \cap N_H(1,2)\right) \leq 1$ is not valid to the maximum 2-graph 2-club problem on $\mathcal{G} = \{G, H\}$.

Proposition 9 gives a necessary and sufficient condition for the validity of these inequalities. We use $\mathcal{G} \setminus \left(\bigcap_{i=1}^{p} N_{G_i}(u,v)\right)$ to denote the graph collection after deleting node subset $\bigcap_{i=1}^{p} N_{G_i}(u,v)$ from each graph in $\mathcal{G}$.

Proposition 9. An inequality (5.2.1) is valid if and only if there does not exist a p-graph 2-club S with $S \supset \{u,v\}$ in $\mathcal{G} \setminus \left(\bigcap_{i=1}^{p} N_{G_i}(u,v)\right)$.

Proof. We prove both directions by contradiction.

($\Leftarrow$) Suppose the inequality is not valid, then there exists some p-graph 2-club S with its incidence vector x^S violating the inequality, i.e., $x_u^S + x_v^S - x^S\left(\bigcap_{i=1}^{p} N_{G_i}(u,v)\right) > 1$. This implies that $S \supset \{u,v\}$ and $S \cap \left(\bigcap_{i=1}^{p} N_{G_i}(u,v)\right) = \varnothing$.

($\Rightarrow$) Suppose there exists a p-graph 2-club $S \supset \{u,v\}$ in $\mathcal{G} \setminus \left(\bigcap_{i=1}^{p} N_{G_i}(u,v)\right)$, then its incidence vector x^S violates the inequality. □

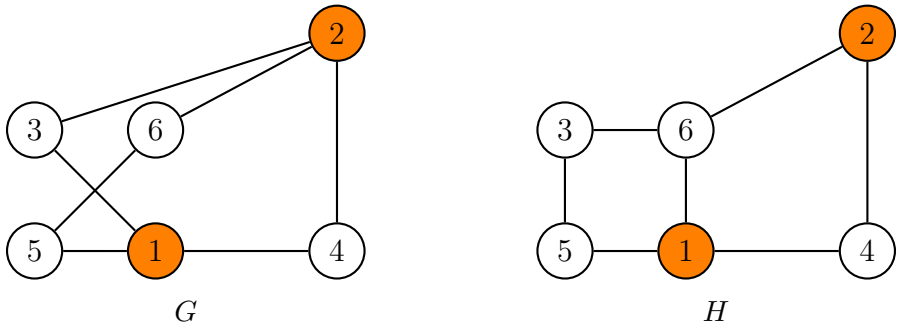


Figure 16: Inequality $x_1 + x_2 - x(N_G(1,2) \cap N_H(1,2)) \leq 1$ is valid to the maximum 2-graph 2-club problem on $\{G, H\}$.

Figure 16 gives an example where the condition in Proposition 9 is satisfied for node pair 1 and 2. See that $N_G(1,2) \cap N_H(1,2) = \{4\}$. In $\{G, H\} \setminus \{4\}$, node 3 and 6 must be selected due to G and H respectively if we want to select 1 and 2 together. The only common neighbor of 1 and 6 is 5 in G, therefore 5 must also be selected. However, $\text{dist}_H(2,5) = 3 > 2$, which means we are not able to select 5. Therefore, there does not exist a 2-graph 2-club which satisfies the necessary and sufficient condition.

5.2.2 The Independent Set Inequality

In this section, we consider the independent set inequality (5.2.2) introduced by Pajouh et al. (2016) for the 2-club polytope. We present a new proof of the validity of inequality (5.2.2), which enables us to extend the result to the p-graph setting. The new proof of Theorem 4 requires Lemma 1.

Theorem 4 ((Pajouh et al., 2016))**.** Given a graph G and an independent set $C \subseteq V(G)$, the inequality,

$$x(C) - \sum_{u \in V(G) \setminus C} \left(|N_G(u) \cap C| - 1\right)^+ x_u \leq 1, \tag{5.2.2}$$

is valid for the maximum 2-club problem on G.

Lemma 1. Given a graph G, an arbitrary 2-club S, and an independent set $C \subset S$, the following inequality is true:

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

Proof. We prove the lemma by induction. If $|C| = 1$, the inequality is trivially true. Assume that the inequality is true for an independent set C such that $|C| \leq \ell$, for some $\ell \geq 2$. Suppose that $|C| = \ell + 1$, and arbitrarily pick a node $a \in C$. We use C_a to denote $C \setminus \{a\}$.

Note that C_a is an independent set and $|C_a| = \ell \geq 2$ and $C_a \subset S$. By induction hypothesis,

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

As $a \in S \setminus C_a$, we obtain the following inequality:

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

Since C is an independent set and contains a, $|N_G(a) \cap C| = 0$, and hence $|N_G(a) \cap C_a| = 0$. As $(|N_G(a) \cap C_a| - 1)^+ = 0$ and given the following inequality,

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

we obtain the inequality,

$$|C| - 1 - \sum_{u \in S \setminus C} (|N_G(u) \cap C_a| - 1)^+ \leq 1. \tag{5.2.3}$$

By hypothesis $C_a \neq \emptyset$, and let b be a node in C_a. Because $a, b \in C \subset S$, $\text{dist}_G(a, b) = 2$, and so $|S \cap N_G(a, b)| \geq 1$. Let w be a node in $S \cap N_G(a, b)$. Since C is an independent set, $w \notin C$, and so $w \in S \setminus C$. Note that $N_G(w) \cap C \supset \{a, b\}$, and therefore $|N_G(w) \cap C| \geq 2$. Since $C = C_a \cup \{a\}$ and $a \in N_G(w)$,

$$|N_G(w) \cap C_a| - 1 = |N_G(w) \cap C| - 2 \geq 0.$$

It follows from inequality (5.2.3) that,

$$\begin{aligned}
& |C| - 1 - \sum_{u \in S \setminus C} \left(|N_G(u) \cap C_a| - 1\right)^+ \\
= \; & |C| - 1 - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a| - 1\right)^+ - \left(|N_G(w) \cap C_a| - 1\right)^+ \\
= \; & |C| - 1 - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a| - 1\right)^+ - \left(|N_G(w) \cap C| - 2\right)^+ \\
= \; & |C| - 1 - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a| - 1\right)^+ - \left(|N_G(w) \cap C| - 2\right) \\
= \; & |C| - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a| - 1\right)^+ - \left(|N_G(w) \cap C| - 1\right)^+ \\
\leq \; & 1.
\end{aligned}$$

As a result,

$$\begin{aligned}
& |C| - \sum_{u \in S \setminus C} \left(|N_G(u) \cap C| - 1\right)^+ \\
= \; & |C| - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C| - 1\right)^+ - \left(|N_G(w) \cap C| - 1\right)^+ \\
\leq \; & |C| \quad - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a| - 1\right)^+ - \left(|N_G(w) \cap C| - 1\right)^+ \\
\leq \; & 1.
\end{aligned}$$

□

Alternate proof of Theorem 4. Let S be an arbitrary 2-club in G and x^S be its incidence vector. It suffices to show that,

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

Note that $C \cap S \subset S$ is an independent set and S is a 2-club. By Lemma 1,

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

Therefore,

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

□

Next, we extend the independent set inequality to the p-graph setting, i.e., showing the p-graph independent set inequality (5.2.4) is valid to the maximum p-graph 2-club problem. We use $N^2_{\mathcal{G}}(u)$ to denote subset of nodes which are within distance 2 from node u in all graphs in $\mathcal{G}$, i.e., $N^2_{\mathcal{G}}(u) = \{v \in V(\mathcal{G}) : \text{dist}_{G_i}(u, v) \leq 2 \quad \forall i = 1, 2, \ldots, p\}$.

Theorem 5. Given a graph collection $\mathcal{G}$ and a set $C \subset V(\mathcal{G})$ that is independent in some graph $G \in \mathcal{G}$, the following inequality,

$$x(C) - \sum_{u \in V(G) \setminus C} (|N_G(u) \cap C \cap N^2_{\mathcal{G}}(u)| - 1)^+ x_u \leq 1 \tag{5.2.4}$$

is valid for the maximum p-graph 2-club problem on $\mathcal{G}$.

Again, we first prove Lemma 2 needed to prove Theorem 5.

Lemma 2. Given a graph collection $\mathcal{G}$, a p-graph k-club S, and a set $C \subset S$ that is independent in some $G \in \mathcal{G}$, the following inequality is true:

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

Proof. We prove this lemma by induction. If $|C| = 1$, the inequality is trivially true. Assume that the inequality is true for $|C| \leq \ell$, for some $\ell \geq 2$. Suppose that $|C| = \ell + 1$. Arbitrarily

pick a node $a \in C$ and let $C_a \coloneqq C \setminus \{a\}$. Note that $C_a \subset S$ is an independent set in G and $|C_a| = \ell \geq 2$. By induction hypothesis,

$$|C_a| - \sum_{u \in S \setminus C_a} \left(|N_G(u) \cap C_a \cap N_{\mathcal{G}}^2(u)| - 1\right)^+ \leq 1.$$

As $a \in S \setminus C_a$, we obtain the following inequality:

$$|C_a| - \sum_{u \in (S \setminus C_a) \setminus \{a\}} \left(|N_G(u) \cap C_a \cap N_{\mathcal{G}}^2(u)| - 1\right)^+ - \left(|N_G(a) \cap C_a \cap N_{\mathcal{G}}^2(a)| - 1\right)^+ \leq 1.$$

Since C is an independent set in G and contains a, $|N_G(a) \cap C| = 0$, and therefore $|N_G(a) \cap C_a| = 0$. Consequently,

$$\left(|N_G(a) \cap C_a \cap N_{\mathcal{G}}^2(a)| - 1\right)^+ = 0,$$

and it follows that,

$$|C_a| - \sum_{u \in (S \setminus C_a) \setminus \{a\}} \left(|N_G(u) \cap C_a \cap N_{\mathcal{G}}^2(u)| - 1\right)^+ \leq 1.$$

Therefore,

$$|C| - 1 - \sum_{u \in S \setminus C} \left(|N_G(u) \cap C_a \cap N_{\mathcal{G}}^2(u)| - 1\right)^+ \leq 1. \tag{5.2.5}$$

Observe that $C_a \neq \emptyset$, and let b be a node in C_a. Because $a, b \in C \subset S$, $\mathrm{dist}_G(a, b) = 2$, and so $|S \cap N_G(a, b)| \geq 1$. Let w be a node in $S \cap N_G(a, b)$. Since C is an independent set in G, $w \notin C$, and so $w \in S \setminus C$. Note that $\{a, b\} \subset N_G(w) \cap C \cap N_{\mathcal{G}}^2(w)$, and so $|N_G(w) \cap C \cap N_{\mathcal{G}}^2(w)| \geq 2$. As a result,

$$|N_G(w) \cap C_a \cap N_{\mathcal{G}}^2(w)| = |N_G(w) \cap C \cap N_{\mathcal{G}}^2(w)| - 1 \geq 1.$$

It follows from inequality (5.2.5) that,

$$\begin{aligned}
& |C| - 1 - \sum_{u \in S \setminus C} \left(|N_G(u) \cap C_a \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ \\
= \; & |C| - 1 - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ - \left(|N_G(w) \cap C_a \cap N^2_{\mathcal{G}}(w)| - 1\right)^+ \\
= \; & |C| - 1 - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ - \left(|N_G(w) \cap C \cap N^2_{\mathcal{G}}(w)| - 2\right)^+ \\
= \; & |C| - 1 - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ - \left(|N_G(w) \cap C \cap N^2_{\mathcal{G}}(w)| - 2\right) \\
\leq \; & 1.
\end{aligned}$$

Therefore,

$$|C| - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ - \left(|N_G(w) \cap C \cap N^2_{\mathcal{G}}(w)| - 1\right)^+ \leq 1 \tag{5.2.6}$$

As a result of inequality (5.2.6),

$$\begin{aligned}
& |C| - \sum_{u \in S \setminus C} \left(|N_G(u) \cap C \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ \\
= \; & |C| - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ - \left(|N_G(w) \cap C \cap N^2_{\mathcal{G}}(w)| - 1\right)^+ \\
\leq \; & |C| - \sum_{u \in (S \setminus C) \setminus \{w\}} \left(|N_G(u) \cap C_a \cap N^2_{\mathcal{G}}(u)| - 1\right)^+ - \left(|N_G(w) \cap C \cap N^2_{\mathcal{G}}(w)| - 1\right)^+ \\
\leq \; & 1.
\end{aligned}$$

□

Proof of Theorem 5. Let S be an arbitrary p-graph 2-club of $\mathcal{G}$ and x^S be its incidence vector.

It suffices to show that the following inequality holds,

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

Note that $C \cap S \subset S$ is an independent set in G and S is a p-graph 2-club. By Lemma 2,

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

Therefore,

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

□

5.2.3 The Cross-Graph Distance-k Independent Set Inequality

In this section, we extend the distance-k independent set inequality that is known to be valid for the maximum k-club problem. The inequality is facet-inducing when the distance-k independent set is maximal by inclusion (Balasundaram et al., 2005; Balasundaram, 2007). In the cross-graph setting, Theorem 6 below, follows trivially from Definition 6 of the p-graph distance-k independent set. However, the arguments showing that it induces a facet do not extend in a straightforward manner. Specifically, for a node $v \notin I$, the maximal cross-graph distance-k independent set, has a path of length at most k to some node $u \in I$ in some graph $G \in \mathcal{G}$. But this u, v-path need not form a cross-graph k-club, make it nontrivial to demonstrate the required number of affinely independent solutions on the face.

Definition 6. A subset of nodes $I \subset V(\mathcal{G})$ is a *p-graph distance-k independent set* if for every pair of distinct nodes $u, v \in I$, there exists a graph $G \in \mathcal{G}$ with $\text{dist}_G(u, v) > k$.

Theorem 6. Given a graph collection $\mathcal{G}$, suppose $I \subseteq V(\mathcal{G})$ is a p-graph distance-k independent set. The *p-graph distance-k independent set inequality* $x(I) \leq 1$ is valid for the

maximum p-graph k-club problem on $\mathcal{G}$.

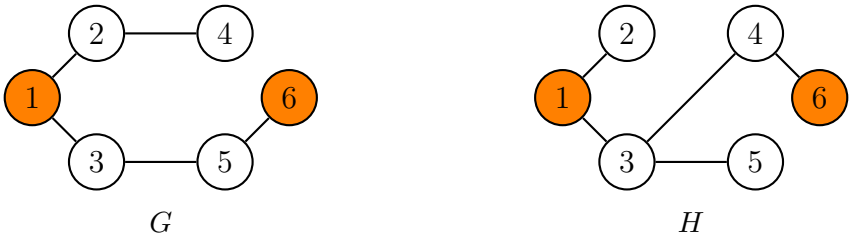


Figure 17: Cross-graph distance-2 independent set $\{1, 6\}$ is maximal, but inequality $x_1 + x_6 \leq 1$ is not facet inducing.

For the maximum p-graph k-club problem, inclusionwise maximality is a necessary but not sufficient facet-inducing condition of a cross-graph distance-k independent set inequality. The necessity is obvious, and Figure 17 gives a counterexample for sufficiency. Observe that distances from node 1 to nodes $2, 3, 4, 5$ are at most 2 in both G and H, and therefore cross-graph distance-2 independent set $\{1, 6\}$ is maximal. However, $x_1 + x_4 + x_6 \leq 1$ is a valid inequality which dominates $x_1 + x_6 \leq 1$, and hence the latter cannot be facet inducing. Because $\text{dist}_G(1, 6) > 2$ and $\text{dist}_G(4, 6) > 2$, it suffices to verify that there does not exist a 2-graph 2-club which simultaneously contains 1 and 4 for the validity of $x_1 + x_4 + x_6 \leq 1$. For 1 and 4 to be contained in a same 2-club in G, node 2 must be selected. However, $\text{dist}_H(2, 4) = 3 > 2$. Therefore, there does not exist a 2-graph 2-club simultaneously containing 1 and 4.

CHAPTER VI

ALGORITHMS AND COMPUTATIONAL STUDY

In this chapter we report results from a computational study to assess and compare techniques we introduce—BC algorithms based on different IP formulations—for solving the maximum p-graph k-club problem (Section 6.1). We also introduce preprocessing procedures for scale reduction in this chapter. In Section 6.2, we verify the computational effectiveness of the techniques we developed for the maximum p-graph k-club problem in solving the maximum k-club signature problem.

6.1 The Maximum p-Graph k-Club Problem

The goal of this section is to compare the performance of a general purpose IP solver when using CCF and PPCF introduced in Chapter V 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 algorithms that use the same master problem based on cross-graph k-cliques defined below.

Definition 7. Given a graph collection $\mathcal{G}$, 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 $\mathcal{G}$.

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

Portions of this chapter are reprinted with permission from:
Balasundaram, B., Borrero, J. S., and Pan, H. (2022). Graph signatures: Identification and optimization. *European Journal of Operational Research*, 296(3):764–775.

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:

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

and we use it in our master IP formulation (6.1.1):

$$\max x(V(\mathcal{G})) \tag{6.1.1a}$$

$$\text{s.t. } x_u + x_v \leq 1 \qquad \forall uv \in \hat{E} \tag{6.1.1b}$$

$$x_u \in \{0, 1\} \qquad \forall u \in V(\mathcal{G}) \tag{6.1.1c}$$

The two decomposition branch-and-cut algorithms, simply referred to as CCF and PPCF henceforth, would add constraints (5.1.1b) and (5.1.5), respectively, whenever a feasible solution to the master problem (6.1.1) that is not a cross-graph k-club is encountered anywhere in the BC tree.

6.1.1 Preprocessing and Extended Formulations

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 (Bourjolly et al., 2000; Salemi and Buchanan, 2020) on the *intersection graph* J with node set $V(\mathcal{G})$ and edge set $\bigcap_{G \in \mathcal{G}} E(G)$. Once a feasible solution is obtained in this manner, we implement *core peeling* (Abello et al., 1999) followed by *community peeling* (Verma et al., 2015) procedures on the power-intersection graph of $\mathcal{G}$, denoted by J^k; the actions are mirrored on $\mathcal{G}$. If node u has fewer than $|S|$ neighbors in J^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^k). Core peeling recursively deletes nodes with degree less than $|S|$ in J^k, and also from every graph in $\mathcal{G}$.

After core-peeling, as long as $V(J^k)$ is nonempty, 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, during community peeling step, the edge uv can be deleted from J^k and from every graph in the collection where u and v are adjacent.

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 of all graphs in $\mathcal{G}$

2 compute a k-club S of J using the 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** $\mathcal{G}$

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).

To avoid unnecessary constraints (6.1.1b) added for pairs of nodes which reside in different components of J^k, we extend formulation (6.1.1) to (6.1.2) by introducing a binary variable y for each of the connected components of J^k and enforce that nodes selected must belong

to the same component. We use $\mathcal{C}(J^k)$ to denote the set of components of J^k, and short form $\forall(u, v, C)$ to denote "for each component C in J^k and each nonadjacent node pair $u, v \in C$". 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.

$$\max\ x(V(\mathcal{G})) \tag{6.1.2a}$$

$$\text{s.t. } x_u + x_v \leq 1 \qquad \forall(u, v, C) \tag{6.1.2b}$$

$$y\left(\mathcal{C}(J^k)\right) \leq 1 \tag{6.1.2c}$$

$$x_u \leq y_C \qquad \forall u \in C \text{ and } C \in \mathcal{C}(J^k) \tag{6.1.2d}$$

$$x_u \in \{0, 1\} \qquad \forall u \in V(\mathcal{G}) \tag{6.1.2e}$$

$$y_C \in \{0, 1\} \qquad \forall C \in \mathcal{C}(J^k). \tag{6.1.2f}$$

6.1.2 Experimental Settings

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 (Gurobi Optimization, LLC, 2020) and the algorithms are implemented in C++.

We consider the following parameter values in our experiments: $k \in \{2, 3, 4\}$ and $p \in \{2, 3, 4, 5\}$. For each problem, we allow a time limit of 7200 seconds. Our test bed is generated from three sets of graphs, the Tenth DIMACS Implementation Challenge (DIMACS-10) benchmark graphs (Bader et al., 2013), uniform random graphs from Veremyev and Boginski (2012) (VB graphs), and graphs generated by the algorithm described in (Bourjolly et al., 2002) (BG graphs), which are known to be challenging for the (single-graph) maximum k-club problem. Generation of graph collections based on these three sets of graphs are

different, and details are given below.

DIMACS-10 graphs. We take 12 graphs from DIMACS-10. Each of these graphs serves as a seed graph to generate an instance for the maximum p-graph k-club problem. We create a collection of 10 graphs generated by the same procedure used to generate graph sequences in (Balasundaram et al., 2022). We summarize these 12 graphs in Table 1. We use ρ to denote edge density of a graph. In Section 6.1.3, each graph collection is simply refered to by its corresponding graph name.

Table 1: DIMACS-10 graphs

G	$\|V(G)\|$	$\|E(G)\|$	$\rho(\%)$
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	2025	1.98
email	1133	5451	0.85
polblogs	1490	16715	1.51
netscience	1589	2742	0.22
power	4941	6594	0.05
hep-th	8361	15751	0.05
PGPgiantcompo	10680	24316	0.04

VB graphs. For each VB instance, we directly take 10 VB graphs with same setting of edge density parameters a and b from Veremyev and Boginski (2012) as a graph sequence. We consider three different settings of a and b summarized in Table 2. We refer to the three graph collections of different edge density parameters in Table 2 as VB_1, VB_2, and VB_3 respectively.

BG graphs. BG instances are generated same way as VB instances. And we consider four different setting of parameters summarized in Table 3. We refer to the four graph collections of different edge density parameter in Table 3 as BG_1, BG_2, BG_3, and BG_4 respectively.

Table 2: The setting of $|V(G)|$, and edge density parameters a and b.

$\|V(G)\|$	a	b
300	0.3%	0.7%
300	0.5%	1.5%
300	1.0%	2.0%

Table 3: The setting of $|V(G)|$, and edge density parameter ρ.

$\|V(G)\|$	ρ
200	15%
200	5%
200	2.5%
200	10%

6.1.3 Computational Results

From our preliminary results, BG_1 and BG_4 are challenging instances when $k = 2$. Collection BG_2 and BG_3 are challenging instances when $k = 3$ and 4 respectively. For each p, k pair, we report computational results for the corresponding challenging instances in Table 4.

Each row of Table 4 reports results averaged over $11 - p$ runs, i.e., on graph collections $\{G_1, \ldots, G_p\}, \{G_2, \ldots, G_{p+1}\}, \ldots, \{G_{11-p}, \ldots, G_{10}\}$. Except for BG_1 instance when $p = 2$ and $k = 2$, where both approaches did not reach optimality, PPCF based branch-and-cut takes significantly shorter running times than CCF for the rest 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.

The columns labeled #NCT in Table 4 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 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

Table 4: 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.

five instances under PPCF. The lazy constraints of this type are actually conflict constraints with no negative terms on the left hand side.

For PPCF, column #SLC reports average number of strengthened lazy constraints added. These are constraints which belong to (5.1.5) but not (5.1.1b). For PPCF, over 52% of lazy constraints added are of this type. The foregoing observations strongly suggest that PPCF approach based on pairwise peeling constraints significantly improves our ability to find maximum p-graph k-clubs.

Tables 10 to 21 in Appendix A contain results on DIMACS-10 and VB instances. Each row of these tables reports results on collection $\{G_1, \ldots, G_p\}$ only instead of an average like in Table 4. Observe that there are few or even zero lazy constraints added by both CCF and PPCF when solving most of these instances. This implies that the master problem is "nearly sufficient" to describe the convex hull, therefore no significant improvements by using PPCF over CCF are observed on these instances. In these 12 tables, we take those instances where over 100 lazy constraints are reported using either CCF or PPCF, and rerun

Table 5: Comparison of CCF and PPCF on DIMACS and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
3	2	email	127.22	1246.08	2403.78	3.24	127.22	1101.27	2125.00	136.78	3.06
4	2	email	462.89	11.26	237.33	1.62	462.89	11.63	234.44	10.56	1.52
3	3	email	113.00	1629.76	5375.38	3.60	113.00	1120.17	4086.63	468.88	3.21
4	3	email	432.50	17.08	353.63	1.79	432.50	17.88	358.25	31.00	1.68
3	4	email	104.14	1540.83	7813.00	3.86	104.14	1188.20	4217.57	681.86	3.14
4	4	email	411.14	24.40	505.71	1.96	411.14	25.93	467.43	54.57	1.75
3	5	email	97.33	1824.62	10716.67	4.13	97.33	1076.25	6878.00	1340.00	3.33
4	5	email	396.17	32.08	753.00	1.97	396.17	34.46	718.67	97.50	1.72
3	2	football	30.67	0.46	448.11	2.18	30.67	0.44	400.67	32.11	1.89
3	3	football	26.13	0.47	485.75	2.35	26.13	0.52	391.88	62.88	1.80
3	4	football	25.14	0.48	404.43	2.47	25.14	0.47	307.43	67.57	1.79
3	5	football	24.67	0.41	334.50	2.53	24.67	0.40	217.50	61.00	1.77
4	3	hep-th	175.88	256.12	496.00	2.81	175.88	280.60	457.75	57.13	2.50
4	4	hep-th	154.57	293.25	872.71	2.84	154.57	261.54	871.57	155.71	2.41
4	5	hep-th	139.50	939.83	1240.00	3.04	139.50	811.85	1189.33	253.50	2.49
4	2	VB_2	4.00	5.55	632.44	1.14	4.00	4.63	594.56	525.44	0.00
4	2	VB_3	4.78	195.94	19245.78	2.05	4.78	122.62	14857.78	14672.89	0.00
4	3	VB_3	4.00	99.93	9695.38	1.92	4.00	75.09	8217.63	7987.38	0.00
4	4	VB_3	4.00	49.03	4359.71	1.83	4.00	43.83	3962.71	3754.14	0.00
4	5	VB_3	4.00	15.37	1803.67	1.72	4.00	11.32	1727.83	1592.67	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.

the two algorithms and report results averaged over $11 - p$ runs in Table 5. Because a fair number of lazy constraints are used in solving these instances, the advantage of PPCF over CCF is revealed again.

Figure 18 shows the performance profiles (Dolan and Moré, 2002) based on running times of CCF and PPCF algorithms for solving the maximum p-graph k-club problem on all BG instances and those DIMACS-10 and VB instances listed in Table 5. 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 indicate that the computational benefits of using lazy constraints strengthened by pairwise peeling is quite significant. Same conclusions can be drawn if we group running times by p or k; see Figures 19 and 20.

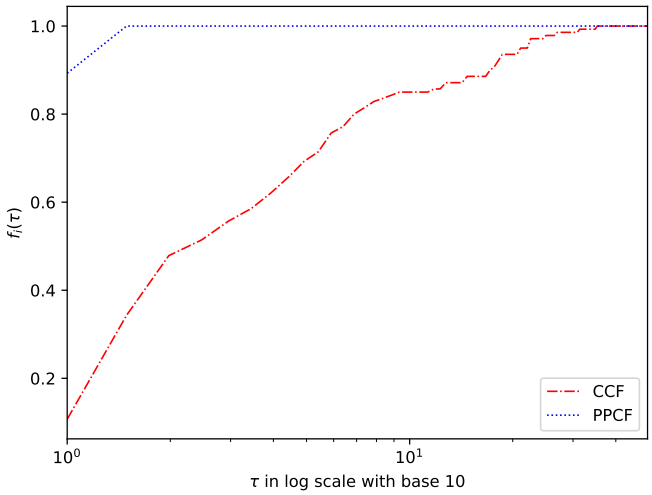


Figure 18: Performance profiles of CCF and PPCF algorithms.

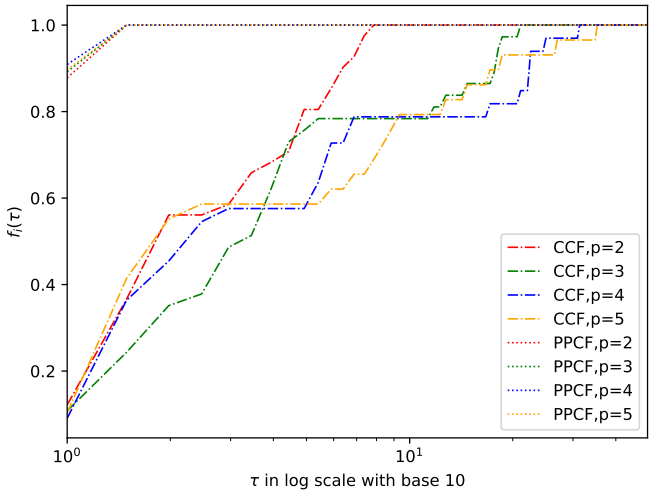


Figure 19: Performance profiles of CCF and PPCF algorithms differentiated by p.

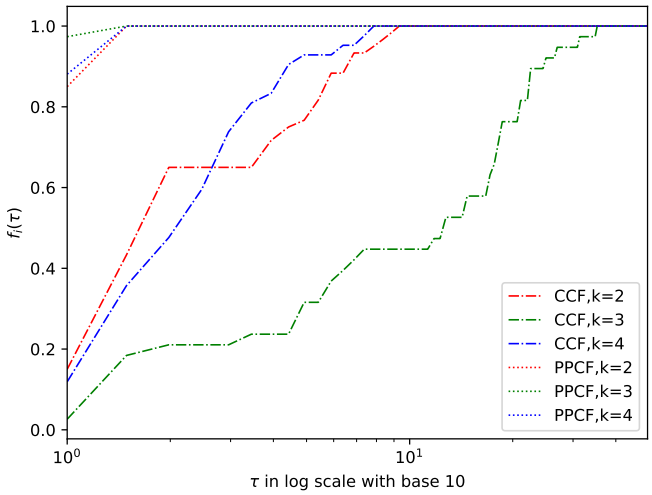


Figure 20: Performance profiles of CCF and PPCF algorithms differentiated by k.

6.2 The Maximum k-Club Signature Problem

This section serves as a case study to assess the computational effectiveness of the approaches developed for solving the maximum p-graph k-club problem in solving the maximum k-club signature problem. Recall from Chapter I that the k-club signature problem and the cross-graph k-club problem are closely related because a maximum k-club signature problem can be decomposed into a sequence of maximum cross-graph k-club problems. Given a graph sequence $\mathcal{G} = (G_t, t \in [T])$ and positive integers k and τ, the maximum k-club signature problem seeks to find a maximum cardinality τ-persistent k-club signature of $\mathcal{G}$. We use $[T]$ to denote the index set $\{1, 2, \ldots, T\}$. By definition, each τ-persistent k-club signature on a subsequence $\mathcal{H}$ of τ graphs is also a τ-graph k-club on graph collection $\mathcal{H}$. We regard $\mathcal{H}$ as a graph collection in the cross-graph setting because a τ-graph k-club of $\mathcal{H}$ must be a k-club in each graph in $\mathcal{H}$ and thus order of graphs in $\mathcal{H}$ is of no significance.

In Balasundaram et al. (2022), a monolithic IP formulation (6.2.1) and a dynamic programming based moving window (MW) method are introduced to solve the maximum 2-club signature problem. Given a graph sequence $\mathcal{G}$, the MW method involves solving $T - \tau + 1$

maximum τ-graph k-club problems on $T - \tau + 1$ graph collections of length τ. Each consecutive subsequence $\mathcal{H}$ of τ graphs is referred to as a window, and solving for a maximum τ-graph k-club of a window is referred to as a window problem.

$$\max\ x(V(\mathcal{G})) \tag{6.2.1a}$$

$$\text{s.t.} \quad x_u + x_v - x(N_G(u,v)) \leq 2 - z_t \quad \forall uv \in E(\overline{G}), G \in \mathcal{G}, t \in [T] \tag{6.2.1b}$$

$$y_t \leq z_j \quad \forall j = t, t+1, \ldots, t+\tau-1, \quad t = 1, 2, \ldots, T-\tau+1 \tag{6.2.1c}$$

$$\sum_{t=1}^{T-\tau+1} y_t \geq 1 \tag{6.2.1d}$$

$$x_u \in \{0,1\} \quad \forall u \in V(\mathcal{G}) \tag{6.2.1e}$$

$$y_t \in \{0,1\} \quad \forall t = 1, 2, \ldots, T-\tau+1 \tag{6.2.1f}$$

$$z_t \in \{0,1\} \quad \forall t \in [T]. \tag{6.2.1g}$$

In Balasundaram et al. (2022), two versions of MW methods are compared for solving the maximum 2-club signature problem, MW-2CLB and MW-F2. 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 scale reduction techniques based on specific properties of 2-clubs (see Algorithm 3). By the reported computational results, MW-2CLB is preferable over MW-F2 and the monolithic IP formulation (6.2.1) for the problem. Due to this reason, we do not extend MW-F2 and the monolithic formulation to the p-graph setting.

Proposition 10 that follows forms the basis for Algorithm 3 for finding a maximum τ-persistent 2-club signature in $\mathcal{G}^{t,\tau}$, which denotes the subsequence $\{G_t, G_{t+1}, \ldots, G_{t+\tau-1}\}$. First we define a τ-persistent k-clique signature, which is necessary to formulate the master relaxation problem.

Definition 8. A subset of nodes $S \subseteq V(\mathcal{G})$ is called a *τ-persistent k-clique signature* if S is a k-clique in at least τ consecutive graphs in $\mathcal{G}$.

Proposition 10. Consider a sequence of graphs $\mathcal{G} = (G_r, r \in [T])$, and a pair of integers $t, \tau \in [T]$. Define the *square intersection graph* $J_2^{t,\tau}$ as,

$$J_2^{t,\tau} := \left(V(\mathcal{G}), \bigcap_{r=t-\tau+1}^{t} E^{r,2} \right), \tag{6.2.2}$$

where $E^{r,2} := \big\{\{u, v\} : 1 \leq \text{dist}_{G_r}(u, v) \leq 2\big\}$. A subset of nodes S is a τ-persistent 2-clique signature in $\mathcal{G}$ if and only if there exists a $t = \tau, \ldots, T$ such that S is a clique in the square-intersection graph $J_2^{t,\tau}$.

Algorithm 3: (MW-2CLB) Finding a maximum cardinality 2-club signature using the moving window method.

Input: $\mathcal{G} = (G^t, t \in [T])$, $\tau \geq 1$, $\ell = 0$.

Output: A maximum cardinality 2-club signature S.

1 $v \leftarrow \arg\max\{|N_{H^{\tau,\tau}}(u)| : u \in V(\mathcal{G})\}$

2 $S \leftarrow \{v\} \cup N_{H^{\tau,\tau}}(v)$

3 **for** $t = \tau, \ldots, T$ **do**

4 $\quad \ell \leftarrow |S|$

5 $\quad J_2^{t,\tau} \leftarrow \textsc{CorePeel}(J_2^{t,\tau}, \ell)$

6 $\quad J_2^{t,\tau} \leftarrow \textsc{CommunityPeel}(J_2^{t,\tau}, \ell)$

7 $\quad S' \leftarrow$ solve for a maximum τ-persistent 2-club in $\mathcal{G}^{t,\tau}$ using delayed generation of common neighborhood constraints applied to the maximum clique master relaxation formulation (6.1.2) on $J_2^{t,\tau}$

8 $\quad S \leftarrow \arg\max\{|S|, |S'|\}$

9 **return** S

Step 7 of Algorithm 3 starts by solving the master relaxation, the maximum clique problem on $J_2^{t,\tau}$, using a general purpose branch-and-cut solver. Observe that by Proposition 10,

such a maximum clique is a τ-persistent 2-clique on $\mathcal{G}^{t,\tau}$. Whenever an integral solution is encountered, it must be verified if this τ-persistent 2-clique signature in $\mathcal{G}^{t,\tau}$ is indeed a τ-persistent 2-club signature in $\mathcal{G}^{t,\tau}$. If it is, then the corresponding node of the search tree can be pruned by feasibility; if not, lazy constraints will be added to cut off this infeasible integral solution. The infeasible solution can be removed by noting that it must violate some common neighbor constraint for some graph G_r in $\mathcal{G}^{t,\tau}$, and we add the first violated constraint detected into the lazy constraint pool, and the node subproblem is re-solved.

The effectiveness of the core and community peeling steps relies on having a good initial heuristic solution. Because a 2-club that exists in the intersection graph $H^{\tau,\tau}$ of $\mathcal{G}^{\tau,\tau}$ exists in every graph in $\mathcal{G}^{\tau,\tau}$, a node of maximum degree and its neighbors in $H^{\tau,\tau}$ is used as an initial heuristic solution in step 2 of Algorithm 3.

Algorithm 3 can be extended and improved for solving the maximum k-club signature problem by borrowing a few preprocessing ideas from Algorithm 2. We use $J_k^{t,\tau}$ to denote the power-intersection graph of $\mathcal{G}^{t,\tau}$, that is,

$$J_k^{t,\tau} \coloneqq \left(V(\mathcal{G}), \bigcap_{r=t-\tau+1}^{t} E^{r,k} \right), \tag{6.2.3}$$

where $E^{r,k} \coloneqq \big\{\{u,v\} : 1 \leq \mathrm{dist}_{G_r}(u,v) \leq k\big\}$. In step 1 of Algorithm 4, we compute a heuristic k-club instead of 2-club. Each iteration inside the for-loop (step 2 of Algorithm 4) involves solving a maximum τ-graph k-club problem. In step 6, if a node pair u, v are in disinct connected components of some graph G in the extended collection $\mathcal{G}^{t,\tau} \cup \{J_k^{t,\tau}\}$, we recursively remove edge uv from every graph $G \in \mathcal{G}^{t,\tau} \cup \{J_k^{t,\tau}\}$ in which it is present. Eventually, all graphs in $G \in \mathcal{G}^{t,\tau} \cup \{J_k^{t,\tau}\}$ share a same set of connected components. In step 7, different inequalities can be chosen as lazy constraints. If inequalities (5.1.1b) are added as lazy constraints, we refer to the algorithm MW-CCF , and call it MW-PPCF if inequalities (5.1.5) are added. 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 more effective to not treat them

independent of each other. This is because the peeling procedures between two consecutive window problems may effectively reduce the size of the subsequent window problems, and hence, reduce the scale of the problem as the window slides further along the sequence.

Algorithm 4: (MW-kCLB) Finding a maximum cardinality k-club signature using the moving window method.

Input: $\mathcal{G} = (G^t, t \in [T])$, $\tau \geq 1$, $\ell = 0$.

Output: A maximum cardinality k-club signature S.

compute a k-club S of $H^{\tau,\tau}$ using DROP heuristic

for $t = \tau, \ldots, T$ **do**

  $\ell \leftarrow |S|$

  $J_k^{t,\tau} \leftarrow \textsc{CorePeel}(J_k^{t,\tau}, \ell)$

  $J_k^{t,\tau} \leftarrow \textsc{CommunityPeel}(J_k^{t,\tau}, \ell)$

  $(\mathcal{G}^{t,\tau}, J_k^{t,\tau}) \leftarrow \textsc{EdgePeel}(\mathcal{G}^{t,\tau}, J_k^{t,\tau})$

  $S' \leftarrow$ solve for a maximum τ-persistent k-club in $\mathcal{G}^{t,\tau}$ using delayed generation of CCF constraints (5.1.1b) or PPCF constraints (5.1.5) applied to the maximum clique master relaxation (6.1.2) on $J_k^{t,\tau}$

  $S \leftarrow \arg\max\{|S|, |S'|\}$

return S

In the rest of this section, we compare the performance of MW-CCF and MW-PPCF in solving the maximum k-club signature problem. We consider $k = 2, 3, 4$ and $\tau = 2, 3, 4, 5$ in our experiments. For each window problem, we allow a time limit of 3600 seconds. We terminate the algorithm if two consecutive window problems are not solved to optimality. We first test the two algorithms on the challenging instances identified for each k in Section 6.1.3. Tables 6 and 7 summarize the results. MW-CCF failed to solve three instances to optimality, while MW-PPCF did not solve one to optimality; a better lower bound of 17 was reported by MW-PPCF than that of 15 reported by MW-CCF. Overall, MW-PPCF is over 6.4 times faster than MW-CCF on BG instances, and over 1.09 times faster 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 than on BG instances.

Table 6: Comparison of MW-CCF and MW-PPCF on BG instances.

			MW-CCF		MW-PPCF	
k	τ	**Instance**	obj	time(s)	obj	time(s)
2	2	BG_4	10	791.4	10	487.22
2	3	BG_4	5	437.18	5	304.52
2	4	BG_4	3	341.48	3	308.55
2	5	BG_4	2	238.28	2	215.66
3	2	BG_2	≥ 13[a]	26475.81	14	5194.38
3	3	BG_2	4	14265.58	4	844.06
3	4	BG_2	3	7744.9	3	320.33
3	5	BG_2	3	5121.59	3	236.92
4	2	BG_3	11	1738.8	11	357.91
4	3	BG_3	4	748.36	4	238.95
4	4	BG_3	4	379.32	4	188.99
4	5	BG_3	4	242.02	4	164.16
2	2	BG_1	≥ 15	7203.65	≥ 17	7204.42
2	3	BG_1	≥ 10	24144.99	10	6074.36
2	4	BG_1	5	13602.42	5	2343.89
2	5	BG_1	3	9023.71	3	1183.58

[a] Instance not solved to optimality.

The DIMACS-10 instances in Section 6.1.3 are generated in a different way from VB and BG instances. Because BG instances are consistently challenging to the problem based on results reported in Tables 4 and 6. Due to these two observations, we are interested in running MW-CCF and MW-PPCF on more BG instances. We use the instance generator from Balasundaram et al. (2022) to generate sequences BG_5, BG_6, BG_7, BG_8, each based on a BG graph with edge densities of 15%, 5%, 2.5%, and 10%, respectively. Each graph in these sequences contains 200 nodes. Results are reported in Table 8. Overall, MW-PPCF is 2.03 times faster than MW-CCF on average. It is over 3 times faster on BG_6 with $k = 3$ and $\tau = 5$ and BG_7 with $k = 4$ and $\tau = 5$, which stand out in particular.

We also consider instances with long graph sequences. We use the instance generator from Balasundaram et al. (2022) to generate 12 sequences based on the 12 DIMACS-10 graphs. Each instance is a sequence containing 100 graphs. For these instances, we consider

Table 7: Comparison of MW-CCF and MW-PPCF on DIMACS-10 and VB instances.

			MW-CCF		MW-PPCF	
k	τ	**Instance**	obj	time(s)	obj	time(s)
3	2	email	135	6563.29	135	5557.29
4	2	email	473	76.13	473	76.46
3	3	email	123	7245.76	123	6145.28
4	3	email	440	92.22	440	92.6
3	4	email	114	5769.38	114	4405.68
4	4	email	419	109.82	419	111.78
3	5	email	107	4848.66	107	3943.77
4	5	email	403	120.98	403	140.93
3	2	football	35	3.55	35	3.76
3	3	football	28	4.06	28	4.13
3	4	football	26	3.64	26	3.57
3	5	football	25	2.16	25	2.14
4	3	hep-th	198	1343.74	198	1469.24
4	4	hep-th	171	676.92	171	706.27
4	5	hep-th	143	952.09	143	926.79
4	2	VB_2	4	28.71	4	26.44
4	2	VB_3	7	1700.82	7	1098.69
4	3	VB_3	4	734.61	4	572.61
4	4	VB_3	4	313.66	4	297.6
4	5	VB_3	4	59.34	4	46.21

Table 8: Comparison on additional BG instances.

			MW-CCF		MW-PPCF	
k	τ	**Instance**	obj	time(s)	obj	time(s)
2	2	BG_8	24	92.64	24	113.41
2	2	BG_5	35	15981.49	35	7612.34
2	3	BG_8	22	66.35	22	64.07
2	3	BG_5	30	7058.1	30	3749.54
2	4	BG_8	19	58.25	19	31.73
2	4	BG_5	25	5516.52	25	2351.31
2	5	BG_8	18	58.77	18	49.62
2	5	BG_5	22	2683.21	22	1274.49
3	2	BG_6	45	2970.63	45	1549.62
3	3	BG_6	39	2046.51	39	984.33
3	4	BG_6	35	1659.85	35	692.59
3	5	BG_6	29	1633.83	29	435.69
4	2	BG_7	56	116.45	56	79.19
4	3	BG_7	45	218.69	45	116.18
4	4	BG_7	36	229.67	36	115.88
4	5	BG_7	27	239.93	27	64.39

$\tau = 10$ and $k = 2, 3, 4$. Results are reported in Table 9. To differentiate these instances from the DIMACS-10 instances generated in Section 6.1.2, we use 100 as a suffix in instance names, e.g., adjnoun_100. For most instances in this group, we do not observe an obvious advantage

Table 9: Comparison of MW-CCF and MW-PPCF on long sequences.

			CCF		PPCF	
k	τ	**Instance**	obj	time(s)	obj	time(s)
2	10	adjnoun_100	12	2.35	12	2.27
3	10	adjnoun_100	53	3.71	53	4.47
4	10	adjnoun_100	89	1.8	89	1.91
2	10	celegans_metabolic_100	40	10.19	40	10.39
3	10	celegans_metabolic_100	185	52.58	185	61.58
4	10	celegans_metabolic_100	352	76.18	352	81.76
2	10	email_100	21	27.08	21	27.54
3	10	email_100	92	56867.72	92	26963.66
4	10	email_100	371	3379.66	371	3682.71
2	10	football_100	12	0.75	12	0.71
3	10	football_100	23	42.89	23	45.56
4	10	football_100	89	1028.32	89	666.18
2	10	hep-th_100	24	84.22	24	85.14
3	10	hep-th_100	40	525.12	40	519.3
4	10	hep-th_100	≥ 124[a]	23577.59	≥ 124	19765.49
2	10	karate_100	8	0.45	8	0.42
3	10	karate_100	16	0.33	16	0.31
4	10	karate_100	26	0.46	26	0.44
2	10	lesmis_100	14	0.69	14	0.69
3	10	lesmis_100	33	0.9	33	0.97
4	10	lesmis_100	50	0.78	50	0.76
2	10	netscience_100	20	7.49	20	6.63
3	10	netscience_100	26	7.71	26	7.57
4	10	netscience_100	41	9.04	41	8.6
2	10	PGPgiantcompo_100	64	414.86	64	418.27
3	10	PGPgiantcompo_100	191	491.98	191	500.75
4	10	PGPgiantcompo_100	392	1163.71	392	1134.68
2	10	polblogs_100	164	987.33	164	1002.6
3	10	polblogs_100	579	597.37	579	571.13
4	10	polblogs_100	947	808.9	947	720.98
2	10	polbooks_100	16	1.16	16	1.09
3	10	polbooks_100	35	1.19	35	0.77
4	10	polbooks_100	53	1.92	53	1.66
2	10	power_100	9	35.57	9	35.06
3	10	power_100	17	64.36	17	61.02
4	10	power_100	27	78.31	27	66.42

[a] Instance not solved to optimality.

of MW-PPCF over MW-CCF, except on email_100 with $k = 3$ and $\tau = 10$ and football_100 with $k = 4$ and $\tau = 10$. MW-PPCF is 1.5 and 2.1 times faster on these two instances, respectively. Notice that τ is set to 10 in these instances, and this could significantly reduce the size of the master relaxation problems on the power-intersection graphs. Recall from Section 6.1.3, the master problem constraints appear to provide very tight relaxations on DIMACS-10 instances, which could explain the limited impact of lazy constraints.

From these observations we can conclude that computational benefits of using lazy constraints strengthened by pairwise peeling is quite significant when solving the maximum k-club signature problem. Same conclusions can be drawn by looking at the performance profiles in Figures 21, 22, and 23.

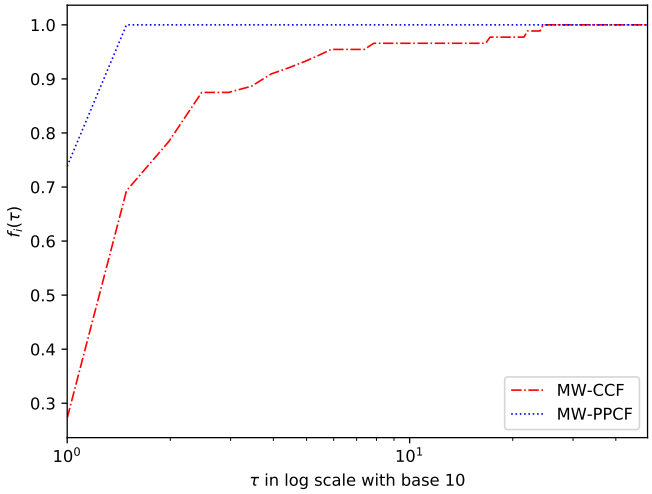


Figure 21: Performance profiles of MW-CCF and MW-PPCF algorithms.

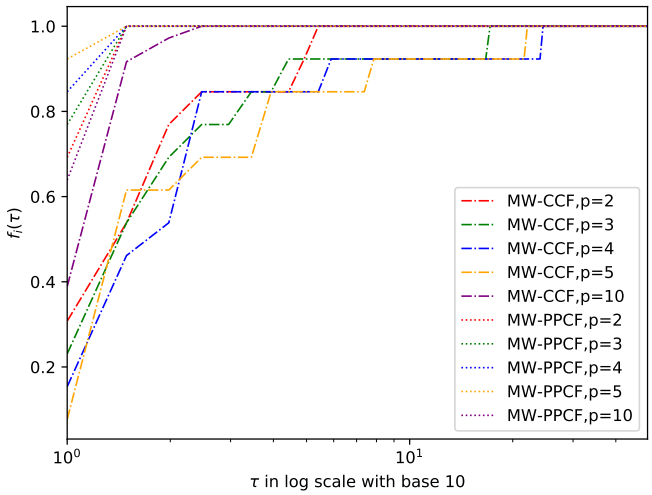


Figure 22: Performance profiles of MW-CCF and MW-PPCF algorithms differentiated by p.

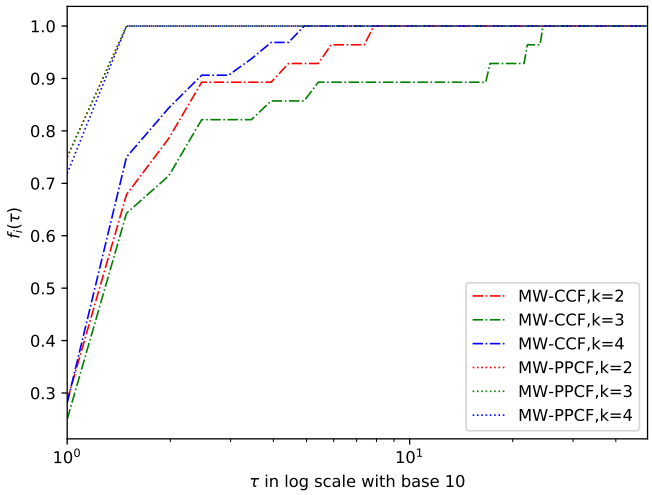


Figure 23: Performance profiles of MW-CCF and MW-PPCF algorithms differentiated by k.

CHAPTER VII

CONCLUSION AND FUTURE WORK

The p-graph k-club model is designed to mine low-diameter clusters conserved in graph collections. A graph collection may represent a time-varying graph or a graph where node relationships change under different conditions. This dissertation develops integer programming approaches to solve for a maximum p-graph k-club given a graph collection. In this chapter, we summarize our theoretical and algorithmic contributions and identify possible future research directions.

7.1 Contributions

We started by establishing NP-completeness of several bounded-enlargement problems in Chapter IV. The BEkC problem, when $\ell = 1$, always assumes an opposite answer to the MTkC problem which is proved to be coNP-complete for every $k \geq 2$ by Pajouh and Balasundaram (2012). Therefore, BEkC is NP-complete for every fixed $k \geq 2$. We offered an alternate proof of this result by reduction from the Clique problem, and borrowed the instance construction procedure from Balasundaram et al. (2005). We then showed that BEkC is NP-complete when $k = 1$. Consequently, BEkC is NP-complete for every fixed $k \geq 1$. We also extended this result to the p-graph case, i.e., showing that the BECkC problem is NP-complete for every fixed integer $k \geq 1$. As a corollary, we obtained the same result for the CkC problem, the decision counterpart of the maximum cross-graph k-club problem.

In Chapter V, we explored IP approaches to solving the maximum p-graph k-club prob-

lem. We presented a naive conjunctive formulation (CCF) based on cut-like formulation for the maximum k-club problem proposed by Salemi and Buchanan (2020), and offered ideas to strengthen this formulation. Eventually, we reached a new formulation (PPCF) based on a "pairwise peeling" algorithm. In addition, we identified a special class of inequalities fo the maximum p-graph 2-club problem, and provided a necessary and sufficient condition for its validity. For the independent set inequality introduced by Pajouh et al. (2016) for the maximum 2-club problem, we gave an alternative proof of its validity, which enabled us to extend the result to the p-graph setting. And lastly, we extended the distance-k independent set inequality introduced by Balasundaram (2007) to the p-graph setting.

In Chapter VI, we conducted a computational study to compare the performance of a general purpose IP solver when using CCF and PPCF formulations to solve the maximum p-graph k-club problem. For scale reduction, we introduced preprocessing procedures and an extended formulation. For preprocessing, we implemented core peeling (Abello et al., 1999) followed by community peeling (Verma et al., 2015). Then, we recursively deleted edges until all graphs of interest share a common set of connected components, enabling a component-wise extended formulation. We conducted a computational study and concluded that PPCF based on pairwise peeling significantly improves our ability to solve for a maximum p-graph k-club. Our computational study on the maximum k-club signature problem also led us to the same conclusions.

7.2 Future Work

In the p-graph setting, identifying facet-inducing conditions of valid inequalities is more challenging due to the difficulty of finding enough affinely independent incidence vectors of p-graph k-clubs. For a nonadjacent node pair u, v, there might not exist a cross-graph k-club containing them even if $\text{dist}_G(u, v) \leq k$ for all $G \in \mathcal{G}$. Identifying facet-inducing conditions would offer insights into the strength of inequalities. One possible direction of future work is to find sufficient conditions under which our proposed valid inequalities to the maximum

p-graph k-club problem induce facets of the associated convex hull.

In Chapter V, we introduced a special class of inequalities (5.2.1) and offered a necessary and sufficient condition for its validity. However, complexity of testing this condition remains open. Similarity of this testing problem with MTkC can be observed, and we conjecture it is NP-hard to check if this condition holds. Establishing the complexity of testing this condition is of interest.

Although our focus in this dissertation is on clusters that induce low-diameter subgraphs, one may investigate any clique relaxation or another graph property in the p-graph setting. A third direction of future work is to investigate the p-graph extension of other cluster models, including those whose single-graph counterparts are polynomially solvable. This entails establishing computational complexity of the new problems, finding IP formulations, identifying valid inequalities, developing algorithmic approaches to solve the new problems. Once computational tool-kits are available for a variety of cross-graph cluster models, we may explore applications of these models and algorithms in the analysis of social and biological network collections, including temporal networks.

APPENDICES

Appendix A

Table 10: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
2	2	adjnoun 1-2	34	0	-	-	34	0	-	-	-
2	2	celegans_metabolic 1-2	142	0.04	-	-	142	0.04	-	-	-
2	2	email 1-2	50	0.71	1	1	50	0.73	1	0	1
2	2	football 1-2	12	0.33	9	1.11	12	0.32	9	2	0.89
2	2	hep-th 1-2	34	0.57	-	-	34	0.61	-	-	-
2	2	karate 1-2	13	0	-	-	13	0	-	-	-
2	2	lesmis 1-2	26	0	-	-	26	0	-	-	-
2	2	netscience 1-2	27	0.03	-	-	27	0.05	-	-	-
2	2	PGP 1-2	133	0.64	-	-	133	0.91	-	-	-
2	2	polblogs 1-2	220	14.61	5	1.6	220	14.85	5	0	1.2
2	2	polbooks 1-2	20	0.02	3	0.33	20	0.02	3	0	0.33
2	2	power 1-2	13	0.26	-	-	13	0.26	-	-	-
2	2	VB_1 1-2	3	0.01	0	0	3	0.01	0	0	0
2	2	VB_2 1-2	2	0.32	0	0	2	0.32	0	0	0
2	2	VB_3 1-2	3	1.37	12	1	3	1.56	15	13	0

[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.

Table 11: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
3	2	adjnoun 1-2	66	0.06	4	1	66	0.06	4	0	1
3	2	celegans_metabolic 1-2	283	0.74	4	0.5	283	0.74	4	0	0.5
3	2	email 1-2	124	1148.39	4797	3.22	124	727.52	1212	75	2.87
3	2	football 1-2	31	0.44	454	2.13	31	0.38	270	14	1.84
3	2	hep-th 1-2	72	16.42	8	3.25	72	16.69	8	0	2.88
3	2	karate 1-2	21	0	-	-	21	0	-	-	-
3	2	lesmis 1-2	44	0.01	0	0	44	0.01	0	0	0
3	2	netscience 1-2	39	0.03	-	-	39	0.04	-	-	-
3	2	PGP 1-2	257	6.07	7	1.14	257	5.88	7	0	1.14
3	2	polblogs 1-2	660	7.7	9	1.22	660	7.72	9	0	1.22
3	2	polbooks 1-2	40	0.05	1	1	40	0.06	1	1	0
3	2	power 1-2	20	0.6	0	0	20	0.87	0	0	0
3	2	VB_1 1-2	3	0.01	0	0	3	0.01	0	0	0
3	2	VB_2 1-2	3	2.31	15	1.2	3	1.97	15	13	0
3	2	VB_3 1-2	4	5.98	715	1.32	4	5.77	794	789	0

[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.

Table 12: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
4	2	adjnoun 1-2	94	0.05	0	0	94	0.06	0	0	0
4	2	celegans_metabolic 1-2	402	0.92	2	1	402	0.96	2	0	1
4	2	email 1-2	457	11.12	204	1.69	457	12	211	9	1.56
4	2	football 1-2	111	0.04	1	1	111	0.04	1	0	1
4	2	hep-th 1-2	214	37.6	10	3.6	214	37.23	10	0	3.5
4	2	karate 1-2	28	0	-	-	28	0	-	-	-
4	2	lesmis 1-2	59	0.01	0	0	59	0.01	0	0	0
4	2	netscience 1-2	59	0.13	0	0	59	0.11	0	0	0
4	2	PGP 1-2	663	56.93	7	1.57	663	56.59	7	0	1.57
4	2	polblogs 1-2	1032	8.88	0	0	1032	8.9	0	0	0
4	2	polbooks 1-2	56	0.07	3	1.33	56	0.07	3	2	0.67
4	2	power 1-2	31	1.02	0	0	31	1.01	0	0	0
4	2	VB_1 1-2	3	0.02	0	0	3	0.02	0	0	0
4	2	VB_2 1-2	4	9.04	719	1.23	4	6.47	687	637	0
4	2	VB_3 1-2	6	218.02	21273	2.22	6	119.75	16075	15768	0

[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.

Table 13: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
2	3	adjnoun 1-3	27	0.02	0	0	27	0.02	0	0	0
2	3	celegans_metabolic 1-3	115	0.06	-	-	115	0.06	-	-	-
2	3	email 1-3	41	0.2	0	0	41	0.21	0	0	0
2	3	football 1-3	12	0.31	1	1	12	0.32	1	0	1
2	3	hep-th 1-3	29	0.72	-	-	29	0.81	-	-	-
2	3	karate 1-3	10	0	-	-	10	0	-	-	-
2	3	lesmis 1-3	19	0	-	-	19	0	-	-	-
2	3	netscience 1-3	26	0.05	-	-	26	0.07	-	-	-
2	3	PGP 1-3	104	0.81	-	-	104	1.2	-	-	-
2	3	polblogs 1-3	182	75	32	2.16	182	66.47	26	3	2.31
2	3	polbooks 1-3	16	0.02	7	1.14	16	0.02	7	2	0.86
2	3	power 1-3	12	0.18	-	-	12	0.18	-	-	-
2	3	VB_1 1-3	1	0	-	-	1	0	-	-	-
2	3	VB_2 1-3	1	0.24	0	0	1	0.22	0	0	0
2	3	VB_3 1-3	1	0.37	0	0	1	0.37	0	0	0

[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.

Table 14: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
3	3	adjnoun 1-3	58	0.1	13	1.23	58	0.1	11	3	0.73
3	3	celegans_metabolic 1-3	239	1.19	7	1.43	239	1.24	7	1	1.29
3	3	email 1-3	109	1246.25	8406	3.54	109	699.4	2401	333	2.92
3	3	football 1-3	26	0.44	342	2.39	26	0.67	550	84	1.77
3	3	hep-th 1-3	57	9.71	34	2.91	57	8.95	35	5	2.37
3	3	karate 1-3	20	0.01	0	0	20	0.01	0	0	0
3	3	lesmis 1-3	39	0.01	0	0	39	0.01	0	0	0
3	3	netscience 1-3	37	0.06	-	-	37	0.03	-	-	-
3	3	PGP 1-3	221	7.83	2	0.5	221	7.23	2	0	0.5
3	3	polblogs 1-3	629	11.93	10	2.2	629	11.13	10	0	2.2
3	3	polbooks 1-3	39	0.02	0	0	39	0.02	0	0	0
3	3	power 1-3	18	0.94	1	1	18	0.84	1	1	0
3	3	VB_1 1-3	1	0	-	-	1	0	-	-	-
3	3	VB_2 1-3	1	0.37	0	0	1	0.42	0	0	0
3	3	VB_3 1-3	3	5.05	14	1.21	3	6.72	14	14	0

[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.

Table 15: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
4	3	adjnoun 1-3	92	0.07	2	1	92	0.06	2	0	1
4	3	celegans_metabolic 1-3	395	1.33	5	1	395	1.32	5	0	1
4	3	email 1-3	423	15.58	191	1.73	423	15.85	230	22	1.7
4	3	football 1-3	107	0.07	11	1.18	107	0.09	9	2	1
4	3	hep-th 1-3	161	1110.12	1399	2.66	161	1184.37	524	62	2.35
4	3	karate 1-3	27	0.01	0	0	27	0.01	0	0	0
4	3	lesmis 1-3	53	0.01	0	0	53	0.01	0	0	0
4	3	netscience 1-3	50	0.19	0	0	50	0.17	0	0	0
4	3	PGP 1-3	561	44.47	3	2	561	46.04	3	0	1.67
4	3	polblogs 1-3	995	10.48	1	0	995	10.44	1	0	0
4	3	polbooks 1-3	53	0.07	2	1.5	53	0.08	2	1	1
4	3	power 1-3	28	1.27	10	1.2	28	1.25	9	3	1
4	3	VB_1 1-3	1	0	-	-	1	0	-	-	-
4	3	VB_2 1-3	1	0.65	0	0	1	0.71	0	0	0
4	3	VB_3 1-3	4	100.44	10039	2.04	4	69.72	8705	8417	0

[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.

Table 16: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
2	4	adjnoun 1-4	20	0.04	5	1.6	20	0.04	5	1	1.4
2	4	celegans_metabolic 1-4	94	0.07	-	-	94	0.07	-	-	-
2	4	email 1-4	32	0.23	0	0	32	0.2	0	0	0
2	4	football 1-4	12	0.22	4	2.25	12	0.22	4	2	1.75
2	4	hep-th 1-4	24	1.01	-	-	24	0.95	-	-	-
2	4	karate 1-4	9	0.01	4	0	9	0.01	4	0	0
2	4	lesmis 1-4	17	0	-	-	17	0	-	-	-
2	4	netscience 1-4	23	0.09	-	-	23	0.09	-	-	-
2	4	PGP 1-4	80	6.01	0	0	80	5.92	0	0	0
2	4	polblogs 1-4	175	42.59	40	3.73	175	48.51	32	3	3.62
2	4	polbooks 1-4	15	0.03	3	1.67	15	0.03	3	1	1.33
2	4	power 1-4	12	0.25	-	-	12	0.27	-	-	-
2	4	VB_1 1-4	1	0	-	-	1	0	-	-	-
2	4	VB_2 1-4	1	0.01	-	-	1	0.01	-	-	-
2	4	VB_3 1-4	1	0.38	0	0	1	0.37	0	0	0

[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.

Table 17: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
3	4	adjnoun 1-4	54	0.09	12	1.42	54	0.1	12	4	1
3	4	celegans_metabolic 1-4	223	1.29	7	1.71	223	1.32	7	1	1.43
3	4	email 1-4	100	1426.61	13617	3.93	100	863.25	4170	723	3.08
3	4	football 1-4	25	0.81	790	2.37	25	0.78	569	112	1.72
3	4	hep-th 1-4	51	211.27	24	3.54	51	169.33	76	11	2.16
3	4	karate 1-4	20	0.01	0	0	20	0.01	0	0	0
3	4	lesmis 1-4	37	0.01	0	0	37	0.01	0	0	0
3	4	netscience 1-4	32	0.06	0	0	32	0.06	0	0	0
3	4	PGP 1-4	213	6.67	0	0	213	6.74	0	0	0
3	4	polblogs 1-4	613	12.85	10	3.2	613	13.74	13	1	2.54
3	4	polbooks 1-4	36	0.05	0	0	36	0.08	0	0	0
3	4	power 1-4	18	0.66	2	0	18	1.04	2	0	0
3	4	VB_1 1-4	1	0.01	-	-	1	0.01	-	-	-
3	4	VB_2 1-4	1	0.66	0	0	1	0.65	0	0	0
3	4	VB_3 1-4	1	1.37	0	0	1	1.41	0	0	0

[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.

Table 18: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
4	4	adjnoun 1-4	88	0.08	2	1	88	0.08	2	0	1
4	4	celegans_metabolic 1-4	384	1.4	6	1.17	384	1.4	6	0	1.17
4	4	email 1-4	403	21.82	344	1.78	403	22.91	391	50	1.68
4	4	football 1-4	104	0.13	21	1.95	104	0.13	18	2	1.67
4	4	hep-th 1-4	147	375.4	864	3.04	147	313.11	937	164	2.5
4	4	karate 1-4	27	0.01	0	0	27	0.01	0	0	0
4	4	lesmis 1-4	50	0.02	0	0	50	0.02	0	0	0
4	4	netscience 1-4	48	0.19	0	0	48	0.19	0	0	0
4	4	PGP 1-4	487	66.8	19	2.26	487	61.93	17	1	2.18
4	4	polblogs 1-4	979	12.56	1	0	979	12.5	1	0	0
4	4	polbooks 1-4	49	0.09	7	2.29	49	0.1	7	3	1.57
4	4	power 1-4	25	2.55	4	0.5	25	2.19	6	2	0.33
4	4	VB_1 1-4	1	0.01	-	-	1	0.01	-	-	-
4	4	VB_2 1-4	1	0.45	0	0	1	0.76	0	0	0
4	4	VB_3 1-4	4	64.61	4307	1.95	4	49.06	3913	3751	0

[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.

Table 19: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
2	5	adjnoun 1-5	18	0.04	7	2.14	18	0.05	7	1	1.86
2	5	celegans_metabolic 1-5	76	0.06	-	-	76	0.09	-	-	-
2	5	email 1-5	27	4.21	15	2.67	27	4.22	13	7	1.92
2	5	football 1-5	12	0.08	0	0	12	0.09	0	0	0
2	5	hep-th 1-5	24	2.58	0	0	24	2.69	0	0	0
2	5	karate 1-5	8	0.01	0	0	8	0.01	0	0	0
2	5	lesmis 1-5	15	0.01	0	0	15	0.02	0	0	0
2	5	netscience 1-5	21	0.08	-	-	21	0.1	-	-	-
2	5	PGP 1-5	64	5.98	0	0	64	6.09	0	0	0
2	5	polblogs 1-5	162	52.24	55	2.27	162	55.57	62	6	2.16
2	5	polbooks 1-5	13	0.04	5	1.4	13	0.03	5	3	0.8
2	5	power 1-5	10	0.29	-	-	10	0.27	-	-	-
2	5	VB_1 1-5	1	0.01	-	-	1	0.01	-	-	-
2	5	VB_2 1-5	1	0.01	-	-	1	0.01	-	-	-
2	5	VB_3 1-5	1	0.01	-	-	1	0.01	-	-	-

[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.

Table 20: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
3	5	adjnoun 1-5	54	0.08	4	1.5	54	0.08	4	1	1.25
3	5	celegans_metabolic 1-5	200	1.79	5	1.8	200	1.76	5	1	1.6
3	5	email 1-5	91	1444.98	8993	3.85	91	1510.47	10639	2126	3.32
3	5	football 1-5	24	0.54	424	2.51	24	0.36	116	34	1.75
3	5	hep-th 1-5	47	110.88	28	2.04	47	121.08	45	12	1.53
3	5	karate 1-5	16	0.01	6	1.5	16	0.01	6	4	0.83
3	5	lesmis 1-5	34	0.02	1	4	34	0.02	1	0	4
3	5	netscience 1-5	31	0.12	0	0	31	0.08	0	0	0
3	5	PGP 1-5	204	11.76	0	0	204	11.27	0	0	0
3	5	polblogs 1-5	599	14.16	15	2.73	599	15.25	15	1	2.33
3	5	polbooks 1-5	34	0.11	5	1.2	34	0.12	5	1	1
3	5	power 1-5	15	1.03	0	0	15	1.08	0	0	0
3	5	VB_1 1-5	1	0.01	-	-	1	0.01	-	-	-
3	5	VB_2 1-5	1	0.95	0	0	1	0.98	0	0	0
3	5	VB_3 1-5	1	0.92	0	0	1	0.91	0	0	0

[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.

Table 21: Comparison of CCF and PPCF on DIMACS-10 and VB instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
4	5	adjnoun 1-5	85	0.09	4	1.25	85	0.09	4	0	1
4	5	celegans_metabolic 1-5	375	1.68	4	1	375	1.48	4	0	1
4	5	email 1-5	388	36.07	780	2.07	388	42.54	798	114	1.81
4	5	football 1-5	103	0.14	27	1.74	103	0.14	22	2	1.55
4	5	hep-th 1-5	137	614.78	1390	3.38	137	564.66	1213	220	2.72
4	5	karate 1-5	26	0.01	0	0	26	0.01	0	0	0
4	5	lesmis 1-5	49	0.03	0	0	49	0.02	0	0	0
4	5	netscience 1-5	44	0.22	0	0	44	0.27	0	0	0
4	5	PGP 1-5	437	134.87	14	1.93	437	123.16	8	1	1.75
4	5	polblogs 1-5	960	13.85	1	0	960	15.02	1	0	0
4	5	polbooks 1-5	48	0.16	4	3.5	48	0.19	9	3	1.56
4	5	power 1-5	21	1.23	37	1.46	21	1.34	37	19	0.73
4	5	VB_1 1-5	1	0.01	-	-	1	0.01	-	-	-
4	5	VB_2 1-5	1	0.88	0	0	1	0.96	0	0	0
4	5	VB_3 1-5	4	13.77	2153	1.86	4	12.33	1987	1863	0

[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.

REFERENCES

Abello, J., Pardalos, P. M., and Resende, M. G. C. (1999). On maximum clique problems in very large graphs. In *External memory algorithms and visualization*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, Boston, MA, USA.

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

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

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

Balasundaram, B. (2007). *Graph Theoretic Generalizations Of Clique: Optimization and Extensions*. PhD thesis, Texas A&M University, College Station, Texas, USA.

Balasundaram, B., Borrero, J. S., and Pan, H. (2022). Graph signatures: Identification and optimization. *European Journal of Operational Research*, 296(3):764–775.

Balasundaram, B., Butenko, S., and Hicks, I. V. (2011). Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142.

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

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

Bourjolly, J.-M., Laporte, G., and Pesant, G. (2000). Heuristics for finding k-clubs in an undirected graph. *Computers & Operations Research*, 27:559–569.

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

Charikar, M., Naamad, Y., and Wu, J. (2018). On finding dense common subgraphs. *arXiv preprint arXiv:1802.06361*.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Gurobi Optimization, LLC (2020). Gurobi optimizer reference manual.

Hellmann, T. and Staudigl, M. (2014). Evolution of social networks. *European Journal of Operational Research*, 234(3):583–596.

Himmel, A.-S., Molter, H., Niedermeier, R., and Sorge, M. (2017). Adapting the Bron–Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35.

Hu, H., Yan, X., Huang, Y., Han, J., and Zhou, X. J. (2005). Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21(suppl_1):i213–i221.

Jethava, V. and Beerenwinkel, N. (2015). Finding dense subgraphs in relational graphs. In Appice, A., Rodrigues, P. P., Santos Costa, V., Gama, J., Jorge, A., and Soares, C.,

editors, *Machine Learning and Knowledge Discovery in Databases*, pages 641–654, Cham. Springer International Publishing.

Jiang, C., Coenen, F., and Zito, M. (2013a). A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105.

Jiang, C., Coenen, F., and Zito, M. (2013b). A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105.

Jiang, D. and Pei, J. (2009). Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data*, 2(4):16:1–42.

Junker, B. H. and Schreiber, F., editors (2008). *Analysis of Biological Networks*. Wiley, New York.

Kuramochi, M. and Karypis, G. (2005). Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery*, 11(3):243–271.

Latapy, M., Viard, T., and Magnien, C. (2018). Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61.

Li, W., Liu, C.-C., Zhang, T., Li, H., Waterman, M. S., and Zhou, X. J. (2011). Integrative analysis of many weighted co-expression networks using tensor computation. *PLOS Computational Biology*, 7(6):1–13.

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

Luce, R. D. (1950). Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15(2):169–190.

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

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

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

Pajouh, F. M., Balasundaram, B., and Hicks, I. V. (2016). On the 2-club polytope of graphs. *Operations Research*, 64(6):1466–1481.

Paranjape, A., Benson, A. R., and Leskovec, J. (2017). Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 601–610, New York, NY, USA. Association for Computing Machinery.

Pattillo, J., Veremyev, A., Butenko, S., and Boginski, V. (2013a). On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1–2):244–257. 10.1016/j.dam.2012.07.019.

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

Pei, J., Jiang, D., and Zhang, A. (2005a). Mining cross-graph quasi-cliques in gene expression and protein interaction data. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 353 – 356. IEEE.

Pei, J., Jiang, D., and Zhang, A. (2005b). On mining cross-graph quasi-cliques. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 228–238, New York, NY, USA. ACM.

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

Seidman, S. B. and Foster, B. L. (1978). A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6(1):139–154.

Semertzidis, K., Pitoura, E., Terzi, E., and Tsaparas, P. (2019). Finding lasting dense subgraphs. *Data Mining and Knowledge Discovery*, 33(5):1417–1445.

Shahinpour, S. and Butenko, S. (2013). Distance-based clique relaxations in networks: s-clique and s-club. In *Models, Algorithms, and Technologies for Network Analysis*, volume 59, pages 149–174. Springer, New York.

Sim, K., Liu, G., Gopalkrishnan, V., and Li, J. (2011). A case study on financial ratios via cross-graph quasi-bicliques. *Information Sciences*, 181(1):201–216.

Trukhanov, S., Balasubramaniam, C., Balasundaram, B., and Butenko, S. (2013). Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations. *Computational Optimization and Applications*, 56(1):113–130.

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

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

Viard, T., Latapy, M., and Magnien, C. (2016). Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252.

Viard, T., Magnien, C., and Latapy, M. (2018). Enumerating maximal cliques in link streams with durations. *Information Processing Letters*, 133:44–48.

Yan, X., Zhou, X. J., and Han, J. (2005). Mining closed relational graphs with connectivity constraints. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 324–333, New York, NY, USA. ACM.

VITA

Hao Pan

Candidate for the Degree of

Doctor of Philosophy

Thesis: MINING LOW-DIAMETER CLUSTERS CONSERVED IN GRAPH COLLECTIONS

Major Field: Industrial Engineering

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Industrial Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2021.

Completed the requirements for the Bachelor of Science in Industrial Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2016.

Completed the requirements for the Bachelor of Science in Industrial Engineering at Southwest Jiaotong University, Chengdu, Sichuan in May, 2016.

Experience:

Data Scientist at Variant/U.S. Xpress since August 2021.

Operations Research Intern at Norfolk Southern Corporation, June - August 2020.

Professional Membership:

Institute for Operations Research and the Management Sciences.