# DECOMPOSITION ALGORITHMS FOR DETECTING LOW-DIAMETER CLUSTERS IN GRAPHS

By

ESMAEEL MORADI

Bachelor of Science in Industrial Engineering
Sharif University of Technology
Tehran, Iran
2006

Master of Science in Industrial Engineering
Amirkabir University of Technology (Tehran Polytechnic)
Tehran, Iran
2009

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 2016

DECOMPOSITION ALGORITHMS FOR DETECTING LOW-DIAMETER CLUSTERS IN GRAPHS

Dissertation Approved:

Dr. Balabhaskar Balasundaram

Dissertation Advisor

Dr. Manjunath Kamath

Dr. Chaoyue Zhao

Dr. Ramesh Sharda

*Dedicated to my parents*

## ACKNOWLEDGMENTS

I would like to express my exceptional appreciation and thanks to my advisor Dr. Balabhaskar Balasundaram, who has been a wonderful mentor for me. I would like to thank him for guiding me to grow as a research scientist, for his support and encouragement, for his patience, and his excellent knowledge. His advice in each step of my Ph.D. study were priceless. He has not only been a superb researcher model for my professional development, but also a great friend.

I would also like to thank my committee members, Dr. Manjunath Kamath, Dr. Ramesh Sharda, and Dr. Chaoyue Zhao for serving as my committee members and for their invaluable guidance during my Ph.D. process.

I wish to thank the School of Industrial Engineering and Management (IEM) at Oklahoma State University (OSU) for the continuous support of my Ph.D. work. I would like to especially thank Dr. Sunderesh Heragu, head of IEM department, for all his efforts to support my Ph.D. study. I also owe a warm thank you to Laura Brown, Megan Hughes, and Sarah Lewis for their friendly attitude and great administrative support.

My Ph.D. work was supported in part by grants from the U.S. Department of Energy (DE-SC0002051), Air Force Office of Scientific Research (FA9550-12-1-0103), and the National Science Foundation (CMMI-1404971). The support from these grants is sincerely appreciated. The computational experiments reported in this dissertation were conducted at the Oklahoma State University High Performance Computing Center. I am thankful to Dr. Dana Brunson and Jesse Schafer for their support in conducting these computational experiments. Also, I would like to thank Dr.

Alexander Veremyev for sharing his experimental test-bed with me.

A special thanks to my family. Words cannot express how grateful I am to my mother, father, brother, and sisters for all their support and encouragement. Your prayer for me was what sustained me thus far.

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

Name: Esmaeel Moradi

Date of Degree: May, 2016

Title of Study: DECOMPOSITION ALGORITHMS FOR DETECTING LOW-DIAMETER CLUSTERS IN GRAPHS

Major Field: Industrial Engineering and Management

Detecting low-diameter clusters in graphs is an effective graph-based data mining technique, which has been used to find cohesive subgraphs in a variety of graph models of data. Low pairwise distances within a cluster can facilitate fast communication or good reachability between vertices in the cluster. A $k$-club is a subset of vertices, which induces a subgraph of diameter at most $k$. For low values of the parameter $k$, this model offers a graph-theoretic relaxation of the clique model that formalizes the notion of a low-diameter cluster. The maximum $k$-club problem is to find a $k$-club with maximum cardinality in a given graph. The goals of this study are focussed on developing decomposition and cutting plane methods for the maximum $k$-club problem for arbitrary $k$.

Two compact integer programming formulations for the maximum $k$-club problem were presented by other researchers. These formulations are very effective integer programming approaches presently available to solve the maximum $k$-club problem for any given value of $k$. Using model decomposition techniques, we demonstrate how the fundamental optimization problem of finding a maximum size $k$-club can be solved optimally on large-scale benchmark instances. Our approach circumvents the use of complicated formulations in favor of a simple relaxation based on necessary conditions, combined with canonical hypercube cuts introduced by Balas and Jeroslow. Next, we demonstrate that by using a delayed constraint generation approach in a branch-and-cut algorithm, we can significantly speed-up the performance of an integer programming solver over the direct solution of the implementation of either formulation.

Then, we study the problem of detecting large risk-averse 2-clubs in graphs subject to probabilistic edge failures. To achieve risk aversion, we first model the loss in 2-club property due to probabilistic edge failures as a function of the decision (chosen 2-club cluster) and randomness (graph structure). Then, we utilize the conditional value-at-risk of the loss for a given decision as a quantitative measure of risk, which is bounded in the stochastic optimization model. A sequential cutting plane method that solves a series of mixed integer linear programs is developed for solving this problem.

ADVISOR'S APPROVAL: ______________________________

TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

The need for extracting hidden information from large data-sets in different applications, such as social network analysis, bioinformatics, text-mining, internet analytics, and finance, increased a need for scalable techniques in data mining [70, 96, 60]. Generally, the process of analyzing data and summarizing it into useful information (knowledge) is called *data mining* [95]. Graph theory can be used to model the components of a system and the relations among them in different data mining applications. In these applications, graphs can also be used as visualization tools that represent the interconnected information about the elements of a system which are shown by points (vertices, nodes) in space that are connected by lines (edges, arcs) [65]. A variety of graph algorithms and optimization techniques are then used to find specific patterns from a graph model in various data mining applications [32, 70, 104].

## 1.1 Graph models of data

A graph $G$ is defined by a set of vertices $V$ in which some pairs of vertices are connected by a set of edges $E$ and is denoted by $G = (V, E)$. Figure 1.1 illustrates a simple graph.

Graphs are used to model real-world systems in a variety of applications, such as social network analysis (SNA) [105], bioinformatics, internet analytics, text-mining, etc. In these graph models, vertices represent the entities of a system and edges show similarities/dissimilarities between them.

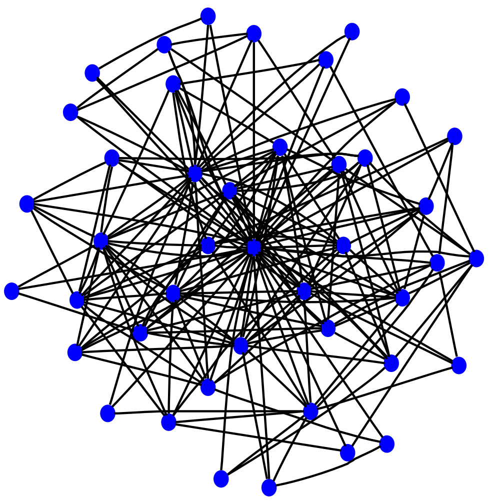
Figure 1.1: A graph example

*Social networks* are graph models in which sociological information, such as acquaintance or friendship among people, is represented. In these networks, people are considered as vertices and edges represent the relationship between them [105]. *Scientific collaboration networks* show the collaborations between authors in a specific scientific community in which authors are shown by vertices and co-authorship between them is modeled by edges [43, 48, 51, 92]. The connections between IP addresses are modeled in *internet graphs* in which IP addresses are represented by vertices and the flow of information between those IP addresses are shown by edges [25]. A *call graph* has vertices representing telephone numbers; edges in such graphs indicate a call placed between two vertices in a specific time interval [1]. *Stock-market graphs* represent the correlation between stock prices. In these graphs, stocks are shown by vertices and an edge between two stocks exists if their prices are positively correlated over a period of time in the stock-market history [20, 18, 19]. *Protein interaction networks* (PIN), *gene co-expression networks* (GCE), and *metabolic networks* are biological networks that model relationships between biological entities. PINs model protein interaction information. In these networks, vertices represent proteins and an edge between two proteins exists if they interact with each other [52, 39, 93]. GCEs have vertices representing genes and an edge exists between two genes if they are co-expressed with a correlation higher than a specific threshold in micro-array experiments [80]. In metabolic networks, metabolites are shown by vertices and conversion of metabolite $i$ to metabolite $j$ is represented by the directed edge from $i$ to $j$.

## 1.2 Cluster models in graphs

Detecting graph-theoretic clusters is an effective graph-based data mining technique, which has been used to find tightly-knit subgraphs in a variety of graph models of data. The notion of a graph-theoretic cluster could be described by considering one or more of the following structural properties: (1) Each vertex is adjacent to large number of vertices inside the cluster (vertex degree), (2) The pairwise distances between the vertices in the subgraph induced by the cluster is small (distance), (3) A large number of edges with both endpoints is included in the cluster (edge density), and (4) The minimum number of vertices or edges whose removal causes a disconnected cluster is large [79].

A *clique* is an ideal cluster model according to the aforementioned properties. A clique is a subset of vertices for which the induced subgraph is complete. In other words, a clique is a subgraph in which all vertices are pairwise adjacent. Evidently, deletion of any node in a clique does not violate the clique structure of the remaining nodes. Cliques have been used in several real-world applications in which detecting "tightly-knit" subgraphs, called *cohesive subgraphs*, is necessary [38, 86]. However, the requirement of complete pairwise adjacency in a clique makes it overly restrictive for most real-world problems and led to the use of clique relaxation structures [16]. These relaxations are based on the aforementioned structural properties, which are as follows: vertex degree ($k$-plex), distance ($k$-clique and $k$-club), and edge density ($\gamma$-quasi-clique) [16]. This dissertation investigates $k$-clubs, which are distance-based relaxations that model low-diameter clusters in graphs.

The $k$-clubs have been the choice for clusters in a variety of applications. For example, $k$-club models have been used to detect protein complexes in PINs. Protein complexes, which are identified as functional modules, are cellular entities that perform certain biological functions. Researchers in biology need to find these modular structures in order to understand how cells function [15]. Terveen et al. [96] used the 2-club model in order to cluster web sites. They studied approaches to enable faster search among web sites to gain specific information by using the 2-club models. Miao and Berleant [70] used the concept of the $k$-club models in document graphs in which vertices represent documents and edges represent links between paragraphs of the documents. They investigated retrieving information from documents and found that the two-hop transitivity captured by the 2-club concept is very intuitive in this application.

## 1.3 Notations and definitions

We consider a simple, undirected, connected graph $G = (V, E)$ with a set of vertices $V = \{1, \dots, n\}$ and a set of edges $E$. $G[S] = (S, E \cap (S \times S))$ denotes the subgraph induced by $S \subseteq V$. The *distance* between a pair of vertices $i$ and $j$ in $G$, denoted by $d_G(i, j)$, is the length of the shortest path in number of edges. The maximum distance between any pair of vertices in $G$ is called the *diameter* of $G$ and denoted by $diam(G)$, i.e., $diam(G) = \max\{d_G(i, j) \mid i, j \in V\}$. The edge density of $G$, denoted by $\rho$, measures how many edges are in set $E$ compared to the maximum possible number of edges between vertices in set $V$. A power graph is defined as follows: $k$-th power of $G$ is denoted by $G^k = (V, E^k)$ where $E^k = \{(i, j) | d_G(i, j) \leq k, i < j\}$. The neighborhood of vertex $i$ is a set of vertices adjacent to $i$ in $G$, which is denoted by $N_G(i)$. The closed neighborhood of vertex $i$ is the set of neighbors of $i$ denoted by $N_G[i]$, i.e., $N_G[i] = N_G(i) \cup \{i\}$.

The $k$-neighborhood of $i$ in $G$ is defined as $N_G^k(i) = \{j \in V : 1 \leq d_G(i, j) \leq k\}$. The 2-neighborhood of node 1 in Figure 1.2 is $N_G^2(1) = \{2, 3, 4, 5\}$. The set $I \subseteq V$

is called an independent set if no two vertices of $I$ are adjacent. It is known as a $k$-independent set if for every distinct pair of vertices $i, j \in I$, $d_G(i, j) \geq k + 1$. In Figure 1.2, the sets $I = \{1, 4, 7\}$ and $I' = \{1, 7\}$ are independent set and 2-independent set respectively.

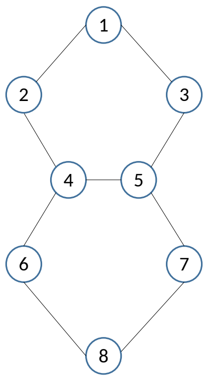

Figure 1.2: In this graph, $\{1, 4, 7\}$ is an independent set while $\{1, 7\}$ is 2-independent. It is also a 2-dominating set

The set $S \subseteq V$ is a $k$-dominating set if for any $v \in V \backslash S$ there exist $v' \in S$ such that $d_G(v, v') \leq k$. When $k = 1$, it is simply called a dominating set. The set $S = \{1, 7\}$ is a 2-dominating set in the graph in Figure 1.2. A subset of vertices in which every pair of vertices are adjacent is a clique. Figure 1.3 illustrates a clique.

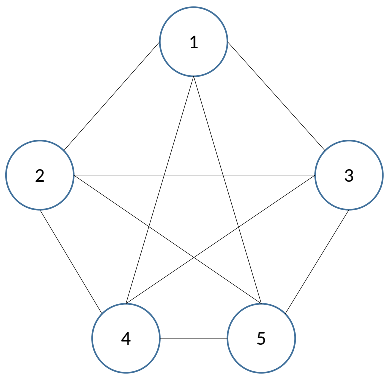

Figure 1.3: A clique

The maximum clique problem is to find a clique with maximum cardinality in a graph. The clique number of $G$ is denoted by $\omega(G)$, which is the size of a clique with maximum cardinality in $G$.

**Definition 1 ([15, 3, 71])** *A subset of vertices $S$ is called a $k$-clique if $d_G(u,v) \leq k$ for all $u, v \in S$. A subset $S$ is a $k$-clique in $G$ if and only if $S$ is a clique in the power graph $G^k$.*

**Definition 2 ([15, 3, 71])** *A subset of vertices $S$ is called a $k$-club if $diam(G[S]) \leq k$.*

Invariant $\widetilde{\omega}_k(G)$ denotes the $k$-clique number of $G$ that is the size of a $k$-clique with maximum cardinality in $G$. The size of a $k$-club with maximum cardinality in $G$ is called the $k$-club number of $G$, and is denoted by $\overline{\omega}_k(G)$. The maximum $k$-club problem (MkCP), which is the focus of this dissertation, is to find a $k$-club with maximum cardinality in $G$.

When $k = 1$, $k$-cliques and $k$-clubs represent a clique, and a relaxation when $k \geq 2$. Since $d_{G[S]}(i,j) \geq d_G(i,j)$, a $k$-club is always a $k$-clique but the converse is not true. For example, in the graph illustrated in Figure 1.4, the set $S = \{1, 2, 3, 4, 5\}$ is a 2-clique but not a 2-club since $d_G(1,5) = 2$ and $d_{G[S]}(1,5) = 3$.

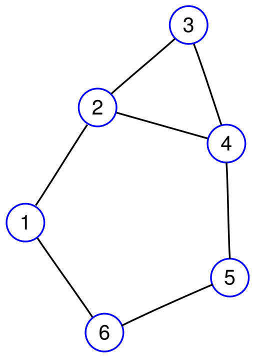

Figure 1.4: 2-cliques vs. 2-clubs: $S = \{1, 2, 3, 4, 5\}$ is a 2-clique that is not a 2-club

In the next chapter, we review related literature on solving the maximum $k$-club problem and provide the required background to highlight the challenges involved. The specific research objectives of this dissertation are also identified in the next chapter.

# CHAPTER 2

# BACKGROUND AND LITERATURE REVIEW

A review of complexity, approximation, polyhedral results, and algorithms for the distance-based clique relaxations ($k$-clique and $k$-club) is presented in this chapter.

## 2.1 Nonhereditary nature of $k$-clubs

The maximum clique problem has been the subject of extensive research. Several surveys [77, 21, 54] document work on this central combinatorial optimization problem. Also, many researchers have studied clique relaxations and their applications [16, 79]. Recently, Shahinpour and Butenko [91] surveyed distance-based clique relaxations, namely $k$-cliques and $k$-clubs. In this chapter, we limit our focus to complexity, approximation, polyhedral results, and algorithms for the distance-based clique relaxations. Before reviewing the relevant literature, we highlight a combinatorial feature of $k$-clubs that makes it both challenging and interesting to study.

After the clique model was used to find tightly-knit groups in social network analysis by Luce and Perry [63], a generalized version of clique, called $k$-clique, was introduced by Luce [62]. Although the concept of $k$-clique was pioneered by Luce, it lacked a computational procedure for detecting one [3]. Other researchers [22, 11] worked on developing procedures to find these clusters. The $k$-clique model relaxed the requirement of adjacency in the clique to a shortest path requirement. This model led to another clique relaxation, the $k$-club [3, 71]. The difference between these two models is discussed in detail by Balasundaram et al. [15]. In the $k$-clique model, a vertex from outside the group can be considered in the shortest path between vertices of the group, which could be a drawback for some applications in social network analysis. The $k$-club model overcomes this drawback. Recalling that $\omega(G)$ denotes the clique number, $\widetilde{\omega}_k(G)$ represents the $k$-clique number, and $\overline{\omega}_k(G)$ indicates the $k$-club number, the following inequality is valid for all graphs in general:

$$\omega(G) \leq \overline{\omega}_k(G) \leq \widetilde{\omega}_k(G).$$

Notice that finding a maximum $k$-clique in $G$ is equivalent to finding a maximum clique in $G^k$. For example, Figure 2.1 represents the square of the graph illustrated in Figure 1.4. In this graph, set $S = \{1, 2, 3, 4, 5\}$ is a clique which is a 2-clique in the graph presented in Figure 1.4.

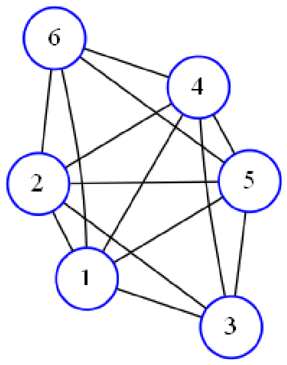

Figure 2.1: Graph $G^2$

The $k$-clique model is hereditary in the sense that every subset of a $k$-clique is a $k$-clique. In contrast, the $k$-club model is not hereditary. For example, the set $S = \{1, 2, 4, 5, 6\}$ in the graph in Figure 1.4 is a 2-club in which every subset is a 2-clique but not necessarily a 2-club. For instance, subset $\{1, 2, 4, 5\}$ is a 2-clique but not a 2-club. This property of the $k$-club model makes it extremely challenging to develop exact and heuristic algorithms for the MkCP [65].

## 2.2 Complexity and approximation

Bourjolly et al. [24] proved that the MkCP is NP-hard for any fixed positive integer $k$. Balasundaram et al. [15] showed that the MkCP is NP-hard in graphs of fixed diameter with $diam(G) > k$ and thus is NP-hard even in graphs with diameter $k+1$. Kahruman-Anderoglu et al. [55] demonstrated that the problem of recognizing a gap between $\overline{\omega}_k$ and $\overline{\omega}_\ell$ (if there is a positive gap between them) is NP-hard for any positive integers $k$ and $\ell$ ($k \neq \ell$). This result was then used to show that, unless P = NP, a polynomial-time algorithm cannot be designed to detect a $k$-club of size greater than $\Delta(G)+1$ in any graph $G$ with $\overline{\omega}_k > \Delta(G)+1$ and for any fixed positive integer $k \geq 2$, where $\Delta(G)$ denotes the maximum degree of a vertex in $G$.

Marincek and Mohar [68] proved that the MkCP is inapproximable within a factor of $n^{\frac{1}{3}-\epsilon}$ for any $\epsilon > 0$ as long as P $\neq$ NP. Following which, Asahiro et al. [10] improved this result by showing that it is inapproximable even within a factor of $n^{\frac{1}{2}-\epsilon}$. Further, Asahiro et al. [10] developed approximation algorithms of factor $n^{\frac{1}{2}}$ and $n^{\frac{1}{3}}$ for even and odd $k$, respectively.

Downey and Fellows [35] proved that detecting cliques of a particular size is not fixed-parameter tractable and, in fact, it is a basic $W[1]$-hard problem. In stark contrast to cliques, $k$-clubs (for $k \geq 2$) are fixed-parameter tractable with respect to the size of a solution. The MkCP is solvable in $O(nk^2)$ on trees, in $O(n^2)$ on interval graphs, and when $k = 2$, in $O(n^5)$ on bipartite graphs [87].

In this dissertation, the maximality and minimality of sets are defined based on inclusion and exclusion, respectively. Meaning that, a maximal set is one that is not a proper subset of another set [65] also satisfying a required property. Mahdavi Pajouh and Balasundaram [66] derived results on the intractability of testing inclusion-wise maximality of $k$-clubs. Since $k$-clubs are non-hereditary, their results were in contrast to polynomial-time verifiability of maximal cliques. They also found a class of graphs for which maximality testing is polynomial-time solvable.

## 2.3 Formulations

The MkCP for general positive integer $k$ can be formulated as follows [24, 15]:

$$\bar{\omega}_k(G) = \max \sum_{i \in V} x_i \tag{2.1}$$

*subject to:*

$$x_i + x_j \leq 1 + \sum_{\ell : P_{ij}^{\ell} \in \mathbb{P}_{ij}} y_{ij}^{\ell} \qquad \forall (i,j) \notin E, \tag{2.2}$$

$$x_p \geq y_{ij}^{\ell} \qquad \forall p \in V(P_{ij}^{\ell}),\ P_{ij}^{\ell} \in \mathbb{P}_{ij},\ (i,j) \notin E, \tag{2.3}$$

$$x_i \in \{0,1\} \qquad \forall i \in V, \tag{2.4}$$

$$y_{ij}^{\ell} \in \{0,1\} \qquad \forall P_{ij}^{\ell} \in \mathbb{P}_{ij},\ (i,j) \notin E, \tag{2.5}$$

where $\mathbb{P}_{ij}$ indicates an indexed collection of all paths of length at most $k$ between vertices $i$ and $j$ in $G$, and $P_{ij}^{\ell}$ denotes a path with index $\ell$ between vertices $i$ and $j$. According to this formulation, if vertices $i, j$ are selected in a $k$-club, all the vertices in at least one of the paths of length at most $k$ between them should be selected. The special cases where $k = 2, 3$ admit compact formulations, which has facilitated more detailed analysis [32, 28, 6, 7]. Note that $|\mathbb{P}_{ij}|$ could be exponentially large with respect to the number of vertices and $k$. Therefore this formulation is difficult to handle for large instances, and for large $k$.

To address this drawback, Veremyev and Boginski [101] formulated the MkCP as a binary polynomial programming problem. They showed that the linearization of their mathematical formulation yields a compact formulation that is far more practical than the path-based formulation. This binary polynomial programming formulation is presented next. In the following, $a_{ij}$ is used to denote the $i$-th and $j$-th element of the adjacency matrix of graph $G$.

$$\begin{aligned}
\bar{\omega}_k(G) &= \max \sum_{i \in V} x_i && (2.6)\\
\textit{subject to:}\\
x_i &+ x_j - 1 \le a_{ij} + \sum_{\ell \in V} a_{i\ell} a_{\ell j} x_\ell + \sum_{\ell \in V} \sum_{m \in V} a_{i\ell} a_{\ell m} a_{mj} x_\ell x_m\\
&+ \sum_{\ell \in V} \sum_{m \in V} \sum_{t \in V} a_{i\ell} a_{\ell m} a_{mt} a_{tj} x_\ell x_m x_t\\
&+ \cdots\\
&+ \sum_{i_1 \in V} \cdots \sum_{i_{k-1} \in V} a_{ii_1} \ldots a_{i_{k-2} i_{k-1}} a_{i_{k-1} j} x_{i_1} \ldots x_{i_{k-1}} \; \forall i < j \in V, && (2.7)\\
x_i &\in \{0, 1\} \; \forall i \in V. && (2.8)
\end{aligned}$$

Note that if $x_i = x_j = 1$, then at least one of the terms in the right-hand side of the constraint in this formulation must be 1. If $a_{ij} \neq 1$, then some binary product term must be 1. The correctness of the formulation follows from the observation that a term $a_{ii_1} a_{i_1 i_2} \ldots a_{i_{\ell-2} i_{\ell-1}} a_{i_{\ell-1} j} x_{i_1} \ldots x_{i_{\ell-1}}$ is 1 if and only if $G$ contains a path $i - i_1 - \cdots - i_{\ell-1} - j$ of length $\ell$, and all the internal vertices on this path are also selected in the solution $x$. This "nonlinear" approach to formulating this problem immediately yields a linearization formulation presented next, which is referred as F1 in this document.

The trick used in this linearization is not the textbook technique for linearizing binary polynomial products [73]. This formulation recursively forces the linearizing variables $z_{ij}^{\ell}$ to take on the correct value using the "connectedness" property in these linearizing variables that is required by the induced path requirements of a $k$-club. This mathematical formulation uses $O(kn^2)$ decision variables and constraints, which is significantly less than the formulation (2.1)-(2.5) that uses $O(n^{k+1})$ entries.

$$\textbf{(F1)}\quad \bar{\omega}_k(G) = \max \sum_{i \in V} x_i \tag{2.9}$$

*subject to:*

$$x_i + x_j - 1 \leq a_{ij} + \sum_{\ell=2}^{k} z_{ij}^{\ell} \ \forall i < j \in V, \tag{2.10}$$

$$z_{ij}^2 \leq x_i, \quad z_{ij}^2 \leq x_j, \quad z_{ij}^2 \leq \sum_{\ell=1}^{n} a_{i\ell} a_{\ell j} x_{\ell} \ \forall i < j \in V, \tag{2.11}$$

$$z_{ij}^2 \geq \frac{1}{n}\Big(\sum_{\ell=1}^{n} a_{i\ell} a_{\ell j} x_k\Big) + (x_i + x_j - 2) \ \forall i < j \in V, \tag{2.12}$$

$$z_{ij}^{\ell} \leq x_i, \quad z_{ij}^{\ell} \leq \sum_{t=1}^{n} a_{it} z_{tj}^{\ell-1} \ \forall \ell = 3, \ldots, k; \ \forall i < j \in V, \tag{2.13}$$

$$z_{ij}^{\ell} \geq \frac{1}{n}\Big(\sum_{t=1}^{n} a_{it} z_{tj}^{\ell-1}\Big) + (x_i - 1) \ \forall \ell = 3, \ldots, k; \ \forall i < j \in V, \tag{2.14}$$

$$x_i \in \{0, 1\} \ \forall i \in V, \tag{2.15}$$

$$z_{ij}^{\ell} \in \{0, 1\} \ \forall i < j \in V; \ \ell = 2, \ldots, k. \tag{2.16}$$

Another important point to note here is that unlike the path-based formulation that is not suitable for direct-solution, this formulation is compact using $O(kn^2)$ variables and constraints. This means a general-purpose solver could be leveraged to solve relatively larger-scale instances, especially for larger values of $k$. This was one of the key computational advantages demonstrated by Veremyev and Boginski [101]. Recently, Veremyev et al. [102] introduced a slightly different formulation, which improves the performance of solvers over using formulation F1. This formulation is presented next and referred to as F2 in this document.

$$\textbf{(F2)} \quad \bar{\omega}_k(G) = \max \sum_{i \in V} x_i \tag{2.17}$$

*subject to:*

$$x_i + x_j - 1 \leq u_{ij}^k, \ \forall i, j \in V \mid i \neq j, \tag{2.18}$$

$$u_{ij}^1 = 0, \ \forall (i,j) \notin E \mid i \neq j, \tag{2.19}$$

$$u_{ij}^1 = u_{ij}^\ell, \ \forall (i,j) \in E, \ \ell = 2, \ldots, k, \tag{2.20}$$

$$u_{ij}^\ell \leq \sum_{t=1}^{n} a_{it} u_{tj}^{\ell-1}, \ \forall (i,j) \notin E \mid i \neq j, \ \ell = 2, \ldots, k, \tag{2.21}$$

$$u_{ij}^\ell \leq x_i, \quad u_{ij}^\ell \leq x_j, \ \forall i, j \in V \mid i \neq j, \ \ell = 1, \ldots, k, \tag{2.22}$$

$$u_{ij}^\ell = u_{ji}^\ell, \ \forall i, j \in V \mid i \neq j, \ \ell = 1, \ldots, k, \tag{2.23}$$

$$u_{ij}^\ell \in [0,1], \ \forall i, j \in V \mid i \neq j, \ \ell = 1, \ldots, k, \tag{2.24}$$

$$x_i \in \{0,1\}, \ \forall i \in V. \tag{2.25}$$

## 2.4 Polyhedral results

Denoted by $Q_k(G)$, the $k$-club polytope of a graph $G$ is the convex hull of the incidence vectors of $k$-clubs in $G$. Balasundaram et al. [15] developed the first family of facet of $Q_2(G)$. Theorems 1 and 2 present their results.

**Theorem 1 ([15])** *Consider the 2-club polytope $Q_2(G)$ of a graph $G = (V, E)$.*

1. *$dim(Q_2(G)) = n$.*
2. *$x_i \geq 0$ induces a facet of $Q_2(G)$ for every $i \in V$.*
3. *For $i \in V$, $x_i \leq 1$ induces a facet of $Q_2(G)$ if and only if $d_G(i,j) \leq 2 \ \forall j \in V$.*

**Theorem 2 ([15])** *The inequality $\sum_{i \in I} x_i \leq 1$ induces a facet of $Q_2(G)$ if and only if $I$ is a maximal 2-independent set in $G$.*

Carvalho and Almeida [28] presented valid inequalities for 2-club polytope and developed necessary and sufficient conditions under which these inequalities will define facets for the polytope. In another study, Almeida and Carvalho [5, 4] introduced new mathematical formulations for the MkCP. A new upper bound for the maximum 3-club problem is defined in their study by using a compact formulation as a relaxation of their proposed formulations. Also, they derived new families of inequalities for the 3-club polytope and used them to strengthen the linear programming (LP) relaxation of their mathematical formulations. More recently, Mahdavi Pajouh et al. [67] introduced a family of facet inducing inequalities that unify the results presented in [28] and [15]. Theorem 3 presents this result.

**Theorem 3 ([67])** *Let $I$ be an independent set in $G$. Then the inequality*

$$\sum_{i \in I} x_i - \sum_{j \in V \setminus I} (|N_G(j) \cap I| - 1)^+ x_j \leq 1$$

*is valid for the 2-club polytope, where $(t)^+ = \max\{0, t\}$. If, in addition, $I$ is a 2-dominating set, i.e., if $I$ is an independent 2-dominating set, then this inequality induces a facet of the 2-club polytope.*

## 2.5 Exact algorithms and heuristics

Well-known exact algorithms for cliques like the algorithms developed by Carraghan and Pardalos [27] and Östergård [76] are not directly extended to $k$-clubs since they are nonhereditary and testing inclusionwise maximality is NP-hard. Bourjolly et al. [24], Schäfer [87], and Mahdavi Pajouh and Balasundaram [66] introduced combinatorial algorithms for the problem. A $k$-clique number is used by Bourjolly et al. [24] as an upper bound at each node of the search tree, while Mahdavi Pajouh and Balasundaram [66] use the distance-$k$ coloring problem to obtain upper bounds at each node of the search tree. The former approach provides tighter bounds but needs to solve an NP-hard problem (the maximum $k$-clique problem) at each node of the search tree. The upper bounds provided by the latter approach are weaker, but this approach employs a heuristic coloring algorithm, which is faster, at each node to obtain the bounds. Chang et al. [30] implemented a combinatorial branch-and-bound algorithm, which is shown to solve the MkCP in $O(1.62^n)$. Schäfer [87] introduced a fixed-parameter tractable algorithm to find a $k$-club of size $c$ (if it exists) in $G$ in $O((c-2)^c \times c! \times c^3 n + nm)$ time. Mahdavi Pajouh et al. [67] presented a new family of facet inducing inequalities for the 2-club polytope (see Theorem 3) and developed a branch-and-cut algorithm to find a maximum 2-club in a graph.

Heuristic algorithms for finding $k$-cliques and $k$-clubs have been widely studied recently [23, 28, 36, 55]. Three heuristic algorithms, which were named CONSTELLATION, DROP, and $k$-Clique & DROP were described in [23]. The principle of the CONSTELLATION algorithm stemmed from the idea that a star graph provides a 2-club. This idea was then generalized to find $k$-clubs for $k \geq 3$. The DROP algorithm computes the length of the shortest path between every pair of vertices in $G$ then, removes a vertex with smallest size of $k$-neighborhood until $G$ becomes a $k$-club. The $k$-Clique & DROP algorithm applies the DROP algorithm in a subgraph induced by a largest $k$-clique in $G$. Carvalho and Almeida [28] presented a heuristic algorithm, which was similar to DROP. The main difference was that the optimal solution from the LP relaxation of the $k$-club formulation is used to guide the deletion of vertices from the graph. Chang et al. [30] introduced a heuristic algorithm called iterative DROP (iDROP). For each vertex $i$ in the graph, iDROP applies the DROP algorithm to find a $k$-club in the subgraph induced by the $k$-neighborhood of $i$ and then, returns the $k$-club with the largest cardinality among them. A sufficient condition for testing maximality of a given $k$-club is provided by Shahinpour and Butenko [90]. Following which, they develop a variable neighborhood search that is used to provide a lower bound for initializing the exact algorithm from [66]. The heuristic algorithm introduced in [28] used the LP relaxation of the formulation for the 2-club problem to compute an upper bound on the 2-club number of the graph and guide the generation of feasible solutions. In the heuristic algorithm presented in [36], the $k$-clique model was used to separate a graph into several subgraphs and lead the algorithm to find a $k$-club. The heuristic algorithms could be used to improve the lower bound in exact approaches like the combinatorial branch-and-bound algorithms as reported in [90].

## 2.6 CVaR in optimization

In this dissertation, we also explore the maximum 2-club problem in a probabilistic setting, specifically, assuming independent probabilistic edge failures. In the probabilistic setting, we model the low-diameter cluster detection problem as a Conditional Value-at-Risk (CVaR) constrained optimization problem.

Rockafellar and Uryasev [84, 85] defined the notion of CVaR, which is a measure of downside risk of heavy losses. A loss function $L(x, Y)$ quantifies losses as a function of a decision vector $x$ and uncertainty, modeled by a random vector $Y$. Then, for a given decision vector $x^0$ and $\alpha \in (0, 1)$, $\alpha$-Value-at-Risk (VaR) is the $\alpha$-quantile of the loss distribution $\Psi(x^0, \ell) = Pr\{L(x^0, Y) \leq \ell\}$, and $\alpha$-CVaR is the mean of the (1-$\alpha$)-tail of the loss distribution. Rockafellar and Uryasev [84, 85] pioneered the approach minimizing CVaR rather than VaR in the context of portfolio optimization, and laid the mathematical foundations for this approach. Uryasev [98] also found through empirical studies that portfolios that were CVaR optimal for their test-bed were also near-optimal in terms of VaR. CVaR is also considered a coherent measure of risk in the sense of Artzner et al. [9]. Consequently, this approach has been adapted in a variety of ways for designing optimal portfolios and in other financial applications (See [8, 81]).

Rockafellar and Uryasev [84] showed that the computation of CVaR can be stated as the following univariate minimization problem for a given decision $x$:

$$\alpha\text{-CVaR}[L(x,Y)] = \min_{\zeta\in\mathbb{R}}\Big\{\zeta + \frac{1}{1-\alpha}\mathbb{E}\big[[L(x,Y)-\zeta]^+\big]\Big\}. \tag{2.26}$$

If the distribution of random vector $Y$ is approximated by a set of $N$ samples $y^1,\ldots,y^N$ with probability $\pi_l$ for $l = 1,\ldots,N$, this optimization problem can be approximated as follows:

$$\alpha\text{-CVaR}[L(x,Y)] \simeq \min_{\zeta\in\mathbb{R}}\Big\{\zeta + \frac{1}{1-\alpha}\sum_{l=1}^{N}\pi_l[L(x,y^l)-\zeta]^+\Big\}. \tag{2.27}$$

Based on this observation, the generic CVaR-constrained optimization problem with a user specified limit $d$ can be stated as follows:

$$\max_{x\in P} \quad c^Tx \tag{2.28}$$

*subject to:*

$$\min_{\zeta\in\mathbb{R}}\Big\{\zeta + \frac{1}{1-\alpha}\sum_{l=1}^{N}\pi_l[L(x,y^l)-\zeta]^+\Big\} \leq d. \tag{2.29}$$

Following Krokhmal et al. [58], the CVaR constraint (2.29) can be replaced with the equivalent constraint (2.31) resulting in the following formulation:

$$\max_{\zeta\in\mathbb{R},x\in P} \quad c^Tx \tag{2.30}$$

*subject to:*

$$\zeta + \frac{1}{1-\alpha}\sum_{l=1}^{N}\pi_l[L(x,y^l)-\zeta]^+ \leq d. \tag{2.31}$$

For the sake of argument, let us assume that the loss function is linear in $x$, and $P$ is a polyhedron in formulation (2.30)-(2.31). Then, direct solution of this formulation using large-scale linear optimization methods is possible after using additional variables to linearize constraint (2.31). Typically, a good approximation of the loss function's probability distribution requires a large number of samples, which in turn results in the linearization (2.31) requiring a large number of variables. Thus, solving a direct reformulation/linearization of problem (2.30)-(2.31) can be a challenging task. If the loss function is convex and piecewise linear in $x$ (as is the case with our loss function in Chapter 5), after linearizing constraints (2.31), we would have to linearize $L(x, y^l)$, which would require additional variables. Finally, when $P$ is nonconvex (as it is in Chapter 5 due to binary restrictions on $x$), other issues need to be addressed to solve (2.30)-(2.31).

## 2.7 Research gaps and objectives

The lack of exact sophisticated algorithms, such as decomposition algorithms to address the MkCP for general $k$ is evident. The nonhereditary nature of the $k$-club model is a fundamental challenge in developing exact and heuristic algorithms for the MkCP. Even though the nonhereditary nature of $k$-clubs has been considered in the literature (see [15, 66]), developing exact algorithms for solving the MkCP for general $k$ is very much open. The known exact algorithms to solve the MkCP are limited to the two branch-and-bound algorithms presented in [24] and [66]. These algorithms show weaknesses to solve the MkCP for instances larger than 200-vertex graphs. There are two compact and linear integer programming formulations available in the literature (see [101, 102]). The performance of these formulations reduces when size of graph and value of $k$ increase.

Besides, the existing literature on the $k$-club model is limited to deterministic graphs. There is no study of the MkCP on random graphs in which vertices or edges have probabilities for failure.

These facts have motivated us to conduct an algorithmic study of the $k$-club model on deterministic and random graphs in order to address the gaps in literature for this distance-based clique relaxation model. Our research objectives in this dissertation are as follows:

- *Objective 1.* Investigate model decomposition and relaxation techniques for solving the MkCP for arbitrary $k$.
- *Objective 2.* Investigate the performance of the decomposition algorithms based on different type of cuts.
- *Objective 3.* Explore the maximum 2-club problem in graphs subject to probabilistic edge failure using a CVaR-constrained optimization model; extend algorithmic techniques from literature to solve this model.

## 2.8 Organization of the dissertation

A basic decomposition algorithm for solving the MkCP for general $k$ is presented in Chapter 3. Chapter 4 shows that adding the constraints (2.10)-(2.14) in a delayed fashion as cuts in a branch-and-cut algorithm results in a more effective decomposition algorithm, which equips us to solve the MkCP faster for general $k$. Chapter 5 investigates detecting large 2-clubs in graphs subject to probabilistic edge failures. We quantify the risk of losing the desired 2-club property in detected clusters due to edge failures using the concept of conditional value-at-risk. Finally, Chapter 6 concludes this dissertation and presents some interesting research directions for future work.

# CHAPTER 3

# A BASIC DECOMPOSITION APPROACH[1]

A new approach to solve the MkCP for general $k$ is proposed in this chapter, which is of particular interest for $k \geq 3$. The path-based integer programming (IP) formulation of the MkCP discussed in Section 2.3 introduced by Bourjolly et al. [24] has exponentially many variables and constraints. Mahdavi Pajouh et al. [67], Almeida and Carvalho [6, 7], and Carvalho and Almeida [28] have studied the special cases of this IP formulation where $k = 2$ and 3, which admit compact formulations. The two IP formulations introduced by Veremyev and Boginski [101] and Veremyev et al. [102] discussed in Section 2.3 are promising for arbitrary $k$ for direct solution using integer programming solvers. These two IP formulations are compared in our computational studies to show the effectiveness of our proposed approach.

## 3.1 Decomposition and branch-and-cut

Our basic approach for solving the MkCP using decomposition is presented in this section. Since every $k$-club is also a $k$-clique, the simple edge-based formulation of the maximum $k$-clique problem, which is presented next, is used as a relaxation of

[1]Contents of this chapter are reproduced with permission from E. Moradi and B. Balasundaram. Finding a maximum $k$-club using the $k$-clique formulation and canonical hypercube cuts. Optimization Letters, page 1-11; 2015. DOI: 10.1007/s11590-015-0971-7 © Springer

the MkCP in our proposed approach.

$$(\mathcal{P}^0) \qquad \tilde{\omega}_k(G) = \max \sum_{i \in V} x_i \tag{3.1}$$

*subject to:*

$$x_i + x_j \leq 1 \ \forall i, j \in V, d_G(i,j) > k, \tag{3.2}$$

$$x_i \in \{0,1\}, \quad \forall i \in V. \tag{3.3}$$

Our approach starts by solving the relaxation $(\mathcal{P}^0)$ using a branch-and-bound (BB) approach. Suppose an IP feasible solution $\hat{x} \in \{0,1\}^n$ is encountered in some node of the BB tree. If $\hat{x}$ corresponds to a $k$-club, the BB node will be pruned by feasibility. Otherwise, $\hat{x}$ corresponds to a $k$-clique, which is not a $k$-club, a cut will be added that eliminates $\hat{x}$ without cutting off solutions corresponding to $k$-clubs. This algorithm is called the *Decomposition and Branch-and-Cut* (DBC) algorithm for the MkCP. The DBC algorithm is shown in Algorithm 1. In this chapter, the *canonical hypercube cut* (CHC), which was proposed by Balas and Jeroslow [12], is used as a cutting plane in Step 9 of Algorithm 1. This cutting plane is presented below:

$$\sum_{i=1}^{n} x_i(1 - \hat{x}_i) + (1 - x_i)\hat{x}_i \geq 1. \tag{3.4}$$

Suppose $\hat{x} \in \{0,1\}^n$ is an incidence vector of $k$-clique that is encountered and the decision vector $x \in \{0,1\}^n$ is an incidence vector of a $k$-club. Notice that all the components of $\hat{x} \in \{0,1\}^n$ and $x \in \{0,1\}^n$ cannot be the same. Therefore, the $k$-clique $\hat{x}$ encountered at the BB node is cut-off and a tighter relaxation is re-solved at the same BB node. As a consequence, the number of cuts added in DBC is exactly the number of $k$-cliques encountered in all BB nodes, which are not $k$-clubs. This interesting quantity is measured and reported in the computational results.

**Algorithm 1** Generic Decomposition and Branch-and-Cut Approach

```
1: procedure DBC($G, k$)
2:     $lb \leftarrow 0$, $t \leftarrow 0$, $x^* \leftarrow \vec{0}$, and $\mathcal{F} := \{\mathcal{P}^0\}$.
3:     while $\mathcal{F} \neq \emptyset$ do
4:         Select the problem $\mathcal{P}^t$ from $\mathcal{F}$; $\mathcal{F} := \mathcal{F} \setminus \mathcal{P}^t$
5:         Solve $\mathcal{P}^t$
6:         Let $x_t$ be the optimal solution found for $\mathcal{P}^t$ and $z^t$ the value of the objective function
7:         if $x_t \in \{0,1\}^n$ and $z^t > lb$ then
8:             if $x_t$ is not a $k$-club then
9:                 Add violated cuts to $\mathcal{P}^t$
10:                Add this node with appropriate node-ID to $\mathcal{F}$
11:            else
12:                $x^* \leftarrow x_t$ and $lb \leftarrow z^t$
13:            end if
14:        else
15:            if $x_t \notin \{0,1\}^n$ and $z^t > lb$ then
16:                Branch on some fractional component of $x_t$
17:                Generate and add child nodes with appropriate node-IDs to $\mathcal{F}$
18:            else
19:                Prune the node
20:            end if
21:        end if
22:    end while
23:    Return $x^*$ and $lb$
24: end procedure
```

It should be noted that the model decomposition algorithms developed in this dissertation when combined with graph decomposition techniques [14, 72] can enable us to solve the MkCP on very large power-low graph instances.

## 3.2 Computational results

The computational performance of the DBC algorithm described in Section 3.1 in solving the MkCP is studied in this section. The direct solution of our implementations of the IP formulations F1 and F2 from Section 2.3 serve as our comparison. The DBC, F1, and F2 were implemented in C++, and Gurobi$^{\text{TM}}$ optimizer 6.0 was used to solve the IP formulations [44]. All numerical experiments were conducted on a 64-bit Linux® computer node with dual quad-core Intel® Xeon® E5620 2.40GHz processor and 96 GB RAM.

We conduct our study on the same test-bed of 540 randomly generated instances used by Veremyev and Boginski [101], summarized in Table 3.1. The results reported are average running times in seconds measured over ten instances of each order ($n$) and edge density ($\rho$) considered. The MkCP was solved on these instances for $k = 2, 3, \ldots, 7$ using Algorithm 1 with CHC (3.4) in Step 9, and compared against the direct solution of implementations F1 and F2. In Table 3.1, the fastest average running time for every pair of $n$ and $\rho$ is highlighted in bold font. We can observe that the decomposition approach becomes more effective as $k$ gets larger; F2 is consistently the fastest when $k = 2$, but we can see the model decomposition ideas starting to pay-off computationally for the DBC algorithm when $k = 3, 4$, and 5. We also observe that implementation F2 is consistently faster than the F1 implementation it was meant to replace for all $k$, $n$, and $\rho$ considered.

Table 3.1: A comparison of average running times (secs) for $k = 2$ and 3 on the test-bed from Veremyev and Boginski [101].

| $n$ | $\rho$ | DBC | F1 | F2 | DBC | F1 | F2 | DBC | F1 | F2 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $k = 2$ | | | $k = 4$ | | | $k = 6$ | |
| | 2% | 1.63 | 0.92 | **0.87** | 3.64 | 4.89 | **3.21** | **6.04** | 11.04 | 7.80 |
| 100 | 3% | 1.73 | 0.90 | **0.76** | **9.52** | 14.78 | 9.86 | 16.79 | 25.62 | **15.74** |
| | 4% | 1.67 | 0.95 | **0.85** | **34.64** | 46.25 | 35.34 | **17.96** | 33.65 | 23.07 |
| | 1% | 7.47 | 6.01 | **4.47** | **11.02** | 18.21 | 15.24 | 93.36 | 132.85 | **89.41** |
| 200 | 1.5% | 9.17 | 7.11 | **5.15** | **43.02** | 65.44 | 53.32 | 769.22 | 912.17 | **596.52** |
| | 2% | 9.69 | 7.93 | **6.33** | **519.80** | 925.29 | 750.68 | 319.51 | 417.79 | **274.16** |
| | 0.5% | 21.82 | 17.85 | **13.55** | **30.15** | 40.52 | 33.57 | **45.29** | 71.58 | 48.34 |
| 300 | 1% | 23.78 | 20.48 | **14.44** | 130.82 | 145.36 | **121.83** | 7213.84 | 10806.20 | **6898.40** |
| | 1.5% | 28.73 | 27.17 | **18.59** | 2664.82 | 3173.93 | **2499.35** | **1075.26** | 1844.03 | 1330.49 |
| | | | $k = 3$ | | | $k = 5$ | | | $k = 7$ | |
| | 2% | 3.05 | 2.81 | **2.24** | **4.83** | 7.74 | 5.56 | **6.16** | 13.46 | 7.85 |
| 100 | 3% | 5.10 | 6.07 | **4.49** | **20.57** | 28.14 | 22.53 | **13.75** | 39.70 | 19.18 |
| | 4% | 12.34 | 14.26 | **10.82** | **18.36** | 28.85 | 21.50 | **25.28** | 56.51 | 27.91 |
| | 1% | **9.45** | 13.72 | 11.06 | **41.55** | 69.75 | 52.21 | **96.95** | 316.55 | 150.47 |
| 200 | 1.5% | **18.17** | 24.59 | 19.53 | **354.28** | 646.81 | 501.52 | **141.34** | 465.75 | 241.18 |
| | 2% | **51.78** | 70.11 | 55.44 | **589.66** | 816.57 | 633.91 | **248.58** | 811.13 | 422.30 |
| | 0.5% | 30.86 | 33.13 | **29.61** | **31.77** | 43.31 | 33.55 | **25.44** | 98.50 | 54.38 |
| 300 | 1% | **46.42** | 51.34 | 47.18 | 735.11 | 968.11 | **725.84** | **3557.13** | 14086.39 | 8181.91 |
| | 1.5% | 253.01 | 258.27 | **217.29** | 10557.38 | 13380.10 | **9781.28** | **834.63** | 3706.48 | 2181.13 |

The next group of 60 test instances are 200-vertex graphs that we generated using the randomized generation procedure introduced in [24]; 20 instances were generated for each ‘challenging edge density’ associated with each of the three values of $k$ we considered. Since determining challenging edge densities for each $k$ is a time consuming process, we limit ourselves to only three values for parameter $k$ in these experiments. The edge densities considered are 15% and 5% for the maximum 2-club and the maximum 3-club problems, respectively. These values were shown to be challenging densities in the numerical experiments of Mahdavi Pajouh and Balasundaram [66] for the respective $k$ values. For the maximum 4-club problem, edge density 2.5% produced challenging test instances based on our preliminary experiments. These instances for $k = 2$ and 3 seem to be much harder than the previous group of instances based on the average running times on 200-vertex instances in both groups. Since the generation procedures are similar, this illustrates the sensitivity of the difficulty level of the test-bed to edge density and vertex degree variance controlled by parameters in the generation procedure. Table 3.2 reports average metrics over the 20 instances generated for each value of $k$. On this test-bed we see that DBC is several times faster than solving the implementations of F1 and F2 using Gurobi. Notably, F2 can be solved in half the time taken to solve F1, or less, on this test-bed.

Table 3.2: A comparison of average metrics over 20 instances with 200 vertices for each $k$ value.

| | | DBC | | F1 | F2 |
|---|---|---|---|---|---|
| $k$ | $\bar{\omega}_k(G)$ | #CHC | Time (secs) | Time (secs) | Time (secs) |
| 2 | 53.15 | 499.95 | **144.51** | 2713.12 | 1313.31 |
| 3 | 34.55 | 6615.20 | **139.82** | 3720.06 | 1739.96 |
| 4 | 64.30 | 2389.10 | **133.06** | 1268.99 | 597.43 |

In addition to the randomly generated test instances, Table 3.3 lists the DIMACS clustering benchmark instances that were also used in our experiments; these graphs model real data and/or real-life situations as described in [42]. Observations for the DIMACS clustering benchmarks are in many ways similar to the synthetic test-bed from [101]. Tables 3.4, 3.5, and 3.6 report the $k$-club numbers, the number of canonical hypercube cuts, #CHC, and the running times for DBC, F1, and F2, for $k = 2, 3$, and 4, respectively.

For $k = 2$, we find F2 to be fastest when the instances are small enough ($|V| < 200$). But for the larger instances with $|V| > 1000$, the DBC algorithm is faster than solving the F2 implementation with speed-ups by a factor of 3 to 14 observed; with one instance ('netscience') being solved 35-times faster using the DBC algorithm. When $k = 3, 4$, the DBC algorithm is clearly the favorite for the smaller instances. At the same time, considering only the larger instances, DBC offers speed-ups anywhere in the range of 1.75–4 times faster in comparison to solving the F2 implementation, which is consistently the faster of the two implementations by a very significant margin. Two notable exceptions being the instances 'netscience' and 'power' on which the speed-ups are actually much higher than the others; when $k = 3$ these two instances exhibit speed-ups by a factor of 32 and 16 respectively when using DBC over solving F2, and when $k = 4$ the speed-ups are by a factor of 117 and 21 respectively when DBC is used over solving F2.

As guidelines for a practitioner, at the risk of oversimplification, we can state that when seeking 2-clubs, direct solution of the formulation F2 is more effective among the approaches considered here unless we are solving large-scale instances with 1000+ vertices. For such larger instances, or for larger values of the parameter $k$, the DBC algorithm proposed here is relatively a better choice.

Table 3.3: Number of vertices, edges, and edge density for DIMACS clustering benchmarks used in this study.

| Graph | $\|V\|$ | $\|E\|$ | $\rho$ |
|---|---:|---:|---:|
| 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-metabolic | 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 | 4.3e-2% |

Table 3.4: The 2-club number, the number of cuts added, and the running time in seconds for DBC, F1, and F2 on the DIMACS test-bed.

| | | DBC | | F1 | F2 |
|---|---:|---:|---:|---:|---:|
| Graph | $\bar{\omega}_2(G)$ | #CHC | Time | Time | Time |
| karate | 18 | 22 | 1.76 | 0.68 | **0.31** |
| lesmis | 40 | 71 | 5.85 | 2.49 | **1.17** |
| polbooks | 28 | 257 | 12.38 | 7.11 | **2.63** |
| adjnoun | 50 | 98 | **2.60** | 5.23 | 2.98 |
| football | 16 | 321 | 14.91 | 10.53 | **5.16** |
| celegans-metabolic | 238 | 410 | **217.51** | 334.89 | 217.68 |
| email | 72 | 410 | **605.14** | 10324.91 | 3923.47 |
| polblogs | 352 | 2646 | **6784.01** | 44675.08 | 19657.04 |
| netscience | 35 | 196 | **179.51** | 10804.17 | 6374.46 |
| power | 20 | 244 | **341.35** | 11211.45 | 4596.69 |
| hep-th | 51 | 708 | **527.22** | 14625.92 | 5411.59 |
| PGPgiantcompo | 206 | 285 | **867.55** | 19907.45 | 11745.40 |

Table 3.5: The 3-club number, the number of cuts added, and the running time in seconds for DBC, F1, and F2 on the DIMACS test-bed.

| | | DBC | | F1 | F2 |
|---|---:|---:|---:|---:|---:|
| Graph | $\bar{\omega}_3(G)$ | #CHC | Time | Time | Time |
| karate | 25 | 55 | **3.71** | 67.19 | 28.22 |
| lesmis | 59 | 146 | **15.05** | 106.32 | 48.91 |
| polbooks | 53 | 943 | **32.41** | 119.03 | 61.90 |
| adjnoun | 82 | 470 | **14.69** | 165.38 | 105.84 |
| football | 58 | 866 | **44.49** | 202.91 | 105.51 |
| celegans-metabolic | 371 | 2085 | 629.45 | 998.49 | **449.32** |
| email | 111 | 5173 | **3618.30** | 15096.08 | 6491.31 |
| polblogs | 398 | 15276 | **7348.60** | 66652.89 | 27994.21 |
| netscience | 54 | 370 | **296.87** | 17032.56 | 9708.56 |
| power | 30 | 1462 | **698.47** | 19121.44 | 11281.65 |
| hep-th | 67 | 2214 | **7992.10** | 39976.19 | 15990.48 |
| PGPgiantcompo | 302 | 1092 | **17270.12** | 45089.13 | 23897.24 |

Table 3.6: The 4-club number, the number of cuts added, and the running time in seconds for DBC, F1, and F2 on the DIMACS test-bed.

| | | DBC | | F1 | F2 |
|---|---:|---:|---:|---:|---:|
| Graph | $\bar{\omega}_4(G)$ | #CHC | Time | Time | Time |
| karate | 33 | 54 | **3.01** | 120.23 | 42.08 |
| lesmis | 75 | 128 | **7.92** | 214.59 | 120.17 |
| polbooks | 68 | 522 | **15.93** | 291.08 | 163.00 |
| adjnoun | 107 | 268 | **4.12** | 486.11 | 243.06 |
| football | 115 | 405 | **53.40** | 1367.15 | 615.22 |
| celegans-metabolic | 432 | 571 | **3725.71** | 10897.36 | 5993.55 |
| email | 468 | 2680 | **6789.00** | 26945.82 | 11856.16 |
| polblogs | 970 | 10405 | **8604.65** | 81635.49 | 31021.49 |
| netscience | 68 | 405 | **126.21** | 31675.24 | 14253.86 |
| power | 45 | 683 | **654.59** | 33069.42 | 11904.99 |
| hep-th | 184 | 1868 | **11537.96** | 58607.19 | 30475.74 |
| PGPgiantcompo | 1161 | 408 | **28774.66** | 73972.05 | 42903.79 |

# CHAPTER 4

# A DELAYED CONSTRAINT GENERATION APPROACH[1]

As discussed in Chapter 2 (Section 2.3), a compact integer programming formulation for this problem was presented by Veremyev and Boginski [101]. Solving this formulation (F1) and an improvement of this formulation (F2), published recently [102], are effective integer programming approaches presently available to solve the MkCP for any given value of $k$. In this chapter, we demonstrate that by using a delayed constraint generation approach we can significantly speed-up the performance of an IP solver. We demonstrate performance gains on the same test-bed of instances used by Veremyev and Boginski [101], as well as benchmark instances from the recent DIMACS clustering challenge. We also compare against the CHC based DBC algorithm introduced in Chapter 3.

The main contribution of this chapter is to show that adding the constraints in the formulation developed by [101] in a delayed fashion as cuts in a branch-and-cut (BC) algorithm results in a more effective decomposition algorithm that is much faster than solving a complete, direct implementation of the same. This approach further enhances our DBC approach from Chapter 3 in which we showed that the MkCP can be solved optimally for arbitrary values of $k$ on the same test-bed without using the MkCP formulations. Specifically, we began with (3.1)-(3.3) of the maximum $k$-clique problem as the master relaxation. Whenever a $k$-clique that is not a $k$-club was detected as an IP feasible solution in the BC tree, we added a naive cutting-plane that requires the IP feasible solutions encountered subsequently to differ in at

[1]Contents of this chapter have been submitted to *European Journal of Operational Research* for publication.

least one component, from the IP feasible solution detected at that node of the BC tree. While even this naive cutting-plane in a decomposition algorithm allowed us to improve over solving the direct implementation, the potential for further improvement with stronger cutting-planes was open. Addressing this gap by using the constraints developed in Veremyev and Boginski [101] is the main theme of this chapter.

In our computational studies, we employ the implementation of F2 presented in Chapter 2, solved directly using a commercial IP solver, and the DBC algorithm using CHC introduced in Chapter 3, in comparisons against two variations of a delayed constraint generation approach in a BC using the constraints of F1. We elaborate on our proposed approaches in the next section.

## 4.1 Delayed constraint generation in a branch-and-cut

Similar to the DBC algorithm in Chapter 3, the master relaxation that we commence solving with a BC algorithm is the maximum $k$-clique problem. Note that every $k$-independent set intersects every $k$-clique or $k$-club in $G$ in at most one vertex. In fact, $S \subset V$ is a $k$-clique if and only if $|S \cap I| \leq 1$ for every $k$-independent set $I$ in $G$. The following maximum $k$-clique formulation that serves as our master relaxation (henceforth referred to as MkCPR) is based on combining the aforementioned equivalent characterization of $k$-cliques via $k$-independent sets and the more elementary Definition 1 presented in Chapter 1.

$$\textbf{(MkCPR)} \quad \max \sum_{i \in V} x_i \tag{4.1}$$

*subject to:*

$$\sum_{t \in I_{ij}} x_t \leq 1, \ \forall i,j \in V \mid d_G(i,j) \geq k+1, \tag{4.2}$$

$$x_i \in \{0,1\}, \ \forall i \in V, \tag{4.3}$$

where $I_{ij}$ is a $k$-independent set containing vertices $i$ and $j$ that is also maximal by inclusion. Note that instead of (4.2), we could simply use (3.2), resulting in the "edge formulation" of the maximum $k$-clique problem used in Chapter 3. But constraint (4.2) dominates the edge constraint (3.2) and is a facet inducing inequality (when $I_{ij}$ is maximal) for the convex hull of incidence vectors of $k$-clubs in $G$.

The basic principle behind the delayed constraint generation approaches we propose in this chapter are similar to the approach introduced in Chapter 3. In the previous chapter, we solve the edge formulation for $k$-cliques using a BC algorithm, and whenever a BC node encounters IP feasibility, we verify if the detected $k$-clique is a $k$-club. If it is not, we add the canonical hypercube cut which disallows precisely this single binary vector from becoming feasible again, and every other binary vector distinct from it will satisfy this cut. Hence, the potential for improvement using stronger cuts is arguably quite significant. But the challenge is that the only family of facet-inducing inequalities presently known for arbitrary $k$ are the aforementioned maximal $k$-independent set inequalities. We have taken advantage of some of these strong inequalities in formulating our initial master relaxation. Hence, any IP feasible solution encountered during the progress of the BC will correspond to a $k$-clique, and will not violate any of the maximal $k$-independent set inequalities generated in constraints (4.2). Hence, in place of facets of the convex hull that are unknown to us, we instead use the constraints developed in [101] as we seek to cut-off integral solutions, not fractional optima of the LP relaxation.

Suppose we encounter $\hat{x} \in \{0,1\}^n$ as the LP relaxation optimum in some node of the BC tree. If $\hat{x}$ corresponds to a $k$-club, then we can allow the BC node to be pruned, and update the incumbent if $\hat{x}$ is a better solution. Suppose $\hat{x}$ does not correspond to a $k$-club, then because of the master relaxation (MkCPR) used at the root node of this BC tree, we are guaranteed that $\hat{x}$ corresponds to a $k$-clique. Now let $S \triangleq \{i \in V \mid \hat{x}_i = 1\}$ be the detected $k$-clique that is not a $k$-club. Hence,

$diam(G[S]) \geq k+1$ and the set of "violated pairs," defined as:

$$\mathcal{V} \triangleq \{(i,j) \mid i,j \in S, i<j, d_{G[S]}(i,j) \geq k+1\},$$

is nonempty. Note that however, for every $(i,j) \in \mathcal{V}, d_G(i,j) \leq k$. Now for every $(i,j) \in \mathcal{V}$ there exists a constraint (2.7) in the original Veremyev-Boginski formulation, enforcing any one of which will eliminate $\hat{x}$ from further consideration. Furthermore, the polynomial constraint corresponding to one such violated pair can also eliminate other binary solutions corresponding to $k$-cliques that are not $k$-clubs, containing the same violated pair. These observations clearly continue to hold if we enforce a group of constraints corresponding to every violated pair in $\mathcal{V}$. Of course, the polynomial constraint is not added explicitly, instead the corresponding linearized constraints are added. The two variations of the delayed constraint generation approach we propose in this chapter, referred to as *One-VP* ("one violated pair") and *All-VP* ("all violated pairs") are summarized next.

1. *One-VP*: Select the lexicographically smallest $(i,j) \in \mathcal{V}$, add the corresponding constraints (2.10)-(2.14) at that node of the BC tree.

2. *All-VP*: For every $(i,j) \in \mathcal{V}$ detected, generate and add all the corresponding constraints (2.10)-(2.14) at that node of the BC tree.

We illustrate some observations about *One-VP* and *All-VP* schemes with the example graph in Figure 4.1. Suppose that the IP feasible solution obtained as optimum to the LP relaxation at some node of the BC tree corresponds to the set $S = \{1,2,3,4,7,8\}$ in Figure 4.2 and suppose that it is a 2-clique in the graph considered in this example.

Associated with this solution are the violated pairs given by $\mathcal{V} = \{(1,7),(1,8),(7,8)\}$. This solution will be eliminated by adding the constraints for any one of the violated

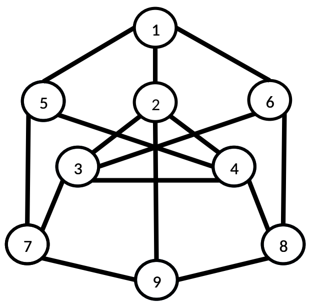

Figure 4.1: A graph of diameter two.

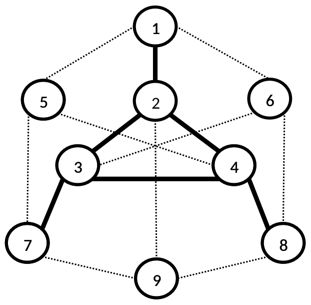

Figure 4.2: The graph induced by the 2-clique $\{1, 2, 3, 4, 7, 8\}$ (shown with bold edges) has diameter three.

pairs. But the 2-clique in Figure 4.3 will not be cut-off by adding constraints corresponding to $(1, 7) \in \mathcal{V}$, but those in Figures 4.4 and 4.5 will be. Similarly, the 2-clique in Figure 4.4 will not be eliminated by using constraints for $(1, 8) \in \mathcal{V}$, but those in Figures 4.3 and 4.5 will be. The constraints corresponding to $(7, 8) \in \mathcal{V}$ will eliminate the 2-cliques in Figures 4.3 and 4.4, but not the one in Figure 4.5. All the 2-cliques in Figures 4.3–4.5 will be eliminated in one pass of the *All-VP* scheme.

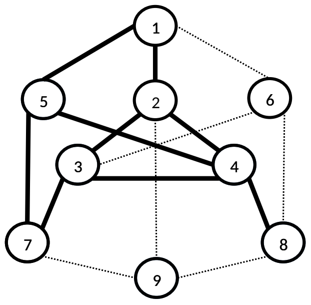

Figure 4.3: For $S = \{1, 2, 3, 4, 5, 7, 8\}$, $d_{G[S]}(1, 7) = 2$, but $d_{G[S]}(1, 8) = d_{G[S]}(7, 8) = 3$.

In the computational study presented in this chapter, either *One-VP* cuts or *All-VP* cuts are used in Step 9 of Algorithm 1, which was presented in Chapter 3, and

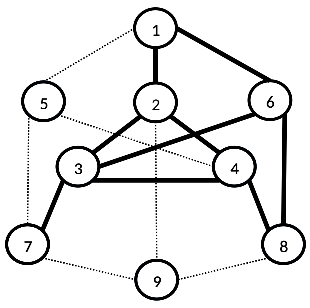

Figure 4.4: For $S = \{1, 2, 3, 4, 6, 7, 8\}$, $d_{G[S]}(1, 8) = 2$, but $d_{G[S]}(1, 7) = d_{G[S]}(7, 8) = 3$.

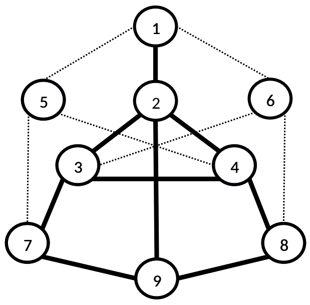

Figure 4.5: For $S = \{1, 2, 3, 4, 7, 8, 9\}$, $d_{G[S]}(7, 8) = 2$, but $d_{G[S]}(1, 7) = d_{G[S]}(1, 8) = 3$.

the master relaxation used in the root node is MkCPR (4.1)-(4.3).

Note that *One-VP* requires less effort in the sense that the strength of the cutting-planes are not considered as we add the constraints corresponding to the first violated pair detected, and no other violated pairs are used (or even detected). By contrast, *All-VP* represents the most effort to tighten the relaxation within the scope of our delayed constraint generation framework. Naturally, further strengthening is possible by the addition of problem-specific cutting-planes at the nodes of the BC tree, especially at those with fractional LP relaxation optima. However, that digresses from the purpose of this study, which is to demonstrate how much more effectively the compact formulation introduced in [101] can be used to solve the MkCP by employing delayed constraint generation. We do so in the next section via computational experiments on an test-bed of benchmark instances introduced in Chapter 3. Although problem-specific cuts are not used, the IP solver we use by default employs a variety of general-purpose cutting planes in the BC algorithm.

## 4.2 Computational study

As emphasized earlier, the objective of our computational study is to assess the performance gains (measured in terms of the running time) that can be achieved by using the delayed constraint generation frameworks *One-VP* and *All-VP* described in Section 4.1. Between the two approaches, there is a basic trade-off in the sense that the former adds fewer constraints in any one pass, resulting in a relatively smaller sized LP that is potentially easier to re-solve. The latter adds the maximum possible number of constraints in any one pass, resulting in a relatively tighter LP relaxation that potentially produces better bounds. The baseline for comparison is the computational results for directly solving the implementations of F1 and F2 described in Section 4.1. These complete formulations start with the largest possible system in comparison, at the root node of the BC tree as all the constraints are added *a priori* at the root node. The downside is the need to solve this very large LP relaxation at the root node and at all descendants subsequently. In fact, in our preliminary experiments with F1 and F2, when simplex was used as the root node solver, the size and potential degeneracies in the formulation resulted in much longer running times in comparison to using an interior-point LP solver at the root node. This is a significant computational overhead that can be avoided by the delayed constraint generation approaches that begin with the MkCPR formulation at the root node. We expect our master relaxation (4.1)-(4.3) to offer a better compromise between smaller upper-bounds and a smaller sized LP relaxation at the root node.

All approaches (*One-VP*, *All-VP*, F1, F2) were implemented in C++, and Gurobi$^{\text{TM}}$ optimizer 6.0 was used to solve the mathematical models [44]. *One-VP* and *All-VP* were implemented using the 'lazy constraints' feature of Gurobi. F1 and F2 used a barrier method as the root node LP solver, set using the Gurobi parameter 'NodeMethod'. All other Gurobi default settings were unchanged. All numerical experiments were conducted on a 64-bit Linux$^{®}$ compute node with dual quad-core

Intel® Xeon® E5620 2.40GHz processor and 96 GB RAM. The performance of all the approaches are compared for $k = 2, \ldots, 7$ on these instances. We use the same test-bed used by Veremyev and Boginski [101] and selected DIMACS clustering benchmark instances [42] in our experiments similar to Chapter 3.

Table 4.1: A comparison of average running times (secs) on the test-bed from Veremyev and Boginski [101] for $k = 2$ and 3.

| $n$ | $\rho$ | *One-VP* | *All-VP* | DBC | F1 | F2 |
|---|---|---|---|---|---|---|
| | | | | $k = 2$ | | |
| | 2% | 1.48 | 1.64 | 1.63 | 0.92 | **0.87** |
| 100 | 3% | 1.60 | 1.65 | 1.73 | 0.90 | **0.76** |
| | 4% | 1.54 | 1.80 | 1.67 | 0.95 | **0.85** |
| | 1% | 7.27 | 7.13 | 7.47 | 6.01 | **4.47** |
| 200 | 1.5% | 8.68 | 8.93 | 9.17 | 7.11 | **5.15** |
| | 2% | 9.38 | 9.32 | 9.69 | 7.93 | **6.33** |
| | 0.5% | 19.12 | 19.95 | 21.82 | 17.85 | **13.55** |
| 300 | 1% | 21.03 | 21.02 | 23.78 | 20.48 | **14.44** |
| | 1.5% | 25.28 | 25.62 | 28.73 | 27.17 | **18.59** |
| | | | | $k = 3$ | | |
| | 2% | 2.42 | 2.68 | 3.05 | 2.81 | **2.24** |
| 100 | 3% | **4.22** | 4.43 | 5.10 | 6.07 | 4.49 |
| | 4% | **9.82** | 11.22 | 12.34 | 14.26 | 10.82 |
| | 1% | **8.55** | 8.88 | 9.45 | 13.72 | 11.06 |
| 200 | 1.5% | **15.36** | 16.74 | 18.17 | 24.59 | 19.53 |
| | 2% | **45.95** | 46.96 | 51.78 | 70.11 | 55.44 |
| | 0.5% | **24.35** | 26.88 | 30.86 | 33.13 | 29.61 |
| 300 | 1% | **38.07** | 40.59 | 46.42 | 51.34 | 47.18 |
| | 1.5% | **202.13** | 219.89 | 253.01 | 258.27 | 217.29 |

Tables 4.1, 4.2, and 4.3 report the average running time for the synthetic test-bed from [101], and the fastest running times are highlighted in bold font. The first clear observation is that *One-VP* consistently performed better on this test-bed than *All-VP* indicating that the time needed to re-build and re-solve the relaxation in *All-VP* outweighs any savings gained by the tighter relaxation potentially leading to less enumeration. At the same time, *One-VP* is only slightly faster than *All-VP* until we get to the larger $n$ $(= 200, 300)$ and $k$ $(= 6, 7)$, where the savings are noticeable. Nonetheless, it is also possible that the balance may shift in favor of *All-VP* on

Table 4.2: A comparison of average running times (secs) on the test-bed from Veremyev and Boginski [101] for $k = 4$ and 5.

| $n$ | $\rho$ | *One-VP* | *All-VP* | DBC | F1 | F2 |
|---|---|---|---|---|---|---|
| | | $k = 4$ | | | | |
| 100 | 2% | **2.97** | 3.20 | 3.64 | 4.89 | 3.21 |
| | 3% | **7.71** | 7.94 | 9.52 | 14.78 | 9.86 |
| | 4% | **28.10** | 29.84 | 34.64 | 46.25 | 35.34 |
| 200 | 1% | **8.62** | 10.01 | 11.02 | 18.21 | 15.24 |
| | 1.5% | **33.24** | 38.05 | 43.02 | 65.44 | 53.32 |
| | 2% | **409.60** | 455.28 | 519.80 | 925.29 | 750.68 |
| 300 | 0.5% | **22.71** | 25.75 | 30.15 | 40.52 | 33.57 |
| | 1% | **99.46** | 108.97 | 130.82 | 145.36 | 121.83 |
| | 1.5% | **2062.62** | 2180.59 | 2664.82 | 3173.93 | 2499.35 |
| | | $k = 5$ | | | | |
| 100 | 2% | **3.71** | 3.91 | 4.83 | 7.74 | 5.56 |
| | 3% | **15.77** | 17.04 | 20.57 | 28.14 | 22.53 |
| | 4% | **13.95** | 15.01 | 18.36 | 28.85 | 21.50 |
| 200 | 1% | **31.69** | 35.68 | 41.55 | 69.75 | 52.21 |
| | 1.5% | **280.19** | 303.24 | 354.28 | 646.81 | 501.52 |
| | 2% | **469.62** | 515.73 | 589.66 | 816.57 | 633.91 |
| 300 | 0.5% | **22.68** | 22.77 | 31.77 | 43.31 | 33.55 |
| | 1% | **537.04** | 550.26 | 735.11 | 968.11 | 725.84 |
| | 1.5% | **7452.14** | 7501.50 | 10557.38 | 13380.10 | 9781.28 |

a different test-bed. Finally, addressing the main objective of this computational study, we find that *One-VP* is faster than DBC, which is faster than solving F2 for all $k \geq 3$, but slower when $k = 2$. This is acceptable since the maximum 2-club problem has a more compact common neighborhood formulation that has been the subject of other polyhedral studies [15, 28, 67]. So better IP approaches and combinatorial algorithms [46] are available for the case $k = 2$. However, for the larger values of $k$, *One-VP* was $1.06 - 4.52$ times faster than solving F2 except when $n = 100, \rho = 2\%, k = 3$ and $n = 200, \rho = 1.5\%, k = 6$ when it was slower by a factor of 1.08 and 1.01, respectively. The DBC algorithm was $1.02 - 2.61$ times faster than solving F2 for the 56% of instances. For the 44% of instances, it was $1.01 - 2.30$ times slower. It should also be noted that *All-VP* was faster than solving F1 and F2 for $k = 4, \ldots, 7$. While these observations are indicative, a performance profile

Table 4.3: A comparison of average running times (secs) on the test-bed from Veremyev and Boginski [101] for $k = 6$ and 7.

| $n$ | $\rho$ | *One-VP* | *All-VP* | DBC | F1 | F2 |
|---|---|---|---|---|---|---|
| | | $k = 6$ | | | | |
| 100 | 2% | **4.26** | 4.97 | 6.04 | 11.04 | 7.80 |
| | 3% | **11.56** | 13.53 | 16.79 | 25.62 | 15.74 |
| | 4% | **12.33** | 14.62 | 17.96 | 33.65 | 23.07 |
| 200 | 1% | **69.06** | 79.75 | 93.36 | 132.85 | 89.41 |
| | 1.5% | 604.49 | 643.21 | 769.22 | 912.17 | **596.52** |
| | 2% | **239.89** | 275.03 | 319.51 | 417.79 | 274.16 |
| 300 | 0.5% | **29.76** | 33.55 | 45.29 | 71.58 | 48.34 |
| | 1% | **4588.58** | 5427.19 | 7213.84 | 10806.20 | 6898.40 |
| | 1.5% | **679.58** | 825.12 | 1075.26 | 1844.03 | 1330.49 |
| | | $k = 7$ | | | | |
| 100 | 2% | **3.91** | 4.53 | 6.16 | 13.46 | 7.85 |
| | 3% | **8.84** | 10.23 | 13.75 | 39.70 | 19.18 |
| | 4% | **16.22** | 18.73 | 25.28 | 56.51 | 27.91 |
| 200 | 1% | **60.93** | 74.03 | 96.95 | 316.55 | 150.47 |
| | 1.5% | **84.68** | 105.49 | 141.34 | 465.75 | 241.18 |
| | 2% | **148.19** | 187.29 | 248.58 | 811.13 | 422.30 |
| 300 | 0.5% | **16.20** | 17.82 | 25.44 | 98.50 | 54.38 |
| | 1% | **2093.17** | 2434.86 | 3557.13 | 14086.39 | 8181.91 |
| | 1.5% | **482.86** | 562.84 | 834.63 | 3706.48 | 2181.13 |

across the entire test-bed for all $k$ values can better summarize the performance of each approach.

A performance profile is defined as the empirical cumulative distribution function $f(\tau)$ of an appropriate performance ratio $\tau$ [34]. The profiles offer an effective way to summarize, visualize, and compare the performance of different solvers. In our case, the performance metric is the running time of the different approaches, namely, *One-VP*, *All-VP*, DBC, F1, and F2, and the performance ratio is the ratio of the running time of each solver on a given instance to the minimum running time among all solvers for that instance. Figure 4.6 plots the performance profiles across all instances, for all values of $k$ considered in our study to present a broad picture of their effectiveness as a general purpose MkCP solver.

Tables 4.4, 4.5, and 4.6 present the results for the DIMACS clustering benchmark

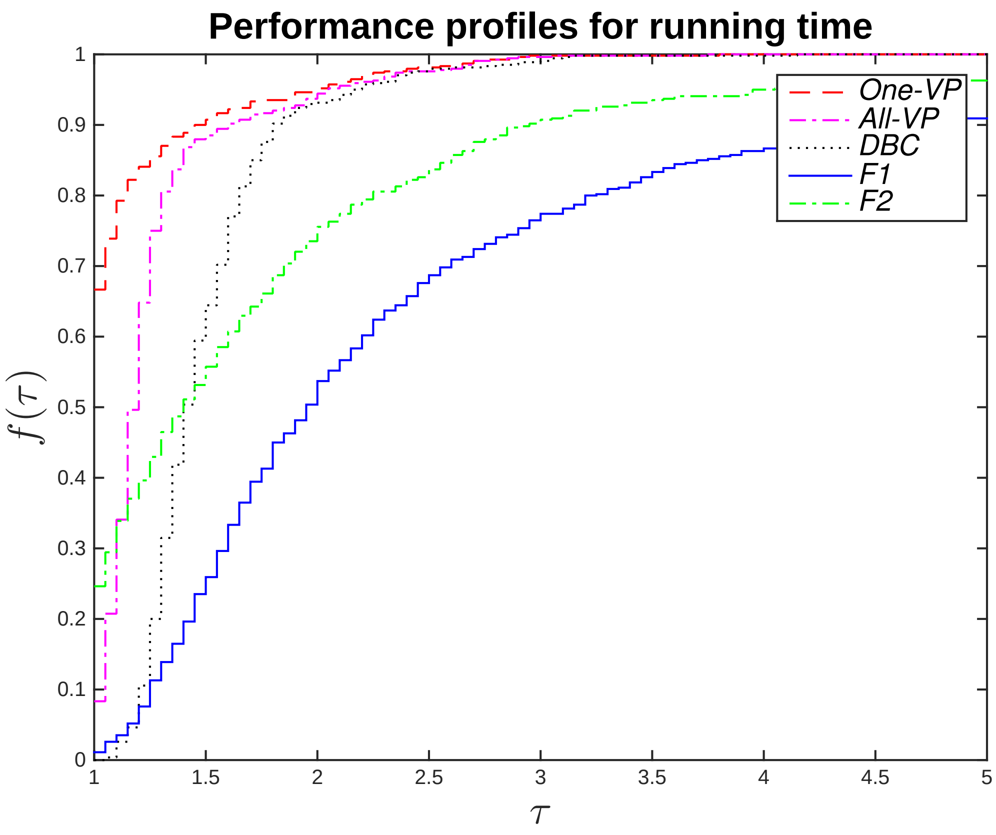

Figure 4.6: Performance profiles comparing running time ratios of *One-VP*, *All-VP*, DBC, F1, and F2. For a given $\tau$, the larger $f(\tau)$ implies that a larger fraction of the instances were solved under a factor-$\tau$ of the instance-wise minimum running time.

instances for $k = 2, 3$, and 4, respectively. For these instances we only report the running times of *One-VP*, DBC and F2, and the fastest of the three is highlighted in bold font. We excluded *All-VP* and F1 running times as they were consistently slower than their counterparts, similar to what we observed with the test-bed of [101]. These tables again demonstrate that *One-VP* was the fastest approach for $k = 3, 4$ and it was the fastest for $k = 2$ on the DIMACS instances with $n > 400$. Although not reported, we observed that again *All-VP* was consistently faster than DBC, which is faster than solving F1 and F2 when $k = 3, 4$ and for $k = 2$ when the instances have 1000+ vertices. It is also important to note that on the instances with 1000+ vertices, *One-VP* is at least 3 times faster when $k = 2$, reducing the running time by a factor of 37 for one instance, at least twice as fast as solving F2 for $k = 3, 4$

with some instances showing speed-ups by a massive factor, e.g., 'netscience' is solved 44-times faster with *One-VP* when $k = 3$, and 150-times faster when $k = 4$. The DBC algorithm is at least 1.38 times faster than F2 for the instances with 1000+ vertices. When $k = 4$, DBC is solved 113 times faster than F2 for 'netscience'.

Table 4.4: The 2-club number and running time (secs) for *One-VP*, DBC, and F2 for the DIMACS clustering instances.

| Graph | $\bar{\omega}_2(G)$ | *One-VP* | DBC | F2 |
|---|---:|---:|---:|---:|
| karate | 18 | 1.61 | 1.76 | **0.31** |
| lesmis | 40 | 5.26 | 5.85 | **1.17** |
| polbooks | 28 | 11.02 | 12.34 | **2.63** |
| adjnoun | 50 | **2.26** | 2.60 | 2.98 |
| football | 16 | 14.01 | 14.91 | **5.16** |
| celegans-metabolic | 238 | **204.46** | 217.51 | 217.68 |
| email | 72 | **574.88** | 605.14 | 3923.47 |
| polblogs | 352 | **6444.81** | 6784.01 | 19657.04 |
| netscience | 35 | **172.33** | 179.51 | 6374.46 |
| power | 20 | **320.87** | 341.35 | 4596.69 |
| hep-th | 51 | **485.04** | 527.22 | 5411.59 |
| PGPgiantcompo | 206 | **772.12** | 867.55 | 11745.40 |

Table 4.5: The 3-club number and running time (secs) for *One-VP*, DBC, and F2 for the DIMACS clustering instances.

| Graph | $\bar{\omega}_3(G)$ | *One-VP* | DBC | F2 |
|---|---:|---:|---:|---:|
| karate | 25 | **2.71** | 3.71 | 28.22 |
| lesmis | 59 | **11.59** | 15.05 | 48.91 |
| polbooks | 53 | **26.25** | 32.41 | 61.90 |
| adjnoun | 82 | **10.72** | 14.69 | 105.84 |
| football | 58 | **31.59** | 44.49 | 105.51 |
| celegans-metabolic | 371 | 478.38 | 629.45 | **449.32** |
| email | 111 | **3147.92** | 3618.30 | 6491.31 |
| polblogs | 398 | **5217.51** | 7348.60 | 27994.21 |
| netscience | 54 | **219.68** | 296.87 | 9708.56 |
| power | 30 | **544.81** | 698.47 | 11281.65 |
| hep-th | 67 | **6153.92** | 7992.10 | 15990.48 |
| PGPgiantcompo | 302 | **12434.49** | 17270.12 | 23897.24 |

Based on a comprehensive computational study, this chapter has demonstrated that a decomposition algorithm (*One-VP*) that uses the constraints from a well-known compact formulation (F1) is a very effective approach for solving the MkCP

Table 4.6: The 4-club number and running time (secs) for *One-VP*, DBC, and F2 for the DIMACS clustering instances.

| Graph | $\bar{\omega}_4(G)$ | *One-VP* | DBC | F2 |
|---|---:|---:|---:|---:|
| karate | 33 | **1.84** | 3.01 | 42.08 |
| lesmis | 75 | **4.75** | 7.92 | 120.17 |
| polbooks | 68 | **12.26** | 15.93 | 163.00 |
| adjnoun | 107 | **2.80** | 4.12 | 243.06 |
| football | 115 | **34.71** | 53.40 | 615.22 |
| celegans-metabolic | 432 | **2570.74** | 3752.71 | 5993.55 |
| email | 468 | **4820.19** | 6789.00 | 11856.16 |
| polblogs | 970 | **5851.16** | 8604.65 | 31021.49 |
| netscience | 68 | **94.65** | 126.21 | 14253.86 |
| power | 45 | **484.40** | 654.59 | 11904.99 |
| hep-th | 184 | **8653.47** | 11537.96 | 30475.74 |
| PGPgiantcompo | 1161 | **20142.26** | 28774.66 | 42903.79 |

for arbitrary large values of $k$, especially for $k = 2 \ldots 6$, that are likely more interesting in real-life networks. In the next chapter, we proceed to study the special case, $k = 2$, but in graphs that are subject to probabilistic edge failures. We focus on the 2-clubs since this is the first study of this kind on the MkCP in a stochastic setting.

# CHAPTER 5

# RISK-AVERSE 2-CLUBS IN RANDOM GRAPHS[1]

In this chapter, we change our focus to the special case of $k = 2$ for which F1, F2, and the path-based formulation (2.1)-(2.5) coincide, but in a stochastic setting. In graph models where the edge relationships are uncertain, cluster detection approaches discussed in this chapter will be useful. Specifically, we are interested in detecting large 2-clubs in graphs subject to probabilistic edge failures, in such a way that the risk of losing the desired 2-club property in detected clusters due to edge failures is limited. To quantify risk, we employ the notion of CVaR introduced in Section 2.6.

To model "risk-averse" solutions to our problem, we first define the loss in 2-club property due to random edge failures as a function of the decision vector $x$ (chosen 2-club) and random vector $Y$ (indicating edge failures). Then, to limit the risk of heavy losses, we bound the CVaR of the loss function for a given decision in the optimization model. The formal description of our problem is presented next.

## 5.1 Problem description

Consider a random graph $\tilde{G} = (V, \tilde{E})$ in which the vertex set $V$ is deterministic and the edge set $\tilde{E}$ is random. Assume that $V = \{1, \dots, n\}$ and each edge $(i, j) \in \tilde{E}$ exists independently of the others with probability $p_{ij}$. Let $G = (V, E)$ denote the support graph, where $(i, j) \in E$ if and only if $p_{ij} > 0$, and let $|E| = m$. Let $A = [a_{ij}]_{n \times n}$ denote the symmetric, zero-diagonal, adjacency matrix of the support graph $G$ in which $a_{ij} = a_{ji} = 1$ if and only if $(i, j) \in E$. We use a random vector $Y$ with image

[1]Results of this chapter have been submitted to *Annals of Operations Research* for publication.

in $\{0,1\}^m$ to indicate the existence of an edge, and hence, $Pr\{Y_{ij} = 1\} = p_{ij} > 0$ for all $(i,j) \in E$.

In our model, the decision vector $x$ must be an incidence vector of a 2-club in the support graph $G$. The set $P$ containing incidence vectors of all 2-clubs in graph $G$ can be formulated as,

$$P \triangleq \Big\{x \in \{0,1\}^n : x_i + x_j - \sum_{k \in V} a_{ik} a_{jk} x_k \leq 1 + a_{ij},\ \forall i, j \in V \mid i < j\Big\}. \tag{5.1}$$

Given a decision $x^0 \in P$ corresponding to a 2-club $S^0$ in $G$ and a realization $y^0$ of the random vector $Y$, our loss function $L(x^0, y^0)$ counts the total number of non-adjacent pairs of vertices in $S^0$ without a common neighbor in $S^0$. Note that $S^0$ is a 2-club in the support graph $G$, but it may not necessarily form a 2-club under the realization $y^0$. Given a real number $\beta$, let $[\beta]^+ = \max\{0, \beta\}$. Then, for a 2-club $x^0 \in P$, the random loss function $L(x^0, Y)$ is defined as,

$$L(x^0, Y) \triangleq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [\phi(x^0, Y, i, j)]^+, \tag{5.2}$$
$$\text{where } \phi(x^0, Y, i, j) \triangleq x_i^0 + x_j^0 - 1 - a_{ij} Y_{ij} - \sum_{t \in V} a_{it} a_{jt} Y_{it} Y_{jt} x_t^0.$$

Therefore, given a graph subject to probabilistic edge failures, we are interested in finding a largest cardinality 2-club in the support graph such that the CVaR of the loss function defined above does not exceed a user-defined threshold $d$. As stated next, this is a CVaR-constrained, single-stage stochastic program in decision vector $x \in P$; and henceforth, is referred to as the *CVaR-constrained maximum 2-club problem.*

$$\max_{x \in P} \Big\{\mathbf{1}^T x \mid \alpha\text{-CVaR}[L(x, Y)] \leq d\Big\} \tag{5.3}$$

Figure 5.1 illustrates the CVaR concept, where $L(x^0, Y)$ is given by (5.2). Assume that $G = (V, E)$ is a star with $|V| = 20$, and $x^0$ is the incidence vector of $V$. We further assume that $p_{ij} = 0.75$ for all $(i, j) \in E$ and $\alpha = 0.90$. For these parameter values, $\alpha$-VaR$[L(x^0, Y)] = 112$ and $\alpha$-CVaR$[L(x^0, Y)] = 119.77$.

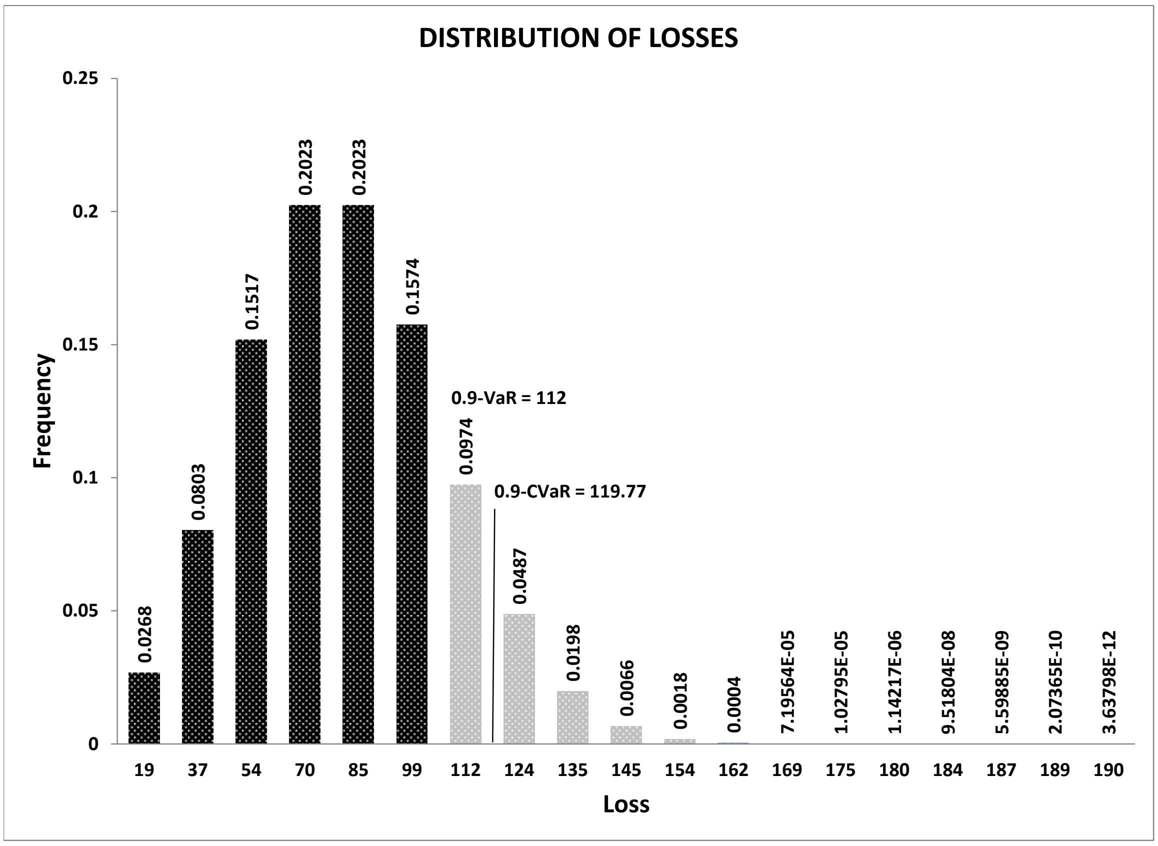

Figure 5.1: Illustration of the CVaR concept using loss function (5.2) on a star support graph of order 20 in which all vertices are selected as the 2-club solution. The probability of existence of each edge is 0.75 and $\alpha = 0.90$.

The remainder of this chapter is organized as follows. In Section 5.2, we provide a brief background on the optimization of CVaR and summarize literature closely related to solving the CVaR-constrained maximum 2-club problem (5.3). The details of the proposed algorithm for solving (5.3), based on an existing decomposition approach for solving CVaR-constrained mixed integer programs is presented in Section 5.3. Section 5.4 contains the numerical results we obtained by solving (5.3) using Algorithm 2 on a test-bed of randomly generated graphs, and real-life biological and social networks. We study the computational performance of Algorithm 2 in terms of solution quality, running time and number of iterations, averaged over multiple replications.

## 5.2 Review of literature on CVaR optimization

The literature on handling CVaR in optimization models can be divided into two groups based on how CVaR is incorporated into the optimization model. One group consists of models that find a feasible solution that minimizes CVaR; and the other group optimizes a different objective (e.g., total cost) while ensuring that CVaR of a suitable loss function is bounded from above by a user-specified parameter. The models could also be classified as single-stage decision models and two-stage decision models that accommodate recourse actions. The algorithmic approaches, specifically decomposition techniques, while related in principle, are not necessarily interchangeable. Of course, a model could also employ multiple loss functions and incorporate multiple CVaR constraints.

Decomposition techniques form the vast majority of exact approaches for CVaR optimization due to the model structure. However, non-differentiable convex optimization based approaches are also suitable whenever the loss function is convex in $x$. For instance, Lim et al. [61] developed a convergent algorithm based on variable target value method for non-differentiable optimization, for CVaR minimization applied to portfolio optimization problems. The overall algorithm includes a perturbation technique as a pre-processing step, and for linear loss functions, an optional post-processing step using the simplex method to facilitate finite convergence. We next highlight several existing decomposition approaches for CVaR optimization with linear loss functions in continuous variables.

Künzi-Bay and Mayer [59] studied solving single-stage CVaR minimization by reformulating it as a two-stage stochastic programming problem and specializing the L-shaped method [99] for this form, leading to a new algorithm for minimizing CVaR. A central result in their work was the polyhedral reformulation of the

CVaR constraint (2.31) that facilitated their decomposition approach. This polyhedral representation was in turn based on the work of Haneveld and van der Vlerk [45] dealing with expectation constrained stochastic optimization. Schultz and Tiedemann [89] investigated the use of CVaR to quantify the uncertain second stage cost in two-stage stochastic optimization problems with integer recourse. They studied its convexity properties, developed a Lagrangian decomposition strategy and solved it using a subgradient-based method. Their modeling approach is also related to the investigation of mean-risk stochastic programs by Ahmed [2].

Several authors (Fábián [37], Subramanian and Huang [94], and Huang and Subramanian [50]) have developed decomposition techniques and other iterative schemes based on the result of [59], as well as extended their approach to solve single-stage and two-stage stochastic optimization problems involving CVaR constraints or objective, typically with *linear* loss functions and *continuous* variables. Very recently Ma et al. [64] developed a polyhedral reformulation for piecewise linear losses and used the linear constraints in a decomposition and branch-and-cut approach for a stochastic network design problem. The decomposition ideas we explore in Section 5.3 were introduced in [64], designed to handle our convex piecewise linear loss function without additional linearization variables. We then combine the reformulation ideas from [64] with the cutting plane method from [94] leading to our algorithm for the CVaR-constrained maximum 2-club problem.

## 5.3 Row-generation-based decomposition algorithm

Künzi-Bay and Mayer [59] developed a polyhedral reformulation approach and cutting plane method for linear loss functions and continuous variables. Subramanian and Huang [94] used their approach to solve single-stage CVaR-constrained mixed integer linear programming problems with linear loss functions. Ma et al. [64] extended the results of [59] to convex piecewise linear losses and showed how the linear reformulation constraints can be used as cutting planes decomposition branch-and-cut. This section presents our row-generation approach based on [59, 94, 64] for solving the CVaR-constrained maximum 2-club problem.

We use the following equivalent representation introduced by Künzi-Bay and Mayer [59] for the generic CVaR constraint (2.31):

$$(1-\alpha)\zeta + \max_{\phi\in\Phi}\Big\{\sum_{l\in\phi}\pi_l[L(x,y^l)-\zeta]\Big\} \leq d(1-\alpha), \tag{5.4}$$

where $\Phi$ is the power set of $\{1,\ldots,N\}$ and the sum is taken to be zero when $\phi$ is empty. Constraint (5.4) is equivalent to the following (exponentially many) constraints:

$$(1-\alpha)\zeta + \sum_{l\in\phi}\pi_l[L(x,y^l)-\zeta] \leq d(1-\alpha),\ \forall\phi\in\Phi. \tag{5.5}$$

Therefore, problem (5.3) can be reformulated as follows:

$$\max_{\zeta\in\mathbb{R},x\in P} \quad \mathbf{1}^T x \tag{5.6}$$

*subject to:*

$$(1-\alpha)\zeta + \sum_{l\in\phi}\pi_l[L(x,y^l)-\zeta] \ \leq\ d(1-\alpha),\ \forall\phi\in\Phi. \tag{5.7}$$

Formulation (5.6)-(5.7) is a large-scale mixed integer program with exponentially many convex, piecewise linear constraints. Note that the loss function in our problem is piecewise linear and convex. To solve this problem, we devise Algorithm 2, which adopts and extends the approach proposed by Subramanian and Huang [94] in conjunction with the approach introduced in Ma et al. [64] for replacing the piecewise linear constraints with linear constraints.

For a given realization of $Y$, say $y^0$, $L(x,y^0)$ is a piecewise linear convex function of $x$. Now, for a given 2-club $x^0$, let $h_{(x^0,y^0)}(x)$ denote the linear function describing a piece of $L(x, y^0)$ in a region near $x^0$ described as follows:

$$h_{(x^0,y^0)}(x) = \sum_{(i,j)\in S(x^0,y^0)} \phi(x, y^0, i, j), \tag{5.8}$$

where

$$S(x^0, y^0) = \Big\{(i, j) : i, j \in V, i < j, \text{ and } \phi(x^0, y^0, i, j) > 0\Big\}. \tag{5.9}$$

Therefore, in iteration $t$ of Algorithm 2, an inequality of type (5.7) violated by the optimum solution $x^*(t)$ that is detected by solving the constructed problem in Step 1 of Algorithm 2, is replaced by the following linear constraint:

$$(1-\alpha)\zeta + \sum_{l\in\phi} \pi_l[h_{(x^*(t),y^l)}(x) - \zeta] \leq d(1-\alpha). \tag{5.10}$$

The constraints of type (5.10) are preferred since the problem remains a linear integer program throughout the iterations of the algorithm and avoids the use of additional variables to linearize the piecewise linear constraint (5.7).

Algorithm 2 is a two-phase procedure. In Phase 1, a continuous relaxation of formulation (5.6)-(5.7) is obtained by removing constraints (5.7) and relaxing the binary restriction on $x$. This relaxation is solved using a sequential cutting plane method that adds cutting planes of type (5.10). The goal of Phase 1 is to find cutting planes of type (5.10) such that the continuous relaxation optimum in the presence of these cutting planes does not violate any of the constraints of type (5.7). This effectively solves the continuous relaxation of formulation (5.6)-(5.7) to optimality. This is a warm-up procedure to initialize the problem for Phase 2, which in each iteration solves a mixed-integer linear program using a sequential cutting plane method that adds cutting planes of type (5.10) until the integer optimum does not violate any of the constraints of type (5.7). This solves the original CVaR-constrained problem to optimality whenever an optimal solution exists.

In both phases, given the optimal solution to the relaxation under consideration, we scan the scenarios to find a $\phi \subseteq \{1,\ldots,N\}$ that corresponds to the most violated constraint of type (5.7), if one exists. This is in turn used to construct a cutting plane of type (5.10). If no violated constraint of type (5.7) exists in Phase 1 the algorithm proceeds to Phase 2, and if no violated constraint of type (5.7) exists in Phase 2 the algorithm terminates optimally. If the relaxation becomes infeasible during an iteration of Phase 1 or Phase 2, then the problem (5.6)-(5.7) is infeasible and Algorithm 2 terminates by infeasibility.

**Algorithm 2** A sequential cutting plane method for solving the CVaR-constrained maximum 2-club problem

```
Procedure SCPM($G, (\pi_1,\ldots,\pi_N), (y_1,\ldots,y_N), d, \alpha$)
 1: Phase 1. Construct RP1 by removing all constraints of type (5.7) from formulation (5.6)-(5.7) and relaxing $x$ to be continuous; $t \leftarrow 1$
 2: Solve RP1; if RP1 is infeasible, then return infeasibility of formulation (5.6)-(5.7); otherwise, suppose $x^*(t), \zeta^*(t)$ is the optimal solution found
 3: $\phi(t) \leftarrow \{l \in \{1,\ldots,N\} \mid L(x^*(t), y^l) - \zeta^*(t) > 0\}$
 4: If the constraint of type (5.7) associated with $\phi(t)$ is violated by $x^*(t), \zeta^*(t)$, then go to Step 5; otherwise, go to Step 7
 5: Update RP1 by adding the cutting plane $(1-\alpha)\zeta + \sum_{l\in\phi(t)} \pi_l[h_{(x^*(t),y^l)}(x) - \zeta] \leq d(1-\alpha)$
 6: $t \leftarrow t+1$ and go to Step 2
 7: Phase 2. Initialize RP2 with RP1 and restrict $x$ to be binary; $t \leftarrow t+1$
 8: Solve RP2; if RP2 is infeasible, then return infeasibility of formulation (5.6)-(5.7); otherwise, suppose $x^*(t), \zeta^*(t)$ is the optimal solution found
 9: $\phi(t) \leftarrow \{l \in \{1,\ldots,N\} \mid L(x^*(t), y^l) - \zeta^*(t) > 0\}$
10: If the constraint of type (5.7) associated with $\phi(t)$ is violated by $x^*(t), \zeta^*(t)$, then go to Step 11; otherwise, return by $x^*(t), \zeta^*(t)$ as the optimal solution to formulation (5.6)-(5.7)
11: Update RP2 by adding the cutting plane $(1-\alpha)\zeta + \sum_{l\in\phi(t)} \pi_l[h_{(x^*(t),y^l)}(x) - \zeta] \leq d(1-\alpha)$
12: $t \leftarrow t+1$ and go to Step 8
End-Procedure
```

## 5.4 Numerical experiments

In this section, we study the computational performance of Algorithm 2 and we assess the performance of the algorithm based on the solution quality, number of iterations, and total running time.

Algorithm 2 was implemented in C++ and all numerical experiments were conducted on a 64-bit Linux compute node with dual quad-core Intel Xeon E5620 2.40GHz processors and 96GB RAM. IBM ILOG CPLEX Optimizer 12.6$^{\circledR}$ was used to solve the relaxation in single-threaded mode.

The test-bed consists of randomly generated instances along with selected real-life biological and social networks. Randomly generated instances include ten 50-vertex graphs and ten 100-vertex graphs generated by the algorithm described in [24]. The edge density of the graphs produced by this algorithm is controlled by two parameters $a$ and $b$. The expected edge density is $(a + b)/2$ and vertex degree variance increases as a function of $b - a$. We considered density 10% with $a = b = 10\%$, which is known to be a challenging density for the deterministic maximum 2-club problem as demonstrated in [66]. If the degree variance is high, the maximum degree vertex and its neighbors are more likely to form a maximum cardinality 2-club in the support graph, a situation we try to avoid to the extent possible in our randomly generated test-bed by setting $a = b$.

For the real-life networks, we use protein interaction networks of two organisms, and a coauthor network of mathematicians. These three networks are large, but very sparse networks that exhibit a power-law degree distribution [31]. The protein interaction data for organism *S. cerevisiae*, commonly known as baker's yeast [53], which has 2114 proteins and 2203 known interactions was obtained from [29]. The other protein interaction network is for a gastric bacterium called *H. pylori* that has 1570 proteins and 1399 interactions [82, 57]. More information on the Erdös collaboration network ($|V| = 511, |E| = 1604$) can be found in [43].

The failure probability for each edge in each test instance was generated randomly from a uniform distribution between 0 and 1. The value of parameter $\alpha = 0.95$ in all our experiments, and the right-hand side of the CVaR constraint (parameter $d$) is set based on preliminary experiments to find a meaningful value for parameter $d$. If $d$ is too large, the CVaR constraint is redundant and if it is too small, the problem becomes infeasible. For each test-instance, 1000 scenarios are sampled based on the edge probabilities; five replications were carried out based on five different sets of samples, and average performance is reported for each performance metric. The running time limit for each algorithm is 10800 seconds. If the algorithm terminates reaching the time-limit, an upper-bound on the optimal solution is reported.

Table 5.1 presents results of our experiments with the 50-vertex test instances for Algorithm 2. This table shows the performance of the algorithm based on the average number of iterations and average running time. The computational results obtained by solving the CVaR-constrained maximum 2-club problem on randomly generated 100-vertex instances are reported in Table 5.2. For these instances, our algorithm terminated reaching the time limit. Since, our algorithm is sequential cutting plane method solving a relaxation in each iteration, feasibility cannot be guaranteed under this type of termination. Nonetheless, a valid upper bound on the optimal objective value is available from the algorithm. Table 5.3 reports the results of our experiments with the social and biological networks.

The preliminary computational study conducted in this chapter is indicative of the possibilities, as well as the challenges in solving this problem in a stochastic setting. We reach the limits of our ability to solve the problem to optimality, rather quickly. A decomposition branch-and-cut instead of a sequential cutting plane method can help improve this situation.

Table 5.1: Average solution size (Sol. Size), number of iterations and running time (in seconds) over the five replications using Algorithm 2 on ten randomly generated 50-vertex instances are reported. In these experiments, $d = 10$, $N = 1000$, and $\alpha = 0.95$. The size of a maximum 2-club in the support graph of each instance (Det. Sol. Size) is also reported.

| Instance # | Det. Sol. Size | Sol. Size | Iterations # | Running Time (Sec.) |
|---|---|---|---|---|
| 1 | 11 | 7 | 294.4 | 409.7 |
| 2 | 12 | 5.2 | 2359.2 | 8567.6 |
| 3 | 12 | 5.8 | 2556.3 | 10133.6 |
| 4 | 12 | 7 | 723.4 | 1413.1 |
| 5 | 11 | 6 | 800.4 | 1641.1 |
| 6 | 11 | 6 | 741.2 | 1202.1 |
| 7 | 11 | 6 | 789.2 | 1753.6 |
| 8 | 10 | 6 | 408.8 | 526.9 |
| 9 | 15 | 6 | 1426.2 | 5364.7 |
| 10 | 15 | 7 | 1932.6 | 10227.0 |

Table 5.2: Average number of iterations and upper-bound over the five replications using Algorithm 2 on ten randomly generated 100-vertex instances are reported. In these experiments, $d = 20$, $N = 1000$, and $\alpha = 0.95$. The size of a maximum 2-club in the support graph of each instance (Det. Sol. Size) is also reported.

| Instance # | Det. Sol. Size | Iterations# | Upper-Bound |
|---|---|---|---|
| 1 | 21 | 481.4 | 16 |
| 2 | 22 | 401.2 | 15 |
| 3 | 23 | 359.7 | 15 |
| 4 | 22 | 318.0 | 16 |
| 5 | 22 | 453.2 | 15 |
| 6 | 19 | 600.4 | 14 |
| 7 | 20 | 565.4 | 13 |
| 8 | 20 | 680.0 | 13 |
| 9 | 22 | 444.2 | 13 |
| 10 | 20 | 425.2 | 15 |

Table 5.3: Solution size (Sol. Size), number of iterations and running time (in seconds) in each of the five replications using Algorithm 2 on real-life biological and social networks. In these experiments, $N = 1000$, and $\alpha = 0.95$. The size of a maximum 2-club in the support graph of each instance (Det. Sol. Size) is reported along with values for parameter $d$.

| Graph | Det. Sol. Size | Sol. Size | Iterations # | Running Time (Sec.) |
|---|---|---|---|---|
| H. pylori ($d$=1300) | 56 | 54 | 47 | 2922.5 |
| | | 55 | 40 | 2913.5 |
| | | 55 | 41 | 2904.0 |
| | | 54 | 43 | 3023.0 |
| | | 55 | 34 | 2816.7 |
| Yeast ($d$=1350) | 57 | 56 | 59 | 7433.7 |
| | | 55 | 63 | 8632.7 |
| | | 56 | 57 | 7668.4 |
| | | 56 | 55 | 6924.3 |
| | | 56 | 58 | 7012.8 |
| Erdös ($d$=700) | 52 | 51 | 41 | 512.7 |
| | | 51 | 44 | 549.2 |
| | | 51 | 38 | 418.5 |
| | | 50 | 76 | 806.8 |
| | | 51 | 52 | 573.0 |

# CHAPTER 6

# CONCLUSIONS AND FUTURE WORK

Detecting low-diameter clusters is an important graph-based data mining technique used in social network analysis, bioinformatics, text-mining, and internet analytics. This chapter summarizes our contributions in developing exact algorithms for detecting $k$-clubs in graphs. At the end, some directions for future work are identified in this chapter.

## 6.1 Contributions

New approaches for exactly solving the MkCP through model decomposition and cutting planes is developed in this dissertation. We demonstrated that by using a delayed cutting plane/constraint generation approach in a branch-and-cut algorithm, we can solve the MkCP optimally for arbitrary values of parameter $k$. We showed in Chapter 3 that complicated formulations of the MkCP can be disregarded in favor of a simple relaxation based on necessary conditions, combined with canonical hypercube cuts introduced by Balas and Jeroslow. The canonical hypercube cut is used as a "lazy" cutting plane in the proposed decomposition algorithm enabling us to show that this approach is effective even when a naive valid inequality is used. As a consequence of using the canonical hypercube cut, we can see how often $k$-cliques that are not $k$-clubs are encountered by the decomposition and branch-and-bound algorithm.

Using the constraints developed in [101] in two different ways (*One-VP* and *All-VP*), this study showed that using delayed constraint generation in a branch-and-cut outperformed directly solving F1 and F2 formulations. Although, using *All-VP* cuts eliminates more infeasible integral solutions at each iteration of the proposed approach, they are not favorable compared to using *One-VP* cuts in terms of overall running time.

In large-scale networks that are error-prone, the uncertainty associated with the existence of an edge between two vertices can be modeled by assigning a failure probability to that edge. In Chapter 5, we study the problem of detecting large risk-averse 2-clubs in graphs subject to probabilistic edge failures. To achieve risk aversion, we first model the loss in 2-club property due to probabilistic edge failures as a function of the decision (chosen 2-club cluster) and randomness (graph structure). Then, we utilize the CVaR of the loss for a given decision as a quantitative measure of risk for that decision, which is bounded in the model. We adapt and apply ideas introduced in the recent literature [59, 94, 64] to design a sequential cutting plane method for the CVaR constrained maximum 2-club problem.

## 6.2 Future work

Studying the $k$-club polytope to discover new families of facet inducing inequalities for the general parameter $k$ would be an interesting direction for future work. Using the discovered families of facet inducing inequalities in the proposed decomposition and branch-and-cut algorithms to solve the MkCP would likely be computationally beneficial, which is another interesting research direction. In particular, integrating the decomposition ideas for general $k$-clubs and CVaR 2-clubs developed in this dissertation for the general CVaR-constrained maximum $k$-club problem for arbitrary $k$ is an interesting direction for immediate future research.

A generalization of $k$-clubs was introduced by Veremyev and Boginski [101] called $r$-robust $k$-clubs. An $r$-robust $k$-club is a subset of vertices in which there are at least $r$ node-disjoint paths of length at most $k$ between every pair of vertices. The motivation for $r$-robust $k$-clubs rose from the fact that $k$-clubs may be vulnerable to failure of the edges/nodes of the induced subgraph. For example, a star is a 2-club in which failure of the hub node will destroy the 2-club property of the subgraph.

The maximum $r$-robust $k$-club problem is to find an $r$-robust $k$-club with maximum cardinality in $G$. There is a lack of compact mathematical formulations analogous to F1 and F2 for this problem in the literature. Veremyev and Boginski [101] presented a formulation that guarantees the distinctness of the $r$ paths, but cannot ensure the node-disjointedness requirement of $r$-robust $k$-clubs.

Developing compact formulations for the maximum $r$-robust $k$-club problem is a good future research direction. Using decomposition techniques proposed in this dissertation for developing a decomposition and branch-and-cut algorithm to solve this problem for general values of parameters $r$ and $k$ would be another interesting research direction. It would also be interesting to conduct polyhedral studies to discover strong valid inequalities for this problem.

## BIBLIOGRAPHY

[1] J. Abello, P. M. Pardalos, and M. G. C. Resende. On maximum clique problems in very large graphs, 1999.

[2] S. Ahmed. Convexity and decomposition of mean-risk stochastic programs. *Mathematical Programming*, 106:433–446, 2006.

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

[4] M. T. Almeida and F. D. Carvalho. Integer models and upper bounds for the 3-club problem. Technical report, CIO-Centro de Investigação Operacional, 2008.

[5] M. T. Almeida and F. D. Carvalho. The $k$-club problem: New results for $k = 3$. Technical report, CIO-Centro de Investigação Operacional, 2008.

[6] M. T. Almeida and F. D. Carvalho. Integer models and upper bounds for the 3-club problem. *Networks*, 60:155–166, 2012.

[7] M. T. Almeida and F. D. Carvalho. An analytical comparison of the LP relaxations of integer models for the $k$-club problem. *European Journal of Operational Research*, 232:489–498, 2014.

[8] F. Andersson, H. Mausser, D. Rosen, and S. Uryasev. Credit risk optimization with conditional value-at-risk criterion. *Mathematical Programming*, 89:273–291, 2001.

[9] P. Artzner, F. Delbaen, J. M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9:203–228, 1999.

[10] Y. Asahiro, E. Miyano, and K. Samizo. Approximating maximum diameter-bounded subgraphs. In A. López-Ortiz, editor, *Lecture Notes in Computer Science*, pages 615–626. Springer New York, 2010.

[11] J. G. Augustson and J. Minker. An analysis of some graph theoretical cluster techniques. *Journal of the ACM*, 317:571–588, 1970.

[12] E. Balas and R. Jeroslow. Canonical ccuts on the unit hypercube. *SIAM J. Appl. Math.*, 23:61–69, 1972.

[13] E. Balas and J. Xue. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica*, 15:397–412, 1996.

[14] B. Balasundaram. *Graph Theoretic Generalizations Of Clique: Optimization and Extensions.* PhD thesis, Texas A&M University, 2007.

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

[16] B. Balasundaram and F. Mahdavi Pajouh. Graph theoretic clique relaxations and applications. In P. M. Pardalos, D. Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 1559–1598. Springer New York, 2013.

[17] R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29:610–637, 2001.

[18] V. Boginski, S. Butenko, and P. Pardalos. Statistical analysis of financial networks. *Computational Statistics & Data Analysis*, 48:431–443, 2005.

[19] V. Boginski, S. Butenko, and P. Pardalos. Mining market data: A network approach. *Computers & Operations Research*, 33(11):3171–3184, 2006.

[20] V. Boginski, S. Butenko, and P. M. Pardalos. On structural properties of the market graph. In A. Nagurney, editor, *Innovation in Financial and Economic Networks*, London, 2003. Edward Elgar Publishers.

[21] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic Publishers, 1999.

[22] R. E. Bonner. On some clustering techniques. *IBM Journal of Research and Development*, pages 22–32, 1964.

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

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

[25] A. Broido and K. C. Claffy. Internet topology: connectivity of IP graphs. *Theoretical Biology and Medical Modelling*, 6:1–11, 2009.

[26] A. Buchanan, J. L. Walteros, S. Butenko, and P. M. Pardalos. Solving maximum clique in sparse graphs: an $o(mn + 2^{d/4})$ algorithm for $d$-degenerate graphs. *Optimization Letters*, 8:1611–1617, 2014.

[27] R. Carraghan and P. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.

[28] F. D. Carvalho and M. T. Almeida. Upper bounds and heuristics for the 2-club problem. *European Journal of Operational Research*, 210:489–494, 2011.

[29] Center for Complex Networks Research. Network databases. *http://www3.nd.edu/ networks/resources.htm*, 2007. Accessed: Dec. 2014.

[30] M. S. Chang, L. J. Hung, C. R. Lin, and P. C. Su. Finding large $k$-clubs in undirected graphs. *Computing*, 95:739–758, 2013.

[31] F. Chung and L. Lu. Complex graphs and networks. *CBMS Lecture Series, American Mathematical Society, Providence, RI*, 2006.

[32] D. J. Cook and L. B. Holder. Graph based data mining. *IEEE Intelligent Systems*, 15:32–41, 2000.

[33] F. Corno, P. Prinetto, and M. Sonza Reorda. Using symbolic techniques to find the maximum clique in very large sparse graphs. In *ROC. 1995 Eur. Conf. Design and Test, EDTC95*, IEEE Computer Society, pages 320–324, Washington, DC, 1995.

[34] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profile. *Mathematical programming*, 91:201–213, 2002.

[35] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 114:109–131, 1995.

[36] J. Edachery, A. Sen, and F. J. Brandenburg. Graph clustering using distance-$k$ cliques. Lecture Notes in Computer Science, 1999. DOI: 10.1007/3-540-46648-7.

[37] C. I. Fábián. Handling CVaR objectives and constraints in two-stage stochastic models. *European Journal of Operational Research*, 191:888–911, 2008.

[38] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.

[39] J. Gagneur, R. Krause, T. Bouwmeester, and G. Casari. Modular decomposition of protein-protein interaction networks. *Genome Biology*, 5:1–12, 2004.

[40] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman and Company, 2nd edition, 1979.

[41] M. Gendreau, P. Soriano, and L. Salvail. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, 41:385–403, 1993.

[42] Graph Partitioning and Graph Clustering: Tenth DIMACS Implementation Challenge. DIMACS. *http://www.cc.gatech.edu/dimacs10/index.shtml*, 2012. Accessed: Feb. 2015.

[43] J. Grossman, P. Ion, and R. De Castro. The Erdös number project, 1995.

[44] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual. *http://www.gurobi.com*, 2015. Accessed: Feb. 2015.

[45] W. Haneveld and M. van der Vlerk. Integrated chance constraints: Reduced forms and an algorithm. *Computational Management Science*, 3:245–269, 2006.

[46] S. Hartung, C. Komusiewicz, and A. Nichterlein. Parameterized algorithmics and computational experiments for finding 2-clubs. *Journal of Graph Algorithms and Applications*, 19:155–190, 2015.

[47] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[48] S. Hill, F. Provost, and C Volinsky. Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, 22:256–275, 2006.

[49] D. Houck, E. Kim, G. O'Reilly, D. Picklesimer, and H. Uzunalioglu. A network survivability model for critical national infrastructures. *Bell Labs Tech J*, 8:153–172, 2004.

[50] P. Huang and D. Subramanian. Iterative estimation maximization for stochastic linear programs with conditional value-at-risk constraints. *Computational Management Science*, 9(4):441–458, 2012.

[51] D. Iacobucci and N. Hopkins. Modelling dyadic interactions and networks in marketing. *Journal of Marketing Research*, 24:5–17, 1992.

[52] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences of the USA*, 98:4569–4574, 2001.

[53] H. Jeong, S. P. Mason, A. L. Barabási, and Z.N. Oltvai. Centrality and lethality of protein networks. *Nature*, 411:41–42, 2001.

[54] D. S. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiablility: Second* DIMACS *Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 1996.

[55] S. Kahruman-Anderoglu, A. Buchanan, and S. Butenko O. Prokopyev. On provably best construction heuristics for hard combinatorial optimization problems. *Networks*, 2015. DOI: 10.1002/net.21620.

[56] K. Katayama, A. Hamamoto, and H. Narihisa. An effective local search for the maximum clique problem. *Information Processing Letters*, 95:503–511, 2005.

[57] KEGG BRITE Database. Biomolecular relations in information transmission and expression. *http://www.genome.jp/kegg/brite.html*, 2004. Accessed: Feb. 2014.

[58] P. Krokhmal, J. Palmquist, and S. Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of Risk*, 4:43–68, 2002.

[59] A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value-at-risk. *Computational Management Science*, 3:3–27, 2006.

[60] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets.* Cambridge University Press, 2014.

[61] C. Lim, H. D. Sherali, and S. Uryasev. Portfolio optimization by minimizing conditional value-at-risk via nondifferentiable optimization. *Computational Optimization and Applications*, 46:391–415, 2010.

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

[63] R. D. Luce and A. D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14:95–116, 1949.

[64] J. Ma, F. Mahdavi Pajouh, B. Balasundaram, and V. Boginski. The minimum spanning $k$-core problem with bounded CVaR under probabilistic edge failures. *INFORMS Journal on Computing*, 28:295–307, 2016.

[65] F. Mahdavi Pajouh. *Polyhedral Combinatorics, Complexity & Algorithms for $k$-Clubs in Graphs.* PhD thesis, Oklahoma State University, 2012.

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

[67] F. Mahdavi Pajouh, B. Balasundaram, and I. V. Hicks. On the 2-club polytope of graphs. *Operations Research*, 2015. Accepted.

[68] J. Marincek and B. Mohar. On approximating the maximum diameter ratio of graphs. *Discrete Mathematics*, 244:323–330, 2002.

[69] T. Matisziw and A. Murray. Modeling $s$-$t$ path availability to support disaster vulnerability assessment of network infrastructure. *Computers & Operations Research*, 36:16–26, 2009.

[70] J. Miao and D. Berleant. From paragraph networks to document networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing, 2004 (ITCC 2004)*, volume 1, pages 295–302, april 2004.

[71] R. J. Mokken. Cliques, clubs and clans. *Quality and Quantity*, 13:161–173, 1979.

[72] E. Moradi and B. Balasundaram. Finding a maximum $k$-club using the $k$-clique formulation and canonical hypercube cuts. *Optimization Letters*, pages 1–11, 2015. DOI: 10.1007/s11590-015-0971-7.

[73] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1999.

[74] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[75] M. E. J. Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the National Academy of Sciences*, 101:5200–5205, 2004.

[76] P. R. J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120:197–207, 2002.

[77] P. Pardalos and J. Xue. The maximum clique problem. *Journal of Global Optimization*, 4:301–328, 1993.

[78] G. Pastukhov, A. Veremyev, V. Boginski, and E. L. Pasiliao. Optimal design and augmentation of strongly attack-tolerant two-hop clusters in directed networks. *Journal of Combinatorial Optimization*, 27:462–486, 2014.

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

[80] X. Peng, M. A. Langston, A. M. Saxton, N. E. Baldwin, and J. R. Snoddy. Detecting network motifs in gene co-expression networks through integration of protein domain information, 2007.

[81] A. G. Quaranta and A. Zaffaroni. Robust optimization of conditional value at risk and portfolio selection. *Journal of Banking & Finance*, 32:2046–2056, 2008.

[82] J. C. Rain, L. Selig, H. D. Reuse, V. Battaglia, C. Reverdy, S. Simon, G. Lenzen, F. Petel, J. Wojcik, V. Schachter, Y. Chemama, A. Labigne, and P. Legrain. The protein-protein interaction map of helicobacter pylori. *Nature*, 409:211–215, 2004.

[83] M. Resende and P. Pardalos. *Handbook of Optimization in Telecommunications.* Springer, 3nd edition, 2006.

[84] R. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2:21–41, 2000.

[85] R. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26:1443–1471, 2002.

[86] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27 – 64, 2007.

[87] A. Schäfer. Exact algorithms for s-club finding and related problems. Master's thesis, Friedrich-Schiller-Universit, Jena, 2009.

[88] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, pages 1–9, 2011.

[89] R. Schultz and S. Tiedemann. Conditional value-at-risk in stochastic programs with mixed-integer recourse. *Mathematical Programming*, 105:365–386, 2006.

[90] S. Shahinpour and S. Butenko. Algorithms for the maximum $k$-club problem in graphs. *Journal of Combinatorial Optimization*, 26:520–554, 2013.

[91] S. Shahinpour and S. Butenko. Distance-based clique relaxations in networks: $s$-clique and $s$-club. In B. I. Goldengorin, V. A. Kalyagin, and P. M. Pardalos, editors, *Models, Algorithms, and Technologies for Network Analysis*, pages 149–174. Springer New York, 2013.

[92] T. Smieszek, L. Fiebig, and R. W. Scholz. Models of epidemics: when contact repetition and clustering should be included. *Theoretical Biology and Medical Modelling*, 6:1–11, 2009.

[93] V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100:12123–12128, 2003.

[94] D. Subramanian and P. Huang. An efficient decomposition algorithm for static, stochastic, linear and mixed-integer linear programs with conditional-value-at-risk constraints. Technical report, IBM, 2009.

[95] P.-N. Tan, M.Steingach, and V. Kumar. *Introdiction to Data Mining.* Addison Wesley, 2nd edition, 2006.

[96] L. Terveen, W. Hill, and B. Amento. Constructing, organizing, and visualizing collections of topically related, web resources. *ACM Trans. Comput. Hum. Interact*, 6:67–94, 1999.

[97] E. Tomita and T. Kameda. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *Journal of Global Optimization*, 37:95–111, 2007.

[98] S. Uryasev. Conditional value-at-risk: optimization algorithms and applications. *Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on, IEEE*, pages 49–47, 2000.

[99] R. Van Slyke and R. Wets. L-shaped linear programs with applications to control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[100] A. Veremyev. *Modeling and optimization approaches for ensuring robustness in networked and financial systems.* PhD thesis, University of Florida, 2011.

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

[102] A. Veremyev, O. A. Prokopyev, and E. L. Pasiliao. Critical nodes for distance-based connectivity and related problems in graphs. *Networks*, 66(3):170–195, 2015.

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

[104] T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5:59–68, 2003.

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

[106] D. R. Wood. An algorithm for finding a maximum clique in a graph. *Operations Research Letters*, 21:211–217, 1997.

[107] M. Yannakakis. Node-and edge-deletion NP-complete problems. In *STOC '78: Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 253–264. ACM Press, New York, NY, 1978.

[108] B. Zhou, X. Cai, and N. Trinajstic. On Harary index. *J Math Chem*, 44:6011–618, 2008.

VITA

Esmaeel Moradi

Candidate for the Degree of

Doctor of Philosophy

Dissertation: DECOMPOSITION ALGORITHMS FOR DETECTING LOW-DIAMETER CLUSTERS IN GRAPHS

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Shahriar, Tehran, Iran on May 18$^{th}$.

Education:

- Received the B.S. degree from Sharif University of Technology, Tehran, Iran, 2006, in Industrial Engineering.
- Received the M.S. degree from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, 2009, in Industrial Engineering.
- Completed the requirements for the degree of Doctor of Philosophy with a major in Industrial Engineering and Management at Oklahoma State University, 2016.

Selected Publication:

- Moradi, E., and Balasundaram, B. (2015). Finding a maximum $k$-club using the $k$-clique formulation and canonical hypercube cuts. Optimization Letters, DOI 10.1007/s11590-015-0971-7.
- Moradi, E., and Balasundaram, B. (2016). A delayed constraint generation approach for the maximum $k$-club problem. European Journal of Operational Research, Under First Revision.
- Mahdavi Pajouh, F., and Moradi, E., and Balasundaram, B. (2016). Detecting large risk-averse 2-clubs in graphs with random edge failures. Annals of Operations Research, Under First Revision.

Experience:

- Logistic Planner, Walmart, Bentonville, AR, USA, 2015-Present.