GRAPH THEORETIC GENERALIZATIONS OF CLIQUE: OPTIMIZATION AND EXTENSIONS

A Dissertation

by

BALABHASKAR BALASUNDARAM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2007

Major Subject: Industrial Engineering

UMI Number: 3281026

UMI Microform 3281026

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

GRAPH THEORETIC GENERALIZATIONS OF CLIQUE: OPTIMIZATION AND EXTENSIONS

A Dissertation

by

BALABHASKAR BALASUNDARAM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Sergiy I. Butenko
Committee Members,	Illya V. Hicks
	Wilbert E. Wilhelm
	Catherine H. Yan
Head of Department,	Brett A. Peters

August 2007

Major Subject: Industrial Engineering

ABSTRACT

Graph Theoretic Generalizations of Clique:

Optimization and Extensions. (August 2007)

Balabhaskar Balasundaram, B.Tech., Indian Institute of Technology – Madras

Chair of Advisory Committee: Dr. Sergiy Butenko

This dissertation considers graph theoretic generalizations of the *maximum clique problem*. Models that were originally proposed in *social network analysis* literature, are investigated from a mathematical programming perspective for the first time. A social network is usually represented by a graph, and cliques were the first models of "tightly knit groups" in social networks, referred to as *cohesive subgroups*. Cliques are idealized models and their overly restrictive nature motivated the development of *clique relaxations* that relax different aspects of a clique. Identifying large cohesive subgroups in social networks has traditionally been used in criminal network analysis to study organized crimes such as terrorism, narcotics and money laundering. More recent applications are in clustering and data mining wireless networks, biological networks as well as graph models of databases and the internet. This research has the potential to impact homeland security, bioinformatics, internet research and telecommunication industry among others.

The focus of this dissertation is a degree-based relaxation called *k-plex*. A distance-based relaxation called *k-clique* and a diameter-based relaxation called *k-club* are also investigated in this dissertation. We present the first systematic study of the complexity aspects of these problems and application of mathematical programming techniques in solving them. Graph theoretic properties of the models are identified and used in the development of theory and algorithms.

Optimization problems associated with the three models are formulated as bi-

nary integer programs and the properties of the associated polytopes are investigated. Facets and valid inequalities are identified based on combinatorial arguments. A branch-and-cut framework is designed and implemented to solve the optimization problems exactly. Specialized preprocessing techniques are developed that, in conjunction with the branch-and-cut algorithm, optimally solve the problems on real-life *power law graphs*, which is a general class of graphs that include social and biological networks. Computational experiments are performed to study the effectiveness of the proposed solution procedures on benchmark instances and real-life instances.

The relationship of these models to the classical maximum clique problem is studied, leading to several interesting observations including a new compact integer programming formulation. We also prove new continuous non-linear formulations for the classical maximum independent set problem which maximize continuous functions over the unit hypercube, and characterize its local and global maxima. Finally, clustering and network design extensions of the clique relaxation models are explored.

Dedicated to my parents

ACKNOWLEDGMENTS

I consider myself truly lucky to have worked with Dr. Sergiy Butenko for my doctoral research. Sergiy has been a resourceful, insightful and patient advisor, a valuable guide in my professional development and most importantly, a true friend and colleague. I am grateful to Sergiy for making my doctoral experience a rich and memorable one, and my admiration and respect go to him.

I would like to express my sincere thanks to Dr. Illya Hicks, my committee member and collaborator, for taking a keen interest in my research and professional development. His expertise in polyhedral combinatorics was a tremendous support for me and guided several research directions taken in this dissertation. My thanks are also due to my wonderful committee members Dr. Wilbert Wilhelm and Dr. Catherine Yan, for their patience and constant support.

I am ever grateful to Dr. G. Srinivasan, my undergraduate mentor, for introducing me to the fascinating field of operations research, and for encouraging me to pursue a doctorate.

The ISE department at Texas A&M provided me with a wonderful learning atmosphere and opportunities to develop the skills I need in academia. I would especially like to thank Drs. Brett Peters, Guy Curry and Richard Feldman, for providing me with several opportunities to teach, and for their guidance and support.

I am also indebted to Drs. Amarnath Banerjee, Gautam Natarajan, Yu Ding, Lewis Ntaimo, Gary Gaukler, Eylem Tekin, Andrew Johnson and Eric Bickel, for guiding me and supporting me through my search for a faculty position.

Without the support from the efficient and friendly administrative and technical staff at ISE, my doctoral program would have been a lot more difficult. In particular, my special thanks are due to Judy Meeks, Michele Bork, Claudia Samford, Katherine

Edwards, Mark Henry, Mark Hopcus and Dennis Allen. I would also like to thank the ISE department for financially supporting my graduate studies.

I would like to thank Deepak Warrier, Sharat Bulusu, Brijesh Vasudeva Rao, Svyatoslav Trukhanov, Oleksii Ursulenko, Reza Seyedshohadaie, Sera Kahruman, Sandeep Sachdeva, Homarjun Agrahari, Abhishek Shrivastava, Elif Kolotoglu and Jung Jin Cho, for being wonderful colleagues, and friends.

I would also like to thank Sujan Dan, Gabriel Krishnamoorthy, Smriti Jayaraman, Vijay Ramakrishnan, Taraka Donti and Karambir Kalsi, for their lasting friendships which made my life in College Station, a memorable one.

There are numerous others who have helped me personally and professionally, and it would be impossible for me to name all of them. But my sincere thanks are due to them all.

Words cannot express the love and pride I have for my parents for making me who I am. It is their ambition, encouragement and support that has always kept me on the right track. I am ever grateful and indebted to my parents for always giving me more than I wanted, more than I deserved and more than they could.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

TABLE Page

TABLE Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Network is a popular term that immediately reminds the reader of *points* in space that are connected by lines, representing a physical network or used as a visualization tool representing interconnected information. A *graph* G is the mathematical abstraction of our visual fix, defined by a pair (V, E) where V is the vertex set and E is the edge set. Fig. 1 illustrates *the Petersen graph*. Graphs are simple and effective tools that can be used to model many real-life situations. Typically the vertex set represents entities and the edges indicate the presence or absence of pairwise relationships.

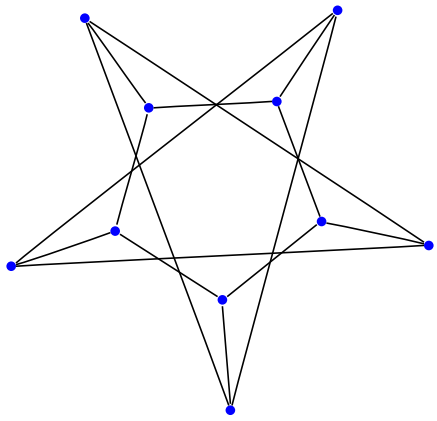


Fig. 1 The Petersen graph (consult [116] for more information about this graph, its unique position and historical significance in graph theory)

Modeling information as a graph presents several advantages. Firstly, it can globally capture massive amounts of information starting with local pairwise interconnections. Secondly, it places the problems of interest in the intersection of Graph Theory, Combinatorial Optimization, Algorithms and Complexity Theory thereby

---

The journal model is Mathematical Programming.

providing rigorous ways to characterize the tractability or intractability of the problems, and to effectively cope with them. Lastly, the information and results can be presented in an intuitive and user-friendly setting using good visualization tools.

Real-life graphs are usually "complex" and are based on various novel models to represent the underlying information graphically. However, several real-life graphs of interest to us, despite arising in diverse application domains share some interesting properties. In the following paragraphs, we discuss the underlying modeling ideas and the structural commonalities that have been observed in real-life graphs.

Complex Graphs and Power Laws. In internet research, typically two models are employed. An *internet graph* has vertices representing IP addresses while a *web graph or WWW graph* has vertices representing websites. Edges in such graphs are determined based on information from routing protocols or using traceroute probes [48]. In *call graphs*, vertices represent telephone numbers and an edge represents a call placed from one vertex to another in a specified time interval [5]. *Stock-market graphs* have vertices representing stocks and two stocks are connected by an edge if they are positively correlated over some threshold value based on calculations over a period of time in history [38]. Biological networks such as *protein interaction networks* and *gene co-expression networks* are used to model biological information. A protein interaction network is represented by a graph with the proteins as vertices, and an edge exists between two vertices if the proteins are known to interact based on two-hybrid analyses and other biological experiments [120]. In gene co-expression networks, vertices represent genes and an edge exists between two vertices if the corresponding genes are co-expressed with correlation higher than a specified threshold in microarray experiments [163]. *Social networks* are graph models representing sociological information such as acquaintance among people. Vertices usually represent people and an edge indicates a "tie" between two people. A tie could mean that they know

each other, they visited the same place or any other sociological connection. *Scientific collaboration networks* with vertices representing authors (in a particular field or with publications in a particular journal) and edges indicating co-authorship fall under this category [105].

One should however not be deceived by the "innocent picture" of the Petersen graph since all of the real-life graphs that we referred to above are massive. One such massive call graph representing telecommunications traffic data over one 20-day period studied in [5] had 290 million vertices and 4 billion edges. Although we do not handle graphs this massive, we do investigate the protein interaction network of gastric pathogen *H. Pylori* which has 1570 vertices and 1403 edges that are almost all inside one giant component with 706 vertices (see Fig. 2). Apart from the fact that graphs are used to conveniently model large amounts of data from real applications, such real networks also exhibit another interesting property introduced next.

Power laws have a long history, that dates as far back as the late 1800s (see Section 1.4 in [68]). However recently, *power law degree distributions* have been observed among networks that arise from diverse real world situations [31]. If $X(t)$ denotes the number of vertices in the network with degree t, it was observed in empirical studies that $X(t) \propto t^{-\gamma}$ where $\gamma \geq 1$ is a constant. Power law has been observed in natural and man-made networks, such as internet and web graphs [32], biological networks [10], stock-market graphs [38] and social networks [101] among others. Such power law graphs are also known in the literature as *scale-free graphs*.

In such graphs, a large number of vertices have very few neighbors while a small number of vertices have an extremely large number of neighbors. An immediate consequence of this property in these real-life graphs is that they are massive in terms of number of vertices, but extremely sparse in number of edges. It is believed that a *principle of preferential attachment* operates in the evolution of such networks where

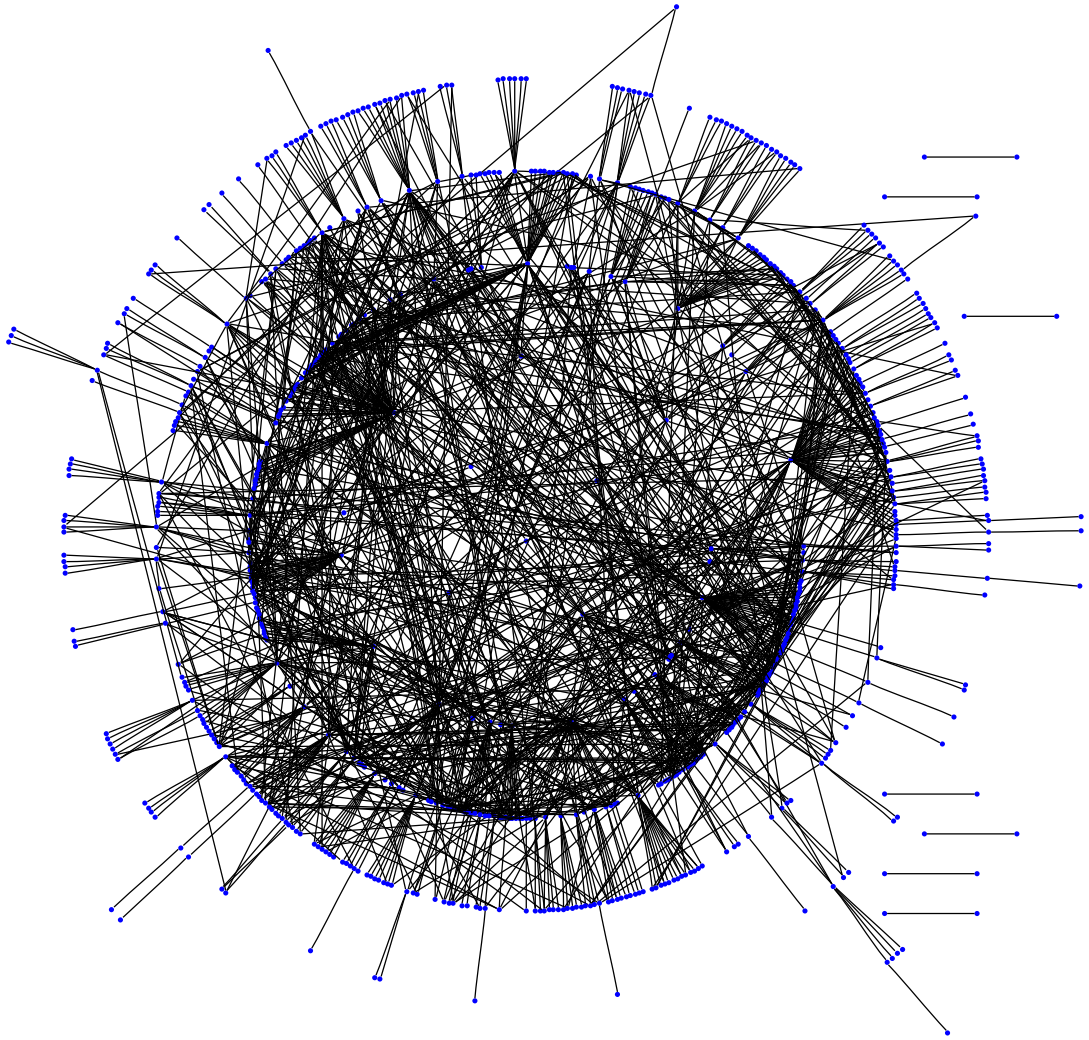


Fig. 2 Protein-protein interaction map of H. Pylori

the "rich get richer", *i.e.*, vertices that already have a large number of neighbors, attract more new neighbors compared to vertices with fewer neighbors. For this reason, they are different from the graphs studied in the classical random graph theory of Erdös and Rényi [87, 88]. Here, the $G(n, p)$ model describes the probability space containing all labeled graphs on n vertices and $p \in [0, 1]$ is the probability that an edge exists between any two vertices and it is *independent* of other edges. In the evolution of such random graphs, a "giant component" emerges when $np > 1$ [11].

This behavior is also observed in power law graphs, and the conditions are studied in [68] using a generalized model that includes $G(n, p)$ and power law models.

Another popular concept is the *small world phenomenon* identified by social psychologist Stanley Milgram in 1967 [145] in his famous experiments where the subjects were given a letter each, all of them supposed to reach the same destination, but the letters can only be personally handed over by each person to a friend who they think would successfully pass it on to the destination. He observed that the average number of links in successfully completed letter chains were about 6, giving rise to the phrase *six degrees of separation*. Although his initial study was heavily debated for its experimental set-up and interpretation of results, it is widely accepted as the earliest attempt at observing the small world phenomenon. In the late 1990s, small world phenomenon was shown to exist in various networks, not just social networks by Watts and Strogatz [186]. However, these observations are empirical in nature. Discussions on theoretical and empirical studies on the small world phenomenon can be found in the recent book [68], that provides a comprehensive mathematical treatment of the subject extending classical random graph theory and its techniques to specifically address large, sparse graphs such as power law graphs. In this dissertation, real-life graphs of interest to us exhibit power law degree distribution, and hence the aforementioned properties are exploited in algorithm development.

Cliques and Social Network Analysis. The study of social networks referred to as *Social Network Analysis* (SNA) has several critical applications that rely on identifying "tightly knit" subgraphs called *cohesive subgroups*. Mathematical modeling of cohesive subgroups has long been a subject of interest in SNA with the earliest models being *cliques*. A clique in an acquaintance or friendship network is a group of people in which everyone knows everyone. Mathematically, a clique in a graph is a subset of pairwise adjacent vertices. Cliques, thought to be perfect models for

cohesive subgroups were later found to be too idealized and restrictive in practice, leading to graph models that generalized the definition of cliques.

This dissertation research deals with such graph theoretic clique relaxation models introduced in SNA called *k-cliques, k-clubs* and *k-plexes*. These parameterized models represent cliques when parameter $k = 1$ and provide a relaxation for $k > 1$. These models have several critical applications that impact *homeland security, biological emergency response, telecommunication, internet research, genomics, proteomics* and *drug-discovery*. Despite their importance, these clique relaxations have hardly been studied by mathematicians, computer scientists or operations researchers. This fact is surprising for two reasons. Firstly, the problem of finding large cliques which is a *classical* problem in all the above mentioned fields of study was originally motivated by applications in social networks– however its relaxations were scarcely noticed. Secondly, the *need existed* for models that relax cliques for a long time and was addressed by methods that were often not systematic and with other drawbacks. The important characteristics of a clique when viewed as a cohesive subgroup are complete *familiarity* within the group, quickest *reachability* enabling fast communication within the group and a structural *robustness* that cannot be destroyed by removing members of the group. These observations essentially mean the same when viewed from a mathematical perspective, the outcome of having all possible edges. However, when viewed from a social perspective, they help motivate different relaxations. In this light, it is easy to see why the operations research and the computer science community was drawn towards models that relaxed the notion of "all possible edges" to look for "dense enough" subgraphs. It is important to note that, it is by looking at cliques through a social perspective, that we can motivate the relaxations. This explains why such interesting relaxations were first developed in SNA and not in other areas.

Optimization problems addressed in this dissertation are to find a largest k-

clique, k-club or k-plex in a graph. We focus more on the k-plex model due its advantage over the other models and its systematic nature. Broadly, this dissertation makes the following contributions. Firstly, we have shown that these problems are *NP-hard*, making our efforts to solve these problems difficult, but worthwhile. The rest of our results utilize classical techniques from *mathematical programming* in developing theory and algorithms for solving the problems. The optimization problems are formulated as integer programs and the properties of the associated polytopes are investigated. We have produced several families of combinatorial valid inequalities and facets for the problems. Selected inequalities are incorporated in branch-and-cut procedures to solve the problems to optimality. Furthermore, we develop specialized preprocessing techniques that work extremely well for large and sparse graphs enabling us to solve these problems to optimality on real-life power law graphs.

The relationship of these models to the classical maximum clique problem is studied, leading to several interesting observations including a new compact integer programming formulation. We also prove new continuous non-linear formulations for the classical maximum independent set problem which maximize continuous functions over the unit hypercube and characterize their local and global maxima. Finally, *clustering* and *network design* extensions of the clique relaxation models are explored.

Organization. The remainder of this dissertation is organized as follows. Chapter II presents an extensive background on SNA and its applications; required notations and definitions from graph theory, basics of complexity theory, polyhedral theory and branch-and-cut approaches; and a review of relevant literature on cliques, independent sets and successful applications of branch-and-cut from literature. In Chapter III we formally define the models and the associated optimization problems. Comparison of the models with respect to the social characteristics of cliques suggested before is carried out revealing the k-plex model to be most attractive.

Chapter IV presents the complexity results for our problems. Meaningful tractability questions are identified and answered in this chapter. Integer programming formulations, polyhedral results, facets and valid inequalities are developed in Chapters V and VI. Algorithms that combine pre-processing techniques and branch-and-cut approaches for the problems are also presented in these chapters. The k-clique and k-club models are studied in Chapter V while the k-plex model is studied extensively in Chapter VI. Summary of computational experiments performed using the algorithms developed in Chapter VI is given in Chapter VII and the details are included in Appendix A.

In Chapter VIII, we switch gears from combinatorial optimization to continuous optimization. But we still address a classical combinatorial optimization problem that is tied to our theme: The *maximum independent set problem* which is equivalent to finding a largest clique. In the past, some of the most important breakthroughs in operations research and optimization that have gone on to impact a wide variety of applications, have resulted from continuous approaches to combinatorial optimization problems. New formulations for the maximum independent set problem and results characterizing their local and global maxima are identified, in our contribution to this emerging research field.

As suggested by the title of this dissertation, we not only study the natural optimization problems that arise from the models we borrowed from SNA, but we also propose several new extensions that utilize the basic idea to develop other optimization problems with different, but related applications. These models and algorithms are proposed in Chapter IX.

This dissertation studies problems that are rich and deep for both theory and practice. Naturally, we only lay the foundations for a systematic study of these important problems. In the course of our research, we have identified several important

open problems and research directions that need to be addressed in the future. We conclude this dissertation with Chapter X where some of these issues are discussed.

Publications. Some results in Chapters IV and V have appeared in [30] and some from Chapters IV and VI have been submitted for publication [29]. Results from Chapter VIII have appeared in [24, 26]. During the course of this dissertation research, two survey articles of graph optimization problems in telecommunication were written [25, 28] and an expository article on network clustering [27] was also completed. These efforts brought to our attention and motivated some of the modeling extensions introduced in Chapter IX.

All the figures in this dissertation, except Fig. 6 and Fig. 7 were generated using GRAPHVIZ® [104].

CHAPTER II

BACKGROUND

This chapter reviews the background material required by this dissertation. In Section II.1 we review concepts from social network analysis and various applications of the area. Notations and definitions from graph theory are presented in Section II.2. A brief background on complexity theory and polyhedral theory are provided in Section II.3 and Section II.4 respectively. Review of results from literature related to branch-and-cut approaches, cliques and independent set is presented in Sections II.5, II.6.

II.1. Social Network Analysis

A social network is usually represented by a graph, in which the set of vertices represent the *actors* in a social network and the edges represent *ties* between them [171]. Typically, actors are people, and examples of a tie between two actors include acquaintance, friendship, or other type of association between them, such as visiting the same social event or place at the same time. Alternately, actors can be companies, with ties representing business transactions between them.

Cliques and their graph theoretic generalizations were proposed in SNA to model *cohesive subgroups* in social networks. *Social cohesion* is often used to explain and develop sociological theories. Members of a cohesive subgroup tend to share information, have homogeneity of thought, identity, beliefs, behavior, even food habits and illnesses [185]. Social cohesion is also believed to influence emergence of consensus among group members. Examples of cohesive subgroups include religious cults, terrorist cells, criminal gangs, military platoons and sports teams among others.

Modeling a cohesive subgroup mathematically has long been a subject of interest in SNA. One of the earliest graph models used for studying cohesive subgroups was *cliques* [141]. A clique is a subset of vertices in which every pair has an edge between them. Cliques are *ideal* structures for modeling cohesive subgroups. They have three important structural properties that are expected of a cohesive subgroup, namely, *familiarity* (all neighbors and no strangers in the group), *reachability* (direct communication inside the group) and *robustness* (difficult to destroy the group by removing members). However, the clique approach has been criticized for its overly restrictive nature [8, 185] and modeling disadvantages [173, 93].

A clique requires all possible edges to exist between vertices for them to be considered a cohesive subgroup. This is a very restrictive definition for two basic reasons. Firstly, in reality a cohesive subgroup that has high familiarity, reachability and robustness, need not have all possible edges *i.e.*, it could include some "strangers". Such strangers would be excluded if cliques were used to model cohesive subgroups. Secondly, real-life networks are often constructed based on data that could be erroneous or incomplete. Even if the cohesive subgroup was indeed a clique, but some edges were missed due to experimental errors, the restrictive definition of cliques would again exclude members that should belong in the cohesive subgroup. Hence cliques are ideal, but not practical models of cohesive subgroups. Using cliques makes the cohesive subgroup extremely robust to "false-positives" (edges *included* in error) in the data since if a node belongs to a clique, it is unlikely that all (or many) of its neighbors are results of experimental errors. On the other hand, clique is overly sensitive to "false-negatives" (edges *excluded* in error) since a node could be excluded from a clique for missing a single edge to one of the clique members which could be due to an error in the data. This motivated the development of clique relaxations that relax different aspects of a cohesive subgroup and try to overcome the inherent

difficulties in the clique model.

Luce [140] introduced a distance-based model called *k-clique* and Alba [8] introduced a diameter-based model called *k-club*. These models were also studied along with a variant called *k-clan* by Mokken [148]. However their definitions required some modifications from the original definition to be mathematically more meaningful. These drawbacks are pointed out and the models are appropriately redefined in Chapter III. Seidman and Foster [173] introduced a degree-based model called *k-plex*, which is the third model we study along with the k-clique and k-club models. This model is the main focus of this dissertation and is also formally defined in Chapter III.

The parameter k in all three parameterized models controls the "extent of relaxation". They represent cliques when $k = 1$ and represent relaxations for $k > 1$. While the distance-based and diameter-based models emphasize the need for high reachability (small k) inside a cohesive subgroup, they may not be suitable models of cohesiveness with respect to familiarity and robustness. The degree-based model relaxes familiarity within a cohesive subgroup and implicitly achieves reachability and robustness. This model is a more systematic relaxation of cliques. These issues are described in greater detail in Section III.3.

In the wake of the information revolution, the interest in studying the network structure of organizations, especially criminal in nature, has increased manifold. Social network concepts, despite their versatility, have come to the forefront especially for these applications. We will now cite some direct application areas of social networks, as well as several emerging areas where we believe our research will have significant impact.

Terrorist Recruitment Models. The recent world events have sparked interest in studying recruitment within terrorist groups. The tools of SNA have been used extensively in this respect [167, 36]. Based on the understanding that recruitment

is significantly due to the influence of actors in a social group, cliques have been used in the development of simulation toolkits for studying recruitment models [36]. However, clique models of cohesive subgroups will underestimate the influence on a particular actor X by the actor's social group as it may exclude some neighbors of X that are not fully connected to the group. This could lead to inaccurate simulation results. Our research on clique relaxations aims to contribute to the improvement of such models.

Criminal Network Analysis. Study of terrorist networks is essentially a special case of criminal network analysis that is intended to study organized crimes such as terrorism, narcotics and money laundering [143, 80]. Concepts of SNA provide suitable data mining tools for this purpose [61, 60]. Given that these massive amounts of intelligence and law enforcement data are plagued by errors, repetitions, incomplete information etc. [178], a restrictive model like clique is not practical. Clique relaxations, especially k-plexes will be of great practical value since they will uncover groups in which some links are absent, either due to incomplete information or errors in the data. Similarly, *link analysis* is also an important tool used in wire transfer screening to detect money laundering [183]. Link analysis essentially constructs the graph with the information from a wire transfer database. *Database networks* in general are often constructed by first designating a field as *matching field*. Then vertices representing records in the database are connected by an edge if the two matching fields are "close". But these records are known to be incomplete, faulty and fragmented [183]. Finding clique relaxations on such *wire transfer graphs* can yield different types of information depending on the choice of the matching field. Choice for matching fields could include address, beneficiary/originator of the wire transfer, origin/destination routing numbers, etc. and will yield different insights into the activities. *Telephone toll analysis* on call graphs introduced in Chapter I has also

been suggested as an useful tool to monitor criminal organizations [178]. Approaches for finding clique relaxations are better suited for such purposes and would greatly aid crime prevention and prosecution by their better representation of cohesive subgroups.

Epidemic Control, Internet Research and Marketing. Target applications from these seemingly diverse areas are tied together by a single concept called *viral marketing*. This marketing jargon describes any strategy that encourages individuals to pass on messages to others, leading to rapid growth in the message's exposure and influence [127]. Whether it is a marketing virus or a malicious computer virus spreading through the internet or a biological virus spreading through a social network of people, the phenomena naturally share the same network dynamics [9, 32]. Identifying clique relaxations can provide important information about the structure that can be used to contain the computer/biological virus by identifying the largest cohesive subgroup containing the infected websites/known virus carriers. On the other hand, it can be used to maximize the influence of a marketing virus by finding large cohesive subgroups or by clustering using the clique relaxations as cluster models to identify which actors need to be chosen to start the viral marketing campaign. In a related application, Terveen et al. [181] utilize 2-clubs to represent collections of densely connected web sites (referred to as *clans* in [181]). Topically related web sites are thus identified and organized to facilitate faster search and retrieval of information from the web. Clique relaxations such as a k-plex present a realistic approach for modeling and studying such phenomena.

Network Clustering and Data Mining. Clustering can be loosely defined as the process of grouping objects into subsets. Each subset is a *cluster* modeled by structures such as cliques or clique relaxations. The model used as a cluster represents the *cohesiveness* required of the cluster. Clique and other low diameter models have

been popular in the area of wireless communication [62, 132]. Clustering the *connectivity graph* of a wireless network introduces a hierarchy which facilitates routing of information through the network resulting in efficient resource management and better throughput performance. A similar principle is also used in *organizational management* where SNA is used to study organizational structure to suggest better work practices and improve communication and work flow [81]. Graph based data mining [71] has been used in studying structural properties of social networks [151], unraveling molecular structures to facilitate drug discovery and compound synthesis [91] and for identifying frequently occurring patterns in data sets (modeled as graphs) [184]. Clustering using cohesive units such as k-plexes in these graphs provides a valuable tool to organize data as well as indicate interesting structures in the graph that are of practical importance.

Biological Systems. In this post-genomic era, classical biology is undergoing a transformation into *systems biology* in order to study organisms as biological systems. Biologists aiming at a systems-level understanding are approaching biology from a top-down perspective with the objective of understanding and predicting the behavior of biological systems [126]. This is often accomplished by taking a "snapshot" of all the elements and their interactions at various levels– genes, transcripts, proteins, metabolites. This has been made possible by technological advances such as microarrays that facilitate parallel high-throughput experiments resulting in enormous amounts of experimental data, such as genomic and proteomic data. This has led to a phenomenon that is very imaginatively termed the "data avalanche". As stated in the introductory chapter, graphs present a simple and effective tool for modeling such massive amounts of data. Several biological interactions are well captured by networks such as protein-protein interaction networks and gene co-expression networks introduced in Chapter I.

Identifying large clusters or functional modules in biological networks can aid different objectives depending on the nature of these networks. Clique models have been most popular in this area. Cliques have been used to cluster gene co-expression networks [163, 124], while cliques and high-density subgraphs have been used to cluster protein interaction networks [180, 95]. For instance, finding cliques in a protein interaction network is used to identify protein complexes and functional modules [180]. Protein complexes are groups of proteins that act as a multi-molecular machine by interacting at the same time, and in the same place in the cell. Functional modules on the other hand is a group of proteins that participate in a cellular process while interacting with each other at a different time and place. Protein complexes and modules have been shown to be responsible for several important cell functions (see [180]).

Graph theoretic clique relaxations can provide interesting insights into these networks and provide more information than what is revealed by cliques. The voluminous data generated by microarray experiments and other procedures are bound to contain significant percentage of errors. There could also be missing edges if interactions between elements are not yet known. Relaxing the restrictions imposed by clique models could reveal new protein interactions. In particular, structures where interactions of proteins occur through a central protein, which are likely to be found in similar biological processes can be identified [17]. Several important applications of cliques in computational biochemistry and genomics such as integration of genome mapping data, non-overlapping local alignments and matching three-dimensional molecular structures are also well known [53].

This brief survey of SNA and its wide applicability is sufficient to motivate our study of the aforementioned clique relaxations. For a detailed introduction to the area, see the texts by Wasserman and Faust [185] and Scott [171]. Since the introduction of these models, there has been almost no literature that deals with these important

application oriented problems in the areas of mathematics, operations research or computer science. This dissertation will lay the foundations for a systematic study of these problems from the perspectives of mathematical programming and theoretical computer science.

II.2. Graph Theory

For an introduction to graph theory readers are referred to texts by West [188] or Diestel [83]. We only provide the notations and basic definitions for the sake of clarity.

In this dissertation, we always consider finite, simple, undirected graphs denoted by $G = (V, E)$, where $V = \{1, \ldots, n\}$ and $(i, j) \in E$ when vertices i and j are adjacent (assume $i < j$) with $|E| = m$. By *order* we mean the number of vertices n and by *size*, the number of edges m. We also use the notation $V(G)$ and $E(G)$ to denote the vertex set and edge set, respectively, of a given graph G. Whenever the graph under consideration is obvious, we sometimes use n instead of $|V|$ to denote its order. The *complement graph* is denoted by $\bar{G} = (V, \bar{E})$. Given $X \subseteq V$, *induced subgraph* $G[X]$ is obtained by deleting from G, all vertices (and incident edges) in $V \setminus X$. The graph obtained by the deletion of a vertex i or a set of vertices I from G is denoted respectively by $G - i$ and $G - I$. By $G_1 \cup G_2$ we denote the union graph $G = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. G is called a *null graph* if $V = E = \emptyset$ and a trivial graph if $E = \emptyset$. The *adjacency matrix* A_G is a symmetric 0,1-matrix of order $n \times n$ with $a_{i,j} = 1 \Leftrightarrow (i, j) \in E$. The *complete graph* on n vertices is denoted by K_n and the *complete bipartite graph* with bipartitions of sizes p, q is denoted by $K_{p,q}$. In particular, the graph $K_{1,n}$ is called a *star*. *Cycle* and *path* on n vertices are denoted by C_n and P_n respectively.

For a vertex $i \in V, N(i)$ denotes the set of vertices adjacent to i in G, called

the *neighborhood* of i and $N[i] = \{i\} \cup N(i)$ denotes the *closed neighborhood* of i. The set of *non-neighbors* of i is given by $V \setminus N[i]$. Denote by $deg_G(i)$, the degree of vertex i in G given by $|N(i)|$. Degree of a vertex $i \in X \subseteq V$ in $G[X]$ is denoted by $deg_{G[X]}(i)$ where $deg_{G[X]}(i) = |N(i) \cap X|$. The neighborhood of a vertex i in a subgraph $\tilde{G} \subseteq G$ is denoted by $N_{\tilde{G}}(i)$. The maximum and minimum degrees in a graph are denoted respectively by $\Delta(G)$ and $\delta(G)$. For two vertices $i, j \in V$, $d_G(i, j)$ denotes the length of the shortest path between i and j in G. The distance in the induced subgraph, for $i, j \in X \subseteq V$ is denoted by $d_{G[X]}(i, j)$. By convention, when no path exists between two vertices, the shortest distance between them is infinity. If the graph G under consideration is obvious, we sometimes drop the subscript in the neighborhood, degree and distance notations.

For a graph $G = (V, E)$ and positive integer k, the k^{th} power of G is defined as $G^k = (V, E^k)$, where $E^k = \{(i, j) : i, j \in V,\ i < j,\ d_G(i, j) \leq k\}$. G^k is constructed from the original graph by adding edges corresponding to all pairs of vertices with distance no more than k between them in G.

Definition 1. The *diameter* of a graph is defined as $diam(G) = \max_{i,j \in V} d_G(i, j)$.

Definition 2. The *vertex connectivity* $\kappa(G)$ of a graph is the minimum number of vertices whose removal results in a disconnected or trivial graph.

Definition 3. A *clique* is a subset of vertices that are pairwise adjacent in the graph.

Definition 4. An *independent set* (stable set, vertex packing) is a subset of vertices that are pairwise non-adjacent in the graph.

The *maximum clique problem* is to find a clique of maximum cardinality. The *clique number* $\omega(G)$ is the cardinality of a maximum clique in G. The maximum cardinality of an independent set of G is called the *independence number* of the graph

G and is denoted by $\alpha(G)$. The associated problem of finding a largest independent set is the *maximum independent set problem*. Note that I is an independent set in G *if and only if* I is a clique and $\bar{G}$ and consequently $\alpha(G) = \omega(\bar{G})$.

Always, *maximality* and *minimality* of sets are defined based on inclusion and exclusion respectively. A *maximal independent set* is one that is not a proper subset of another independent set. A *maximal clique* is one that is not contained in a larger clique. For simplicity, we write MIS in place of *maximal* independent set(s) in this dissertation.

Definition 5. A *proper coloring* of a graph is one in which every vertex is colored such that no two vertices of the same color are adjacent.

A graph is said to be *k-colorable* if it admits a proper coloring with k colors. Vertices of the same color are referred to as a *color class* and they induce an independent set. The *chromatic number* of the graph, denoted by $\chi(G)$ is the minimum number of colors required to properly color G. Note that for any graph G, $\omega(G) \leq \chi(G)$, as different colors are required to color the vertices of a clique.

Definition 6. A *dominating set* is a subset of vertices such that every vertex in the graph is either in this set or has a neighbor in this set.

The minimum cardinality of a dominating set is called the *domination number*, denoted by $\gamma(G)$. Note that every MIS is also a minimal dominating set.

Definition 7. A graph is *perfect* if the chromatic number is equal to the clique number for every vertex-induced subgraph.

Definition 8. A *hole* is an induced chordless cycle of length at least four and an *antihole* is the complement of a hole.

II.3. Complexity Theory

Theory of NP-completeness plays a very important role in optimization. It provides a framework for comparing the hardness of different problems and broadly classify them as easy and hard. In this section, we give a brief review of the topic. Interested readers are referred to [98, 159] for a comprehensive treatment of the subject.

The framework of NP-completeness is designed to address the tractability of *decision problems*. These are problems such that every instance can be answered by either a "yes" or a "no". Instances that can be answered with a "yes" are referred to as *yes-instances* for the problem and instances answered by a "no" are *no-instances* for the problem. Consider for example, the maximum clique problem. This can be phrased as a decision problem as follows.

CLIQUE : Given a graph $G = (V, E)$ and a positive integer c, does there exist a clique of size $\geq c$ in G?

Each instance of CLIQUE problem is defined by (G, c) and we can solve the maximum clique problem on G by varying c from 1 to n and identifying the largest c for which (G, c) is a yes-instance.

Definition 9. A decision problem $\mathcal{Q}$ is said to be in *class P* if an algorithm exists that can answer it correctly in a running time that is polynomially bounded by its input size.

Polynomial-time algorithms are considered "efficient" and hence problems in class P and their optimization variants are considered to be "easy". However, most optimization problems do not belong to this class. They belong to a larger class and are some of the hardest problems in that class.

Definition 10. A decision problem $\mathcal{Q}$ is in *class NP* if there exists a polynomial-time algorithm $\mathcal{A}$ such that:

1. Given a yes-instance x and a binary string y polynomially bounded in length representing a solution, $\mathcal{A}$ returns "yes" as the answer for some y that solves x;
2. Given a no-instance x', for any string y, $\mathcal{A}$ returns "no" as the answer.

In other words, decision problem $\mathcal{Q}$ is placed in class NP if there exists a polynomial-time algorithm $\mathcal{A}$ that responds with a yes for a yes-instance x when solution y that correctly solves x is given as a hint and for a no-instance x', $\mathcal{A}$ cannot be fooled by any string y representing a solution and it always responds with a no. Note that $\mathcal{A}$ does not know how to construct a correct y for a given yes-instance x and can only test it for its correctness. This non-deterministic nature of "guess" y can be included in the description of $\mathcal{A}$ and class NP can be redefined as follows.

Definition 11. A decision problem $\mathcal{Q}$ is said to be in *class NP* if there exists a *non-deterministic* polynomial-time algorithm $\mathcal{A}$ that can correctly guess a proof y for a yes-instance x and accept it; while on a no-instance x' for any guess y returns no.

Clearly, $P \subseteq NP$ since any decision problem that can be solved in polynomial time by an algorithm $\mathcal{A}$ is also included in class NP by treating $\mathcal{A}$ as the non-deterministic algorithm that does not have to guess, but can construct a proof y if it exists. Before we define the concept of NP-completeness, we introduce the notion of reducibility among decision problems.

Definition 12. Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be two decision problems. We say $\mathcal{Q}_1$ is *polynomial-time reducible* to $\mathcal{Q}_2$ (written as $\mathcal{Q}_1 \propto \mathcal{Q}_2$) if there exists a polynomial-time algorithm $\mathcal{A}$ that, given an instance x of $\mathcal{Q}_1$ constructs an instance $\mathcal{A}(x)$ of $\mathcal{Q}_2$ such that: x is a yes-instance of $\mathcal{Q}_1$ *if and only if* $\mathcal{A}(x)$ is a yes-instance of $\mathcal{Q}_2$.

The reduction is also qualified as a many-one reduction as many instances x can give rise to the same instance $\mathcal{A}(x)$. Note that this relationship sets computational

bounds on the problems relative to each other. Up to polynomial time computations, $\mathcal{Q}_1$ is not harder than $\mathcal{Q}_2$ (equivalently $\mathcal{Q}_2$ is not easier than $\mathcal{Q}_1$). Consequently if $\mathcal{Q}_2$ is in class P, then so is $\mathcal{Q}_1$. On the other hand, if $\mathcal{Q}_1$ is intractable, then so is $\mathcal{Q}_2$. A meaningful definition of intractability was made possible by the following result that is considered the cornerstone of NP-completeness theory.

Theorem 1 (Cook's Theorem)**.** *Every decision problem in the class NP is polynomial-time many-one reducible to the* SATISFIABILITY *problem.*

A proof of this theorem and the definition of the SATISFIABILITY problem can be found in [98]. However, for our purposes it suffices to note that no problem in NP is harder than a particular problem. Hence there exists a problem that can be considered as the hardest problem in this class. In fact, many such problems exist and it is formalized by the following definitions.

Definition 13. A decision problem $\mathcal{Q}$ is *NP-hard* if every problem in class NP is polynomial-time many-one reducible to $\mathcal{Q}$. An NP-hard problem $\mathcal{Q}$ is said to be *NP-complete* if $\mathcal{Q}$ is also in class NP.

Note than in optimization literature when the decision version of a problem is NP-complete, the corresponding optimization problem is NP-hard. Clearly, NP-complete problems are the hardest problems in class NP. If any such problem can be solved by a polynomial-time algorithm, then by polynomial-time reducibility, every problem in this class can be solved in polynomial time. However, no such efficient algorithm has been found for an NP-complete problem after decades of research by the most brilliant minds of our time. The working *conjecture* in NP-completeness theory is that $P \neq NP$ and there exist problems in NP for which there are no efficient algorithms. Note that it is still an open problem and no proof establishing that $P \subset NP$ or that

$P = NP$ is known. It is widely believed by computer scientists that the former is true.

Whenever a new optimization problem is encountered, the first meaningful step is to establish its complexity. That is to either develop a polynomial-time algorithm for it, or show that its decision version is NP-complete. To prove NP-completeness of a problem $\mathcal{Q}$, it suffices to show that,

1. $\mathcal{Q} \in NP$;

2. Some known NP-complete problem is polynomially reducible to $\mathcal{Q}$.

A compendium of known NP-complete problems can be found in [98]. For the clique relaxations studied in this dissertation, such results are unknown and we utilize the aforementioned CLIQUE problem as the known NP-complete problem in our complexity results presented in Chapter IV.

II.4. Polyhedral Theory and Combinatorial Optimization

A combinatorial optimization (CO) problem can be simply stated as follows. Given a finite set of objects, select an object that maximizes or minimizes some objective. Clearly, a large number of problems fall into this category. CO problems appear in every field of science and engineering and a wide array of techniques exist to "cope with them". This is because not all CO problems can be solved easily. A majority of them are actually NP-hard which is an indication that they may not be efficiently solvable. Exact combinatorial algorithms, heuristic methods, approximation algorithms, randomized algorithms even continuous global optimization techniques are available for approaching CO problems. In this section, we briefly review the use of polyhedral techniques in CO. Numerous books have been written on CO that emphasize different aspects of the study. A classical and comprehensive reference for

integer programming and its role in CO is [154]. See [169] for a background on linear and integer programming with an emphasis on polyhedral aspects. A good introduction to basic and advanced techniques in CO can be found in [72]. Algorithmic and complexity aspects of CO have been discussed in several excellent texts over the years [136, 160, 130], each adding newer material given the active research in this area. Possibly the most comprehensive study of polyhedral combinatorics is presented in three volumes in [170].

Polyhedral theory enables us to utilize linear programming (LP) techniques to solve CO problems. The LP problem can be stated in its general form as:

$$\max\{c^T x : Ax \leq b\}$$

where column vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times n}$ are given and we are interested in finding x that optimizes the above problem. The feasible region defined by the solution set of a finite system of linear inequalities is called a *polyhedron*.

Definition 14. A set $P \subseteq \mathbb{R}^n$ is called a *polyhedron* if for some matrix $A \in \mathbb{R}^{m \times n}$ and a column vector $b \in \mathbb{R}^m$,

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}.$$

If A, b are rational, then P is said to be a *rational polyhedron*. P is said to be *integral* if its extreme points are integral vectors. If P is bounded, then it is a *polytope*.

Most CO problems can be formulated as integer programs whose feasible solutions are in bijection with the combinatorial objects of interest to us. We will now define a term that we will use often in this dissertation.

Definition 15. Given a graph $G = (V, E)$, an *incidence vector* of a subset of vertices C is a binary vector $x \in \{0, 1\}^{|V|}$ such that $x_i = 1$ if and only if $i \in C$.

An integer programming (IP) problem has a general form:

$$\max\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\}$$

where c, b and A are as before. Let the set of feasible solutions to the above IP be denoted by $Q = \{x \in \mathbb{Z}^n : Ax \leq b\}$ and let $P_I = conv(Q)$ denote its convex hull. A useful theorem in this regard is the following.

Theorem 2 (**see [72]**). *A set P is a polytope if and only if there exists a finite set Q such that P is the convex hull of Q.*

Assuming that we have finite set of feasible solutions to the integer program, P_I is a polytope (similar results also hold for polyhedra, see [169]). If we had the complete linear description of P_I, the CO problem can be solved by using LP approaches. However, finding the complete linear description is not an easy task and such descriptions are known only for a small number of problems. On the other hand, consider the LP relaxation polyhedron P_L of the above integer program. Clearly, $P_L = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_I = \{x \in \mathbb{Z}^n : Ax \leq b\}$. Maximizing over P_L gives an upper-bound on the maximum over P_I which is what we seek. Thus, strengthening the LP relaxation with additional constraints that do not cut off of any feasible integer solutions can help us get closer to our goal. Different algorithmic frameworks have exploited this idea in different ways with remarkable success especially given the computational intractability of such problems. This leads us to the following definition.

Definition 16. Inequality $\alpha^T x \leq \beta$ is *valid* for a polyhedron P if $P \subseteq \{x : \alpha^T x \leq \beta\}$.

Definition 17. A valid inequality $\alpha^T x \leq \beta$ is said to *dominate* a valid inequality $\gamma^T x \leq \theta$ if $\{x : \alpha^T x \leq \beta\} \subseteq \{x : \gamma^T x \leq \theta\}$. If the containment is proper, it is said to *strongly dominate*.

A valid inequality when restricted to be an equation ($\alpha^T x = \beta, \alpha \neq 0$) represents a hyperplane. From an algorithmic perspective, these are also called *cutting planes* or simply, *cuts*.

Definition 18. A valid inequality $\alpha^T x \leq \beta$ for polyhedron P is said to be *supporting* if $P \cap \{x : \alpha^T x \leq \beta\} \neq \emptyset$. The intersection is called a *face* of the polyhedron.

By convention, the empty set and P itself are also considered faces of P. Hence, a face that is neither P nor $\emptyset$ is called a *proper face*. Another characterization of faces of a polyhedron is given by the following theorem, which shows that the number of faces of P is finite.

Theorem 3 (**see [72]**). *A nonempty set $F \subseteq P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is a face of P if and only if for some subsystem $A^o x \leq b^o$ of $Ax \leq b$ we have $F = \{x \in P : A^o x = b^o\}$.*

Definition 19. A maximal proper face of P is called a *facet* of P. An inequality $\alpha^T x \leq \beta$ is said to induce a facet F of P if $F = \{x \in P : \alpha^T x = \beta\}$.

Definition 20. A system of inequalities and equations defining P is *minimal* if no inequality can be turned into an equation without reducing the size of P and no inequality or equation can be omitted without enlarging P.

Facets are important because they correspond to inequalities that are necessary in a minimal definition of a polyhedron P. This is formalized by the following theorem.

Theorem 4 (**see [72]**). *Let $P = \{x \in \mathbb{R}^n : A'x = b', A''x \leq b''\}$ be a nonempty polyhedron. Then the defining system is minimal if and only if the rows of A' are linearly independent and for each row i of A'' the inequality $(a_i'')^T x \leq b_i''$ induces a distinct facet of P.*

But it should be noted that, even if the complete minimal defining system can be obtained, it could be much larger (say exponentially larger) than the size of the

original CO problem. Hence, for a polynomial-time solvable problem, taking a cutting plane approach may not be meaningful. This significant theoretical hurdle was overcome by the application of *ellipsoid method* [128] to CO resulting in the concepts of *separation* and *optimization*. This is a deep and powerful result with vast implications, especially regarding complexity and polyhedra [106]. We present the basic ideas here before concluding this section.

Instead of looking for a complete polyhedral description of the CO problem, viewing it from a cutting plane perspective leads us to pose the following *separation problem* that goes along with an *optimization problem*.

Definition 21. *Separation problem*: Given a rational polytope $P \subseteq \mathbb{R}^n$ and a rational vector $v \in \mathbb{R}^n$, either conclude that v belongs to P or, if not, find an inequality $\alpha^T x \leq \beta$ satisfied by all $x \in P$ but violated by v, $\alpha^T v > \beta$.

Definition 22. *Optimization problem*: Given a rational polytope $P \subseteq \mathbb{R}^n$ and a rational vector $c \in \mathbb{R}^n$, either find $x^* \in P$ that maximizes $c^T x$ over all $x \in P$, or conclude that $P = \emptyset$.

The basic result that guides the search for efficient algorithms using cutting plane approaches is the following theorem. We present here a simpler version of this equivalence result from [72], for more details see [169, 170, 106]. The following result indicates that the "difficulty" in optimization is equivalent to the "difficulty" in separation, not the number of facets of the convex hull of interest. If a polynomial-time algorithm exists for a CO problem, cutting-plane approaches can be used to find one.

Theorem 5 (**see [72]**)**.** *For any proper class of polyhedra, the optimization problem is polynomially solvable if and only if the separation problem is polynomially solvable.*

To reiterate, valid inequalities for P_I that are not valid for P_L help us strengthen the LP relaxation, and the tighter bounds obtained could be utilized in an algorithmic framework to solve the CO problem. Facets of P_I are more important (and often harder to identify) since we know that they need to be present in any minimal defining system of P_I. These two ideas are put to use in an algorithm such as a *branch-and-cut* which is expected to be worst-case exponential if the CO problem is known to be NP-hard. However, this approach has been found to be practically effective in solving a wide variety of problems in literature. We discuss this approach in the next section.

II.5. Branch-and-cut

Many developments have appeared in the areas of IP and CO, especially encouraged by the development of robust software programs capable of tackling large IP problems. Originating from classical *cutting plane algorithms* for integer programs, polyhedral studies have led to the development of exact branch-and-cut algorithms that facilitated resolution of large instances of several CO problems formulated as binary, pure integer and mixed integer programs. We will give a brief description of branch-and-cut approaches and their successful application in exactly solving problems from literature.

Branch-and-cut (BC) methods are popular and effective in optimally solving a wide variety of CO and general IP problems. These methods incorporate cutting planes in solving the LP relaxation at the nodes of a branch-and-bound tree to get tighter bounds. Note that this is different from the popular variant called *cut-and-branch* that adds cutting planes only at the root-node of the branch-and-bound tree and then proceeds with enumeration starting with a stronger root-node LP relaxation. However, this approach has been found to be lacking, especially on hard CO

problems [74].

Algorithm 1 presents a general framework for BC algorithms. Consider the following general mixed integer program (MIP).

$$z^* = \max\{c^T x : Ax \leq b, x \geq 0, x_i \in \mathbb{Z} \ \ \forall i \in \Theta\}$$

Here A, b, c are defined as before and Θ is the index set of variables restricted to be integers. Note that this general form encompasses pure integer and binary formulations as well. The active node list of the search tree is denoted by L and the incumbent objective is denoted by $\underline{z}$ which is also the best known lower-bound on z^*. Upper-bound for each node is denoted by $\bar{z}_t$ which is from the LP relaxation of the original problem with branching constraints and cutting planes added to strengthen the relaxation. An upper-bound on z^* in case of early termination of the algorithm can be calculated as $\max_{MIP^t \in L} \bar{z}_t$. It can be easily seen that the only difference in Algorithm 1 from a classical LP relaxation based branch-and-bound for MIP is the fact that when desired, cuts are added and the subproblems are re-solved in the nodes of the branch-and-bound tree. In addition to the quality of cuts added, the frequency of adding cuts and number of cuts added are two important parameters that greatly influence the performance of a BC implementation. The cuts generated at a node are valid for the subtree rooted at that node, referred to as *local cuts*. Cuts added at any node that are valid for the entire search tree are referred to as *global cuts*. Local cuts are sometimes *lifted* to be made global [154]. It is easy to see that the cuts added increase the size of the system and hence can result in memory problems if left uncontrolled. It is for this reason, the decisions regarding when (at what nodes) to add cuts and how many cuts to add, should be experimentally studied and tuned. Care must also be taken to manage the cut pool– decisions regarding keeping versus dropping cuts from the pool of generated cuts, which cuts from among the pool should

be added if the decision to add is made and which cuts should be lifted. Next we briefly survey some classical and successful applications of BC algorithms to a variety of problems from literature.

BC methods for the *traveling salesman problem* (TSP) are studied in [158, 15]. Polyhedral techniques for the maximum clique problem that uses cutting planes for tightening the LP relaxation are presented in [21]. General cutting planes in a *lift-and-project* [18] framework for mixed 0,1-programs are studied in [19]. Gomory cutting planes are used within a BC framework for mixed 0,1-programs in [20] and lifting techniques are developed to ensure that cuts generated at some node in a BC tree are valid globally. BC methods for integer programs with general integer variables are studied where lifting procedures are developed for *knapsack inequalities* and *Gomory's mixed integer cuts* in [58]. Several other successful applications of these techniques also exist in literature. For more information on BC methods and for other useful references, see [147]. An introduction to theory and computations in CO using polyhedral techniques is presented in [1, 2]. Although BC methods have been successfully applied to solve several hard CO problems, tailoring a BC algorithm to effectively solve a specific problem is a delicate task that requires attention in itself in terms of extensive experimentation and tuning.

II.6. Cliques and Independent Sets

The maximum clique problem is a classical CO problem for several important reasons. Although the early algorithms for finding cliques in graphs were original motivated by social network applications [110], it has been found to be applicable in a variety of fields. We already discussed some of these applications in the context of clique relaxations in Section II.1. Other application areas include coding theory, fault di-

Algorithm 1 General Branch-and-Cut Framework For MIP Problems

procedure BRANCH-AND-CUT(MIP^1) ▷ original MIP instance
  $L \leftarrow \{MIP^1\}, \bar{z}_1 = \infty, \underline{z} = -\infty$ ▷ initialization
  if $L = \emptyset$ **then** ▷ termination
    if $\underline{z} > -\infty$ **return** $(x^*, \underline{z})$ **else return** *infeasible*
  end if
  select $MIP^t \in L$ and delete from L ▷ best-bound/depth/breadth first order
  solve LP relaxation of MIP^t
  if optimum exists **then**
    denote optimum by $(x^t, \bar{z}_t)$
  else
    $\bar{z}_t \leftarrow -\infty$, go to Step 16 ▷ LP relaxation is infeasible
  end if
  if node chosen to add cuts **then**
    identify violated cuts, if they exist add cuts to MIP^t, go to Step 7
  end if
  if $\bar{z}_t \leq \underline{z}$ **then**
    go to Step 3 ▷ node is fathomed by bound or infeasibility
  end if
  if $\bar{z}_t > \underline{z}$ and x^t is integer feasible **then**
    $\underline{z} \leftarrow \bar{z}_t, \quad x^* \leftarrow x^t$
    delete every $MIP^j \in L$ such that $\bar{z}_j \leq \underline{z}$ ▷ fathoming by bound
    go to Step 3 ▷ current node fathomed by feasibility
  end if
  create and add $MIP^{|L|+1}, \ldots, MIP^{|L|+r}$ to L based on the branching rule
  set $\bar{z}_{|L|+1} = \cdots = \bar{z}_{|L|+r} \leftarrow \bar{z}_t$, go to Step 3
end procedure

agnosis models, computer vision and pattern recognition [42]. It has also become a well-known problem due to rich and deep results surrounding the problem in a variety of research areas.

Maximum clique problem is NP-hard [98] and it is hard to approximate within $n^{1-\epsilon}$ for any $\epsilon > 0$ as established by Håstad [113]. The best known polynomial-time approximation algorithm was developed by Bopanna and Halldórsson in [43]. Exact algorithms for the problem include the implicit enumeration algorithm developed by Carraghan and Pardalos [57], the branch-and-bound developed by Balas and Yu [23] and approaches developed in [161, 21, 22, 189, 156]. Several algorithms and heuristics for the maximum clique problem were put to test in the *Second* DIMACS *Implementation Challenge* and their performance on benchmark instances documented in [125]. Several detailed surveys on the problem already exist [42, 162] that cover different formulations, complexity and inapproximability results as well as algorithms and heuristics for the problem on arbitrary and restricted graph classes. Before concluding this chapter we review classical polyhedral results and continuous approaches to the problem. Given the close connection between maximum clique and independent set problems, we review their results together and as presented in the literature.

II.6.1. Polyhedral Results

The *clique polytope* $C(G)$ is the convex hull of incidence vectors of cliques in G. Alternately, $C(G) = conv\{x \in \{0,1\}^{|V|} : x_i + x_j \leq 1 \; \forall \; (i,j) \notin E\}$. The maximum clique problem can then be written as $\omega(G) = \max\{\sum_{i \in V} x_i : x \in C(G)\}$. The closely related *independent set polytope* is $IS(G) = conv\{x \in \{0,1\}^{|V|} : x_i + x_j \leq 1 \; \forall \; (i,j) \in E\}$. In literature, polyhedral properties of $IS(G)$ (and hence $C(G)$) and its LP relaxation have been extensively studied since the 1970s.

Facets and Valid Inequalities. Padberg [157] derived facets of $IS(G)$ from maxi-

mal cliques. Padberg [157] also showed that facets can be derived for $IS(H)$ from an odd hole H and presented lifting procedures to obtain facets of $IS(G)$ from the facets of $IS(H)$ when H is a vertex-induced subgraph of G. Nemhauser and Trotter [152] present systematic lifting procedures for obtaining facets of $IS(G)$ from arbitrary vertex-induced subgraphs of G generalizing the results of Padberg [157]. They also show that similar to odd holes, odd antiholes can be used to generate facets of $IS(G)$. These results are summarized next.

Maximal Clique Facets: Let C denote a maximal clique in G. $\sum_{i \in C} x_i \leq 1$ induces a *facet* of $IS(G)$.

Odd Hole Inequalities: Let H denote an odd hole in G. $\sum_{i \in H} x_i \leq \frac{|H|-1}{2}$ is *valid* for $IS(G)$ and facet defining for $IS(G[H])$.

Odd Antihole Inequalities: Let A denote an odd antihole in G. $\sum_{i \in A} x_i \leq 2$ is *valid* for $IS(G)$ and facet defining for $IS(G[A])$.

Web inequalities were identified by Trotter [182] to produce facets of $IS(G)$ for graphs called *webs* that generalize clique, odd hole and odd antihole inequalities. A web $W(p,q)$ is defined on vertex set $V = \{1, \ldots, p\}$ and edge set $E = \{(i,j) : j = i+q(\mod p), \ldots, i+p-q(\mod p), \forall i \in V\}$. Trotter [182] showed that the *web inequality*, $\sum_{i \in V} x_i \leq q$ is valid for $IS(W(p,q))$ and it is facet inducing *if and only if* p and q are relatively prime, and $q > 1$. $W(2q+1, q)$ is an odd hole and $W(2p+1, 2)$ is an odd antihole there by generalizing the previous results of Padberg [157] and Nemhauser and Trotter [152]. A clique can also be considered as a degenerate web $W(p, 1)$. Trotter [182] also established that the antiwebs $\overline{W}(p,q)$ (complement of webs) produce valid inequalities of the form $\sum_{i \in V} x_i \leq \lfloor \frac{p}{q} \rfloor$ for $IS(\overline{W}(p,q))$.

Wheel inequalities for the independent set polytope are presented in [63] while *antiweb-wheel inequalities* are studied in [64, 65]. In [54], a new class of facet defining graphs called *fans* are introduced and new methods to obtain facets are identified.

An *independence system* $(S, \mathcal{I})$ consists of a finite set of elements S and a non-empty collection $\mathcal{I}$ of subsets of S satisfying the property $I \in \mathcal{I} \Rightarrow I' \in \mathcal{I}, \forall I' \subseteq I$. Numerous CO problems including cliques and independent sets fall into this general category. For independence systems, the notion of cliques, odd holes and odd antiholes are generalized and facets and valid inequalities are identified in [89]. Antiwebs are generalized and studied in the context of independence systems in [135]. Apart from valid inequalities and facets, other interesting properties of $IS(G)$ are also known.

Extreme Points and Adjacency. Nemhauser and Trotter [152] show that the extreme points of the LP relaxation polytope of $IS(G)$ have only $\{0, \frac{1}{2}, 1\}$ components. Nemhauser and Trotter [153] also show that from an optimal solution x^* to the LP relaxation of the maximum independent set problem, if $I = \{i \in V : x_i^* = 1\}$, then I is contained in some maximum independent set of G.

Chvátal [69] characterized adjacent extreme points of $IS(G)$ as follows. The incidence vectors of two independent sets R, S in G are adjacent extreme points of $IS(G)$ *if and only if* $G[S \triangle R]$ is connected, where $S \triangle R = (R \setminus S) \cup (S \setminus R)$.

Perfect Graphs and Stable Set Polytope. The *perfect graph theorem* conjectured by Berge [34] and proved by Lovàsz [138] states that *a graph is perfect if and only if its complement is perfect.* Clearly, odd holes and antiholes are not perfect. In 1960, Berge conjectured (along with his "weak perfect graph conjecture" of the perfect graph theorem) what came to be known as the *strong perfect graph conjecture*: *A graph is perfect if and only if it contains no odd hole and no odd antihole* (see [35] for its history). Graphs containing no odd holes or antiholes are now called *Berge graphs* in honor of Claude Berge. The *strong perfect graph theorem* proved recently, four decades after the conjecture itself by Chudnovsky et al. [67] states that *a graph is perfect if and only if it is Berge.*

Study of the independent set polytope also led to an interesting polyhedral char-

acterization of graph perfection. Combining the results of Lovàsz [138, 137], Fulkerson [94] and Chvátal [69], for a graph G the following are equivalent [78]:

1. G is perfect;
2. The polytope $P(G) = \{x \in \mathbb{R}_+^{|V|} : \sum_{i \in K} x_i \leq 1 \ \forall \text{ maximal cliques } K\}$ is integral;
3. $\bar{G}$ is perfect.

Perfect graphs are precisely those graphs for which the independent set polytope is completely described by non-negativity and maximal clique inequalities. These results establish the close connection between polyhedral combinatorics and the theory of perfect graphs.

II.6.2. Continuous Approaches

The maximum clique and independent set problems have also been addressed using continuous approaches. We will now present some classical results from this active area of research. Let A_G be the adjacency matrix of G and let $\mathbf{1}$ be the n-dimensional vector with all components equal to 1. For a non-empty subset C of vertices, let its *characteristic* vector x^C be defined by $x_i^C = 1/|C|$ if $i \in C$, $x_i^C = 0$, otherwise, for $i = 1, \ldots, n$. The following is a classical result due to Motzkin and Straus [150].

Theorem 6 (**Motzkin-Straus [150]**)**.**

$$1 - \frac{1}{\omega(G)} = \max\{x^T A_G x : \mathbf{1}^T x = 1, x \geq 0\}$$

Moreover, a subset of vertices $C \subseteq V$ is a maximum clique of G if and only if the characteristic vector x^C of C is a global maximizer of the above problem. In its original form, the Motzkin-Straus formulation has local maximizers which are not in

the form of characteristic vectors. Bomze [41] introduced a regularization of Motzkin-Straus formulation by replacing the objective function in Theorem 6 with the function $g(x) = x^T \left(A_G + \frac{1}{2} I_n\right) x$ where I_n is the $n \times n$ identity matrix. In the regularized formulation, x^* is a *local maximum* if and only if it is the characteristic vector of a *maximal clique* in the graph. First-order and second-order optimality conditions for the Motzkin-Straus formulation and its extension to weighted maximum clique problem are presented in Gibbons et al. [100]. Characterization of maximal cliques in terms of local solutions is also provided in [100].

Recall the description of $IS(G)$ from the previous section. The binary and edge constraints can be easily rephrased using non-linear equivalents. Consider the following quadratically constrained global optimization formulation for maximum independent set problem.

$$\alpha(G) = \max\{\mathbf{1}^T x : x_i x_j = 0,\ \forall\ (i,j) \in E,\ \ x_i(x_i - 1) = 0\ \forall\ i \in V\} \tag{2.1}$$

Shor [176] applied dual quadratic estimates and reported good computational results using a weighted version of (2.1). Other global optimization formulations for the independence number $\alpha(G)$ are as follows.

$$\alpha(G) = \max_{0 \neq x \in [0,1]^n} \frac{(\sum\limits_{i \in V} x_i)^2}{\sum\limits_{i \in V} x_i^2 + 2 \sum\limits_{(i,j) \in E} x_i x_j} \tag{2.2}$$

$$\alpha(G) = \max_{x \in [0,1]^n} \sum_{i \in V} x_i \prod_{j \in N(i)} (1 - x_j) \tag{2.3}$$

$$\alpha(G) = \max_{x \in [0,1]^n} \left\{ \sum_{i \in V} \left(x_i + \frac{(1 - x_i) \prod\limits_{j \in N(i)} (1 - x_j)}{1 + \sum\limits_{j \in N(i)} \prod\limits_{l \in N(j) \setminus N[i]} (1 - x_l)} \right) - \sum_{(i,j) \in E} x_i x_j \right\} \tag{2.4}$$

$$\alpha(G) = \max_{x\in[0,1]^n} \left\{ \sum_{i\in V} x_i - \sum_{(i,j)\in E} x_i x_j \right\} \tag{2.5}$$

$$\alpha(G) = \max_{x\in[0,1]^n} \sum_{i\in V} \left(\frac{x_i}{1+\sum\limits_{j\in N(i)} x_j} + \frac{(1-x_i)\prod\limits_{j\in N(i)}(1-x_j)}{1+\sum\limits_{j\in N(i)}\prod\limits_{l\in N(j)\setminus N[i]}(1-x_l)} \right) \tag{2.6}$$

$$\begin{aligned} \alpha(G) &= \max_{x\in[0,1]^n} h(x), \text{ where} \\ h(x) &= \sum_{i\in V} \left(x_i + \frac{1-x_i}{1+\sum\limits_{j\in N(i)}\prod\limits_{l\in N(j)\setminus N[i]}(1-x_l)} \right) \prod_{j\in N(i)}(1-x_j) \\ &+ \sum_{i\in V'} \frac{x_i(1-\prod\limits_{j\in N(i)}(1-x_j))^2}{1-\prod\limits_{j\in N(i)}(1-x_j)+\sum\limits_{j\in N(i)} x_j} \\ \text{and } V' &= \{i\in V : \sum_{j\in N(i)} x_j > 0\}. \end{aligned} \tag{2.7}$$

Formulation (2.2) can be obtained from the Motzkin-Straus theorem (see [108]). Statements (2.3) and (2.4) were proved using probabilistic methods in [109] and [108] respectively. Formulation (2.5) is clearly implied by (2.4), but this weaker version was algorithmically realizable and used in the development of heuristic for the independence number in [108]. Formulations (2.3) and (2.5) were also proved deterministically in [4] and heuristics were developed based on them. Formulation (2.6) was also established in [108] using a probabilistic approach. In this dissertation we develop a deterministic proof of a weaker version of (2.6) and characterize its local maxima in Chapter VIII. Statement (2.7) is provably stronger than formulations (2.6) and (2.4). However, it is complicated and difficult to realize as the set V' is also variable.

Apart from providing an interesting alternate approach to classical CO problems, the standard quadratic programming formulation due to Motzkin and Straus [150],

continuous formulation due to Gibbons et al. [99] and formulations due to Harant [107, 108, 109] have led to the development of effective heuristics for maximum clique and maximum independent set problems [41, 4, 50, 49, 13].

CHAPTER III

CLIQUE RELAXATIONS

In this chapter, we introduce the distance-based, diameter-based and degree-based relaxations of a clique and define appropriate optimization problems associated with the models. It should be noted that these models were originally introduced in SNA literature. However, certain drawbacks existed in the original definition of the distance-based and diameter-based models. We address these difficulties and redefine the problems to be mathematically more meaningful.

This chapter is organized as follows. In Section III.1, the distance-based and diameter-based models are introduced. Their original definitions from the literature and the new definitions are presented. Section III.2 defines the *k-plex* model which is the focus of this dissertation. In Section III.3, the models are compared based on structural properties they guarantee, which serves to motivate our emphasis on the k-plex model. In the following sections defining the parameterized models, the parameter k is assumed to be a positive integer.

III.1. Distance-Based and Diameter-Based Relaxations

Luce [140] defines a *k-clique* of G as a subset of vertices $C \subseteq V$ such that for all $u, v \in C,\ d_G(u, v) \leq k$ and this subset is maximal by inclusion. In other words, a k-clique C is a set of vertices in which any two vertices are a distance of at most k from each other in G, and there is no vertex outside C that is distance k or less from *every* vertex in C. Thus, if two vertices $u, v \in V$ belong to a k-clique C, then $d_G(u, v) \leq k$, however this does not imply that $d_{G[C]}(u, v) \leq k$. For example, Fig. 3 shows a graph in which the subset of vertices $C_1 = \{1, 2, 3, 4, 5\}$ forms a 2-clique, however the distance

between vertices 1 and 5 in the subgraph induced by C_1 is 3. Hence, the concept of k-clique lacks the requirement of "cohesion" in the subgroup consisting of vertices in the k-clique, while such a requirement is essential to applications in social networks. This observation motivated Alba [8] to introduce the concept of a "sociometric clique", which was later renamed to "k-clan" by Mokken [148]. A k-clique C is called a *k-clan* if the diameter of the induced subgraph $G[C]$ is no more than k. Finally, Mokken [148] defines a *k-club* to be an inclusionwise maximal subset of vertices, $D \subseteq V$ such that the diameter of the induced subgraph $G[D]$ is at most k. To highlight the differences between the three structures, we turn to the graph in Fig. 3. In this graph, the 2-cliques are given by $C_1 = \{1, 2, 3, 4, 5\}$ and $C_2 = \{1, 2, 4, 5, 6\}$. It is easy to see that C_1 is not a 2-clan or 2-club, since the diameter of induced subgraph $G[C_1]$ is 3. Since any k-clan is a k-clique, the only 2-clan in this graph is given by C_2. Lastly, the 2-clubs of this graph are $D_1 = \{1, 2, 3, 4\}$, $D_2 = \{2, 3, 4, 5\}$ and $D_3 = C_2$. A study of relations between cliques, clans and clubs in a graph can be found in [148].

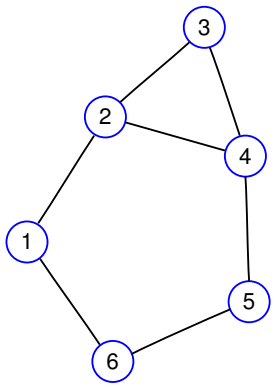


Fig. 3 2-clique Vs. 2-club

Even though the concepts just defined are used quite extensively in social networks analysis and are even covered in standard textbooks (see for instance [185]), their definitions have some deficiencies from a mathematical viewpoint. One consid-

erable drawback of the k-clan definition is that for some graphs a k-clan may not exist. This point is illustrated in Fig. 4, which shows a graph with two 2-cliques $\{1, 2, 3, 4, 5, 6, 7\}$ and $\{1, 2, 3, 5, 6, 7, 8\}$, neither of which is a 2-clan.

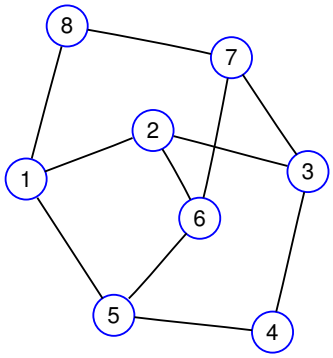


Fig. 4 A graph with no 2-clans

Some other difficulties arise from the requirement of maximality in all three definitions. In particular, this requirement makes checking whether a given subset of vertices is a k-club, a nontrivial matter. Indeed, to check that a set C is a k-clique, we need to check if the pairwise distance between vertices in C is at most k and it is maximal. For verifying maximality, it suffices to show that there is no vertex outside C that could be added to C without violating the pairwise distances condition. A similar criterion would not work for k-clubs. Although it is easy to verify whether the subgraph induced by the given subset D has diameter at most k, verifying its maximality is not straightforward. In this case, maximality by inclusion is *not equivalent* to nonexistence of one vertex that could increase the size of the k-club. As an example, consider the graph in Fig. 3. For the subset of vertices $D = \{1, 5, 6\}$ of this graph, the subgraph induced by D has diameter 2. Adding any one of the vertices 2, 3 or 4 to D would increase the diameter of the induced subgraph, however if both 2 and 4 are added, the diameter of the resulting induced subgraph is

still 2.

Taking into account that the above definitions of 1-clique, 1-clan and 1-club all correspond to the standard definition of a *maximal* clique, we propose to modify the definitions of k-clique and k-club as follows.

Definition 23. A *k-clique* is a subset of vertices C such that for every $i, j \in C$, $d_G(i, j) \leq k$.

Definition 24. A *k-club* is a subset of vertices D such that $diam(G[D]) \leq k$.

A similar definition of k-clan becomes redundant. The example in Fig. 4 suggests the impracticality of such a concept, so we do not consider k-clans any further. By a maximal k-clique (k-club) we will mean a k-clique (k-club) that is not a subset of a larger k-clique (k-club). Finally, a maximum k-clique (k-club) is a k-clique (k-club) of the largest size in the graph.

These definitions retain the basic notions of relaxing pairwise distances in the original graph (k-clique) and in the induced subgraph (k-club), exclude maximality from the basic definition which is consistent with common practice and separates the complications involved in testing maximality from testing if the basic definition is satisfied, especially for k-clubs.

From the definitions and the above example (set C_1 in Fig. 3) it follows that any k-club in G is also a k-clique, but the converse is not true. Note that a shortest path between two vertices in a k-clique C may include vertices outside C. So there can exist two vertices $u, v \in C$ such that $d_G(u, v) \leq k$, but $d_{G[C]}(u, v) > k$ and hence C need not be a k-club. Observe that both these models reduce to a clique when $k = 1$ and are relaxations for $k > 1$. The k-clique model allows for the pairwise distance to be more than one while the k-club model allows the same, with the restriction that some path of length at most k exist, that only uses vertices in the k-club.

Given a graph $G = (V, E)$ and a positive integer k, the **maximum k-clique problem** is to find a largest k-clique in G. We denote the *k-clique number* of graph G, which is the cardinality of a maximum k-clique of G, by $\tilde{\omega}_k(G)$. The **maximum k-club problem** is defined likewise, with $\bar{\omega}_k(G)$ denoting the *k-club number* of G.

III.2. Degree-Based Relaxation

The degree-based relaxation called *k-plex* was introduced by Seidman and Foster [173]. We present here the original definition, paraphrased to be stylistically consistent.

Definition 25. A subset of vertices S is said to be a *k-plex* if the minimum degree in the induced subgraph $\delta(G[S]) \geq |S| - k$.

In other words, for all $v \in S$, $deg_{G[S]}(v) = |N(v) \cap S| \geq |S| - k$ if S is a k-plex. Note that the lower-bound on the induced degree varies with S. A maximal k-plex is one that is not contained in a larger k-plex.

This model also reduces to the clique when $k = 1$ and is a relaxation of the clique requirement for all $k > 1$, allowing for at most $k - 1$ non-neighbors inside the set. Fig. 5 illustrates this concept: The set $\{1, 2, 3, 4\}$ is a 1-plex (clique), sets $\{1, 2, 3, 4, 5\}$ and $\{1, 2, 3, 4, 6\}$ are 2-plexes (maximal and maximum) and the entire graph is a 3-plex. We propose the following definition of a complementary structure for k-plex.

Definition 26. A subset of vertices S is a *co-k-plex* if the maximum degree in the induced subgraph $\Delta(G[S]) \leq k - 1$.

In other words, for a co-k-plex S, $|N(i) \cap S| \leq k - 1$ for all $i \in S$. A maximal co-k-plex is one that is not a proper subset of any co-k-plex in G.

Remark 1. Note that S is a co-k-plex in G if and only if S is a k-plex in the complement graph $\bar{G}$.

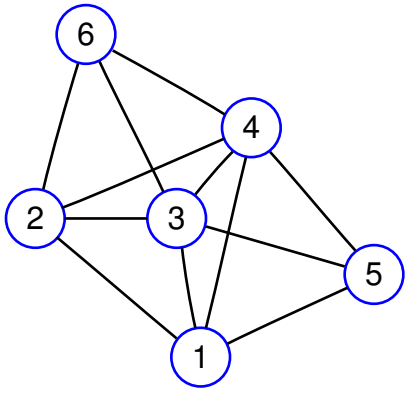


Fig. 5 Illustration of k-plexes for $k = 1, 2, 3$

Given a graph $G = (V, E)$ and a positive integer k, the **maximum k-plex problem** is defined as the problem of finding a largest k-plex in G. The *k-plex number* of G is the cardinality of a maximum k-plex of G denoted by $\omega_k(G)$.

III.3. Comparison of the Models

In Chapter II, we briefly referred to three properties that we require in a cohesive subgroup model: *familiarity, reachability and robustness*. These translate to degree, distance/diameter and connectivity in terms of graph properties and are reasonable measures of cohesiveness of a subgroup. We will now evaluate the clique relaxation models based on these properties.

By definition, k-cliques permit members on the shortest path between two vertices to be outside the set itself. The 2-clique $\{1, 2, 3, 4, 5\}$ in Fig. 3 is an example of this. In a social network, it may be unreasonable to expect a cohesive subgroup to require outside members.

The concept of k-club overcomes the weakness common to k-cliques by bounding the diameter of the induced subgraph. However, k-clubs have certain drawbacks that we illustrate using a simple example involving 2-clubs. It is possible that there exists

one vertex in a 2-club that is adjacent to all other vertices, making it a 2-club, but these neighbors are poorly connected among themselves. This is demonstrated by a star graph $K_{1,n-1}$ which has diameter two as the central vertex is adjacent to all other vertices, but the neighbors of the central vertex have no edges between them. Although k-clique and k-club models ensure reachability, they may lack cohesiveness in terms of degree and connectivity. In particular, removal of just one central vertex in a star graph completely disconnects the graph.

Clearly, every k-clique (k-club) is also a $k+1$-clique ($k+1$-club). Every subset of a k-clique is a k-clique. However, the k-club property is not *hereditary* in the sense that every subset of a k-club need not be a k-club. A star graph is a 2-club but subsets obtained by excluding the central vertex have diameter infinity. In Fig. 3, the set $\{1,2,4,5,6\}$ is a 2-club but $\{1,2,4,5\}$ is not. We pointed out in Section III.1 that verifying maximality of k-clubs is nontrivial. This is also an outcome of the absence of hereditary nature in k-clubs.

Since all three models relax cliques, it is natural to look for a complementary model that relaxes independent sets. For the distance-based models k-clique and k-club, a meaningful complementary definition is unlikely for $k \geq 2$. By edge complementing, we lose control over distances in the graph. Even if G is known to be connected or disconnected, we cannot guarantee that about $\bar{G}$. We provide simple examples to explain this fact. A star graph is a 2-club (and a 2-clique). Its complement however is the union of an isolated vertex and a clique. While the clique component is highly connected, the graph itself has diameter infinity. On the other hand, cycle C_5 is a 2-club whose complement is also isomorphic to C_5 and hence has the same properties. Fortunately, the k-plex relaxation is more systematic in the sense that it permits such a complementary definition.

In fact, the following theorem summarizing the results from the introductory

paper by Seidman and Foster [173] establish that, for low values of k the k-plex model is better than the k-clique and k-club models based on the characteristics that are required of a cohesive subgroup– familiarity, reachability and robustness. For the sake of completeness, we also present a sketch of the original proofs from [173].

Theorem 7 ([**173**])**.** *Let $G = (V, E)$ be a k-plex on n vertices.*

1. *Every induced subgraph of G is a k-plex.*
2. *Any k vertices in V forms a dominating set of G.*
3. *If $k < \frac{n+2}{2}$, then $diam(G) \leq 2$.*
4. *$\kappa(G) \geq n - 2k + 2$.*

Proof.

1. Every induced subgraph of G can be obtained by deleting vertices, one at a time. If G is a k-plex, $\delta(G) \geq n - k$. In G' obtained by deleting some vertex from G, the degree of all the vertices and hence the minimum degree, can drop by at most 1. Hence, the new graph continues to be a k-plex as $\delta(G') \geq (n-1) - k$.
2. Suppose there exist k vertices that do not form a dominating set, then there must be some vertex v distinct from these k vertices that is not adjacent to any of these k vertices. This is not possible when G is a k-plex. Furthermore, Seidman and Foster [173] also prove the converse by showing that if every k vertices form a dominating set in an arbitrary graph G, then G is a k-plex.
3. For any vertex v in the k-plex G, the number of vertices in the closed neighborhood $|N[v]| \geq n - k + 1$. Thus for two arbitrary vertices u, v we have $|N[u]| + |N[v]| \geq 2n - 2k + 2 > n$ by the given condition. Hence $N[u]$ and $N[v]$ are not disjoint. Thus for all u, v, $d_G(u, v) \leq 2 \Rightarrow diam(G) \leq 2$.

4. Suppose we delete t vertices from G. The resulting graph G' is still a k-plex and the above argument still holds. If $k < \frac{n-t+2}{2}$ then $diam(G') \leq 2$. Thus, G' is still connected if $t < n - 2k + 2 \Rightarrow \kappa(G) \geq n - 2k + 2$. □

Remark 2. Note that the condition for a diameter-two bound is sharp. Consider the graph $K_{k-1} \cup K_{k-1}$ which is a k-plex for every k. Thus a k-plex could be disconnected even for $k = \frac{n+2}{2}$. However, no isolated vertex could be present in a k-plex of size $k + 1$ or more.

Remark 3. It is also useful to note that in an arbitrary graph G, any k-element subset of vertices is a k-plex and any k-plex is also a $k + 1$-plex.

By definition, members of a k-plex S can have at most $k-1$ non-neighbors inside S. Thus, k-plexes with low k values ($k = 2, 3$) provide good relaxations of clique that closely resemble the cohesive subgroups that can be found in real-life networks that are often erroneous and incomplete. Recall the discussion in Section II.1 regarding the sensitivity of cliques to false-negatives in the data. Note that k-plexes for low k values provide a good balance by protecting against false-positives in the data, and at the same time not become overly sensitive to false-negatives in the data. In addition, the above results indicate that a k-plex also retains the properties of a clique such as low diameter and high connectivity, for low values of k.

The k-plex model overcomes the disadvantages of k-cliques and k-clubs by directly limiting the number of *non-neighbors* inside the cohesive subgroup. This structure imposes a degree bound that varies with the size of the group and hence ensures a cohesive subgroup even as the size of the group varies. Implicitly, it also achieves reachability and robustness. By allowing some strangers in a social group, k-plex provides a more realistic alternative to model cohesive subgroups in a social network. Furthermore, k-plex is closely related to its complementary model co-k-plex. In par-

ticular, 1-plex is a clique and a co-1-plex is an independent set. Thus, k-plexes and co-k-plexes provide a systematic way to generalize two important graph models. For these reasons, we will pay more attention to the k-plex model in this dissertation.

III.4. Existing Approaches

Before concluding this chapter, we point out some of the approaches from literature that generalize cliques. The discussion in the previous section comparing the clique relaxations from SNA has already brought to our attention the structural properties that need to be studied while evaluating a clique relaxation. It must now be apparent to the reader that any model relaxing a clique, while relaxing a particular aspect must also guarantee several important properties. We will review some of the existing approaches in this context.

Consider a simple, undirected, edge-weighted graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}^+$. Define the weight of a subset of edges E' as $w(E') = \sum_{e \in E'} w(e)$. Given a positive integer $k < n$, the *heaviest k-subgraph problem* (HkS) is to find a subgraph $G' = (V', E')$ such that $|V'| = k$ and $w(E')$ is a maximum. When the edge weights are unity, it is referred to as the *heaviest unweighted k-subgraph problem* (HUkS). In literature, HkS is also known as *densest k-subgraph problem* while HUkS is called the *maximum edge subgraph problem*. A different but closely related problem is the *densest subgraph problem* (DS) which is to choose a subgraph $G' = (V', E')$ such that density, defined as $\frac{w(E')}{|V'|}$ is maximized. This problem can be solved in polynomial time using maximum flow algorithms [96]. On the contrary, HkS and HUkS are clearly NP-complete by reduction from Clique [77]. Hence, approximation algorithms for the problems have been developed [131, 16, 90]. But for our purposes of relaxing a clique, the HUkS and DS models have obvious drawbacks. In the DS model, there

is no guarantee on the size of resulting V' as our objective is to maximize density as defined above. In the HkS/HUkS models, the resulting G' can be disconnected as well as include vertices of low degree. Furthermore, the size of the resulting set is fixed at k. However, these models have been used effectively in facility location [166].

The concept of a *quasi-clique* is used in defining a clique relaxation used in data-mining massive call graphs [5, 7] (see Section II.1 for definitions). A graph $G = (V, E)$ is said to be γ-*dense* if $|E| \geq \gamma\binom{|V|}{2}$. A γ-clique (quasi-clique) S is a subset of vertices S such that $G[S]$ is connected and γ-dense. A γ-clique represents a clique when $\gamma = 1$ and is a relaxation when $0 \leq \gamma < 1$. The maximum γ-clique problem can then be defined as the problem of finding a largest γ-clique. In this case, although we find a largest connected subgraph with density no smaller than a fixed number γ, we are still not guaranteed connectivity or degree. This is because unless γ is sufficiently high, a large clique sharing one vertex with a path could also meet the constraints, but it can be disconnected easily and the vertices on the path have very low degree. However high γ values could result in smaller γ-cliques. Furthermore as γ is reduced, a sudden jump in the size of the γ-clique in power law graphs would be observed when the giant component becomes "dense enough". While using this approach, it is recommended that an appropriate choice of γ be made by proper tuning.

Remark 4. Note that this "jump" would also be observed while using k-cliques and k-clubs due to the small world phenomenon observed in power law graphs. The increase in k-plex number with k is more controlled in power law graphs until the condition for diameter-2 is violated by the maximum k-plex found. In this context the following relationship should be taken into account,

$$\omega(G) \leq \omega_k(G) \leq \bar{\omega}_k(G) \leq \tilde{\omega}_k(G)$$

whenever $\omega_k(G) > 2k - 2$. This is true since each is equal to $\omega(G)$ when $k = 1$ and

$\bar{\omega}_k(G) \leq \tilde{\omega}_k(G)$ holds always by definition. Whenever $\omega_k(G) > 2k-2$, the maximum k-plex is also a 2-club as stated in Theorem 7.

More recently, a related model was proposed for relaxing independent sets in [175] where an upper-bound is placed on the number edges in the induced subgraph. A generalized vertex packing GVP-k is a subset of vertices I such that there are at most k edges in the induced subgraph $G[I]$. When $k = 0$, it represents an independent set and is a relaxation for $k \geq 1$. This approach, used to model problems in air-traffic control and national airspace planning is studied using polyhedral methods in [175]. This model is close to the co-k-plex model proposed in III.2 for relaxing independent sets. Note that every GVP-k is a co-k+1-plex but not vice versa.

Clustering problem on gene co-expression networks is studied in [66]. The authors here, also point out some of the issues that were raised in Section II.1 regarding the restrictive and impractical nature of cliques when studying erroneous data. A model robust enough to deal with this situation is necessary and k-plex is ideal for such purposes. In fact, the authors of [66] propose the use of *paracliques* that are close to the 2-plex model. Starting with a set P initialized to some maximum clique C, the authors find new vertices that have at least g neighbors in P. These new vertices are then added to P and the process is repeated until P can no longer be enlarged. The authors report successful results with $g = |C| - 1$ where g is called the *glom factor*. However, the final P that results need not be a 2-plex since each vertex is allowed at most one non-neighbor in a 2-plex. For the paraclique P in every iteration, it is only ensured that new vertices have at least $|C| - 1$ neighbors in the current set and hence could have more than one non-neighbor in the final P that is produced by the algorithm. However, the notion of allowing one non-neighbor is clearly in the same spirit as a 2-plex approach.

The P that is produced by the algorithm is actually a "g-core", an induced

subgraph with minimum degree g. Although the approach of finding a maximum g-core is only of limited use (as we will see in Chapter IX), the approach here is effective because g is chosen to be $\omega(G)-1$ and we start building around a maximum clique. Maximum k-plex for $k = 2, 3$ as well clustering using 2 or 3-plexes can provide an interesting alternative to paracliques.

It should be noted that our discussion pointing out drawbacks in the existing models for relaxing cliques including k-cliques and k-clubs is not meant to undermine their usefulness. In fact, all these approaches have been applied successfully on real-life data. In applications where reachability is the only consideration, k-cliques and k-clubs models are obviously the most appropriate. In this section, we have only tried to emphasize the fact that not all clique relaxations are "created equal" and one must carefully evaluate the guarantees provided in the context of the application under consideration. The best model for cohesive subgroups is that model which mines the most useful information out of the given data. Often evaluating what is "most useful information" depends on the application and there is usually never a common consensus even among area experts on this issue. Our recommendation is that whenever sound structural guarantees as provided by the clique definition are necessary, k-plex model should be considered as a meaningful alternative.

CHAPTER IV

COMPUTATIONAL COMPLEXITY*

In Chapter II, we introduced the concepts of computational complexity and their usefulness in studying the tractability of CO problems. The optimization problems defined in Chapter III lead to several interesting questions regarding their tractability. Clearly for *arbitrary* k, the problems are at least as hard as the maximum clique problem since any algorithm that can solve the optimization problems for arbitrary k can solve the maximum clique problem. But allowing k to be arbitrary does not convey anything specific about the complexity of these problems. In other words: What is the complexity of solving maximum 2-clique, 2-club and 2-plex problems? This is not answered by our observation that the problems are hard to solve when k is "arbitrary". Furthermore, when $k = n$ or $n-1$ all three optimization problems are solved easily. For example, when $k = n$ or $n-1$, the maximum k-clique and k-club problems reduce to finding the largest connected component in the given graph, which is solvable in polynomial time. On the other hand the maximum k-plex problem is trivially solved when $k = n$ since the graph is itself a k-plex. Similarly when $k = n-1$, the trivial answer to the maximum k-plex problem is the graph itself, minus an isolated vertex if it exists. There seems to be a trend in the tractability of the problem as k goes from 1 to n, becoming easier towards the end.

We need to focus on two aspects with regards to complexity, in order to obtain meaningful results. Firstly, the tractability of the problems when k is a fixed positive integer and secondly, how the transition in complexity occurs when k is neither arbi-

*Parts of this chapter are reprinted with permission from Balasundaram, B., Butenko, S., Trukhanov, S.: Novel approaches for analyzing biological networks. Journal of Combinatorial Optimization **10**(1), 23–39 (2005) © Springer.

trary nor fixed, but viewed in relation to a meaningful graph parameter. In the case of k-cliques and k-clubs, we consider k in relation to the diameter of the instance and in the case of k-plex, we consider k in relation to minimum degree in the graph.

This chapter establishes NP-completeness results that show the problems are hard for every fixed positive integer k. Furthermore, transition results on restricted graph classes are also established. Recall that in order to prove NP-completeness of a problem $\mathcal{P}$, we need to provide a polynomial-time reduction from a known NP-complete problem to $\mathcal{P}$ and show that $\mathcal{P}$ is in NP. The reductions for our problems are from the decision version of the maximum clique problem stated as follows.

Clique : Given a graph $G = (V, E)$ and a positive integer c, does there exist a clique of size $\geq c$ in G?

IV.1. Complexity of k-Clique and k-Club

Before stating the complexity results, we introduce the decision version of each problem. For a fixed positive integer k, the k-Clique (k-Club) problem is defined as follows: Given a graph $G = (V, E)$ and a positive integer c, does there exist a k-clique (k-club) of size $\geq c$ in G?

Theorem 8. *The k-Clique and k-Club problems are NP-complete for any fixed positive integer k.*

Proof. Note that for $k = 1$ both problems coincide with Clique problem, which is a well-known NP-complete problem. So, we consider $k > 1$. Given a "yes" instance of k-Clique (k-Club), any k-clique (k-club) of size $\geq c$ can be used as a certificate to verify that this is indeed a "yes" instance in polynomial time. Thus, k-Clique and k-Club are in NP. To complete the proof, we reduce Clique, which is a well known NP-complete problem [98], to k-Clique (k-Club). Let $G = (V, E)$ be an instance of

CLIQUE which we assume does not contain isolated vertices, as no isolated vertex can be included in any clique of size two or more. We construct a corresponding instance of k-CLIQUE (k-CLUB) which is a $(\lfloor k/2 \rfloor + 2)$-partite graph $G' = (V', E')$. We define the vertex set as a union of $\lfloor k/2 \rfloor$ copies of V, a copy of E and one more auxiliary vertex 0:

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

where $V^{(i)} = \{1^{(i)}, 2^{(i)}, \ldots, |V|^{(i)}\}$ is the i-th copy of V, $i = 1, \ldots, \lfloor k/2 \rfloor$. For any $v \in V$, by $v^{(i)} \in V^{(i)}$ we denote the i-th copy of v. The edge set connects copies of the same vertex in $V^{(i)}$ and $V^{(i+1)}$, $i = 1, \ldots, \lfloor k/2 \rfloor - 1$. A vertex $v^{\lfloor k/2 \rfloor}$ in $V^{\lfloor k/2 \rfloor}$ is connected to a vertex $e \in E$ if v is an endpoint of e in G. Finally, all vertices from E in G' are connected to 0. To summarize,

$$\begin{aligned} E' = & \bigcup_{i=1}^{\lfloor k/2 \rfloor - 1} \{(v^{(i)}, v^{(i+1)}) : v \in V\} \\ & \bigcup \{(v^{(\lfloor k/2 \rfloor)}, e) : v \in V, v \text{ is an endpoint of } e \text{ in } G\} \\ & \bigcup \{(e, 0) : e \in E\}. \end{aligned}$$

Fig. 6 shows graph G' corresponding to graph G from Fig. 3 for $k = 5$. Graph G' contains $\lfloor k/2 \rfloor |V| + |E| + 1$ vertices and can obviously be constructed in time polynomial with respect to the size of G.

Our reduction is based on the observation that G has a clique of size c if and only if G' has an k-clique (k-club) of size $c + (\lfloor k/2 \rfloor - 1)|V| + |E| + 1$. Note that G' is connected and $diam(G') \leq 2(\lfloor k/2 \rfloor) + 2$. Indeed, in G', all vertices of $V' \setminus V^{(1)}$ can be included in any k-clique (k-club). Two vertices $u^{(1)}, v^{(1)} \in V^{(1)}$ belong to the same k-clique (k-club) in G' if and only if $(u, v) \in E$ in G. Thus, k-CLIQUE and k-CLUB are NP-complete problems for any fixed positive integer k. □

Recall from Chapter I, the *small world phenomenon* that is observed in power

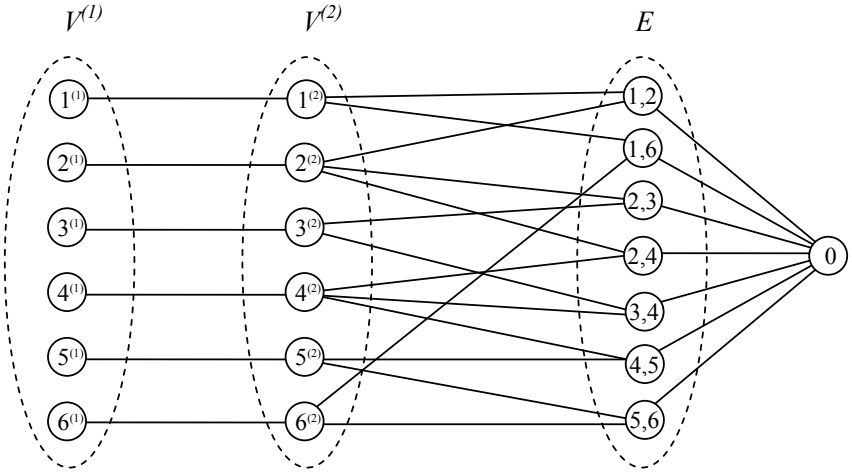


Fig. 6 An illustration to the proof of Theorem 8 for $k = 5$

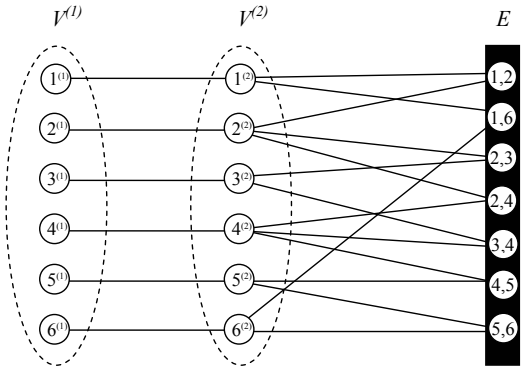


Fig. 7 An illustration to the proof of Theorem 9 for $k = 4$

law graphs. Therefore, the clustering problems on graphs of small diameter are of particular interest. This motivates us to consider the k-Clique and k-Club problems on graphs of fixed diameter. Note that if $diam(G) \leq k$ then both the maximum k-clique problem and the maximum k-club problem are trivial as G is the maximum k-clique (k-club), therefore we are only interested in the case where $diam(G) > k$. For fixed positive integers k, d and $d > k$, we define the k-Clique(d) (k-Club(d)) problem as follows: Given a graph G of diameter d and a positive integer c, does there exist a k-clique (k-club) of size $\geq c$ in G?

Theorem 9. *For any fixed positive integer k, d and $d > k$, the k-*Clique*(d) and k-*Club*(d) problems are NP-complete.*

Proof. Obviously both considered problems are in NP. To complete the proof we reduce Clique to k-Clique(d) and k-Club(d). We first prove the statement for $k = 1$. Given $G = (V, E)$ with no isolated vertices and $d > 1$, we construct a graph $\hat{G} = (\hat{V}, \hat{E})$ of diameter d as follows.

$$\begin{aligned} \hat{V} &= V \cup \{u_i : i = 1, \ldots, d\}; \\ \hat{E} &= E \cup \{(v, u_1) : v \in V\} \cup \{(u_i, u_{i+1}) : i = 1, \ldots, d-1\}. \end{aligned}$$

Then G has a clique of size c if and only if $\hat{G}$ has a clique of size $c + 1$ and the proof is complete for $k = 1$.

If $k > 1$, we consider two cases, for odd and even k. If k is odd, then we use the same construction of graph G' as in the proof of Theorem 8 to reduce Clique to k-Clique(d) and k-Club(d). This is true since $diam(G') \leq k + 1 \leq d$ when k is odd and G has a clique of size c if and only if G' has an k-clique (k-club) of size $c + (\frac{k-1}{2} - 1)|V| + |E| + 1$. If k is even, a similar construction can be used (see Fig. 7) to prove the reduction. As before, we use $k/2$ copies of V and a copy of E for the vertex set of the constructed graph $G'' = (V'', E'')$.

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

The edge set E'' is also similar to E' in the previous construction, but instead of connecting vertices from the copy of E to an auxiliary vertex, we make the subset of

vertices corresponding to E, a clique.

$$\begin{aligned} E'' = & \bigcup_{i=1}^{k/2-1} \{(v^{(i)}, v^{(i+1)}) : v \in V\} \\ & \bigcup \{(v^{(k/2)}, e) : v \in V, v \text{ is an endpoint of } e \text{ in } G\} \\ & \bigcup \{(e_1, e_2) : e_1, e_2 \in E, e_1 \neq e_2\}. \end{aligned}$$

Once again, $diam(G'') \leq k + 1 \leq d$ and G has a clique of size c if and only if G'' has an k-clique (k-club) of size $c + (k/2 - 1)|V| + |E|$. This completes the proof of NP-completeness on fixed diameter graphs. □

These complexity results illustrate two important facts. Firstly, these generalizations are hard to solve not only because they generalize cliques, but because they are hard in their own respect (NP-complete for every fixed positive integer k). Secondly, the transition in complexity is also sudden, while the problems are easily solved under trivial circumstances when diameter is at most k, but immediately become NP-complete on graphs of diameter of the graph is strictly larger than k.

IV.2. Complexity of *k*-Plex

This section presents computational complexity results for the maximum k-plex problem. The results are similar in nature to the ones obtained for the k-Clique and k-Club problems. For a fixed positive integer k, the decision version k-Plex can be stated as follows: Given a graph $G = (V, E)$ and a positive integer c, does there exist a k-plex of size c in G?

Theorem 10. *k-Plex is NP-complete for any fixed positive integer k.*

Proof. We again prove this by reducing Clique to k-Plex. The problem is clearly in NP as a given k-plex of size c or more can be verified in polynomial-time. Given an instance $G = (V, E)$ of Clique, we construct an instance $G' = (V', E')$ in polynomial

time such that G has a clique of size c *if and only if* G' has a k-plex of size c'. To construct G', we expand G by adding $k-1$ copies of the complete graph of order $n = |V|$. Denote the vertex set of the r^{th} such copy by $V_r,\ r = 1, \ldots, k-1$, where $V_r = \{1_r, \ldots, n_r\}$, and let $R = \bigcup_{r=1}^{k-1} V_r$. Put $V' = V \cup R$ and $E' = E \cup \hat{E} \cup \tilde{E}$, where

$$\hat{E} = \{(i, j_r) : i \in V, j_r \in V_r, i \neq j, r = 1, \ldots, k-1\}$$

and

$$\tilde{E} = \{(i_p, j_r) : i_p \in V_p, j_r \in V_r, i \neq j, p, r = 1, \ldots, k-1\}.$$

The set $\hat{E}$ represents the edges between V and R, where every vertex $u \in V$ is connected to every vertex in every complete graph except its copies, *i.e.* u is adjacent to every vertex in $R \setminus \{u_1, \ldots, u_{k-1}\}$. The set $\tilde{E}$ includes the cross edges between distinct V_p and V_r, as well as all possible edges between vertices in V_p, $p = 1, \ldots, k-1$. In other words, every vertex $u_p \in V_p, p = 1, \ldots, k-1$ is adjacent to all the vertices in $V_r \setminus \{u_r\},\ r = 1, \ldots, k-1$. Setting $c' = c + (k-1)n$ completes the reduction. Note that the instance $G' = (V', E')$ can be constructed in polynomial time. Fig. 8 illustrates this transformation for $k = 3$ when G is a path on three vertices. Original graph $G = (\{a, b, c\}, \{(a, b), (a, c)\})$ is shown in box-vertices and bold edges. $\hat{E}$ is denoted by regular solid edges while $\tilde{E}$ is denoted by dashed edges.

We now show that if there exists a clique of size c in G then G' has a k-plex of size c'. Let $C \subseteq V$ induce a clique of size $c = |C|$ in G. We claim that the set $S = C \cup R$, where $|S| = c + n(k-1) = c'$, is a k-plex. For any $u \in C$, there exist $c-1$ neighbors inside C, and $(n-1)(k-1)$ neighbors in R. Thus, for $u \in C,\ deg_{G[S]}(u) = c - 1 + (n-1)(k-1) = c' - k$. For any $v_r \in R$, there exist $(n-1)(k-1)$ neighbors in R and c neighbors in C if $v \notin C$, and $c-1$ neighbors in C if $v \in C$. Again, for $v_r \in R,\ deg_{G[S]}(v_r) \geq c - 1 + (n-1)(k-1) = c' - k$. Hence,

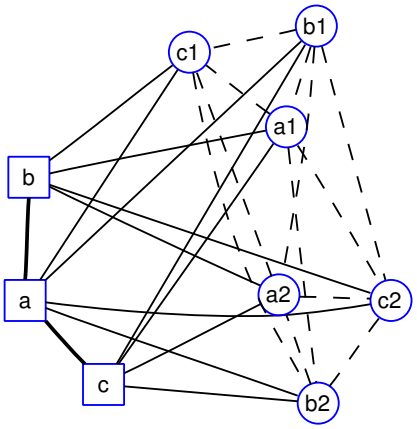


Fig. 8 Illustration of the k-PLEX instance G'

S induces a k-plex of size c'.

We now establish the other direction that if there exists a k-plex of size c' in G' then G has a clique of size c. Let S be a k-plex of size $c' = c+n(k-1)$. Let $P = R \setminus S$ denote the set of vertices from R not included in the k-plex and let $|P| = p$. Then, the c' vertices in S consist of $n(k-1)-p$ vertices in $S \cap R$ and $c+p$ vertices in $S \cap V$. Without loss of generality, suppose that $S \cap V = \{1, \ldots, c+p\}$ and further assume that for each $i \in S \cap V$ there exist q_i copies of i in P that are left out of the k-plex. Since every $i \in S \cap V$ has $p - q_i$ neighbors in P, we know that

$$|N(i) \cap (S \cap R)| = (n-1)(k-1) - (p - q_i).$$

Since S is a k-plex, $\forall\ i \in S \cap V$:

$$deg_{G[S]}(i) = |N(i) \cap (S \cap R)| + |N(i) \cap (S \cap V)| \geq c + n(k-1) - k,$$

$$\Rightarrow |N(i) \cap (S \cap V)| \geq c + p - 1 - q_i. \tag{4.1}$$

Recall that each q_i is a non-negative integer counting copies of vertex $i \in S \cap V$ in P and note that P can contain vertices that are not copies of any vertex in $S \cap V$.

Thus, we have $\sum_{i=1}^{c+p} q_i \leq p$. Hence, there can exist at most p terms, q_i, in that sum that are strictly greater than 0, meaning that there exist at least c terms in that equation, that are equal to 0. Without loss of generality, suppose that $q_i = 0,\ i \in \{1, \ldots, c\}$. Now, let $C = \{1, \ldots, c\}$. We already know from (4.1) that for all $i \in C \subseteq S \cap V = \{1, \ldots, c+p\}$:

$$|N(i) \cap (S \cap V)| \geq c + p - 1 - q_i = c + p - 1.$$

But $|S \cap V| = c + p$, so for all $i \in C$,

$$|N(i) \cap (S \cap V)| = c + p - 1.$$

Thus, every vertex in $C \subseteq S \cap V$ is adjacent to every vertex in $S \cap V$. Hence, every vertex in C is adjacent to every other vertex in C. Therefore C induces a clique of size c in G. This completes the proof. □

This complexity result again demonstrates that the maximum k-plex problem is hard not only because it is a generalization of the maximum clique problem, but it is a hard problem in its own respect, as Theorem 10 states that the decision version of the problem is NP-complete for every fixed positive integer k.

Remark 5. Note that maximum k-plex problem is trivial when $\delta(G) \geq n - k$ as $V(G)$ is itself a k-plex. In the above reduction, whenever G is not complete, $\delta(G') < |V'| - k$.

Remark 6. Furthermore, as a consequence of Theorem 10 we know that the problem of finding a *maximum co-k-plex* in a graph is also NP-hard.

IV.3. Some Special Cases

In Sections IV.1 and IV.2, we established that k-Clique, k-Club and k-Plex are NP-complete for every fixed k. The problems are also trivially NP-hard for arbitrary

k as they all generalize cliques. However, as shown in the beginning of this chapter the problems are easy to solve when $k = n, n-1$. In fact, the problems are polynomial-time solvable whenever $k = n - t$ for a *fixed* integer $t \geq 0$. Note that this does not conflict any of the previous NP-completeness results since all of them consider a fixed positive integer k or an arbitrary k, while k here is a particular function of n.

First we observe that any $k = n - t$ vertices in the graph G on n vertices forms a k-plex. Thus $\omega_k(G) \geq k = n - t$. Hence we try deleting every subset $S_j \subseteq V$ of size $j = 0, 1, \ldots, t$ until we find that $G[V \setminus S_j]$ is a k-plex. This will happen for $j = t$ guaranteeing termination and since we consider every possible subset, optimality is guaranteed. We only need to show that the algorithm runs in polynomial time. We have also implicitly assumed that $t \not\geq n$, this will be removed in the algorithm.

For solving the maximum k-clique and k-club problems, we initially assume that the given graph is connected and $t \not\geq n + 1$. These assumptions are subsequently relaxed. In any connected graph G on n vertices, there exists a connected induced subgraph G' on $n - t + 1$ vertices. G' can be formed by deleting $t - 1$ vertices, by recursively deleting the leaves of a breadth-first search tree of G. Then $diam(G') \leq k = n - t$ since the longest path in G' can be length at most $n - t$. Thus we have $\tilde{\omega}_k(G) \geq \bar{\omega}_k(G) \geq n - t + 1$. Again, we use an algorithm that is similar to previous case. Find the first $G[V \setminus S_j]$ which is a k-clique (k-club) for every possible S_j such that $|S_j| = j$ for $j = 0, 1, \ldots, t - 1$. Algorithm will terminate when $j = t - 1$ at optimality if not earlier. This algorithm can also be used on disconnected graphs by applying it to every component of G with at least $n - t + 1$ vertices and returning the largest k-clique or k-club found among all components. If every component of G has fewer than $n - t + 1$ vertices, then the largest connected component is optimum.

We present a generic polynomial-time algorithm that solves maximum k-clique, k-club and k-plex for $k = n - t$ on a graph of order n when t is a *fixed non-negative*

integer. Let Π denote the model of interest and parameter $M = t$ if Π is the k-plex model and $M = t - 1$ if Π is k-clique or k-club models. Algorithm 2 presents a pseudocode of a generic algorithm that combines the approaches mentioned above as they only differ in the value of parameter M across the problems. Step 2 removes the assumption made in the previous arguments that $t < n$ in the case of k-plex and and $t < n+1$ in the case of k-clique and k-club. When this assumption does not hold, no k-plex, k-clique or k-club exists in G for $k = n-t$. The rest of the algorithm implements the ideas discussed earlier. We try to remove the fewest possible number of vertices ($j = 0, \ldots, M$) to obtain a maximum Π. But since we do not know which j vertices to delete, we try every possible set thus guaranteeing optimality. Furthermore, as a Π is guaranteed to exist when $j = M$, the algorithm will terminate. Now it only remains to show that this algorithm runs in polynomial time. Step 2 runs in constant time. Step 4 runs at most M times and Step 5 runs $\binom{n}{j}$ times for each j. Step 6 can be done in polynomial time, say n^c where c is a fixed constant since the problems are in NP. The running time of this algorithm can then be bounded by a function $f(n, M)$ defined as, $f(n, M) = \sum_{j=0}^{M} n^c \binom{n}{j} \leq \sum_{j=0}^{M} n^c n^j \leq M n^{c+M}$. Thus the algorithm runs in $O(tn^{c+t})$ which is polynomial for *fixed* t.

Summary of Results. This chapter presented NP-completeness results for the k-Clique, k-Club and k-Plex problems for fixed k on arbitrary graphs. For fixed integer k, NP-completeness on graphs of diameter d with $k < d$ is established for k-Clique and k-Club. Special polynomially solvable cases are also identified.

Having established the intractability of maximum k-clique, k-club and k-plex problems, we identify approaches that would help us solve these problems, although via worst case exponential algorithms. The approach taken in this dissertation is that of a polyhedral study leading to branch-and-cut algorithms.

Algorithm 2 Generic Polynomial-Time Bounded Enumeration Algorithm

1: **procedure** MAXIMUM Π PROBLEM$(G, k = n - t, M)$
2: **if** $M \geq n$ **then** No Π exists in G; go to Step 11
3: **end if**
4: **for** $j = 0$ to M **do**
5: **for** each $S_j \subseteq V : |S_j| = j$ **do**
6: **if** $G[V \setminus S_j]$ is a Π **then**
7: **return** $V \setminus S_j$; go to Step 11
8: **end if**
9: **end for**
10: **end for**
11: **end procedure**

CHAPTER V

THE MAXIMUM *k*-CLIQUE AND *k*-CLUB PROBLEMS*

We have already discussed in Chapter II, the successful application of IP techniques and polyhedral combinatorics to the classical maximum clique and maximum independent set problems. In this chapter, we develop some preliminary approaches for the maximum k-clique and k-club problems. In particular, the close correspondence between the maximum clique and k-clique problems is established in an algorithmic, as well as polyhedral sense. The maximum k-club problem is also formulated as a binary integer program and the polyhedral studies are carried out for the special case when $k = 2$. The facets identified for the maximum 2-club problem will be used in developing a BC implementation in Chapter VII.

V.1. The Maximum *k*-Clique Problem

The *maximum k-clique problem* is to find a k-clique of largest cardinality in the given graph. Recall that the cardinality of a maximum k-clique is called the *k-clique number* of G and denoted by $\tilde{\omega}_k(G)$. The following binary integer program finds a maximum k-clique given $G = (V, E)$ and pairwise distances $d_G(i, j)$ $\forall i, j \in V$.

---

*Parts of this chapter are reprinted with permission from Balasundaram, B., Butenko, S., Trukhanov, S.: Novel approaches for analyzing biological networks. Journal of Combinatorial Optimization **10**(1), 23–39 (2005) © Springer.

$$\tilde{\omega}_k(G) = \max \sum_{i \in V} x_i \tag{5.1}$$

subject to:

$$x_i + x_j \leq 1 \quad \forall \ i, j \in V : i < j \text{ and } d_G(i,j) > k \tag{5.2}$$

$$x_i \in \{0,1\} \quad \forall \ i \in V \tag{5.3}$$

Constraint (5.2) ensures that at most one of i, j is included if $d_G(i,j) > k$ thereby ensuring that the feasible solutions are incidence vectors of k-cliques in G. The associated *k-clique polytope* $C_k(G)$ is the convex hull of binary vectors feasible to the above formulation.

Recall the definition of power graphs from Chapter II. Note that, the pairs of i, j over which constraint (5.2) is defined, is exactly the set $\overline{E^k}$, the edge set of the complement of k^{th} power of G. The formulation can then be seen as the maximum clique formulation on G^k. This is in fact true due to the following fact that can be easily verified from the definitions. *A subset of vertices C is a k-clique in G if and only if they form a clique in G^k.* This observation tells us that we can utilize the vast research that already exists for the maximum clique problem to deal with this problem since the power graph can be obtained in polynomial time from the original graph. Furthermore, the k-clique polytope $C_k(G)$ is exactly the clique polytope of the power graph, $C(G^k)$. Hence results regarding properties of the clique polytope are also thus applicable. Thus, for this dissertation, we will not consider the maximum k-clique problem henceforth.

V.2. The Maximum k-Club Problem

The *maximum k-club* problem is to find a largest k-club in a given graph. Recall that we denote the size of a maximum k-club by $\bar{\omega}_k(G)$ and refer to it as the *k-club number* of G. The following integer program finds the k-club number.

$$\begin{aligned}
\bar{\omega}_k(G) &= \max \sum_{i \in V} x_i && (5.4)\\
\textit{subject to:}& \\
x_i + x_j &\leq 1 + \sum_{l: P_{ij}^l \in \mathbb{P}_{ij}} y_{ij}^l \quad \forall\ (i,j) \notin E && (5.5)\\
x_p &\geq y_{ij}^l \quad \forall\ p \in V(P_{ij}^l),\ P_{ij}^l \in \mathbb{P}_{ij},\ (i,j) \notin E && (5.6)\\
x_i &\in \{0,1\} \quad \forall\ i \in V && (5.7)\\
y_{ij}^l &\in \{0,1\} \quad \forall\ P_{ij}^l \in \mathbb{P}_{ij},\ (i,j) \notin E && (5.8)
\end{aligned}$$

where $\mathbb{P}_{ij}$ is an indexed collection of all paths of length at most k between vertices i, j in G and P_{ij}^l is the path with index l between vertices i, j. The formulation essentially ensures that if two vertices are in a k-club, then all the vertices in at least one path between them with length less than or equal to k are also included in the k-club. Note that the size of $\mathbb{P}_{ij}$ could be very large making this formulation difficult to handle. Identifying conditions or restricted graph classes that guarantee a more compact formulation would be of interest.

The *k-club polytope* $W_k(G)$ is the convex hull of incidence vectors of k-clubs in G. Recall that a subset of a k-club need not be a k-club. This property makes it especially difficult to generate combinatorial valid inequalities for the problem. However, valid inequalities can be generated based on necessary conditions for vertices to belong in a k-club. We need the following definition.

Definition 27. Subset $I \subseteq V$ is *k-independent* if for every distinct pair $i, j \in I$, $d_G(i, j) \geq k + 1$.

Clearly, when $k = 1$, this is the classical independent set definition. For $k \geq 2$, the k-independent sets in G are exactly independent sets of the power graph G^k. So maximal k-independent sets can be found using simple greedy algorithms. However, for $k \geq 2$, finding maximum k-independent sets is known to be NP-hard even when the graphs are restricted to be bipartite [59]. It is also important to note that, k-cliques and k-independent sets are not complementary to each other for $k \geq 2$. Furthermore, as k increases, k-independent sets do not relax independent sets, but rather become restrictive in their definition.

k-Independent Set Inequalities. Let $I \subseteq V$ be a maximal k-independent set. Note that any k-club can contain at most one vertex from this set. Then we have the following family of valid inequalities.

$$\sum_{i \in I} x_i \leq 1 \quad \text{for every } k\text{-independent set } I \tag{5.9}$$

Heuristics for the maximum k-club problem were available in literature [44] even before its tractability was known. NP-completeness of the problem was subsequently established in [45] (independent of our result in Theorem 8) with the implicit assumption of a fixed positive integer k, using a slightly different reduction. However, our results with regards to transition in complexity identified by our NP-completeness results on bounded diameter graphs were previously unknown. An exact combinatorial branch-and-bound algorithm for the problem was also developed in [45].

In the next section, we look at the special case when $k = 2$, which lends itself for better analysis in terms of theory and algorithm development. It is also practically important as diameter-two structures are often used in design and analysis of networks

(see applications described in Section II.1).

V.3. The Maximum 2-Club Problem

The *maximum 2-club problem* can be formulated more compactly than the maximum k-club formulation in the previous section for general k. The 2-club number $\bar{\omega}_2(G)$ of a graph $G = (V, E)$ admits the following IP formulation:

$$\bar{\omega}_2(G) = \max \sum_{i \in V} x_i \tag{5.10}$$

subject to:

$$x_i + x_j - \sum_{k \in N^{\cap}(i,j)} x_k \leq 1 \quad \forall \ (i,j) \notin E \tag{5.11}$$

$$x_i \in \{0,1\} \quad \forall i \in V \tag{5.12}$$

where $N^{\cap}(i,j)$ denotes the common neighborhood of vertices i, j in G, *i.e.* $N^{\cap}(i,j) = N(i) \cap N(j)$. The formulation ensures that if two vertices are in a 2-club and they do not have an edge between them, then they have at least one common neighbor inside the 2-club. The *2-club polytope* is the convex hull of feasible solutions to the above formulation. Some interesting properties of this polytope have been identified. For the following results, we assume G has at least two vertices.

Theorem 11. *Let $W_2(G)$ denote the 2-Club polytope of a given graph $G = (V, E)$.*

1. *$dim(W_2(G)) = |V|$.*
2. *$x_i \geq 0$ induces a facet of $W_2(G)$ for every $i \in V$.*
3. *For $i \in V$, $x_i \leq 1$ induces a facet of $W_2(G)$ if and only if $d_G(i,j) \leq 2 \ \forall \ j \in V$.*

Proof. We will use following notations in the proof. Let e_i be the unit vector with i^{th} component 1 and the rest 0; $e_{ij} = e_i + e_j$ and $e_{ijk} = e_i + e_j + e_k$.

1. This is trivial and can shown by demonstrating $|V|+1$ feasible affinely independent points in $W_2(G)$. The points $\mathbf{0}, e_1, e_2, \ldots, e_{|V|}$ are clearly $|V|+1$ affinely independent points in $W_2(G) \subset \mathbb{R}^{|V|}$. Hence $dim(W_2(G)) = |V|$.

2. Let $F_i^0 = \{x \in W_2(G) : x_i = 0\}$. Then $\mathbf{0}, e_k$ for all $k \in V \setminus \{i\}$ form $|V|$ affinely independent points in F_i^0 indicating that $dim(F_i^0) = |V| - 1$ and it is a facet.

3. For a fixed $i \in V$, suppose that $d_G(i,j) \leq 2 \ \ \forall\ j \in V$. We wish to show that $F_i^1 = \{x \in W_2(G) : x_i = 1\}$ is a facet. Then let $S^p = \{j \in V : d_G(i,j) = p\}$. Note that S^0, S^1, S^2, partition V and $S^0 = \{i\}, S^1 = N(i)$. We now establish the maximality of F_i^1 thereby making it a facet. Suppose there exists a valid inequality $\alpha^T x \leq \beta$ such that, $F = \{x \in W_2(G) : \alpha^T x = \beta\} \supseteq F_i^1$. Note that $e_i, e_{ij}\ \forall\ j \in S^1$ are also contained in F_i^1. Also for every $k \in S^2$, there exists a $j \in S^1$ such that $j \in N^{\cap}(i,k)$. Hence $e_{ijk}\ \forall\ k \in S^2$ for some $j \in S^1$ are also in F_i^1. These $|V|$ points are also contained in F. $e_i \in F \Rightarrow \alpha_i = \beta$ (by substituting for x in $\alpha^T x = \beta$). Similarly we get $\alpha_i + \alpha_j = \beta\ \forall j \in S^1 \Rightarrow \alpha_j = 0\ \forall j \in S^1$. From the remaining points we obtain $\alpha_i + \alpha_j + \alpha_k = \beta\ \forall\ k \in S^2$ and for some $j \in S^1$. Since $\alpha_i = \beta, \alpha_j = 0\ \forall j \in S^1$, we get $\alpha_k = 0\ \forall k \in S^2$. This shows that $F_i^1 = F$ is a maximal face, *i.e.* a facet. Alternately, we could argue that the $|V|$ points used are affinely independent.

To establish the other direction, we establish the contrapositive by showing that if there exists a $j \in V$ such that $d_G(i,j) > 2$ then $F_i^1 = \{x \in W_2(G) : x_i = 1\}$ is not a facet. When such a j exists, we know that $(i,j) \notin E$ and $N^{\cap}(i,j) = \emptyset$. Then for this j, the constraint in the system $x_i + x_j - \sum_{k \in N^{\cap}(i,j)} x_k \leq 1$ reduces to $x_i + x_j \leq 1$ which dominates $x_i \leq 1$. Hence F_i^1 cannot be a facet. $\square$

The next result shows that the valid inequality we generated for the k-club problem is actually facet defining when $k = 2$.

Theorem 12. *Let $W_2(G)$ denote the 2-club polytope of $G = (V, E)$ and $I \subseteq V$. The inequality*

$$\sum_{i \in I} x_i \leq 1 \tag{5.13}$$

induces a facet of $W_2(G)$ if and only if I is a maximal 2-independent set in G.

Proof. Suppose I is a maximal 2-independent set. Clearly (5.13) is valid for $W_2(G)$. We now establish the maximality of the face $F_I = \{x \in W_2(G) : \sum_{i \in I} x_i = 1\}$, thereby showing it is a facet. Suppose there exists a valid inequality $\alpha^T x \leq \beta$ such that, $F = \{x \in W_2(G) : \alpha^T x = \beta\} \supseteq F_I$. Since $e_i \in F_I \subseteq F \; \forall \; i \in I$, we have $\alpha_i = \beta \; \forall \; i \in I$. Now for every $j \in V \setminus I$, there exists a vertex $i \in I$ such that at least one of the following two conditions are satisfied:

1. $(i, j) \in E$ and so $e_{ij} \in F_I \subseteq F$;

2. $N^{\cap}(i, j) \neq \emptyset$, *i.e.* they have a common neighbor $k \in V \setminus I$ in which case $e_{ikj}, e_{ik} \in F_I \subseteq F$.

Now for every $j \in V \setminus I$, in the first case we obtain, $\alpha_i + \alpha_j = \beta \Rightarrow \alpha_j = 0$ and in the second case we obtain $\alpha_i + \alpha_k + \alpha_j = \beta$ and $\alpha_i + \alpha_k = \beta \Rightarrow \alpha_j = \alpha_k = 0$. Thus, $F_I = F$ is a facet.

Suppose (5.13) induces a facet of $W_2(G)$. If I is not a 2-independent set then there exist $i, j \in I$ such that either $(i, j) \in E$ or if I is independent there exists $k \in N^{\cap}(i, j)$. In the first case e_{ij} which is a feasible point violates (5.13) and in the second case e_{ijk} which is feasible violates (5.13) contradicting the validity of (5.13). Hence I must be a 2-independent set. If I is not maximal there exists $I \cup \{u\}$ which is 2-independent and $x_u + \sum_{i \in I} x_i \leq 1$ is valid and dominates (5.13) contradicting its facetness. Hence I is maximal. □

Remark 7. Note that these results are analogous to those obtained for the stable set polytope with maximal cliques by Padberg [157]. Part 3 of Theorem 11 can now be

viewed as a special case of Theorem 12.

This result encourages the use of maximal 2-independent set facets in a BC algorithm to solve the maximum 2-club problem. In the next section, we present an algorithm which also incorporates pre-processing techniques with the BC that we will demonstrate to be computationally effective for power law graphs in Chapter VII.

V.4. Solving the Maximum *2*-Club Problem on Power Law Graphs

We use a pre-processing technique called *trimming* followed by a BC algorithm embedded in an iterative scheme that involves fixing variables in each iteration to be included in the maximum 2-club. Algorithm 4 presents the pseudo-code for the *Iterative Trim-and-Branch-and-Cut (ITBC) Algorithm*. Algorithm 3 presents the pseudo-code for the pre-processing procedure *Trim*. The BC algorithm referred to in the pseudo-code incorporates maximal 2-independent set cuts to tighten the LP relaxation bounds in the nodes of a branch-and-bound search tree. A general BC framework was described in Algorithm 1, specific implementation details and parameter settings will be discussed in Chapter VII before presenting the sample numerical experiments. Note that in the pseudo-codes the 2-neighborhood of a vertex v in G, defined as the set $\{i \in V(G) : d_G(v,i) \leq 2\}$ is denoted by $N_2[v]$.

Trimming. The pre-processing procedure *Trim* works with a graph G and a lower-bound on the size of the maximum 2-club. For any vertex $v \in V(G)$, the size of any maximum 2-club containing v is bounded from above by $|N_2[v]|$, as no vertex at distance 3 or more from v in G can be included. Hence, any vertex with a small 2-neighborhood (relative to $|D|$) can be deleted and this procedure can be recursively applied as many times as possible.

ITBC Algorithm. The ITBC Algorithm performs n iterations and in each iter-

Algorithm 3 Trimming Procedure

1: **procedure** TRIM(G, D) ▷ graph to be pre-processed, known 2-club
2: $X \leftarrow \{v \in V(G) : |N_2[v]| < |D|\}$
3: **if** $X \neq \emptyset$ **then**
4: $G \leftarrow G[V \setminus X]$; **go to** step 2 ▷ this will cause $V(G)$ to change
5: **end if**
6: **return** G
7: **end procedure**

ation, fixes a vertex v to be included in the maximum 2-club. In other words, the maximum 2-club containing v is found for each v in every iteration. In step 4 of the algorithm, the vertices could also be selected in non-increasing order of their degrees in $V(G)$. Having fixed v, the only vertices that need to be considered are vertices in $N_2[v]$. Hence, the trimming procedure is called on the graph induced by $N_2[v]$ along with the lower-bound initialized for the very first time with $\Delta(G) + 1$. The lower-bound is updated as larger 2-clubs are identified during the iterations. Following pre-processing, BC is used to find the maximum 2-club containing v, which is enforced by adding the additional constraint $x_v = 1$ to the system. The best known 2-club size is used in subsequent calls to the Trim procedure. Furthermore, at the end of every iteration, vertex v can be removed from the graph as from that point we are interested in maximum 2-clubs not containing v.

ITBC is an exact algorithm and expected to be worst case exponential given the intractability of the maximum 2-club problem. However, the pre-processing techniques are devised with the hope of speeding up computations and to facilitate optimal resolution of larger instances. The frequent checks comparing the size of the incumbent 2-club to the order of $G[N_2[v]]$ before trimming and to the order of $\tilde{G}$ before

Algorithm 4 Iterative Trim-and-Branch-and-Cut (ITBC) Algorithm

1: **procedure** ITBC(G) ▷ the 2-club instance
2: $v \leftarrow \text{argmax}\{deg(i) : i \in V\}$, $D \leftarrow N[v]$ ▷ D is a feasible 2-club
3: $G \leftarrow \textsc{Trim}(G, D)$
4: **for** each $v \in V(G)$ **do** ▷ a maximum 2-club containing v will be found
5: **if** $|N_2[v]| > |D|$ **then**
6: $\tilde{G} \leftarrow \textsc{Trim}(G[N_2[v]], D)$ ▷ only $i \in N_2[v]$ can be included
7: **if** $|V(\tilde{G})| > |D|$ **then**
8: $\tilde{D} \leftarrow \textsc{Branch-and-cut}(\tilde{G}, x_v = 1)$ ▷ v must be included
9: $D \leftarrow \text{argmax}\{|D|, |\tilde{D}|\}$ ▷ update if necessary
10: **end if**
11: **end if**
12: $G \leftarrow G - v$ ▷ delete v from G, this changes $V(G)$
13: **end for**
14: **return** D
15: **end procedure**

invoking BC play an important role in reducing the number of times BC algorithm is called in very large, sparse graphs. It should be noted that this approach is meaningful for power law graphs and not in general. For general graphs, it is better to use the BC algorithm without the iterative scheme, possibly with trimming procedure invoked for fixing variables at the BC nodes. We will substantiate the claims made with regards to computational experience on power law graphs in Chapter VII.

Summary of Results. In this chapter, we established preliminary results for the maximum k-clique and k-club problems. Emphasis was on the maximum 2-club problem for which polyhedral properties were studied and a family of facet-defining inequalities developed. A specialized algorithm incorporating pre-processing and BC approach in a systematic way has been developed that is effective for power law graphs. We have laid the foundations for several important directions that need to be taken in the future, especially for k-clubs with general k. These will be discussed in greater detail in Chapter X. In the next chapter, we study the focus problem of this dissertation: *The maximum k-plex problem.*

CHAPTER VI

THE MAXIMUM *k*-PLEX PROBLEM

In this chapter, we formulate the maximum k-plex problem as a binary integer program and study the associated polytope, for which several combinatorial valid inequalities are identified. Facet results are established for one family of valid inequalities when $k = 2$. Some interesting aspects of this relaxation, and the complementary definition of co-k-plex that we proposed in Chapter III will be identified in this chapter. In the process, a novel formulation for the classical maximum clique problem is identified and studied. A specialized algorithm incorporating preprocessing with a BC framework is presented that is designed to be effective on power law graphs.

VI.1. The Maximum *k*-Plex Problem

The *maximum k-plex problem* is to find a largest k-plex in a given graph, the size of which is called the *k-plex number* denoted by $\omega_k(G)$. Let $deg_{\bar{G}}(i) = |V \setminus N[i]|$ denote the degree of vertex i in the complement graph $\bar{G} = (V, \bar{E})$. The following 0-1 program finds the largest k-plex in G.

$$\omega_k(G) = \max \sum_{i \in V} x_i \tag{6.1}$$

subject to:

$$\sum_{j \in V \setminus N[i]} x_j \leq (k-1)x_i + deg_{\bar{G}}(i)(1 - x_i) \quad \forall\, i \in V \tag{6.2}$$

$$x_i \in \{0, 1\} \quad \forall\, i \in V \tag{6.3}$$

In this formulation, $x_i = 1$ if and only if $i \in V$ is in a k-plex. Constraint (6.2)

ensures that if a vertex i is in the k-plex then it has at most $k-1$ non-neighbors inside the k-plex. The constraint is made redundant if vertex i is not in the k-plex. Hence every feasible solution is an incidence vector of some k-plex in G. The *k-plex polytope* $P_k(G)$, is the convex hull of feasible points of the above formulation. The following theorem establishes some fundamental properties of the k-plex polytope. We assume that $k \geq 2$ for the subsequent results since $k = 1$ corresponds to the well researched maximum clique problem. However, in Section VI.4, we focus on the case when $k = 1$ in greater detail.

Theorem 13. *Let $P_k(G)$ denote the k-plex polytope of a given graph $G = (V, E)$, where $k \geq 2$. Then,*

1. *$dim(P_k(G)) = |V|$.*
2. *$x_i \geq 0$ induces a facet of $P_k(G)$ for every $i \in V$.*
3. *$x_i \leq 1$ induces a facet of $P_k(G)$ for every $i \in V$.*

Proof. We will use following notations as before in the proof. Let e_i be the unit vector with i^{th} component 1 and the rest 0; $e_{ij} = e_i + e_j$.

1. This is shown by demonstrating $|V| + 1$ affinely independent points in $P_k(G)$. The points $\mathbf{0}, e_1, e_2, \ldots, e_{|V|}$ are clearly $|V| + 1$ affinely independent points in $P_k(G) \subset \mathbb{R}^{|V|}$. Hence, $dim(P_k(G)) = |V|$.
2. Let $F = \{x \in P_k(G) : x_i = 0\}$. Since an empty set or any vertex by itself is a k-plex, we have $\mathbf{0}, e_j$ for all $j \in V \setminus \{i\}$ forming $|V|$ affinely independent points in F. This shows that $dim(F) = |V| - 1$ and it is a facet.
3. Let $F' = \{x \in P_k(G) : x_i = 1\}$. We first observe that every vertex and any pair of vertices form a k-plex for any k such that $1 < k < |V|$. Then e_i and e_{ij}

for all $j \in V \setminus \{i\}$ form $|V|$ affinely independent points in F', indicating that $dim(F') = |V| - 1$ and it is a facet. □

VI.2. Facets and Valid Inequalities

Valid inequalities in this section are combinatorial in nature. They are derived by identifying induced subgraphs that are not k-plexes and hence cannot be present in any k-plex. Note that the property that every induced subgraph of a k-plex is also a k-plex permits us to make the previous argument. Recall that this was not the case for k-clubs and the only family of valid inequalities identified used a necessary condition. The following lemmas are needed.

Lemma 1. *Let k be even. Then, there does not exist a co-k-plex that contains a k-plex of size $2k-1$.*

Proof. Let G be a co-k-plex on n vertices. Assume that $n \geq 2k-1$ as the result is trivial otherwise. Now suppose that S is a k-plex of size $2k-1$ in G. Then we have

$$|N(i) \cap S| \geq 2k - 1 - k = k - 1 \quad \forall i \in S.$$

Since G is co-k-plex we have,

$$|N(i) \cap S| \leq |N(i)| \leq k - 1 \quad \forall i \in S.$$

The two conditions then imply that the induced graph $G[S]$ is regular with all degrees equal to $k-1$ and is of order $2k-1$. But $k-1$ is odd and we cannot have an odd number of vertices of odd degree. The contradiction implies that S does not exist. □

Remark 8. Note that this bound is sharp since the graph family $G_k = K_k \cup K_{k-1}$, the union of complete graphs, for each k forms a co-k-plex of size $2k-1$, which contains

$K_{k-1} \cup K_{k-1}$, a k-plex of size $2k-2$. Fig. 9 illustrates this when $k = 4$. The circled vertices form the said 4-plex.

Lemma 2. *Let k be odd. Then, there does not exist a co-k-plex that contains a k-plex of size $2k$.*

Proof. As before, let G be a co-k-plex on n vertices ($n \geq 2k$). Suppose that S is a k-plex of size $2k$ in G. Then we have

$$|N(i) \cap S| \geq 2k - k = k \quad \forall i \in S.$$

Since G is co-k-plex we have,

$$|N(i) \cap S| \leq |N(i)| \leq k - 1 \quad \forall i \in S.$$

This contradiction establishes that S does not exist. □

Remark 9. This bound is also sharp since the following family of graphs have $2k$ vertices forming a co-k-plex containing a k-plex of size $2k-1$. Construct the graph $G_k = (V, E)$, where

$$V = V' \cup \{2k\}, \quad V' = \{1, \ldots, 2k-1\},$$

and

$$E = \{(i, j) : i \in V' \text{ and } j = i+1, \ldots, \left(i + \frac{k-1}{2}\right) \mod (2k-1)\}.$$

Maximum degree in G_k is $k-1$ and hence it is a co-k-plex of order $2k$. The induced subgraph $G_k[V']$ is a $(k-1)$-regular k-plex of order $2k-1$ in which every vertex has exactly $k-1$ neighbors and non-neighbors each. It is also known as an *antiweb* and its complement is known as a *web* (see Section II.6.1 for details). Fig. 9 illustrates this when $k = 5$. The circled vertices form the said 5-plex.

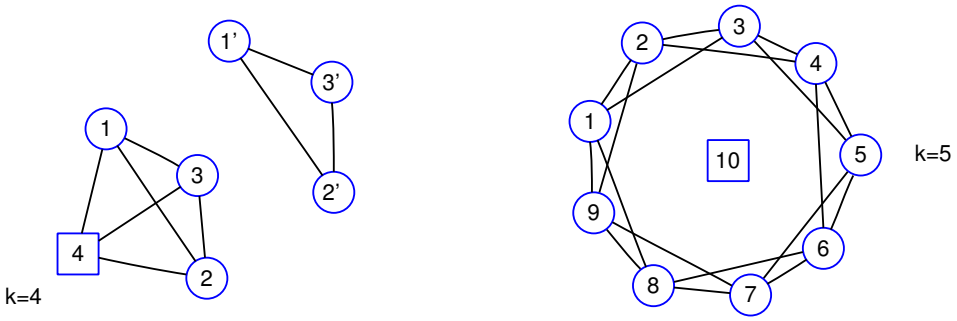


Fig. 9 Graphs demonstrating the sharp bounds in Lemmas 1 and 2

We next present three types of valid inequalities for the k-plex polytope: *Independent set inequalities, hole inequalities,* and *co-k-plex inequalities.*

Independent Set Inequalities. Let $I \subseteq V$ be an independent set. Note that no k-plex can contain an independent set of more than k vertices as $k+1$ or more independent vertices do not form a k-plex. Let $\mathcal{I}_{k+1}$ represent the collection of all MIS of size $k+1$ or more in G. Then we have the following family of valid inequalities.

$$\sum_{i \in I} x_i \leq k \qquad \forall\ I \in \mathcal{I}_{k+1} \tag{6.4}$$

Remark 10. Note that inequality (6.4) is facet-inducing for the k-plex polytope of the trivial graph $G_\emptyset$ with $V = \{1, \ldots, n\}$ and $E = \emptyset$ with $1 < k < n$. When n and k are relatively prime, the incidence vectors of $\{i+1(\mod n), \ldots, i+k(\mod n)\}$ for $i \in V$ form n affinely independent vectors [182]. If n is a multiple of k, the above observation holds for the trivial graph $G_\emptyset - u$ for some u. The facet of $P_k(G_\emptyset - u)$ induced by inequality (6.4), when lifted to $P_k(G_\emptyset)$, will yield a coefficient of one for x_u establishing our claim.

Hole Inequalities. Let $H \subseteq V$ be a hole. If $|H| \leq k+2$, then H is a k-plex. Now suppose $|H| > k+2$, then H is not a k-plex and for every proper subset $S \subset H$, we have $\delta(G[S]) \leq 1$. Hence, if $|S| - k \geq 2$, S is not a k-plex. Thus, any k-plex can

contain at most $k+1$ vertices from the hole and this bound is met when S induces a path of $k+1$ vertices in H. Let $\mathcal{H}_{k+3}$ represent the collection of all holes of size $k+3$ or more in G. Then we have the following family of valid inequalities.

$$\sum_{i \in H} x_i \leq k+1 \qquad \forall\ H \in \mathcal{H}_{k+3} \tag{6.5}$$

Co-k-plex Inequalities. Lemmas 1 and 2, when combined, imply that the size of a maximum k-plex in any co-k-plex is less than or equal to $\rho_k = 2k - 1 - \frac{1+(-1)^k}{2}$. Let $\mathcal{J}_{\rho_k+1}$ represent the collection of all maximal co-k-plexes of size more than ρ_k in G. We have the following family of valid inequalities.

$$\sum_{i \in J} x_i \leq \rho_k \qquad \forall\ J \in \mathcal{J}_{\rho_k+1} \tag{6.6}$$

Remark 11. Note that for $k = 1$, the independent set inequalities and co-1-plex inequalities are identical. For $k = 2$, the co-2-plex inequalities (weakly) dominate the independent set inequalities, although this is not necessarily the case for $k = 3$.

Remark 12. Note that if J is a *maximal* co-k-plex, there does not exist another co-k-plex of which J is a proper subset. Then, for every $v \in V \setminus J$, at least one of the following conditions must hold.

1. $\exists\ j \in J \cap N(v)$ such that $|N(j) \cap J| = k-1$ and including v would cause degree of j in the induced subgraph to be k.
2. $|N(v) \cap J| \geq k$ and hence upon inclusion v would have degree k or more in the induced subgraph.

Note that both conditions coincide when $k = 1$. The next theorem uses this observation to show that for $k = 2$ the co-2-plex inequalities actually form facets for the 2-plex polytope, $P_2(G)$.

Theorem 14. *Let $P_2(G)$ denote the 2-plex polytope. Then, the co-2-plex inequality given by,*

$$\sum_{i \in J} x_i \leq 2, \tag{6.7}$$

where J is a maximal co-2-plex with $|J| \geq 3$, induces a facet of $P_2(G)$.

Proof. First, recall that any 2 vertices from J form a 2-plex. Second, for every $v \in V \setminus J$, the above two conditions for a maximal co-2-plex imply the existence of two vertices $u, w \in J$ such that $\{v, u, w\}$ is a 2-plex. Indeed, if the first case holds, let $u \in J \cap N(v)$, then $N(u) \cap J = \{w\}$ and $\{v, u, w\}$ is a 2-plex. If the second case holds, $\{u, w\} \subseteq J \cap N(v)$ and again $\{v, u, w\}$ is a 2-plex. We use these observations to construct n affinely independent vectors that lie on the face defined by $F = \{x \in P_2(G) : \sum_{i \in J} x_i = 2\}$, so F is $(n-1)$-dimensional and hence it is a facet. Without loss of generality, assume that $J = \{1, \ldots, r\}$ and $V \setminus J = \{r+1, \ldots, n\}$, where $r \geq 3$. As before, let $e_i \in \mathbb{R}^n$ denote the unit vector with i^{th} component one and all others zero. The said affinely independent vectors $x^1, \ldots, x^n$ are constructed as follows.

$x^v = e_v + e_r, \quad \forall\ v = 1, \ldots, r-1;$

$x^r = e_1 + e_2$ (note that x^r is distinct from $x^1, \ldots, x^{r-1}$ as $r \geq 3$);

$x^v = e_v + e_u + e_w, \quad \forall\ v = r+1, \ldots, n,$ where for each $v \in V \setminus J$, $u, w \in J$ are the particular vertices described before. Clearly, $x^v \in F$ and we *claim* that these vectors are affinely independent. Thus, the co-2-plex inequalities produce facets for the 2-plex polytope, provided the claim holds. Next we establish the claim.

Claim : Vectors $x^1, \ldots, x^n$ are affinely independent.

We need to show that the only solution to $\sum_{i=1}^{i=n} \lambda_i x^i = 0$ and $\sum_{i=1}^{i=n} \lambda_i = 0$ is $\lambda_i = 0\ \forall\ i =$

$1, \ldots, n$. This can be rewritten as

$$\mathbf{A}\lambda = \mathbf{0}, \text{ where } \mathbf{A} = \begin{bmatrix} x^1 & \ldots & x^r & x^{r+1} & \ldots & x^n \\ 1 & \ldots & 1 & 1 & \ldots & 1 \end{bmatrix} \text{ and } \lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix}$$

By construction (of x^i), $\mathbf{A}$ has the following structure.

$$\mathbf{A} = \begin{bmatrix} \mathbf{B_{r\times r}} & \mathbf{C_{r\times n-r}} \\ \mathbf{0_{n-r\times r}} & \mathbf{I_{n-r\times n-r}} \\ \mathbf{1_{1\times r}} & \mathbf{1_{1\times n-r}} \end{bmatrix}$$

Now it is easy to see that the identity matrix in the second set of equations forces $\lambda_i = 0,\ i = r+1, \ldots, n$. The system now reduces to,

$$\begin{bmatrix} \mathbf{B_{r\times r}} \\ \mathbf{1_{1\times r}} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_r \end{bmatrix} = \mathbf{0}$$

expanding $\mathbf{B_{r\times r}}$ we get

$$\begin{bmatrix} 1 & 0 & 0 & \ldots & \ldots & 0 & 1 \\ 0 & 1 & 0 & \ldots & \ldots & 0 & 1 \\ 0 & 0 & 1 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \vdots & \vdots & 1 & 1 & 0 \\ 1 & 1 & 1 & \vdots & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_r \end{bmatrix} = \mathbf{0}$$

Now it is easy to see that $\lambda_i = 0,\ i = 1, \ldots, r$ is the only solution to this system. □

The maximal co-k-plex inequalities when $k = 1$ are precisely the MIS inequalities for the clique polytope. Due to the classical results of Padberg [157] that we discussed

in Chapter II, it is well known that they also form facets of the clique polytope. The co-k-plex inequalities hence generalize independent set inequalities and Theorem 14 establishes that they form facets of $P_k(G)$ when $k = 2$. The next natural question is whether co-k-plex inequalities form facets of $P_k(G)$ for $k \geq 3$.

Although co-k-plex inequalities form facets of $P_k(G)$ for $k = 1, 2$, they do not for $k \geq 3$. Consider a graph $G = (V, \emptyset)$ with at least k vertices. Note that G is a co-k-plex and the corresponding inequality $\sum_{i \in V} x_i \leq \rho_k$ is not supporting since $\omega_k(G) = k$ and there is no $x \in P_k(G)$ that satisfies it at equality (Note that $k < \rho_k$ for $k \geq 3$). Hence, these inequalities do not form facets of $P_k(G)$ for all G. This is in contrast to the results known for $k = 1, 2$. The reason is $\rho_k = k$ for $k = 1, 2$ and every graph G with at least k vertices has a k-plex of size ρ_k $(= k)$. The next natural question, if they form facets when G is a co-k-plex with $\omega_k(G) = \rho_k$, $k \geq 3$ is also settled in the negative by the following counterexamples.

Assume that k is even. Construct graphs G of arbitrary order $n \geq \rho_k$ as the union of $n - \rho_k$ clique components of size one and two clique components of size $k - 1 = \rho_k/2$. Then G is a co-k-plex with the two "large" clique components forming a k-plex of size ρ_k. Suppose $F = \{x \in P_k(G) : \sum_{i \in V} x_i = \rho_k\}$ is a facet of $P_k(G)$. Since $P_k(G)$ is an integral polytope, the extreme points of F are also integral and F is a convex hull of those integral vectors. Consider one such binary vector $x^o \in F$. If $x^o_i = 1$ for some i that is a one-vertex clique component of G, for x^o to be feasible we have $\sum_{j \in V \setminus N[i]} x^o_j \leq k - 1$. But since $V \setminus N[i] = V \setminus \{i\}$, we have $\sum_{i \in V} x^o_i \leq k$, which contradicts the fact that $x^o \in F$ as $\rho_k > k$. Hence, the components of extreme points of F corresponding to one vertex components of G are all zeros. Hence, there exists *exactly one extreme point* in $P_k(G)$ that satisfies $\sum_{i \in V} x_i \leq \rho_k$ at equality which is the incidence vector of vertices in $K_{k-1} \cup K_{k-1}$. Thus, F is 0-dimensional and not a facet.

For odd k, we can have arbitrarily large graphs by adding single vertex compo-

nents to the antiweb $G_k[V']$ constructed before. By similar arguments, we can again show that there exists only one point in the k-plex polytope that satisfies the co-k-plex inequality at equality. From these observations we can conclude that $\omega_k(G) = \rho_k$ is only a necessary condition for co-k-plex inequality to induce a facet of $P_k(G)$.

Remark 13. Identifying graph classes for which the co-k-plex inequalities form facets of the k-plex polytope is an important problem for future study. It is also a well-known fact that a graph is perfect if and only if its clique polytope is completely characterized by all the MIS inequalities and non-negativity constraints [78] (see Section II.6.1). Similarly, we could explore *k-plex perfection* of graphs whose k-plex polytope can be completely described by the co-k-plex inequalities and the trivial bound constraints. This is also an interesting topic for future research.

VI.3. Solving the Maximum k-Plex Problem

Algorithms described in this section for the maximum k-plex problem are similar to the ones presented for the maximum 2-club problem in the previous chapter, in terms of goals and motivation. First is a BC algorithm which also forms a part of the second, a specialized algorithm for power law graphs. Note that the two BC based algorithms discussed here are for any k and they make use of the structural properties of a k-plex to perform tasks similar to algorithms in Section V.4.

Cutting Planes and Non-dominated Variable Fixing. The BC implementation developed incorporates *MIS cuts* and *co-k-plex cuts* generated using greedy algorithms. The implementation details and parameter settings will be discussed in Chapter VII, a general BC framework was already presented in Algorithm 1. Furthermore, *variable fixing procedures* are incorporated in the BC tree as follows. Recall from Theorem 7 that any k vertices in a k-plex dominate the graph induced by the k-plex. Thus,

when k variables are fixed to one for the first time in node N of the BC tree, every non-dominated vertex (*i.e.*, vertex not adjacent to *any* of the k vertices) can be fixed to zero in the subtree rooted at node N.

The BC algorithm with MIS or co-k-plex cutting planes and non-dominated variable fixing is used to solve the maximum k-plex problems on general graphs. This BC algorithm is also a part of a specialized algorithm designed to solve maximum k-plex problem optimally on power law graphs. The *Iterative Peel-and-Branch-and-Cut (IPBC) Algorithm* (Algorithm 7) applies *peeling* followed by a BC algorithm, both embedded in an iterative scheme that involves fixing variables in each iteration to be included in the maximum k-plex. The pre-processing procedure *Peel* (Algorithm 6) follows an approach similar to the one proposed in [5] for the maximum clique problem. The peeling procedure requires a feasible k-plex, initially provided by greedily finding a maximal clique (Algorithm 5) which is a k-plex for any k.

Algorithm 5 Greedy Maximal Clique Algorithm

1: **procedure** GREEDYMAXIMALCLIQUE(v, G) ▷ v is the starting vertex
2: initialize $C \leftarrow \emptyset$
3: **while** $V(G) \neq \emptyset$ **do**
4: $C \leftarrow C \cup \{v\}$, $G \leftarrow G[N(v)]$, $v \leftarrow \text{argmax}\{deg_G(i) : i \in V(G)\}$
5: **end while**
6: **return** C
7: **end procedure**

Peeling. The peeling procedure removes vertices of low degree-based on the size of a known k-plex. Suppose we know that there exists a k-plex S in the graph, then vertices with degree smaller than $|S| - k$ in the graph cannot belong to any optimal k-plex and can be removed. Vertices are recursively identified and removed based on

the condition that every vertex in an optimal k-plex has degree at least $|S| - k$.

Algorithm 6 Peeling Procedure

1: **procedure** PEEL(G, S) ▷ graph to be pre-processed, known k-plex
2: $X \leftarrow \{v \in V(G) : deg_G(v) \leq |S| - k\}$
3: **if** $X \neq \emptyset$ **then**
4: $G \leftarrow G[V \setminus X]$; **go to** step 2 ▷ this will cause $V(G)$ to change
5: **end if**
6: **return** G
7: **end procedure**

IPBC Algorithm. The IPBC Algorithm is more involved than the iterative analogue ITBC Algorithm 4 introduced for the maximum 2-club problem. The basic idea of fixing a vertex $v \in V$ in each iteration to find an optimal solution containing v is used again. The difference here is that we *assume* the k-plex number of the graph to be large enough, *i.e.*, $\omega_k(G) > 2k - 2$. If the assumption holds and the fixed vertex v is in S^* where S^* is some maximum k-plex in G, we are guaranteed by Theorem 7 that $S^* \subseteq N_2[v]$ as diameter of $G[S^*]$ is at most two. Recall that the definition of 2-neighborhood of a vertex v in G is given by $N_2[v] = \{i \in V(G) : d_G(v, i) \leq 2\}$. As we iterate over $v \in V$ we only need to consider vertices in $N_2[v]$, assuming that there is a large k-plex containing v. Under this assumption, for each $v \in V$ peeling procedure is called on the graph induced by $N_2[v]$ along with the lower-bound initialized with a maximal clique (size) and then updated during the iterations. BC is used after peeling to find a maximum k-plex containing v by adding the additional constraint $x_v = 1$ to the system. The resulting solution is used to update the current best k-plex S if necessary. At the end of every iteration, vertex v can be removed from the graph as from that point we are not interested in k-plexes containing v.

Algorithm 7 Iterative Peel-and-Branch-and-Cut (IPBC) Algorithm

procedure IPBC(G) ▷ the k-plex instance
  $G^o \leftarrow G$ ▷ needed if assumption fails
  for each $v \in V$, $S_v \leftarrow$ GreedyMaximalClique(v, G) **end for**
  $S \leftarrow \text{argmax}\{|S_v| : v \in V\}$ ▷ S is a feasible k-plex
  for $v \in V(G) : deg_G(v) = \Delta(G)$ **do** ▷ find a maximum k-plex containing v
    if $|N_2[v]| > |S|$ **then**
      $\tilde{G} \leftarrow$ Peel$(G[N_2[v]], S)$ ▷ only $i \in N_2[v]$ can be included
      if $|V(\tilde{G})| > |S|$ **then**
        $\tilde{S} \leftarrow$ branch-and-cut$(\tilde{G}, x_v = 1)$ ▷ v must be included
        $S \leftarrow \text{argmax}\{|S|, |\tilde{S}|\}$ ▷ update if necessary
      end if
    end if
    $G \leftarrow G - v$ ▷ delete v from G
  end for
  if $|S| > 2k - 2$ **then**
    return S
  else ▷ assumption failed, re-solve
    $S \leftarrow$ Branch-and-cut$(G^o, \sum_{v \in V} x_v \leq 2k - 2)$
    return S
  end if
end procedure

Once the iterations are complete, if the best known k-plex was larger than $2k-2$, our assumption was right and it can be returned as the optimal solution. If our assumption was incorrect, the solutions from the iterative procedure are not applicable and we re-solve maximum k-plex problem on the original graph with the additional constraint that $\sum_{v \in V} x_v \leq 2k-2$. This is not especially disadvantageous since even a complete enumeration of all k-plexes of size k to $2k-2$ to find the optimum is still polynomial for fixed k. In this situation, the BC algorithm can easily find the optimum. Moreover, a k-plex of size smaller than $2k-2$ might not be of much practical use since for most applications meaningful $\omega_k(G)$ should be large compared to k. If not, one must reconsider the choice of k used or investigate the data closely.

IPBC Algorithm is exact, but worst-case exponential. As was the case with the ITBC Algorithm, the approach taken in IPBC algorithm is to solve n NP-hard problems of smaller size instead of one large NP-hard problem. Note that the diameter-two assumption is essential to facilitate the iterative scheme, and to focus on smaller graphs induced by the two-neighborhood in each iteration. Without the assumption, even after fixing a vertex we would be forced to consider all the other vertices– both neighbors and non-neighbors. The simple checks on sizes in combination with peeling and the assumption of a large k-plex are designed to enable us handle large sparse instances by decomposing the graph. Avoiding a memory consuming large integer program and making use of the properties established in Theorem 7 will enable us to reduce the number of BC calls and permit us to solve maximum k-plex problem to optimality on very large but very sparse graphs like power law graphs. Our claims will become evident from the detailed computational study of the BC implementation and the IPBC algorithm presented in Chapter VII.

VI.4. On the Maximum Clique Problem

Consider the following 0,1-formulation of the maximum clique problem on $G = (V, E)$. Let $deg_{\bar{G}}(i) = |V \setminus N[i]|$ as before.

$$\omega(G) = \max \sum_{i \in V} x_i \tag{6.8}$$

subject to:

$$\sum_{j \in V \setminus N[i]} x_j \leq deg_{\bar{G}}(i)(1 - x_i) \quad \forall\, i \in V \tag{6.9}$$

$$x_i \in \{0, 1\} \quad \forall\, i \in V \tag{6.10}$$

This formulation is a special case of the maximum-k-plex formulation (6.1) - (6.3) presented before when $k = 1$ as $\omega_1(G) = \omega(G)$. Constraint (6.9) ensures that no non-neighbor of a vertex is included in the clique for every vertex in the clique and becomes redundant for others. We will refer to the formulation (6.8) - (6.10) of the maximum clique problem as the *1-plex formulation*.

This case was excluded in the previous polyhedral study for the following reason. Unlike the maximum-k-plex problem for $1 < k < n$, $x_i \leq 1$ is not always a facet for the maximum clique problem. It induces a facet *if and only if* $V = N[i]$. Since when the condition is satisfied, there exist n affinely independent vectors e_i and $e_i + e_j$ for all $j \in N(i) = V \setminus \{i\}$, that lie on the face defined by $x_i \leq 1$, and when the condition is not satisfied, there exists a $j \in V \setminus N[i]$ such that $x_i + x_j \leq 1$ is valid for the clique polytope. In other words $x_i \leq 1$ is a facet if and only if $\{i\}$ is a MIS. This is just a special case of a classical result due to Padberg [157].

Another interesting observation is that, to the best of our knowledge, this is the most compact *integer programming* formulation of the maximum clique problem with exactly n variables and n constraints. The classical (complement) *edge formulation* (6.11) - (6.13), has n variables and $|\bar{E}|$ constraints which could be $O(n^2)$ in the worst

case. This implies that while solving the maximum clique problem using the 1-plex formulation, the LP solved at the nodes of a BC tree is smaller in size compared to the edge formulation if the same number of cuts are added.

$$\omega(G) = \max \sum_{i \in V} x_i \tag{6.11}$$

subject to:

$$x_i + x_j \leq 1 \quad \forall\ (i,j) \in \bar{E} \tag{6.12}$$

$$x_i \in \{0,1\} \quad \forall\ i \in V \tag{6.13}$$

The other well known *MIS formulation* where constraint (6.12) is replaced by (6.14) has n variables and $O(3^n)$ constraints in the worst case [149].

$$\sum_{i \in I} x_i \leq 1 \quad \text{for each MIS, } I \text{ in } G \tag{6.14}$$

However the compactness of the 1-plex formulation comes at a price which we will make clear now. The edge formulation is closely related to the 1-plex formulation in the following sense. Note that constraint (6.12) can be rewritten as

$$x_i + x_j \leq 1 \quad \forall\ i \in V,\ j \in V \setminus N[i], \tag{6.15}$$

which amounts to repeating constraints in the edge formulation. Now it is easy to see that constraint (6.9) in the 1-plex formulation can be obtained by summing constraint (6.15) over all $j \in V \setminus N[i]$, for each $i \in V$, which results in n constraints. Let $C(G)$ denote the clique polytope of G, $LP_{edge}(G)$ denote the LP relaxation polytope of the edge formulation and let $LP_{1plex}(G)$ denote the LP relaxation polytope of the 1-plex formulation. Then we have, $C(G) \subseteq LP_{edge}(G) \subseteq LP_{1plex}(G)$ since $LP_{1plex}(G)$ is defined by a surrogate system [102] of $LP_{edge}(G)$. Thus, we solve a poorer relaxation in the nodes of a BC tree when we use the 1-plex formulation to find $\omega(G)$.

This trade-off between the compactness of our new 1-plex formulation versus the tight LP relaxation offered by the classical edge formulation, for the maximum clique problem, will be investigated experimentally in Chapter VII.

Summary of Results. In this chapter, we established some important fundamental results for the maximum k-plex problem. The k-plex polytope is defined and its basic properties are established. Three families of combinatorial inequalities, valid for the polytope are identified. Further, the co-k-plex inequality is shown to be facet-inducing for the k-plex polytope when $k = 2$. The potential for generalizing the facet result for $k \geq 3$ is discussed in great detail.

Variable fixing procedures that can be incorporated in a BC algorithm are developed based on the domination property of a k-plex. An iterative version of the BC algorithm is also developed with embedded preprocessing techniques that exploit the structural properties of a k-plex, such as low diameter and high degree for low k. This specialized algorithm is designed to work well on very large, but very sparse, power law graphs such as many real-life social and biological networks.

A surprising new and compact IP formulation for the classical maximum clique problem was identified. This new formulation is studied and its connection to the well-known edge formulation for the problem is established. In the next chapter, extensive computational experiments are conducted to test the effectiveness of the approaches proposed in this chapter. Some preliminary experiments are also conducted to investigate the algorithms developed in Chapter V.

CHAPTER VII

COMPUTATIONAL EXPERIMENTS

This chapter presents our computational experience with the algorithms developed in Chapter VI. Extensive investigation of BC and IPBC algorithms proposed for the maximum k-plex problem is carried out on a large test-bed of instances. In Section VII.1, we describe all the relevant BC parameters and in Section VII.2 we describe the instances used in testing. Section VII.3 presents our parameter settings, cut generation heuristics, numerical results and observations. In Section VII.4, we present a computational study of the *1-plex formulation* and *edge formulation* of the maximum clique problem discussed in Section VI.4.

Implementations of the BC algorithms for finding a maximum 2-club and for the maximum k-clique problem by finding a maximum clique on the power graph, have been developed as part of this dissertation work. However, our experimental results are limited to finding a maximum 2-club on social and biological networks using the ITBC algorithm. It was observed that most established benchmarks have diameter two and hence meaningful experiments could not be conducted. Development of appropriate benchmark instances is an important practical issue for testing algorithms for these problems. Our preliminary results of ITBC algorithm on real-life instances is presented in Section VII.5.

VII.1. General Implementation Details

Details that are common to all our experiments are the following. All numerical experiments were conducted on Dell Optiplex GX620® computers with Intel Pentium D® 3.20GHz, 3.19Ghz processor and 2GB RAM. The core of all our algo-

rithms is a BC approach, implemented using ILOG CPLEX 9.0® [119]. The biggest advantage of using the framework provided by CPLEX® is the effective default settings that take care of the branching process, node selection, variable selection, primal heuristics, pre-solving among others, while the bounding is done by solving the LP relaxation with the user-specified cuts.

Cuts are generated at *every* node of the BC tree, and are called *local cuts* as they are valid at the node in which they are generated and for all its child nodes, not necessarily the entire BC tree. Local cuts are implemented using the CPLEX® *goals* feature. For generating local cuts, the vertices corresponding to variables fixed at zero in that node are deleted from the graph before cuts are found, and only variables with high fractional value ($\geq$ HIFRACVAL, external constant) are used as starting vertices for finding cuts. This approach enables us to find cuts that are likely to be violated since the starting vertex, which is a part of the cut generated is already at a high value. Distinct *round of cuts* generated are tested for violation by the LP optimum at the node where they are generated. Most violated cuts, MAXLOCALCUTSPERNODE (external constant) in number, are added to the system. CPLEX® re-solves the problem at that node and handles the cut management from that point onwards. By limiting the number of cuts added at each node we ensure that the size of the LP solved at any node is never too big, always bounded by MAXLOCALCUTSPERNODE$\times n$.

It should be noted that CPLEX® generates its own classes of cuts to solve any given MIP. All such cuts were turned off and only user defined cuts were used in the BC implementations. A description of relevant CPLEX® parameters that were changed from their default values are as follows. Apart from regular termination of an MIP (optimal or infeasible), CPLEX® can be forced to terminate gracefully returning the best integer feasible solution and objective (if found), as well as a bound on the

optimum, when an upper time limit is reached by setting the CPLEX® parameter *TiLim* to the desired value. If non-optimal termination by reaching the set time limit was observed for a particular order and density, runs were not conducted for higher orders with the same density. Furthermore, in order to control memory usage, we can set an upper limit on RAM usage by setting CPLEX® parameter *WorkMem* to a desired value and once that limit is reached the BC tree is written to the hard disk. By setting CPLEX® parameter *NodeFileInd* to an appropriate value, CPLEX® can be forced to write the BC tree as files on hard disk in different ways. When *NodeFileInd* is 0, files are not written to hard disk, when *NodeFileInd* is 1 files are not written to hard disk but the BC tree is stored in RAM in a compressed fashion (this is the default), when *NodeFileInd* is 2 files are written to hard disk without compression and when *NodeFileInd* is 3 files are written to hard disk after compression.

VII.2. Description of the Test-bed

The test-bed of instances used in our experiments consists of two broad groups. The first group consists of graphs of various order and size generated using *Sanchis generators* [168] and benchmark clique instances from the Second DIMACS challenge [84, 125]. The second group of instances are a set of Erdös collaboration networks [105]; protein interaction networks of the yeast *Saccharomyces cerevisiae* [123] and the gastric pathogen *Helicobacter Pylori* [165, 47]; a collaboration network of authors in computational geometry available from [33]; and a text-mining network based on *Reuters* news reporting following the tragic events of September 11 available from [33]. The second group instances are all large, sparse graphs and the effectiveness of the IPBC algorithm will become evident on this group.

Group I. Since the maximum k-plex problem had not been previously studied,

Table 1 DIMACS benchmarks

Graphs	n	m	d	$\omega(G)$
c-fat200-1.clq	200	1534	0.077	12
c-fat200-2.clq	200	3235	0.163	24
c-fat200-5.clq	200	8473	0.426	58
c-fat500-1.clq	500	4459	0.036	14
c-fat500-2.clq	500	9139	0.073	26
c-fat500-5.clq	500	23191	0.186	64
c-fat500-10.clq	500	46627	0.374	126
hamming6-2.clq	64	1824	0.905	32
hamming6-4.clq	64	704	0.349	4
hamming8-2.clq	256	31616	0.969	128
hamming8-4.clq	256	20864	0.639	16
hamming10-2.clq	1024	518656	0.990	512
hamming10-4.clq	1024	434176	0.829	40
johnson8-2-4.clq	28	210	0.556	4
johnson8-4-4.clq	70	1855	0.768	14
johnson16-2-4.clq	120	5460	0.765	8
johnson32-2-4.clq	496	107880	0.879	16
keller4.clq	171	9435	0.649	11
keller5.clq	776	225990	0.751	27
MANN_a9.clq	45	918	0.927	16
MANN_a27.clq	378	70551	0.990	126
MANN_a45.clq	1035	533115	0.996	345
MANN_a81.clq	3321	5506380	0.999	$\geq$1100
brock200_1.clq	200	14834	0.745	21

the most meaningful choice for benchmark instances was those developed for the maximum clique problem. Table 1 presents information regarding the 24 DIMACS benchmark instances used in our experiments. Note that some of the DIMACS instances are also graphs arising in various applications. *Johnson* and *Hamming* graph families arise in coding theory and *c-fat* graphs arise in fault diagnosis. Description of these and other DIMACS instances can be found in [112, 42, 84].

The Sanchis generator available at [84] produces hard test instances for the maximum clique problem with specified maximum clique size, number of vertices, edges and a *construction parameter*, r. The number of vertices in the generated Sanchis

graphs was varied from 100 to 1000 in steps of 100 and the edge density (d) was varied from 0.4 to 0.9 in steps of 0.1. The number of edges was calculated as $\lfloor \frac{dn(n-1)}{2} \rfloor$, where $\lfloor a \rfloor$ is the largest integer less than or equal to a. In one set of Sanchis instances the maximum clique size was fixed at $\lceil 2\log_{1/d} n \rceil$ (where $\lceil a \rceil$ is the smallest integer greater than or equal to a) and in the second set the maximum clique size was fixed at $\lceil \frac{n}{5} \rceil$. Sanchis generators typically generate hard instances [168] when the specified clique size is the expected clique size in a uniform random graph given by $\omega(G(n,p)) \sim 2\log_{1/p} n$ [40, 11]. The construction parameter r, which has to be an integer from interval $[0, \frac{n}{\omega(G)} - 1]$, also controls the difficulty level of the instances. In general smaller r values are recommended for dense graphs and larger r values are recommended if the clique number is small [168]. In our experiments r was set at $\lfloor 0.75(\frac{n}{\omega(G)} - 1) \rfloor$. Henceforth, Sanchis instances with clique number $\lceil 2\log_{1/d} n \rceil$ would be referred to as "Sanchis-log" instances and Sanchis instances with $\lceil \frac{n}{5} \rceil$ clique number as "Sanchis-linear" instances for simplicity. Group I consists of 24 DIMACS instances, 60 Sanchis instances for each maximum clique size type (log and linear) resulting in a total of 144 instances.

Group II. The collaboration network of authors in computational geometry has vertices representing authors from this area and for every two authors the number of joint works is available. This permits us to use a threshold for the edge to be included in the graph. Two authors are connected by an edge if they have (strictly) more than *threshold* joint works. We consider these graphs for *threshold* $t = 0, 1, 2$ resulting in three instances named COMP-GEOM-t.PAJ. Erdös collaboration networks are also similar, where vertices represent mathematicians and an edge indicates that the mathematicians represented by the endpoints have published together. But these collaboration networks are centered around Paul Erdös and *Erdös number* of a mathematician is his or her shortest distance to Erdös in this network. We used the

following Erdös collaboration networks available from [33, 105] in our experiments: ERDOS$-x-y$.NET, where x represents the last two digits of the year for which the network was constructed, and y represents the largest *Erdös number* of a mathematician in that graph. We considered six such networks for years 1997-1999 and $y = 1$ and 2. We will refer to ERDOS$-x-y$.NET graph as the y-neighborhood Erdös network for year x. Note that in the instances we used the vertex corresponding to Erdös himself is excluded. Apart from the aforementioned social networks the following two biological networks, protein interaction networks of *H. Pylori* and *S. cerevisiae* were also used in testing. Recall the definitions from Chapter I where we also presented a picture of the H. Pylori network. The text-mining network (DAYS.NET) from [33] is based on all stories released during 66 consecutive days beginning at 9:00 AM EST 9/11/01 by the news agency *Reuters* concerning the September 11 attack. The network is based on information compiled by Steve Corman, Kevin Dooley and Robert McPhee at the LOCKS labs in Arizona State University [75, 76]. The vertices of the network are selected words that appeared in the news and there is an edge between two words if they appear in the same text unit (sentence) and the edges are weighted with the number of co-appearances of its end-points. We use a threshold model as before and DAYS-t.PAJ refers to this network with only edges of edge weight at least $t + 1$ included. Graphs were constructed for $t = 3, 4, 5$.

Apart from the fact that Group II graphs are constructed from real-life data, another commonality among all these graphs is that they are extremely large and extremely sparse graphs that obey a power-law degree distribution.

VII.3. Numerical Results: Maximum k-Plex Problem

Our BC implementations incorporate non-dominated variable fixing introduced in Section VI.3 and MIS/co-k-plex cuts for the maximum k-plex problem introduced in Section VI.2 into the general BC framework described in Algorithm 1. Settings for the parameters introduced in Section VII.1 were arrived at after careful tuning in preliminary experiments. These values are detailed in Table 2. We set *WorkMem* to 0MB to force CPLEX® to write the BC tree to the disk immediately and avoid using any RAM. We set the *NodeFileInd* parameter to 2 enabling CPLEX® to write the BC tree to the hard disk without any compression. These measures are taken to avoid any memory shortage as the BC tree grows exponentially in size, without any significant increase in runtime [119].

Table 2 Parameter settings

Parameter	Value
HIFRACVAL	0.75
MAXLOCALCUTSPERNODE	6
TiLim	3 hours
WorkMem	0MB
NodeFileInd	2

Two versions of the BC algorithm for the maximum k-plex problem were implemented and studied on instances from Group I for $k = 1, 2$. Both versions employ the same parameter settings in Table 2 and non-dominated variable fixing, they differ in the cuts used. One version incorporates MIS cuts while the other incorporates co-k-plex cuts. Recall our earlier remarks that for $k = 1$, the MIS inequalities and co-1-plex inequalities are identical. For $k = 2$, the co-2-plex inequalities are facet defining and (weakly) dominate the MIS inequalities.

To find a local cut at some node of the BC tree we delete from the graph, vertices

fixed to 0 at that BC node to form the *residual graph* and we identify variables with high fractional value in the node's LP optimum. We then call the greedy algorithm for MIS (Algorithm 8) on the residual graph with each highly fractional variable as starting vertex to generate the cut. For the second BC version, we call the greedy algorithm for finding maximal co-k-plexes (Algorithm 9) instead of MIS. Distinct cuts from the generated cuts are identified and from them, MaxLocalCutsPerNode many most violated cuts are added to the system if available, or all the violated cuts are added if fewer than MaxLocalCutsPerNode violated cuts are found. The first version of the BC algorithm using MIS cuts is used in the IPBC algorithm presented before (Algortihm 7 in Section VI.3). Henceforth, by "BC-MIS" we refer to the first version of the BC algorithm and by "BC-CkPLX" the second version, and clearly they are different only for $k = 2$ case.

Algorithm 8 Greedy MIS Algorithm

1: **procedure** GreedyMIS(v, G) ▷ v is the starting vertex
2: initialize $I \leftarrow \emptyset$
3: **while** $V(G) \neq \emptyset$ **do**
4: $I \leftarrow I \cup \{v\}$, $G \leftarrow G - N[v]$, $v \leftarrow \text{argmin}\{deg_G(i) : i \in V(G)\}$
5: **end while**
6: **return** I
7: **end procedure**

VII.3.1. BC Algorithms on Group I Instances

In this section we only provide a summary of our experimental results on BC for $k = 1$, BC-MIS and BC-C2PLX for $k = 2$ on Sanchis instances. Complete details of the experiments with Sanchis instances are provided in Appendix A. Complete

Algorithm 9 Greedy Co-k-plex Algorithm

procedure GREEDYCO-k-PLEX(v, G) ▷ v is the starting vertex
  initialize $I \leftarrow$ GREEDYMIS(v, G)
  $G \leftarrow G - I$
  while $V(G) \neq \emptyset$ **do**
    $u \leftarrow \text{argmin}\{deg_G(i) : i \in V(G)\}$
    if $I \cup u$ is a co-k-plex **then** ▷ see Remark 12 in Section VI.2
      $I \leftarrow I \cup u$
    end if
    $G \leftarrow G - u$
  end while
  return I
end procedure

results for DIMACS instances are provided here for the aforementioned algorithms.

The largest order up to which optimal resolution was possible on Sanchis instances within the 3 hour time limit using specified algorithm for each density is presented in Table 3 and Table 4. Exponential growth in the number of BC nodes and hence in running time were observed, which is not surprising given the intractability of the problem. As noted before, Sanchis generators are known to produce hard instances when specified maximum clique size is around the expected clique number of a uniform random graph. This is observed in our case also as we perform consistently better on Sanchis-linear instances compared to Sanchis-log instances for all algorithms. Generally speaking, our ability to solve larger Sanchis-linear instances decreases with increase in edge density. This trend is also observed in Sanchis-log instances with the exception that 90% dense instances appear to be relatively easier.

We also observe that for all algorithms, on all Sanchis instances we perform better when $k = 1$ compared to $k = 2$. This could be explained by noting that the number of feasible solutions, as well as possibility of alternate optima is higher when $k = 2$ compared to $k = 1$. Finally, between the two versions for $k = 2$, BC-MIS is consistently better compared to BC-C2PLX despite the fact that co-2-plex inequalities are theoretically the better choice. This observation clearly demands further investigation and clarification which follows.

Table 3 Summary of results on Sanchis-log instances

Algorithm	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
BC, $k = 1$	700	500	400	400	300	300
BC-MIS, $k = 2$	300	200	200	100	100	200
BC-C2PLX, $k = 2$	100	100	100	100	100	200

Table 4 Summary of results on Sanchis-linear instances

Algorithm	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
BC, $k = 1$	1000	1000	1000	800	500	100
BC-MIS, $k = 2$	900	500	400	200	< 100	100
BC-C2PLX, $k = 2$	300	200	100	100	< 100	< 100

The greedy algorithm for finding maximal co-2-plexes is more time consuming compared to finding MIS. When a vertex is added to an independent set, we can delete all its neighbors and proceed until we find a MIS. On the other hand, only vertices that can be deleted while finding a maximal co-2-plex are the vertices that have already been added and the ones outside that cannot be added to the current set. Both neighbors and non-neighbors that do not belong to either of those cases must be considered until all the vertices have been classified into one of the two groups. Due to this simple fact, even though we take a fast and greedy approach, the cut

generation heuristic is relatively expensive. As a consequence, more time is spent at each node and fewer nodes are enumerated within the specified time limit by the BC-C2PLX algorithm. The performance of BC-MIS is better only due to the fact that more BC nodes can be enumerated within the specified time limit, either leading to optimal resolution or identification of better feasible solution and upper-bound (see Appendix A for details).

As a sample illustration, consider the results in Table 5. The superscript † indicates non-optimal termination when time limit set by parameter *TiLim* was reached and "-" indicates instances not attempted. The running time of BC-MIS is much smaller than BC-C2PLX, which is due to smaller running times at each BC node. However the trend predicted by theory that is illustrated by the experiments is the fact that the number of BC nodes enumerated is significantly smaller for BC-C2PLX as maximal co-2-plex inequalities are facet-inducing and weakly dominate MIS inequalities, providing stronger cuts and hence, better bounds at each BC node. Note that this observation is made from the instances terminating optimally, and it is not meaningful to compare the number of BC nodes enumerated in non-optimal cases as the enumeration would have been cut short by the time limit. This observation is critical for the following reasons. Firstly, our choice of the algorithm depends on our need *i.e.*, BC-MIS is recommended whenever the run must be conducted within a short time limit. Whenever long or no time limit is set, BC-C2PLX is preferable. Secondly, given the intractability of the problem it is obviously more important to reduce the size of the tree by stronger bounding as offered by the BC-2PLX algorithm for large instances. An important topic for future research is to develop faster heuristics to generate co-k-plex cuts, which would greatly alleviate the present problem.

Our computational experience with the DIMACS benchmarks using BC (for $k = 1$), BC-MIS (for $k = 2$) and BC-C2PLX (for $k = 2$) are documented in Table 6, Table 7

Table 5 BC-MIS Vs. BC-C2PLX

Performance Measure	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
	BC-MIS Sanchis-log instances $n = 100$					
Running times (secs)	11.922	29.126	29.235	19.36	1.672	0.015
No. of BC Nodes	4687	17291	16482	11356	978	0
No. of Cuts	1740	4903	6462	6575	544	0
	BC-C2PLX Sanchis-log instances $n = 100$					
Running times (secs)	124.925	307.086	478.996	505.872	56.47	0.016
No. of BC Nodes	2490	7839	6536	4340	304	0
No. of Cuts	1559	4842	5862	5448	362	0
	BC-MIS Sanchis-linear instances $n = 100$					
Running times (secs)	2.016	6.312	21.907	325.18	10800.3†	291.586
No. of BC Nodes	407	2139	12609	224245	1223136†	178159
No. of Cuts	414	1320	5089	53805	318711†	66803
	BC-C2PLX Sanchis-linear instances $n = 100$					
Running times (secs)	18.204	107.096	426.464	4109.14	10800.3†	10800.6†
No. of BC Nodes	273	876	5437	101207	126558†	56697†
No. of Cuts	456	1194	5225	81143	109238†	37941†
	BC-MIS Sanchis-linear instances $n = 200$					
Running times (secs)	16.751	92.909	179.192	2025.52	10800.5†	10800.5†
No. of BC Nodes	678	4574	21258	404025	1225284†	958143†
No. of Cuts	877	5115	16249	260290	1037180†	666585†
	BC-C2PLX Sanchis-linear instances $n = 200$					
Running times (secs)	667.423	5418.7	10802.1†	10800.8†	-	-
No. of BC Nodes	724	2200	2760†	4315†	-	-
No. of Cuts	1091	4142	3724†	5096†	-	-

and Table 8 respectively. In the tables, $[l, u]$ represent lower and upper bounds on $\omega_k(G)$ in case of non-optimal termination and $*$ indicates that the maximum k-plex was found even though termination was not optimal (applicable only for $k = 1$). This is in addition to † and "-" as described before.

VII.3.2. IPBC Algorithm on Group II Instances

The first difficulty in solving instances from Group II using *only* the existing BC implementations was not the worst-case exponential growth that is expected in the

Table 6 Results of BC for $k = 1$ on DIMACS instances

Graph	n	d	Time (secs)	BC Nodes	Cuts	$\omega_1(G)$
c-fat200-1.clq	200	0.0770854	9.126	257	238	12
c-fat200-2.clq	200	0.162563	8.72	408	433	24
c-fat200-5.clq	200	0.425779	6.079	551	809	58
c-fat500-1.clq	500	0.0357435	151.03	608	278	14
c-fat500-2.clq	500	0.0732585	176.044	591	387	26
c-fat500-5.clq	500	0.1859	367.928	3587	7296	64
c-fat500-10.clq	500	0.373764	223.003	3753	7337	126
hamming6-2.clq	64	0.904762	0.016	0	0	32
hamming6-4.clq	64	0.349206	0.219	118	75	4
hamming8-2.clq	256	0.968627	0.016	0	0	128
hamming8-4.clq	256	0.639216	44.379	4639	7203	16
hamming10-2.clq	1024	0.990225	0.031	0	0	512
hamming10-4.clq	1024	0.828934	$10800.4^{\dagger}$	$130904^{\dagger}$	$71067^{\dagger}$	$[34, 385]$
johnson8-2-4.clq	28	0.555556	0.031	32	4	4
johnson8-4-4.clq	70	0.768116	0.594	530	547	14
johnson16-2-4.clq	120	0.764706	2130.31	888801	144445	8
johnson32-2-4.clq	496	0.878788	$10800.7^{\dagger}$	$548602^{\dagger}$	$62204^{\dagger}$	$[16, 104]^{*}$
keller4.clq	171	0.649123	82.817	37102	28985	11
keller5.clq	776	0.751546	$10800.4^{\dagger}$	$140638^{\dagger}$	$86221^{\dagger}$	$[27, 100]^{*}$
MANN_a9.clq	45	0.927273	0.296	59	51	16
MANN_a27.clq	378	0.990148	$10800.4^{\dagger}$	$541716^{\dagger}$	$20429^{\dagger}$	$[125, 148]$
MANN_a45.clq	1035	0.9963	$10800.6^{\dagger}$	$498897^{\dagger}$	$6129^{\dagger}$	$[339, 422]$
MANN_a81.clq	3321	0.998825	$10800.4^{\dagger}$	$202799^{\dagger}$	$1822^{\dagger}$	$[1096, 1387]$
brock200_1.clq	200	0.745427	4499.57	1143351	1590922	21

Table 7 Results of BC-MIS for $k = 2$ on DIMACS instances

Graph	n	d	Time (secs)	BC Nodes	Cuts	$\omega_2(G)$
c-fat200-1.clq	200	0.0770854	25.891	1507	267	12
c-fat200-2.clq	200	0.162563	24.235	799	50	24
c-fat200-5.clq	200	0.425779	90.564	4104	12	58
c-fat500-1.clq	500	0.0357435	1263.81	20702	1605	14
c-fat500-2.clq	500	0.0732585	2985.04	27166	2150	26
c-fat500-5.clq	500	0.1859	10142.8	24446	922	64
c-fat500-10.clq	500	0.373764	$10800.5^{\dagger}$	$17865^{\dagger}$	$258^{\dagger}$	$[126, 150]$
hamming6-2.clq	64	0.904762	0.421	199	0	32
hamming6-4.clq	64	0.349206	4.609	2894	849	6
hamming8-2.clq	256	0.968627	$10800.3^{\dagger}$	$828372^{\dagger}$	$0^{\dagger}$	$[128, 130]$
hamming8-4.clq	256	0.639216	$10800.2^{\dagger}$	$707965^{\dagger}$	$468601^{\dagger}$	$[16, 33]$
hamming10-2.clq	1024	0.990225	$10800.3^{\dagger}$	$127934^{\dagger}$	$0^{\dagger}$	$[512, 533]$
hamming10-4.clq	1024	0.828934	$10800.4^{\dagger}$	$84322^{\dagger}$	$47755^{\dagger}$	$[43, 459]$
johnson8-2-4.clq	28	0.555556	1.952	1983	332	5
johnson8-4-4.clq	70	0.768116	1951.87	883011	194214	14
johnson16-2-4.clq	120	0.764706	$10800.2^{\dagger}$	$631945^{\dagger}$	$146745^{\dagger}$	$[10, 32]$
johnson32-2-4.clq	496	0.878788	$10800.3^{\dagger}$	$521882^{\dagger}$	$40228^{\dagger}$	$[21, 214]$
keller4.clq	171	0.649123	$10800.2^{\dagger}$	$1304605^{\dagger}$	$827419^{\dagger}$	$[15, 21]$
keller5.clq	776	0.751546	$10800.4^{\dagger}$	$79412^{\dagger}$	$49483^{\dagger}$	$[31, 320]$
MANN_a9.clq	45	0.927273	0.262	19	11	26
MANN_a27.clq	378	0.990148	$10800.3^{\dagger}$	$677877^{\dagger}$	$40865^{\dagger}$	$[236, 238]$
MANN_a45.clq	1035	0.9963	$10800.6^{\dagger}$	$421833^{\dagger}$	$138879^{\dagger}$	$[662, 671]$
MANN_a81.clq	3321	0.998825	$10800.2^{\dagger}$	$57013^{\dagger}$	$23655^{\dagger}$	$[2162, 2184]$
brock200_1.clq	200	0.745427	$10800.5^{\dagger}$	$1111998^{\dagger}$	$1059670^{\dagger}$	$[25, 49]$

Table 8 Results of BC-C2PLX for $k = 2$ on DIMACS instances

Graph	n	d	Time (secs)	BC Nodes	Cuts	$\omega_2(G)$
c-fat200-1.clq	200	0.0770854	212.239	2091	1576	12
c-fat200-2.clq	200	0.162563	7636.49	6943	6525	24
c-fat200-5.clq	200	0.425779	5006.05	1778	1526	58
c-fat500-1.clq	500	0.0357435	9587.21	6149	4085	14
c-fat500-2.clq	500	0.0732585	$10800.4^{\dagger}$	$5839^{\dagger}$	$6663^{\dagger}$	[26, 108]
c-fat500-5.clq	500	0.1859	$10821.4^{\dagger}$	$865^{\dagger}$	$964^{\dagger}$	[64, 167]
c-fat500-10.clq	500	0.373764	$10810^{\dagger}$	$465^{\dagger}$	$253^{\dagger}$	[126, 199]
hamming6-2.clq	64	0.904762	1.686	59	127	32
hamming6-4.clq	64	0.349206	6.767	1747	883	6
hamming8-2.clq	256	0.968627	$10803.4^{\dagger}$	$775^{\dagger}$	$2079^{\dagger}$	[128, 134]
hamming8-4.clq	256	0.639216	$10800.3^{\dagger}$	$11082^{\dagger}$	$6514^{\dagger}$	[16, 96]
hamming10-2.clq	1024	0.990225	$11201.9^{\dagger}$	$82^{\dagger}$	$60^{\dagger}$	[512, 537]
hamming10-4.clq	1024	0.828934	$10802.2^{\dagger}$	$395^{\dagger}$	$32^{\dagger}$	[43, 465]
johnson8-2-4.clq	28	0.555556	1.171	1567	638	5
johnson8-4-4.clq	70	0.768116	3283.15	540326	296875	14
johnson16-2-4.clq	120	0.764706	$10800.3^{\dagger}$	$570165^{\dagger}$	$219650^{\dagger}$	[10, 19]
johnson32-2-4.clq	496	0.878788	$10801.1^{\dagger}$	$6832^{\dagger}$	$744^{\dagger}$	[21, 228]
keller4.clq	171	0.649123	$10800.5^{\dagger}$	$27887^{\dagger}$	$26647^{\dagger}$	[15, 61]
keller5.clq	776	0.751546	$11170.8^{\dagger}$	$146^{\dagger}$	$126^{\dagger}$	[30, 364]
MANN_a9.clq	45	0.927273	0.289	21	15	26
MANN_a27.clq	378	0.990148	$10851.3^{\dagger}$	$79^{\dagger}$	$159^{\dagger}$	[236, 242]
MANN_a45.clq	1035	0.9963	$31343^{\dagger}$	$1^{\dagger}$	$6^{\dagger}$	[661, 742]
MANN_a81.clq	3321	0.998825	$10800.2^{\dagger}$	$0^{\dagger}$	$0^{\dagger}$	[2161, 2430]
brock200_1.clq	200	0.745427	$10802.1^{\dagger}$	$8830^{\dagger}$	$8278^{\dagger}$	[24, 81]

BC tree. It was CPLEX® running out of memory required to build a massive integer program resulting in a crash. Group II instances are power law graphs that are very sparse, and recall that our formulation in such a case results in an extremely dense constraint matrix even though it is of size $n \times n$. Note that our memory management in the BC implementation using CPLEX® parameters *WorkMem* and *NodeFileInd* is applicable when the BC tree is being created but the crash occurred while building the IP at the root node even before any solution procedure was attempted. Memory management and data handling are important when dealing with massive amounts of data. In our case we have only utilized real-life power law graphs of moderate size (relatively) that have 1000 to 7000 vertices with very low edge density. Much larger networks have been handled in literature using more advanced techniques [6, 7, 37]. But we observed that even these moderately large instances could not be attempted using a straightforward BC algorithm as CPLEX® crashed due to memory shortage. This hurdle motivated us to develop specialized techniques that exploit the properties of the problem and sufficiently general properties of the data in developing effective exact approaches.

In such situations, the IPBC algorithm (see Algorithm 7 in Section VI.3 for details) is an effective alternative as it invokes BC procedure only on a smaller graph (assuming the original network is very sparse) and only if it is necessary after several preprocessing procedures have been applied. This enables us to keep the memory usage under control and hence solve the instance optimally however long it takes. In our implementation, the only time limit imposed was via CPLEX® parameter *TiLim* which was set to 3 hours as before, but this is for each BC call. Hence the implementation in the worst-case could take $3n$ hours to run. However, this was never observed in practice.

It is also important to note that attempting to solve dense instances (relative to

Group II graphs, Sanchis graphs with $d = 0.4$ are also dense) using IPBC could take prohibitively long time. Since in each iteration, a graph of almost the same size as the original, which does not vary much from iteration to iteration, could be presented to the BC procedure. This is much worse than providing the entire graph at once to the BC procedure. The IPBC algorithm is hence suited only for very large and very sparse graphs such as the instances considered in Group II.

Erdös Collaboration Networks. All Erdös graphs were optimally resolved using the IPBC algorithm. In Table 9, we present the graph properties including the k-plex numbers found by the IPBC algorithm. In Table 10, we present the runtime results from this experiment. The column "IPBC Time" is the total time taken by the IPBC algorithm excluding read-write times. The number of calls made to CPLEX® is given under column "#BC calls" column and "BC Time" gives the cumulative time taken by CPLEX® in those calls. Note that by spending significant amount of time on preprocessing we have drastically cut short the time required by the BC procedure as the instances solved were significantly small. To illustrate this fact we provide the following sample results. Detailed BC call statistics are provided in Table 11 for the $k = 1$ case on ERDOS-98-2.NET which corresponds to the maximum number of BC calls and author names (reproduced verbatim from data file) of members in a maximum 3-plex identified in ERDOS-99-1.NET is provided in Table 12. Complete list of authors in identified k-plexes is provided in Appendix A.

Remark 14. The shifting of computational burden from BC to preprocessing which is extremely effective for power law graphs is the key contribution of this approach, and it will be observed henceforth on all instances from Group II.

Protein Interaction Networks. Protein interaction networks of *H. Pylori* and *S. cerevisiae* were also optimally resolved using IPBC algorithm. The properties of

Table 9 Erdös networks: The number of vertices, edges, edge density, and k-plex numbers for $k = 1, 2, 3$

Graph	n	m	d	$\omega_1(G)$	$\omega_2(G)$	$\omega_3(G)$
ERDOS-97-1.NET	472	1314	0.0118212	7	8	9
ERDOS-98-1.NET	485	1381	0.0117662	7	8	9
ERDOS-99-1.NET	492	1417	0.0117315	7	8	9
ERDOS-97-2.NET	5488	8972	0.0005959	7	8	9
ERDOS-98-2.NET	5822	9505	0.0005609	7	8	9
ERDOS-99-2.NET	6100	9939	0.0005343	8	8	9

Table 10 Results for Erdös networks using IPBC algorithm

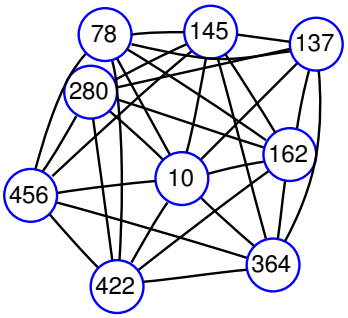
k	Graph	IPBC Time	BC Time	#BC Calls
1	ERDOS-97-1.NET	3.438	0.547	10
	ERDOS-98-1.NET	3.281	0.5	9
	ERDOS-99-1.NET	3.25	0.392	9
	ERDOS-97-2.NET	675.828	2.201	16
	ERDOS-98-2.NET	908.969	1.047	16
	ERDOS-99-2.NET	1058.89	1.123	14
2	ERDOS-97-1.NET	4.031	1.859	6
	ERDOS-98-1.NET	7.734	5.344	6
	ERDOS-99-1.NET	8.547	6.032	6
	ERDOS-97-2.NET	754.266	88.217	7
	ERDOS-98-2.NET	981.704	102.141	7
	ERDOS-99-2.NET	1151.31	101.702	9
3	ERDOS-97-1.NET	6.391	4.141	6
	ERDOS-98-1.NET	7.547	5.124	6
	ERDOS-99-1.NET	8.203	5.639	6
	ERDOS-97-2.NET	1007.7	342.936	7
	ERDOS-98-2.NET	1543.52	653.875	7
	ERDOS-99-2.NET	1891.3	837.015	9

Table 11 BC call statistics of IPBC algorithm on ERDOS-98-2.NET for $k = 1$ (n', m' and d' denote number of vertices, edges and density of the reduced instance solved in that BC call)

Call No.	n'	m'	d'	Fixed vertex	BC time
1	452	2094	0.0205443	182	0.719
2	141	859	0.0870314	10	0.063
3	46	256	0.247343	399	0.016
4	140	829	0.0852004	160	0.047
5	46	236	0.228019	84	0.015
6	92	532	0.12709	244	0.015
7	36	184	0.292064	320	0.016
8	83	476	0.139877	454	0.031
9	60	327	0.184746	277	0.015
10	78	411	0.136863	360	0.031
11	58	298	0.180278	76	0.016
12	34	169	0.301248	67	0.016
13	23	99	0.391304	472	0
14	29	123	0.302956	21	0.015
15	22	90	0.38961	325	0.016
16	10	39	0.866667	221	0.016

Table 12 Members of a maximum 3-plex in ERDOS-99-1.NET

10: Noga Alon
78: Fan Rong K. Chung
137: Peter Frankl
145: Zoltan Furedi
162: Ronald L. Graham
280: Laszlo Lovasz
364: Vojtech Rodl
422: Joel H. Spencer
456: William T. Trotter



these graphs including the k-plex numbers for $k = 1, 2, 3$ are provided in Table 13. The runtime information is not reported since both H. Pylori and S. cerevisiae instances were resolved in under a total of 30 and 70 seconds respectively for all k values. Fig. 10, Fig. 11, Fig. 12 and Fig. 13 illustrate the maximum k-plex identified in each case. Note that vertex pairs that are non-adjacent correspond to protein pairs that could be experimentally investigated for interactions that were previously unknown.

Table 13 Protein interaction networks: The number of vertices, edges, edge density, and k-plex numbers for $k = 1, 2, 3$

Graph	n	m	d	$\omega_1(G)$	$\omega_2(G)$	$\omega_3(G)$
H. Pylori	1570	1399	0.00113586	3	5	6
S. cerevisiae	2112	2203	0.00098824	6	6	7

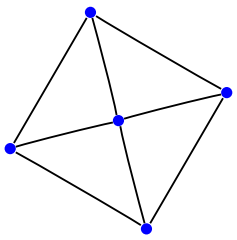


Fig. 10 A maximum 2-plex in H. Pylori

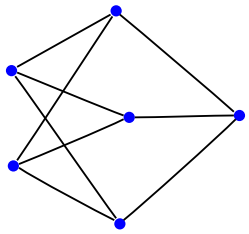


Fig. 11 A maximum 3-plex in H. Pylori

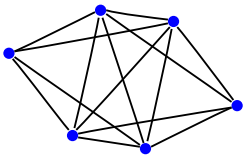


Fig. 12 A maximum 2-plex in S. cerevisiae

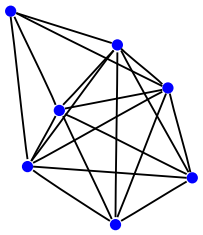


Fig. 13 A maximum 3-plex in S. cerevisiae

Computational Geometers Collaboration Network. COMP-GEOM-t.PAJ represents the collaboration network of authors in computational geometry such that every

edge indicates that the authors represented by the end points have at least $t+1$ joint works. We considered three such networks with $t = 0, 1, 2$ and solved maximum k-plex problem optimally using IPBC algorithm for $k = 1, 2, 3$. The properties of these networks including k-plex numbers found are presented in Table 14. Table 15 presents the runtime results from this experiment. The column labels are as described before. Note that on these instances very few BC calls were necessary to optimally solve the problem and hence detailed BC call statistics are not reported. Complete list of authors in identified k-plexes is provided in Appendix A.

Table 14 Computational geometers collaboration networks: The number of vertices, edges, edge density, and k-plex numbers for $k = 1, 2, 3$

Graph	n	m	d	$\omega_1(G)$	$\omega_2(G)$	$\omega_3(G)$
COMP-GEOM-0.PAJ	7343	11898	0.000441383	22	22	22
COMP-GEOM-1.PAJ	7343	3939	0.000146126	10	10	11
COMP-GEOM-2.PAJ	7343	1976	7.33E-05	8	8	10

Table 15 Results for computational geometers collaboration networks using IPBC algorithm

k	Graph	IPBC Time	BC Time	#BC Calls
1	COMP-GEOM-0.PAJ	7381.83	1.235	1
	COMP-GEOM-1.PAJ	1360.36	0.375	3
	COMP-GEOM-2.PAJ	927.078	0.203	2
2	COMP-GEOM-0.PAJ	7888.33	498.656	1
	COMP-GEOM-1.PAJ	1409.63	58.5	2
	COMP-GEOM-2.PAJ	930.172	6.765	3
3	COMP-GEOM-0.PAJ	7829.72	360.312	1
	COMP-GEOM-1.PAJ	1407.47	54.297	2
	COMP-GEOM-2.PAJ	927.875	8.453	1

Reuters Terror News Networks. Recall that DAYS-t.PAJ represents the text network where vertices represent words and every edge indicates that the words represented by the end points appear in at least $t + 1$ sentences together. We consider

three such networks with $t = 3, 4, 5$ and solve maximum k-plex problem using IPBC algorithm on these networks for $k = 1, 2, 3$. The properties of these networks including k-plex numbers found are presented in Table 16. Table 17 presents the runtime results from this experiment. The columns are labeled as before. These graphs were among the hardest Group II instances. Table 18 lists the words included in maximum k-plexes identified for $k = 1, 2, 3$ on one of the instances. Words belonging to maximum k-plexes identified for $k = 1, 2, 3$ on all instances are detailed in Appendix A.

Table 16 Reuters terror news networks: The number of vertices, edges, edge density, and k-plex numbers for $k = 1, 2, 3$

Graph	n	m	d	$\omega_1(G)$	$\omega_2(G)$	$\omega_3(G)$
DAYS-3.PAJ	13332	9000	0.000101278	8	10	11
DAYS-4.PAJ	13332	5159	5.81E-05	7	8	9
DAYS-5.PAJ	13332	3404	3.83E-05	6	7	8

Table 17 Results for the Reuters terror news networks using IPBC algorithm

k	Graph	IPBC Time	BC Time	#BC Calls
1	DAYS-3.PAJ	7600.34	66.516	33
	DAYS-4.PAJ	5634.09	13.609	23
	DAYS-5.PAJ	5068.78	0.921	17
2	DAYS-3.PAJ	8421.91	1002.91	28
	DAYS-4.PAJ	5842.64	210.423	21
	DAYS-5.PAJ	5046.03	60.375	17
3	DAYS-3.PAJ	9628.45	2605.95	25
	DAYS-4.PAJ	6036.13	505.735	20
	DAYS-5.PAJ	5069.31	68.375	17

VII.4. Maximum Clique Problem: Formulation Study

In this section we study the effect of using *1-plex formulation* (OPF) and *edge formulation* (EF) for the maximum clique problem (see Section VI.4 for details) in a BC algorithm. The OPF has the advantage that it is compact, but it is a theoretically

Table 18 Words belonging to a maximum k-plex identified in DAYS-5.PAJ for $k = 1, 2, 3$

$k = 1$	$k = 2$	$k = 3$
attack	attack	attack
new_york	new_york	new_york
pentagon	pentagon	pentagon
plane	plane	plane
world_trade_ctr	world_trade_ctr	world_trade_ctr
tuesday	tuesday	tuesday
	tower	tower
		twin

weaker relaxation and hence is expected to produce loose bounds in the BC tree. The EF can have a large number of constraints and hence a larger system being solved at the nodes of BC tree but possibly producing tighter bounds. We utilize BC implementations that are identical in every respect except for the formulation used. The implementation has *non-neighbor fixing* (the special case of non-dominated variable fixing introduced in Section VI.3 when $k = 1$), MIS cutting planes found using Algorithm 8 as described in Section VII.3, and parameter settings from Table 2 with the only exception of HiFracVal set to 0. This is because the extreme points of the LP relaxation polytope of EF have $\{0, \frac{1}{2}, 1\}$ components (see Section II.6.1). We solved the maximum clique problem on Dimacs benchmarks using both formulations. Out of the 24 Dimacs instances, we are not reporting the results for instances which were optimally resolved with both formulations under 1 second. Results for the remaining 16 instances is presented in Table 19. For non-optimal cases indicated by †, we report $[l, u]$ where l is the size of best clique found and u is an upper-bound on $\omega(G)$.

Out of the 16 instances considered, 11 were optimally resolved using both formulations. In 8 out of 11 cases OPF ran faster than EF while in 9 out of 11 cases EF enumerated fewer BC nodes compared to OPF. Out of the 16 instances, 4 were

non-optimally terminated for both formulations, with EF and OPF finding the same lower-bound in 2, EF finds a better lower-bound in one and OPF in the other. However, the upper-bound returned by EF was strictly better in 3 cases and was the same as OPF upper-bound in 1 case. One instance that was not optimally solved using EF was solved optimally using OPF. Even on this small set of benchmarks, OPF seems to be better in speed and lower-bounds while EF seems to be better in terms of tight upper-bounds. An efficient way to "mix" the two formulations could be a problem for future study.

VII.5. Numerical Results: Maximum 2-Club Problem

As mentioned before, extensive testing of algorithms for the maximum 2-club problem as conducted for the maximum k-plex problem was not possible since most instances from Group I have diameter two. It is imperative that tough instances from practice are benchmarked for this problem as well as procedures identified that generate difficult instances for the problem with a known optimum. Preliminary results we present here are for the Group II graphs on which we solve the maximum 2-club problem using the ITBC algorithm. See Algorithm 4 in Section V.4 for details. The settings used in the ITBC implementation are as follows: the BC parameter settings are as listed in Table 2, maximal 2-independent set facets (see Section V.3) are used as cutting planes in the BC procedure are generated at every BC node by finding a MIS containing a highly fractional starting vertex using the greedy approach (Algorithm 8) on the square graph. As per the parameter settings, local cuts are generated from vertices that have a high fractional value in that node's LP relaxation and MaxLocalCutsPerNode many most violated cuts are added to re-solve the problem at each BC node.

Table 19 Edge formulation Vs. 1-plex formulation on DIMACS instances

	OPF			EF		
Graph	Time(secs)	$\omega(G)$	BC nodes	Time(secs)	$\omega(G)$	BC nodes
c-fat200-1.clq	85.867	12	333	164.067	12	376
c-fat200-2.clq	47.989	24	422	88.363	24	323
c-fat200-5.clq	16.298	58	557	18.172	58	151
c-fat500-1.clq	6155.93	14	772	$10800.4^{\dagger}$	[14, 165]	742
c-fat500-2.clq	6869.41	26	1659	8293.41	26	988
c-fat500-5.clq	3059.87	64	2591	3889.55	64	727
c-fat500-10.clq	1305.46	126	7522	1411.97	126	476
hamming8-4.clq	183.974	16	1435	401.17	16	2636
hamming10-4.clq	$10801.1^{\dagger}$	[40, 250]	3725	$10800.7^{\dagger}$	[38, 153]	1696
johnson32-2-4.clq	7.703	16	10	0.063	16	0
keller4.clq	138.216	11	9592	84.332	11	3641
keller5.clq	$10800.6^{\dagger}$	[24, 57]	7335	$10801.1^{\dagger}$	[26, 32]	2722
MANN_a27.clq	2243.03	126	346497	31.798	126	6257
MANN_a45.clq	$10800.2^{\dagger}$	[345, 347]	204343	$10800.4^{\dagger}$	[345, 347]	306948
MANN_a81.clq	$10800.5^{\dagger}$	[1098, 1132]	19746	$10800.6^{\dagger}$	[1098, 1120]	22201
brock200_1.clq	2880.43	21	370717	3457.64	21	176683

Table 20 Results of ITBC algorithm on Group II instances

Graph	ITBC Time (secs)	BC Time (secs)	#BC Calls	$\bar{\omega}_2(G)$
ERDOS971.NET	3.313	0.813	3	42
ERDOS981.NET	3.5	0.656	3	43
ERDOS991.NET	3.875	0.876	3	43
ERDOS972.NET	605.313	0	0	258
ERDOS982.NET	774.625	0	0	274
ERDOS992.NET	908.704	0	0	277
H. Pylori	15.859	0	0	56
S. cerevisiae	56.703	0	0	57
COMP-GEOM-0.PAJ	7172.17	0	0	103
COMP-GEOM-1.PAJ	1254.5	0	0	71
COMP-GEOM-2.PAJ	889.141	0	0	50
DAYS-3.PAJ	5818.19	0	0	683
DAYS-4.PAJ	4849.47	0	0	434
DAYS-5.PAJ	4630.09	0	0	308

All instances were solved to optimality and the results are presented in Table 20. Column "ITBC Time" lists the total time taken by the ITBC algorithm excluding read/write times, the number of calls to CPLEX® is in "#BC calls" column and "BC Time" is the cumulative time taken by CPLEX® for those calls. Note that for most instances BC was unnecessary. This is because the closed neighborhood of the maximum degree vertex used as the starting 2-club was optimal given the power law nature of Group II instances and preprocessing procedures were sufficient to establish optimality. In the cases where BC calls were made (3 calls for 1-neighborhood Erdös network for each year), the cumulative BC time of all three calls was under a second. Also note that the 2-club number is significantly larger than the 2-plex number for each instance. Depending on our needs, an optimum as large as these could be an advantage or a disadvantage.

Summary of Results. This chapter presented our computational experience with the maximum k-plex problem. Reasonable parameter settings were identified after

preliminary experimentation with the BC implementation. These settings were used to implement two versions of BC and the IPBC algorithm developed in Chapter VI. Performance of the two versions of the BC algorithm is studied closely on different types of instances and recommendations are made to aid proper selection.

IPBC algorithm was found to be extremely effective for handling and solving to optimality, the maximum k-plex problem on very large and very sparse graphs such as real-life power law graphs. This chapter also presents a computational study of the novel IP formulation for the maximum clique problem proposed in this dissertation. Preliminary results on the ITBC algorithm for solving maximum 2-club problem on large, sparse graphs are also presented.

CHAPTER VIII

CONTINUOUS GLOBAL OPTIMIZATION FORMULATIONS FOR INDEPENDENCE NUMBER OF A GRAPH*

Cliques and independent sets are two of the most researched combinatorial objects with deep and diverse results surrounding them, especially motivated by their vast array of applications. Several of these results and applications were discussed in Chapter II. Our contribution to the literature on cliques thus far is the novel integer program presented in Chapter VI that was studied computationally in Chapter VII. In this chapter we take a continuous optimization approach to the discrete optimization problem of finding maximum independent sets. We present a box-constrained continuous fractional formulation of the maximum independent set problem and characterize its local and global maxima. These results contribute to the recent progress made by studying CO problems using continuous approaches. Some of those results from literature including the classical Motzkin-Straus formulation [150] of maximum clique problem were already discussed in Chapter II.

---

*Parts of this chapter are reprinted with permission from Balasundaram, B., Butenko, S.: Constructing test functions for global optimization using continuous formulations of graph problems. Journal of Optimization Methods and Software **20**(4-5), 439–452 (2005) © Taylor&Francis and from Balasundaram, B., Butenko, S.: On a polynomial fractional formulation for independence number of a graph. Journal of Global Optimization 35(3), 405–421 (2006) © Springer.

VIII.1. Continuous Formulation for Independence Number

Theorem 15. *The independence number $\alpha(G)$ satisfies the following global optimization formulation:*

$$\alpha(G) = \max_{x\in[0,1]^n} \sum_{i\in V(G)} \frac{x_i}{1+\sum\limits_{j\in N(i)} x_j} \tag{8.1}$$

Proof. Formulation (8.1) can be deduced from a stronger result in [108] (see formulation (2.6) in Section II.6.2) that utilizes an elegant probabilistic proof. We provide an alternative deterministic proof, that permits us to study its local maxima properties subsequently. Denote by $f(x)$ the objective function in (8.1), *i.e.*,

$$f(x) = \sum_{i\in V} \frac{x_i}{1+\sum\limits_{j\in N(i)} x_j}.$$

and let $f(G) = \max_{0\le x_i\le 1, i=1,\ldots,n} f(x)$. We need to show that $f(G) = \alpha(G)$. Note that $f(x)$ is a continuous function and $[0,1]^n = \{(x_1, x_2, \ldots, x_n) : 0 \le x_i \le 1, i = 1, \ldots, n\}$ is a compact set. Hence, there always exists $x^* \in [0,1]^n$ such that $f(G) = f(x^*)$.

Next we show that *every* local maximum of (8.1) is binary and hence there exists an optimal 0-1 solution to the problem. Partition V into three disjoint sets as: $V = \{v\} \cup N(v) \cup S$, for some fixed $v \in V$, where $S = V \setminus (v \cup N(v))$. We can rewrite $f(x)$ in the form

$$f(x) = x_v A_v(x) + B_v(x) + C_v(x)$$

where,

$$A_v(x) = \frac{1}{1+\sum\limits_{j\in N(v)} x_j}$$

$$B_v(x) = \sum_{i\in N(v)} \frac{x_i}{x_v + 1 + \sum\limits_{j\in N(i)\setminus\{v\}} x_j}$$

$$C_v(x) = \sum_{i\in S} \frac{x_i}{1+\sum\limits_{j\in N(i)} x_j}$$

We now show that $f(x)$ is a convex function with respect to each variable (x_v). For any $x \in [0,1]^n$, we fix x_i for all $i \neq v$, and treat $f(x)$ as a function of single variable $x_v \in [0,1]$. Observe that $x_v A_v(x) + C_v(x)$ is linear with respect to x_v as $A_v(x), C_v(x)$ are independent of x_v and hence is convex. $B_v(x)$ is a sum of functions of the form

$$g_i(x_v) = \frac{a_i}{x_v + b_i},$$

where $0 \leq a_i \leq 1, b_i = 1 + \sum_{j \in N(i) \setminus \{v\}} x_j$, and a_i, b_i do not depend on x_v. The above function is convex in the region $x_v > -b_i$, so

$$\frac{x_i}{x_v + 1 + \sum_{j \in N(i) \setminus \{v\}} x_j}$$

is convex as a function of x_v for $x_v > -1 - \sum_{j \in N(i) \setminus \{v\}} x_j$, and hence it is convex for $x_v \in [0,1]$. Thus $B_v(x)$ is also convex as a function of x_v. Therefore, $f(x) = x_v A_v(x) + B_v(x) + C_v(x)$ is a convex function with respect to x_v. In fact, $f(x)$ is strictly convex with respect to x_v, unless $x_i = 0$ for all $i \in N(v)$. A strictly convex function over a closed interval can have a local maximum only at an endpoint of the interval (in our case 0 or 1). On the other hand, if $x_i = 0$ for all $i \in N(v)$ then $f(x) = x_v + C_v(x)$ is linear with respect to x_v and its only local maximum as a function of $x_v \in [0,1]$ is attained at $x_v = 1$. Thus, any local maximum of $f(x)$ as a function of x_v would be attained at $x_v = 0$ or 1, the boundary points. This is true for every $v \in V$ and hence any local maximizer x^* of (8.1) is a 0-1 vector, *i.e.* $x^* \in \{0,1\}^n$. We have shown that $\exists\ x^* \in \{0,1\}^n$ such that $f(G) = f(x^*)$. To prove that $f(G) = \alpha(G)$, we establish the inequality in both directions.

1. $f(G) \geq \alpha(G)$.

   Let I^* be a maximum independent set of G. Construct a feasible solution x^0 of

(8.1) as follows,

$$x_i^0 = \begin{cases} 1, & \text{if } i \in I^* \\ 0, & \text{otherwise} \end{cases}$$

Then we have $f(x^0) = \alpha(G)$ and hence $f(G) \geq f(x^0) = \alpha(G)$.

2. $f(G) \leq \alpha(G)$.

Let x^* be an optimal 0-1 solution to (8.1). Let $f(G) = f(x^*)$. Without loss of generality we can assume that the optimal solution is $x_1^* = x_2^* = \ldots = x_r^* = 1; x_{r+1}^* = x_{r+2}^* = \ldots = x_n^* = 0$, for some r. Let $V^* = \{1, \ldots, r\}$. Then we have:

$$f(G) = f(x^*) = \sum_{i \in V^*} \frac{x_i^*}{1 + \sum\limits_{j \in N(i)} x_j^*} = \sum_{i \in V^*} \frac{1}{1 + |N(i) \cap V^*|} \tag{8.2}$$

since $\sum\limits_{j \in N(i)} x_j^* = |N(i) \cap V^*|$.

The following lower-bound on the independence number $\alpha(G)$ of an arbitrary graph G is known in the literature as Wei's lower-bound [56, 187], and also as Turan's theorem [11]:

$$\alpha(G) \geq \sum_{i \in V(G)} \frac{1}{1 + d_i} \tag{8.3}$$

Using (8.3) for $\tilde{G} = G[V^*]$, the subgraph induced by V^* we obtain

$$\alpha(\tilde{G}) \geq \sum_{i \in V^*} \frac{1}{1 + |N_{\tilde{G}}(i)|} = f(G). \tag{8.4}$$

Here we used (8.2) and the observation that $\forall i \in V^*, |N(i) \cap V^*| = |N_{\tilde{G}}(i)|$. But $\alpha(G) \geq \alpha(\tilde{G}) \geq f(G)$ thereby establishing the result in (8.1). □

Remark 15. Note that an optimal solution to the problem does not necessarily correspond to a maximum independent set. Although there does exist at least one that does correspond to a maximum independent set, it is possible for alternate optima to exist. For instance, when $G = K_n$, the complete n-vertex graph, then $x_i^* = 1,\ i = 1, \ldots, n$

is optimal. This may be the case even for graphs that are not complete. Consider the example in Fig. 14. It can be easily verified that $\alpha(G) = 3$ and corresponds to {1,5,7}, {2,5,7}, {3,5,7}, {2,4,6}. But the objective function attains optimum for {1,2,3,5,7}, {1,2,5,7}, {2,3,5,7}, {1,3,5,7} also, although these are not independent.

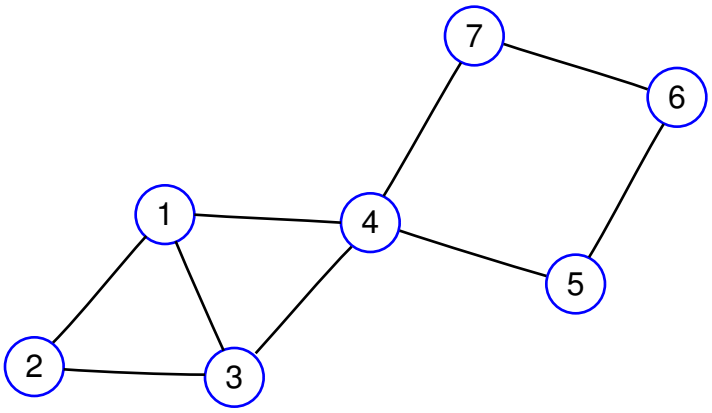


Fig. 14 An example graph illustrating Remark 15

Remark 16. We will say that a 0-1 vector x "corresponds to" a set of vertices $I \subseteq V$, which is constructed as $I = \{i \in V : x_i = 1\}$. We call the induced subgraph $G[I]$ the "support graph" of x and denote it by $G(x)$. Recall that for $I \subseteq V$, the binary vector $x \in \{0,1\}^n$ such that $x_i = 1$ if and only if $i \in I$ is called the incidence vector of I. We denote by $x \equiv I$, the equivalence of a binary vector to a subset of vertices.

Corollary 1. *The support graph of every global maximum of (8.1) is a union of $\alpha(G)$ components, each of which is a complete graph.*

Proof. Recall from the proof of Theorem 15 that $\tilde{G}$ is the support graph of a global maximum of (8.1). From the proof we also know that $\alpha(G) = f(G)$ and $\alpha(G) \geq \alpha(\tilde{G}) \geq f(G)$. Hence we have,

$$\alpha(G) = \alpha(\tilde{G}) = f(G) = \sum_{i \in V^*} \frac{1}{1 + |N_{\tilde{G}}(i)|} \tag{8.5}$$

Since lower-bound (8.3) is sharp on $\tilde{G}$, it is a disjoint union of components each of which is a complete graph [174, 11]. Now it is easy to see that sum of the fractional terms for vertices in each clique component equals one and hence there are $\alpha(G)$ components in $\tilde{G}$.

Henceforth in this chapter, we call a graph such as $\tilde{G}$, an *independent union of cliques* (IUC), since the cliques are disjoint and there is no edge between two vertices in different cliques (each clique is a maximal connected component in the graph). □

VIII.2. Local Maxima

We now identify some properties of local maxima of the above formulation. Note that since any local maximum of (8.1) is a 0-1 vector, it corresponds to a subset of vertices of the graph. We now set up the Karush-Kuhn-Tucker (KKT) conditions for this formulation and a lemma that will be used subsequently.

Denote by $f(x)$ the objective function of (8.1):

$$f(x) = \sum_{i \in V} \frac{x_i}{1 + \sum\limits_{j \in N(i)} x_j}. \tag{8.6}$$

We apply the first order necessary conditions (FONC) to problem (8.1) written as,

$$\max f(x)$$

subject to:

$$\begin{aligned} x_i - 1 &\leq 0 \quad \forall\, i \in V \quad : \lambda_i; \\ -x_i &\leq 0 \quad \forall\, i \in V \quad : \mu_i. \end{aligned}$$

Recall that all local maximizers of (8.1) are binary vectors. And it is easy to check that the Jacobian of all active constraints has full rank at any binary vector,

therefore all feasible binary vectors are regular points for problem (8.1). Thus, any local maximizer x^o of (8.1) satisfies the KKT conditions. This implies that there exist λ^o, μ^o such that the following conditions are satisfied,

$$\begin{aligned} \forall\ v \in V: \quad \left.\frac{\partial f}{\partial x_v}\right|_{x=x^o} &= \lambda_v^o - \mu_v^o; \\ \lambda_v^o(x_v^o - 1) &= 0; \\ \mu_v^o(-x_v^o) &= 0; \\ \lambda_v^o \geq 0; \quad \mu_v^o &\geq 0; \end{aligned}$$

where,

$$\left.\frac{\partial f}{\partial x_v}\right|_{x=x^o} = \frac{1}{1+\sum\limits_{j\in N(v)} x_j^o} - \sum_{i\in N(v)} \frac{x_i^o}{(x_v^o + 1 + \sum\limits_{j\in N(i)\setminus\{v\}} x_j^o)^2}.$$

Consider $I^o \subseteq V$, where $x^o \equiv I^o$. Then $\forall\ v \in I^o,\ x_v^o = 1$ and from the KKT conditions above we have,

$$\begin{aligned} \mu_v^o &= 0; \\ \lambda_v^o &= \frac{1}{1+\sum\limits_{j\in N(v)} x_j^o} - \sum_{i\in N(v)} \frac{x_i^o}{(x_v^o + 1 + \sum\limits_{j\in N(i)\setminus\{v\}} x_j^o)^2} \\ &= \frac{1}{1+|N(v)\cap I^o|} - \sum_{i\in N(v)\cap I^o} \frac{1}{(1+|N(i)\cap I^o|)^2}. \end{aligned}$$

On the other hand, if $v \in V \setminus I^o$ then $x_v^o = 0$ and the KKT conditions yield,

$$\begin{aligned} \lambda_v^o &= 0; \\ \mu_v^o &= \sum_{i\in N(v)} \frac{x_i^o}{(x_v^o + 1 + \sum\limits_{j\in N(i)\setminus\{v\}} x_j^o)^2} - \frac{1}{1+\sum\limits_{j\in N(v)} x_j^o} \\ &= \sum_{i\in N(v)\cap I^o} \frac{1}{(1+|N(i)\cap I^o|)^2} - \frac{1}{1+|N(v)\cap I^o|}. \end{aligned}$$

Next we prove a lemma that will be used in further discussion.

Lemma 3. *Consider the problem*

$$\max\{f(x) : Ax \leq b, x \in \mathbb{R}^n\}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ *is a continuously differentiable function,* $A \in \mathbb{R}^{m\times n}, m > n$. *Assume that a regular point* x^* *satisfies the FONC so that* n *constraints are active in* x^* *and strict complementarity holds, i.e. there exists* $\mu \geq 0$ *such that*

$$\nabla f(x^*) = A^T\mu$$

and the components of μ *corresponding to active constraints in* x^* *are positive. Then* x^* *is a local maximizer of the considered problem.*

Proof. Denote by $\bar{A} \in \mathbb{R}^{n\times n}$ the submatrix of A consisting of rows corresponding to the constraints that are active in x^* and by $\bar{\mu} \in \mathbb{R}^n$ the corresponding Lagrange multipliers $\bar{\mu} > 0$. Then

$$\nabla f(x^*) = \bar{A}^T\bar{\mu}. \tag{8.7}$$

Consider any feasible direction d in x^*. Then $\bar{A}(x^* + d) \leq \bar{b}$, where $\bar{b} \in \mathbb{R}^n$ is the vector of components of b corresponding to the active constraints. Since $\bar{A}x^* = \bar{b}$, we have $\bar{A}d \leq 0$. Moreover, since x^* is regular, $\bar{A}x = \bar{b}$ has a unique solution (given by x^*), thus if $d \neq 0$ then at least one of the components of $\bar{A}d$ has to be negative. Thus, using (8.7) we have $\nabla f(x^*)^T d = \mu^T\bar{A}d < 0$. So, d is a descent direction. Since d is an arbitrary feasible direction, x^* is a local maximizer. □

Theorem 16. *If* x^o *is a point of local maximum of (8.1) and* $x^o \equiv I^o$, *then* I^o *is a dominating set.*

Proof. Suppose that I^o is not a dominating set, then there exists $x_i^o = 0$ such that

$N(i) \cap I^o = \emptyset$. Construct x', such that

$$x'_j = \begin{cases} \epsilon > 0, & \text{if } j = i; \\ x^o_j, & \text{if } j \in V \setminus \{i\}. \end{cases}$$

Then we have, $f(x') = f(x^o) + \epsilon$, which contradicts the fact that x^o is a local maximum. Hence, I^o must be a dominating set. □

Corollary 2. *If x^o is a local maximizer of (8.1) with $x^o \equiv I^o$ and I^o is an independent set, then it is maximal and there exist unique λ^o, μ^o such that (x^o, λ^o, μ^o) solves KKT-FONC given by*

$$\lambda^o_v = \begin{cases} 1, & \text{if } v \in I^o; \\ 0, & \text{if } v \in V \setminus I^o; \end{cases}$$

$$\mu^o_v = \begin{cases} 0, & \text{if } v \in I^o; \\ |N(v) \cap I^o| - \frac{1}{1+|N(v)\cap I^o|}, & \text{if } v \in V \setminus I^o. \end{cases}$$

Remark 17. The conclusion that a local maximum corresponds to a dominating set is only a necessary condition and the converse is obviously not true. For example, consider the graph G_1 in Fig. 15 with $\alpha(G_1) = 2$. The set of vertices $I = \{1, 2, 3\}$

Fig. 15 Graphs illustrating Remarks 17 and 18

is a dominating set. However, $x = [1, 1, 1, 0]^T \equiv I$ is not a local maximum. Let $x' = [1, 1, 1, \epsilon]^T$, where $\epsilon > 0$. Then $f(x') = 1 + \frac{\epsilon(1+\epsilon)}{3(3+\epsilon)} > f(x) = 1$ for any $\epsilon > 0$.

Further, since we know that a global maximum corresponds to an IUC with maximum number of clique components, we can also consider maximal by inclusion IUC (*i.e.*, such that it is not a subset of IUC with larger number of components) to be candidates for local maxima. Note that I is a maximal clique and there does not exist a strict superset which induces an IUC with 2 or more components making this a maximal IUC. So, even a maximal IUC may not correspond to a local maximizer.

Remark 18. Next example shows that even though a global maximizer always corresponds to an IUC, a local maximizer may not have the same property. Consider the graph G_2 with $\alpha(G_2) = 2$. The sets $\{1, 4\}$ and $\{2, 3\}$ are maximal, as well as maximum independent sets. But it can be verified that the point $x = [1, 1, 1, 1]^T \equiv V(G_2)$ is also a local maximum with multipliers $\lambda_k = \frac{1}{9}$ and $\mu_k = 0$ for all $k \in V(G_2)$. So, a local maximizer may correspond to a set that is not an independent set or even an IUC. The following theorem establishes the converse for a special case.

Theorem 17. *If I^o is a MIS and $x^o \equiv I^o$ then x^o is a local maximizer of (8.1).*

Proof. As before x^o is a regular point. FONC are satisfied in x^o as the KKT system has the following unique solution

$$\lambda_v^o = \begin{cases} 1, & \text{if } v \in I^o; \\ 0, & \text{if } v \in V \setminus I^o; \end{cases}$$

$$\mu_v^o = \begin{cases} 0, & \text{if } v \in I^o; \\ |N(v) \cap I^o| - \frac{1}{1+|N(v)\cap I^o|}, & \text{if } v \in V \setminus I^o; \end{cases}$$

and $\mu_v^o > 0$ as $|N(v) \cap I^o| \geq 1$ for all $v \in V \setminus I^o$ since I^o is a MIS.

Problem (8.1) and $x^* = x^o$ satisfy all conditions of Lemma 3, where $m = 2n$ and exactly n constraints are active in x^o. Hence, x^o is a local maximizer of (8.1). □

We now look at local maximum properties of (8.1) in the binary neighborhood.

For a given vector $x \in \{0,1\}^n$, its binary neighborhood consists of all the binary vectors that are at Hamming distance one from x. That is, the set of all binary vectors that can be obtained from x by changing the value of one of its components to the opposite. The results are similar to the continuous case.

Theorem 18. *If x^o is a point of local maximum of (8.1) in the binary neighborhood and $x^o \equiv I^o$, then I^o is a dominating set.*

Proof. Consider x^o, a local maximizer in the binary neighborhood with $x^o \equiv I^o$. Let $v \in V \setminus I^o$, then $x^o_v = 0$. Construct a neighbor of x^o as follows,

$$x''_i = \begin{cases} 1, & \text{if } i = v; \\ x^o_i, & \text{if } i \in V \setminus \{v\}. \end{cases}$$

Since x^o is a local maximizer, $f(x'') \leq f(x^o)$, where

$$f(x^o) \;\; = \;\; \sum_{i \in I^o} \frac{1}{1 + |N(i) \cap I^o|}.$$

If $N(v) \cap I^o = \emptyset$, then

$$f(x'') = \sum_{i \in I^o \cup \{v\}} \frac{1}{1 + |N(i) \cap (I^o \cup \{v\})|} = f(x^o) + 1 > f(x^o),$$

which is a contradiction. Hence, $\forall v \notin I^o,\ N(v) \cap I^o \neq \emptyset$, so I^o is a dominating set. □

Theorem 19. *If I^o is a MIS and $x^o \equiv I^o$, then x^o is a local maximizer of (8.1) in the binary neighborhood.*

Proof. Since I^o is an independent set, $f(x^o) = |I^o|$. Any vector in the binary neighborhood of x^o can be obtained by either changing x^o_v from 1 to 0 for some $v \in I^o$ or changing x^o_v from 0 to 1 for some $v \notin I^o$. We analyze these two cases separately.

First, let $v \in I^o$, then $x^o_v = 1$. Construct x' in the binary neighborhood of x^o as

follows,

$$x'_i = \begin{cases} 0, & \text{if } i = v; \\ x^o_i, & \text{if } i \in V \setminus \{v\}. \end{cases}$$

Then, $f(x') = |I^o| - 1 < f(x^o)$.

Now let $v \in V \setminus I^o$, then $x^o_v = 0$. Construct x'' as follows,

$$x''_i = \begin{cases} 1, & \text{if } i = v; \\ x^o_i, & \text{if } i \in V \setminus \{v\}. \end{cases}$$

Then,

$$\begin{aligned} f(x'') &= \frac{x''_v}{1 + \sum\limits_{j \in N(v)} x''_j} + \sum_{i \in I^o \cap N(v)} \frac{x''_i}{1 + \sum\limits_{j \in N(i)} x''_j} + \sum_{i \in I^o \setminus N(v)} \frac{x''_i}{1 + \sum\limits_{j \in N(i)} x''_j} \\ &= \frac{1}{1 + |N(v) \cap I^o|} + \frac{|N(v) \cap I^o|}{2} + |I^o| - |N(v) \cap I^o| \\ &= |I^o| - \left(\frac{p^2 + p - 2}{2(1+p)} \right), \end{aligned}$$

where $p = |N(v) \cap I^o| \geq 1$ is integer and hence $\frac{p^2+p-2}{2(1+p)} \geq 0$. Thus, $f(x'') \leq f(x^o)$ and for any x in the binary neighborhood of x^o, $f(x) \leq f(x^o)$. □

Remark 19. Note that x^o does not have to be a strict local maximum. For instance when $p = 1$, there is a vertex outside the set that has exactly one neighbor inside and hence including that induces an IUC with the same number of components as in I^o (*i.e.*, $|I^o| - 1$ cliques of size 1 and a clique of size 2). So, the objective function value does not change and the local maximizer is not strict.

VIII.3. Modified Formulation

We now modify the formulation (8.1) to obtain one with more desirable properties. In particular, we are interested in one-to-one correspondence between local maximizers

of the formulation and MIS of the graph.

Given graph $G = (V, E)$ with the adjacency matrix A_G, consider the following function:

$$\begin{aligned} g(x) &= \sum_{i \in V} \frac{x_i}{1 + \sum\limits_{j \in N(i)} x_j} - \frac{1}{2} x^T A_G x \\ &= \sum_{i \in V} x_i \left(\frac{1}{1 + \sum\limits_{j \in N(i)} x_j} - \frac{1}{2} \sum_{j \in N(i)} x_j \right). \end{aligned}$$

Then, $\forall\ x \in [0,1]^n, g(x) \leq f(x) \leq \max_{x \in [0,1]^n} f(x) = \alpha(G)$, and for x^* corresponding to a maximum independent set, $g(x^*) = \alpha(G)$. Hence we have

$$\alpha(G) = \max_{x \in [0,1]^n} \left\{ \sum_{i \in V} x_i \left(\frac{1}{1 + \sum\limits_{j \in N(i)} x_j} - \frac{1}{2} \sum_{j \in N(i)} x_j \right) \right\}. \tag{8.8}$$

As in Section VIII.2, for a given v, we can rewrite $g(x)$ as

$$g(x) = x_v A_v(x) + B_v(x) + C_v(x),$$

where

$$\begin{aligned} A_v(x) &= \frac{1}{1 + \sum\limits_{j \in N(v)} x_j} - \sum_{j \in N(v)} x_j, \\ B_v(x) &= \sum_{i \in N(v)} \frac{x_i}{x_v + 1 + \sum\limits_{j \in N(i) \setminus \{v\}} x_j}, \\ C_v(x) &= \sum_{i \in S} x_i \left(\frac{1}{1 + \sum\limits_{j \in N(i)} x_j} - \frac{1}{2} \sum_{j \in N(i)} x_j \right). \end{aligned}$$

Here $S = V \setminus (\{v\} \cup N(v))$. Using this representation and arguments similar to ones used in Section VIII.2, it is easy to show that $g(x)$ is convex with respect to each variable and every local (and global) maximizer is a binary vector. We now look at

the local maxima of (8.8). Note that every local maximum is a binary vector and is a regular point.

Theorem 20. *x^o is a point of local maximum of (8.8) if and only if I^o is a MIS, where $x^o \equiv I^o$.*

Proof. Let x^o be a local maximum of (8.8). Then the KKT conditions imply the existence of λ^o, μ^o such that

$$\begin{aligned} \forall\ v \in V: \quad \left.\frac{\partial g}{\partial x_v}\right|_{x=x^o} &= \lambda_v^o - \mu_v^o; \\ \lambda_v^o(x_v^o - 1) &= 0; \\ \mu_v^o(-x_v^o) &= 0; \\ \lambda_v^o \geq 0; \quad \mu_v^o &\geq 0; \end{aligned}$$

where,

$$\left.\frac{\partial g}{\partial x_v}\right|_{x=x^o} = \frac{1}{1+\sum\limits_{j\in N(v)} x_j^o} - \sum_{j\in N(v)} x_j - \sum_{i\in N(v)} \frac{x_i^o}{(x_v^o + 1 + \sum\limits_{j\in N(i)\setminus\{v\}} x_j^o)^2}.$$

Let $x^o \equiv I^o$, then $\forall\ v \in I^o,\ x_v^o = 1$ and from KKT-FONC we have,

$$\begin{aligned} \mu_v^o &= 0; \\ \lambda_v^o &= \frac{1}{1+|N(v)\cap I^o|} - |N(v)\cap I^o| - \sum_{i\in N(v)\cap I^o} \frac{1}{(1+|N(i)\cap I^o|)^2} \ \geq\ 0. \end{aligned}$$

Since v is an arbitrary vertex from I^o, in order to show that I^o is an independent set it suffices to prove that $N(v) \cap I^o = \emptyset$. Assume that this is not the case, *i.e.,* $|N(v) \cap I^o| \geq 1$. Then $\lambda_v^o \leq 1/2 - 1 < 0$, which contradicts the nonnegativity of λ_v^o. Hence, $|N(v) \cap I^o| = 0$ for any $v \in I^o$ and I^o is an independent set. Now suppose this independent set is not maximal. Then there exists $x_v^o = 0,\ v \in V \setminus I^o$ such that

$N(v) \cap I^o = \emptyset$. Construct x', such that

$$x'_j = \begin{cases} \epsilon > 0, & \text{if } j = v; \\ x^o_j, & \text{if } j \in V \setminus \{v\}. \end{cases}$$

Then we have, $f(x') = f(x^o) + \epsilon$, which contradicts the fact that x^o is a local maximum. Hence, I^o must be a *maximal* independent set.

To prove the other direction, suppose I^o is a MIS and $x^o \equiv I^o$. In order to show that x^o is a local maximum, we show that it satisfies the KKT-FONC and use Lemma 3. The unique solution to the KKT system is

$$\lambda^o_v = \begin{cases} 1, & \text{if } v \in I^o; \\ 0, & \text{if } v \in V \setminus I^o; \end{cases}$$

$$\mu^o_v = \begin{cases} 0, & \text{if } v \in I^o; \\ 2|N(v) \cap I^o| - \frac{1}{1+|N(v)\cap I^o|}, & \text{if } v \in V \setminus I^o. \end{cases}$$

Note that $\mu^o_v > 0$ as $|N(v) \cap I^o| \geq 1$ for any $v \in V \setminus I^o$ since I^o is a MIS.

Here again, all conditions of Lemma 3 are satisfied for problem (8.8) with $x^* = x^o$, so x^o is a local maximizer of (8.8). □

Corollary 3. *x^* is a global maximum of (8.8) if and only if V^* is a maximum independent set of G, where $x^* \equiv V^*$.*

We now proceed to show that similar properties hold in case of the binary neighborhood for formulation (8.8).

Theorem 21. *x^o is a local maximum of (8.8) in the binary neighborhood if and only if I^o is a MIS, where $x^o \equiv I^o$.*

Proof. Let I^o be a MIS with $x^o \equiv I^o$, then $g(x^o) = |I^o|$. Let x' be a binary neighbor

obtained from x^o by changing a component that was 1 to 0. Then $g(x') = |I^o| - 1 < g(x^o)$ as x' would still correspond to an independent set.

Now, let x'' denote a binary neighbor obtained by changing the component, say v, in x^o from 0 to 1. Let I'' be the corresponding set of vertices. Then,

$$\begin{aligned}
g(x'') &= \sum_{i \in I''} \frac{1}{1 + |N(i) \cap I''|} - |N(v) \cap I^o| \\
&= \frac{1}{1 + |N(v) \cap I''|} + \sum_{i \in I^o \setminus N(v)} \frac{1}{1 + |N(i) \cap I''|} \\
&\quad + \sum_{i \in I^o \cap N(v)} \frac{1}{1 + |N(i) \cap I''|} - |N(v) \cap I^o| \\
&= \frac{1}{1 + |N(v) \cap I^o|} + \sum_{i \in I^o \setminus N(v)} \frac{1}{1 + |N(i) \cap I^o|} \\
&\quad + \sum_{i \in I^o \cap N(v)} \frac{1}{2 + |N(i) \cap I^o|} - |N(v) \cap I^o| \\
&= \frac{1}{1 + |N(v) \cap I^o|} + |I^o \setminus N(v)| + \frac{1}{2}|N(v) \cap I^o| - |N(v) \cap I^o| \\
&= \frac{1}{1 + |N(v) \cap I^o|} + |I^o| - |N(v) \cap I^o| - \frac{1}{2}|N(v) \cap I^o| \\
&= |I^o| - \frac{3}{2}|N(v) \cap I^o| + \frac{1}{1 + |N(v) \cap I^o|} \\
&= |I^o| - \frac{3p^2 + 3p - 2}{2(1+p)},
\end{aligned}$$

where $p = |N(v) \cap I^o| \geq 1$ as I^o is maximal. Note that $\frac{3p^2+3p-2}{2(1+p)} > 0$ if $p \geq 1$ and integer, so we have $g(x'') < g(x^o)$, which establishes one direction.

To show the other direction, suppose that x^o is a local maximum in the binary neighborhood and $x^o \equiv I^o$.

$$g(x^o) = \sum_{i \in I^o} \frac{1}{1 + |N(i) \cap I^o|} - |E \cap (I^o \times I^o)|.$$

Suppose that I^o is not an independent set. Then $\exists\ u, v \in I^o$ such that $(u, v) \in E$.

Construct x' in the binary neighborhood of x^o as follows,

$$x'_i = \begin{cases} x^o_i, & \text{if } i \neq u; \\ 0, & \text{if } i = u; \end{cases} \qquad i \in V.$$

Let I' be the corresponding vertex set, $I' = I^o \setminus \{u\}$. Then we have,

$$g(x') = \sum_{i \in I'} \frac{1}{1 + |N(i) \cap I'|} - |E \cap (I' \times I')|.$$

Note that $|E \cap (I' \times I')| = |E \cap (I^o \times I^o)| - |N(u) \cap I^o|$ and,

$$\sum_{i \in I'} \frac{1}{1 + |N(i) \cap I'|}$$

$$\begin{aligned} &= \sum_{i \in I' \setminus N(u)} \frac{1}{1 + |N(i) \cap I'|} + \sum_{i \in I' \cap N(u)} \frac{1}{1 + |N(i) \cap I'|} \\ &= \sum_{i \in I' \setminus N(u)} \frac{1}{1 + |N(i) \cap I^o|} + \sum_{i \in I' \cap N(u)} \frac{1}{1 + |N(i) \cap I^o| - 1}. \end{aligned}$$

So

$$\begin{aligned} g(x^o) - g(x') &= \frac{1}{1 + |N(u) \cap I^o|} + \sum_{i \in I' \cap N(u)} \Bigg(\frac{1}{1 + |N(i) \cap I^o|} \\ &\qquad - \frac{1}{|N(i) \cap I^o|} \Bigg) - |N(u) \cap I^o| < 0, \end{aligned}$$

since,

$$\frac{1}{1 + |N(u) \cap I^o|} - |N(u) \cap I^o| \;<\; 0$$

as $v \in N(u) \cap I^o$. But x^o was assumed to be a local maximum and hence by contradiction I^o is an independent set.

Now suppose that I^o is not maximal. Then there exists at least one vertex a

that can be added to I^o. That is, $I'' = I^o \cup \{a\}$ is an independent set with the corresponding binary vector x'' given by

$$x''_i = \begin{cases} x^o_i, & \text{if } i \neq a; \\ 1, & \text{if } i = a; \end{cases}$$

and $g(x'') = |I''| = |I^o| + 1 > |I^o| = g(x^o)$. This contradiction with the local maximality of x^o in the binary neighborhood establishes that I^o is a MIS and hence the required result. □

VIII.4. Numerical Experiments

This section presents our preliminary numerical experiments with the continuous formulations. In Section VIII.4.1, we present our results from applying a global optimization algorithm on our formulation along with three other formulations presented in Section II.6.2. In Section VIII.4.2, numerical experiments are carried out to compare the formulations (8.1) and (8.8) in terms of their usefulness with local optimization algorithms.

VIII.4.1. Global Optimization

For sample numerical experiments with box-constrained problems we used the MATLAB® implementation of Multilevel Coordinate Search (MCS) [118] which is available online from [155]. MCS uses function values only and combines global and local search in an attempt to find a global minimum of a given function over a box. Experimentally, MCS is known to have performed better than other global optimization algorithms on many standard test problems and was particularly effective with low-dimensional problems [118, 155]. The numerical results are given in the tables below. We used MCS with default settings in all the experiments. We considered 20 maximum inde-

pendent set problem instances. The number of vertices in the graphs *i.e.*, the number of variables in the corresponding formulations ranged from 15 to 64. In addition to our formulation (8.1), we experimented with the following formulations for independence number from literature (see Section II.6.2 for details).

$$\alpha(G) = \max_{0 \neq x \in [0,1]^n} \frac{(\sum_{i \in V(G)} x_i)^2}{\sum_{i \in V(G)} x_i^2 + 2 \sum_{(i,j) \in E(G)} x_i x_j} \tag{2.2}$$

$$\alpha(G) = \max_{x \in [0,1]^n} \sum_{i \in V(G)} x_i \prod_{j \in N(i)} (1 - x_j) \tag{2.3}$$

$$\alpha(G) = \max_{x \in [0,1]^n} \sum_{i \in V(G)} x_i - \sum_{(i,j) \in E(G)} x_i x_j \tag{2.5}$$

Table 21 contains the results of executing MCS on the maximum independent set problem instances using formulations (2.2)-(2.5) and (8.1). The columns in each row of this table contain the graph name ("Graph") followed by the number of vertices ("$|V|$"), the number of edges ("$|E|$"), and the independence number of the graph ("$\alpha(G)$"). The remaining four columns contain the objective value of the solutions output by MCS for the four different functions corresponding to the different formulations. The extension ".c" in the name of a graph `graphname.c` indicates that this graph is the complement of `graphname`. The graphs `1dc64` and `1et64` are available online from [177] and they arise in coding theory (see also [52]).

As can be seen from the tables, the performance of MCS on the all the formulations was encouraging. In summary, global optima were found in 30 of the 80 cases and the objective was close to optimal in most cases that were not solved to opti-

Table 21 Results of experiments using MATLAB® implementation of MCS

				Formulation			
Graph	$\|V\|$	$\|E\|$	$\alpha(G)$	(2.2)	(2.3)	(2.5)	(8.1)
johnson6-2-4.c	15	60	3	3.0000	3.0000	3.0000	2.3333
johnson6-3-5.c	20	180	2	1.2000	2.0000	2.0000	1.0769
johnson7-5-3.c	21	105	3	2.3333	3.0000	3.0000	2.3333
johnson8-2-4.c	28	168	4	2.4000	4.0000	4.0000	2.6667
johnson7-3-5.c	35	525	2	1.5000	2.0000	2.0000	1.3214
MANN_a9	45	918	3	3.0000	3.0000	3.0000	1.2778
MANN_a9.c	45	72	16	12.0000	16.0000	16.0000	16.0000
hamming6-2.c	64	102	32	32.0000	32.0000	32.0000	32.0000
hamming6-4.c	64	1312	4	2.0000	4.0000	4.0000	2.0220
1dc64	64	543	10	9.0000	9.0000	8.0000	10.0000
1et64	64	264	18	18.0000	18.0000	18.0000	15.6667
san15-1.c	15	70	4	2.0000	4.0000	4.0000	3.0000
san15-2.c	15	20	8	7.0000	7.0000	7.0000	8.0000
san20-1.c	20	60	8	7.0000	6.0000	6.0000	7.0000
san20-2.c	20	70	7	6.0000	6.0000	6.0000	6.0000
san30-1.c	30	43	15	13.0000	13.0000	13.0000	13.0000
san40-1.c	40	78	20	16.0000	16.0000	18.0000	17.0000
san40-2.c	40	73	21	17.2811	17.0000	18.0000	17.0000
san50-1.c	50	125	20	18.0000	18.0000	19.0000	19.0000
san50-2.c	50	65	24	22.0000	22.0000	21.0000	23.0000

mality. Out of 20 instances the highest objective was produced 16 times by (2.5), 13 times by (2.3), 9 times by (8.1) and 7 times by (2.2). We believe that the performance can be improved with a specialized commercial solver such as GAMS® [97].

VIII.4.2. Local Optimization

Numerical experiments were conducted to compare the performance of the original formulation (8.1) and the modified formulation (8.8) as objective functions for a simple local search algorithm and a constrained local optimization procedure available in the MATLAB® Optimization Toolbox. Complements of selected DIMACS clique benchmark graphs [84] were used as instances for testing.

The local search algorithm starts at a random binary vector and reaches a local maximum in the binary neighborhood by successively moving to the first improving neighbor found. Table 22 presents the results that were obtained, where average and the maximum objective function value obtained starting from ten random binary vectors are shown for formulations (8.1) and (8.8).

MATLAB® function `fmincon` uses a sequential quadratic programming approach for solving medium-scale constrained optimization problems. Details and relevant references can be found at [92]. The results tabulated in Table 23 show the average and best objective function value attained in ten runs starting from random initial feasible points inside $[0,1]^n$ with formulations (8.1) and (8.8) as objective functions.

A total of 24 DIMACS clique benchmark graphs with up to 378 vertices were complemented and used in testing. Note that the values could be rounded up to get lower-bounds on $\alpha(G)$. In terms of the average objective function value achieved in 10 runs of the local search algorithm, formulation (8.1) produced better results than formulation (8.8) with 17 instances whereas formulation (8.8) was better in 7 cases. In terms of the best solution obtained in 10 runs, formulation (8.1) beats formulation

(8.8) 14 to 2, with the rest being equal. Similarly with `fmincon`, in terms of average performance, the ratio was 15 to 8 in favor of formulation (8.1) with one instance producing identical results with both. In terms of best solution obtained, the ratio was 11 to 5 again in favor of formulation (8.1), with the rest being equal.

Note that given a graph $G = (V, E)$, every MIS in G corresponds to a local maximum for both formulations. The original formulation can have additional "spurious" local maxima besides these. However, with both algorithms, test results indicate that formulation (8.1) produces better quality solutions more often than formulation (8.8). The main advantage we gain by using the modified formulation is that the locally optimal solution will correspond to a MIS as the spurious local maxima that exist in the other formulation are eliminated here. Note that in the original formulation, although the objective attained is a lower-bound on the independence number, the solution may not even correspond to an independent set.

Summary of Results. In this chapter we propose a new box-constrained continuous fractional formulation for the independence number of a graph. We characterize the local maxima of this continuous formulation in terms of structures in the graph. This formulation is then modified to introduce one-to-one correspondence between local maxima and MIS by eliminating the spurious local maxima in the original formulation. Classical Karush-Kuhn-Tucker conditions and simple combinatorial arguments are found sufficient to deduce several interesting properties of the local and global maxima. These properties can be utilized in developing new approaches to the maximum independent set problem.

Even though we restricted our attention to only the maximum independent set problem and only linearly-constrained formulations, there are many other opportunities for approaching CO problems using continuous global optimization. Since many of CO problems can be expressed in terms of binary integer programs, the constraint

Table 22 Results for local search

				Original Formulation		Modified Formulation	
Graph	Vertices	Edges	$\alpha(G)$	Avg	Best	Avg	Best
c-fat200-1	200	18366	12	9.86	12.00	12.00	12.00
c-fat200-2	200	16665	24	18.12	24.00	23.60	24.00
c-fat200-5	200	11427	58	57.40	58.00	57.90	58.00
johnson16-2-4	120	1680	8	7.68	8.00	8.00	8.00
johnson8-2-4	28	210	4	3.62	4.00	4.00	4.00
johnson8-4-4	70	560	14	14.00	14.00	11.20	14.00
keller4	171	5100	11	10.00	11.00	8.20	10.00
hamming6-2	64	192	32	30.70	32.00	24.30	32.00
hamming6-4	64	1312	4	3.04	4.00	2.80	4.00
hamming8-2	256	1024	128	125.90	128.00	74.60	82.00
hamming8-4	256	11776	16	16.00	16.00	10.20	13.00
san200_0.7_2	200	5970	18	12.30	13.00	12.70	14.00
san200_0.9_1	200	1990	70	46.70	48.00	37.70	47.00
san200_0.9_2	200	1990	60	37.30	40.00	29.10	32.00
san200_0.9_3	200	1990	44	32.60	35.00	27.60	29.00
brock200_1	200	5066	21	17.30	19.00	14.00	16.00
brock200_2	200	10024	12	9.10	11.00	7.90	9.00
brock200_3	200	7852	15	12.00	13.00	10.20	12.00
brock200_4	200	6811	17	12.70	15.00	10.80	12.00
p_hat300-1	300	33917	8	7.20	8.00	5.60	6.00
p_hat300-2	300	22922	25	23.80	25.00	17.60	19.00
p_hat300-3	300	11460	36	30.70	32.00	23.90	28.00
mann_a27	378	702	126	117.20	118.00	117.80	119.00
mann_a9	45	72	16	15.10	16.00	14.30	15.00

$x \in \{0,1\}^n$ can be replaced with the equivalent quadratic constraints $x_i(1-x_i) = 0,\ i = 1,\ldots,n$ [117], thus yielding a continuous, constrained global optimization formulation. For example, Shor [176] used a similar idea to obtain a quadratically constrained global optimization formulation of the maximum weight independent set problem and demonstrated encouraging computational results. Other interesting problems for which continuous approaches exist include graph coloring [51], quadratic assignment and maximum matching problems (see [55] for a convex quadratic ap-

Table 23 Results for MATLAB® `fmincon`

				Original Formulation		Modified Formulation	
Graph	Vertices	Edges	$\alpha(G)$	Avg	Best	Avg	Best
c-fat200-1	200	18366	12	9.86	12.00	11.88	12.00
c-fat200-2	200	16665	24	21.78	24.00	22.40	24.00
c-fat200-5	200	11427	58	54.79	58.00	57.30	58.00
johnson16-2-4	120	1680	8	4.37	4.44	8.00	8.00
johnson8-2-4	28	210	4	2.54	3.00	4.00	4.00
johnson8-4-4	70	560	14	13.30	14.00	11.50	14.00
keller4	171	5100	11	7.29	9.00	7.00	7.00
hamming6-2	64	192	32	30.30	32.00	22.30	32.00
hamming6-4	64	1312	4	2.54	3.33	4.00	4.00
hamming8-2	256	1024	128	104.14	128.00	78.00	90.00
hamming8-4	256	11776	16	14.93	16.00	10.50	16.00
san200_0.7_2	200	5970	18	12.00	12.00	12.00	12.00
san200_0.9_1	200	1990	70	45.39	47.00	45.20	46.00
san200_0.9_2	200	1990	60	35.78	38.00	36.95	40.00
san200_0.9_3	200	1990	44	31.38	34.00	30.90	33.00
brock200_1	200	5066	21	17.20	19.00	16.50	18.00
brock200_2	200	10024	12	8.59	10.00	8.00	9.00
brock200_3	200	7852	15	11.40	13.00	10.20	12.00
brock200_4	200	6811	17	13.40	15.00	12.20	14.00
p_hat300-1	300	33917	8	7.20	8.00	6.30	7.00
p_hat300-2	300	22922	25	22.80	25.00	21.00	24.00
p_hat300-3	300	11460	36	32.02	33.00	29.90	32.00
mann_a27	378	702	126	116.99	117.00	117.10	118.00
mann_a9	45	72	16	15.10	16.00	15.00	16.00

proach to the maximum matching problem). Development of efficient approaches for solving these types of global optimization problems would lead to new algorithms for the mentioned CO problems.

CHAPTER IX

NETWORK CLUSTERING AND DESIGN EXTENSIONS

Cliques have been applied not only in the context of finding a maximum clique or generating all maximal cliques, they have also been the most popular models for representing clusters in network clustering problems. In Section II.1 we described several applications of clustering problems including wireless and biological networks. Naturally, the drawbacks of cliques as practical models of cohesiveness, also follow them into clustering problems. In this chapter, we will explore clustering models and algorithms that utilize clique relaxations instead of cliques as cluster models.

In this chapter, we also propose a novel model for designing a robust network that is based on an extremal version of the k-plex model. First, we will provide a brief introduction to network clustering problems and mention existing approaches using cliques relevant to our discussion.

IX.1. Network Clustering

Clustering can be loosely defined as the process of grouping objects into sets called *clusters*, so that each cluster consists of elements that are similar in some way. The similarity criterion can be defined in several different ways, depending on applications of interest and the objectives that the clustering aims to achieve. For example, in *distance-based clustering* two or more elements belong to the same cluster if they are close with respect to a given *distance metric*. On the other hand, in *conceptual clustering*, which can be traced back to Aristotle and his work on classifying plants and animals, the similarity of elements is based on descriptive concepts.

Network clustering deals with clustering the data represented as a graph. Data points are represented by vertices and an edge exists if two data points are similar or related in a certain way. It is important to note that the similarity criterion used to construct the network model of a data set is based on *pairwise relations*, while the similarity criterion used to define a cluster refers to all elements in the cluster and needs to be satisfied by the cluster as a whole and not just pairs of its elements. In order to avoid confusion, from now on we will use the term "*cohesiveness*" when referring to the cluster similarity. Clearly, the definition of similarity (or dissimilarity) used to construct the network is determined by the nature of the data and based on the cohesiveness we expect in the clusters that result.

In general, network clustering approaches can be used to perform both distance-based and conceptual clustering. In distance-based clustering, the vertices of the graph correspond to the data points, and edges are added if the points are close enough based on some cut-off value. Alternately, the distances could just be used to weight the edges of a complete graph representing the data set. We have already seen the examples of database networks, protein interaction networks and call graphs that illustrate the use of networks in conceptual clustering.

Clustering concepts have been fundamental to data analysis, data reduction and classification. Efficient data organization and retrieval that results from clustering has impacted every field of science and engineering that requires management of massive amounts of data. Cluster analysis techniques and algorithms in the areas of statistics and information sciences are well documented in several textbooks [12, 111, 179, 122, 121]. We will discuss in this chapter the classical problem of network clustering using cliques and propose some novel approaches using clique relaxations.

IX.2. The Clustering Problem

Given a graph $G^0 = (V^0, E^0)$, the *clustering problem* is to find subsets (not necessarily disjoint) $\{V_1^0, \ldots, V_r^0\}$ of V^0 such that $V^0 = \bigcup_{i=1}^{r} V_i^0$. Each subset is a *cluster* modeled by structures such as cliques or other cohesive units. Clustering models can be classified by the constraints on relations between clusters (clusters may be disjoint or overlapping) and the objective function used to achieve the goal of clustering (minimizing the number of clusters or maximizing the cohesiveness). When the clusters are required to be disjoint, $\{V_1^0, \ldots, V_r^0\}$ is a *cluster-partition* and when they are allowed to overlap, it is a *cluster-cover*. For a given G^0, assuming that there is a measure of cohesiveness of the cluster that can be varied, we can define two types of optimization problems:

> Type I: Minimize the number of clusters while ensuring that every cluster formed has cohesiveness over a prescribed threshold;
>
> Type II: Maximize the cohesiveness of each cluster formed, while ensuring that the number of clusters that result is under a prescribed number d (this may be relaxed by setting $d = \infty$).

Hierarchical clustering. After performing clustering, we can abstract the graph G^0 to a graph $G^1 = (V^1, E^1)$ as follows: there exists a vertex $v_i^1 \in V^1$ for every subset V_i^0 and there exists an edge between v_i^1, v_j^1 if and only if there exist $x^0 \in V_i^0$ and $y^0 \in V_j^0$ such that $(x^0, y^0) \in E^0$. We can recursively cluster the abstracted graph G^1 in a similar fashion to obtain a multi-level hierarchy. This process is called *hierarchical clustering.*

IX.3. Clique-based Clustering

Clique is a natural choice for high cohesiveness in a cluster. Cliques have minimum possible diameter, maximum possible connectivity and robustness one can expect in a cluster. Given an arbitrary graph G, Type I approach tries to partition G into (or cover G using) minimum number of cliques. Type II approaches usually work with a weighted complete graph and hence every partition of the vertex set is a clique partition. The objective here is to maximize cohesiveness within the clusters.

IX.3.1. Clique Partitioning and Covering

Type I clique partitioning and clique covering problems are both NP-hard [98]. Consequently, exact approaches to solve these problems that exist in literature are computationally ineffective for large graphs. Heuristic approaches are preferred for large graphs for this reason. Note that the minimum number of clusters produced in clique covering and partitioning are the same. Denote the covering optimum by c and the partition optimum by p. Since every clique partition is also a cover, $p \geq c$. Let $\{V_1^0, \ldots, V_c^0\}$ be an optimal clique cover. Any vertex v present in multiple clusters causing overlaps can be removed from all but one of the clusters to which it belongs, leaving the resulting cover with same number of clusters and one less overlap. Repeating this as many times as necessary would result in a clique partition with the same number of clusters, c. Thus, we can conclude that $p = c$. However, from a practical point of view, clique covering can be more beneficial in some applications compared to partitioning since data points shared between clusters require special attention in biological and social networks.

The minimum clique partitioning problem is also equivalent to the minimum graph coloring problem. A proper coloring of $\bar{G}$ using p colors gives rise to p color

classes which correspond to clique partitioning of G with p clusters. Every clique partitioning in G with p clusters implies that $\bar{G}$ can be colored properly with p colors, one color for each cluster. This bijection shows that the minimum number of cliques into which G can be partitioned is exactly $\chi(\bar{G})$. The graph coloring problem is also NP-hard [98] and has been studied extensively. Several exact algorithms and heuristics exist to solve this problem [129, 46, 133, 144, 134, 125].

The motivation behind clique covering and partitioning are the ideal properties of cliques. However, they are impractical models of cohesiveness in real-life networks that are known to be based on erroneous data. This is an important issue since several edges could be missing due to experimental errors and clique based clustering often produces a large number of clusters even when solved optimally, defeating the purpose of clustering. Clique relaxations can be more meaningful in this setting, and we will discuss these approaches in Section IX.4. Before presenting these new approaches, we briefly survey some popular Type II approaches that utilize cliques. Note that, a clique is already the most ideal cohesive unit, hence Type II approaches usually work with a complete graph whose edges are weighted by measures of similarity or dissimilarity.

IX.3.2. Min-Max d-Clustering

The *min-max d-clustering problem* is a Type II clique partitioning problem with a min-max objective. Consider a weighted complete graph $G = (V, E)$ with weights $w_{e_1} \leq w_{e_2} \leq \cdots \leq w_{e_m}$ where $m = \frac{n(n-1)}{2}$. The problem is to partition the graph into no more than d cliques such that the largest distance between two vertices in a clique is minimized. In other words, the problem is to find $\min\{\max_{i=1,\ldots,p} w(V_i)\}$ where $V_1, \ldots, V_p$ partition V with $p \leq d$ and $w(V_i) = \max_{u,v \in V_i : u<v} w_{uv}$. This problem is NP-hard and it is NP-hard to approximate within a factor less than two, even if the

edge weights obey triangle inequality [115, 103]. The best possible approximation algorithms (factor of two) for this problem (with the triangle inequality satisfied) are available in [115, 103]. Algorithm 10 presents the pseudocode of a *bottleneck* approach for the problem based on [115].

Definition 28. The *bottleneck graph* of a weighted graph $G = (V, E)$ is defined for a given number c as follows: $G(c) = (V, E_c)$ where $E_c = \{e \in E : w_e \leq c\}$.

In Algorithm 10, the procedure $bottleneck(w_{e_i})$ returns the bottleneck graph $G(w_{e_i})$, and $MIS()$ is an arbitrary procedure for finding a MIS in the given graph. Without loss of generality, let $I_t = \{1, \ldots, p\}$ be the output of the above algorithm terminating in iteration t. Form $V_1, \ldots, V_p$ as follows: $V_1 = \{j \in V \setminus I_t : w_{1j} \leq w_{e_t}\} \cup \{1\}$ and $V_l = \{j \in V \setminus I_t : w_{lj} \leq w_{e_t} \text{ and } j \notin V_1 \cup \ldots \cup V_{l-1}\} \cup \{l\}$ for $l = 2, \ldots, p$. In other words, $V_1, \ldots, V_p$ are the closed neighborhoods of $1, \ldots, p$ in the last bottleneck graph $G(w_{e_t})$, but vertices included in $V_1, \ldots, V_{l-1}$ are not included in V_l for $l = 2, \ldots, p$ to guarantee a partition (if they are not excluded, it will result in a cover). $G[V_1], \ldots, G[V_p]$ is a partition of G into p cliques and any edge in any clique $G[V_l]$ has weight at most $2w_{e_t}$ if the edge weights satisfy the triangle inequality. The algorithm runs in polynomial time if the $MIS()$ procedure does and $\max_{i=1,\ldots,p} w(V_i) \leq 2OPT$ where OPT represents the minimum objective [115].

The approach of using bottleneck graphs is a powerful technique that can be applied to many other related problems such as *k-center*, *weighted k-center* and *k-supplier problems*. More information about this approach and its applicability can be found in [115, 114].

Algorithm 10 Bottleneck Min-Max d-Clustering Algorithm

procedure MIN-MAX d-CLUSTER($G = (V, E)$, sorted edges, d)
    initialize $i \leftarrow 0$, $stop \leftarrow false$
    while $stop = false$ **do**
        $i \leftarrow i + 1$
        $G_b^i \leftarrow bottleneck(w_{e_i})$
        $I_i \leftarrow MIS(G_b^i)$
        if $|I_i| \leq d$ **then**
            $stop \leftarrow true$
        end if
    end while
    return I_i
end procedure

IX.4. Clique Relaxations in Clustering

In this section, we define new Type I and Type II clustering problems based on the clique relaxations. The preliminary results comprise of model and problem definition along with approaches for addressing some of them.

IX.4.1. *k*-Clique and *k*-Club Clustering

Type I k-clique clustering problem can be defined along the same lines as clique partitioning and covering. Given an arbitrary graph G and a fixed positive integer k, partition G into (or cover G using) the minimum possible number of k-cliques. As before, this problem can be reduced to clique partitioning and covering on the power graph G^k. Hence approaches developed in the literature for clique partitioning, covering and graph coloring are applicable here. On the other hand, we can define

a novel Type II approach based on k-cliques that does not require edge weights or a complete graph, since the parameter k is itself a measure of cohesiveness of the k-clique.

The *Type II min-max k-clique d-clustering problem* can be defined as follows. Given an arbitrary graph $G = (V, E)$, partition V into subsets $V_1, \ldots, V_p$, $p \leq d$ such that V_i is a k_i-clique for $i = 1, \ldots, p$ and $\max\limits_{i=1,\ldots,p} k_i$ is a minimum. In other words, we wish to partition the graph into no more than d subsets such that each is a k-clique with the smallest possible k. The covering version can also be defined similarly. Note that we do not have formal proof of complexity for these problems although we suspect they are NP-hard.

A Factor Two Approximation. We propose the following bottleneck approach to approximately solve min-max k-clique d-clustering problem. Construct a complete graph $G_C = (V, E_C)$ from $G = (V, E)$ where the edges are weighted using the shortest path distance between the end points in G, $\forall i, j \in V : i < j, w_{ij} = d_G(i, j)$. Define as before, $w(V_i) = \max\limits_{u,v \in V_i : u<v} w_{uv}$ for any $V_i \subseteq V$ and we have V_i to be a $w(V_i)$-clique in G. Thus a clique partition $V_1, \ldots, V_p$ of G_C is a k_i-clique partition of G with $k_i = w(V_i)$ for $i = 1, \ldots, p$ and vice versa. Based on this observation, we solve the min-max d-clustering problem on G_C using the bottleneck Algorithm 10.

Let $I_t = \{1, \ldots, p\}$ be the output of Algorithm 10 terminating in iteration t. Partition V into $V_1, \ldots, V_p$ as follows: $V_1 = \{j \in V \setminus I_t : w_{1j} \leq w_{e_t}\} \cup \{1\}$ and $V_l = \{j \in V \setminus I_t : w_{lj} \leq w_{e_t} \text{ and } j \notin V_1 \cup \ldots \cup V_{l-1}\} \cup \{l\}$ for $l = 2, \ldots, p$. As before $V_1, \ldots, V_p$ are the closed neighborhoods of $1, \ldots, p$ in the bottleneck graph $G_C(w_{e_t})$ excluding vertices in $V_1, \ldots, V_{l-1}$ from V_l for $l = 2, \ldots, p$. Note that a cover can be obtained if they are not excluded. Next we argue that the partition constructed has an objective that is at most twice the optimum following the proof in [115].

Denote the optimum objective by k^*. Since the algorithm did not terminate in

iteration $t-1$, we know that there exists an independent set I_{t-1} such that $|I_{t-1}| > d$ in $G_C(w_{e_{t-1}})$. We claim that there is no clique partition in G_C with at most d cliques and an objective value of $w_{e_{t-1}}$ or less. Suppose the claim was false. Let G'_C denote the union of clique components of such a partition, and every edge weight in G'_C is at most $w_{e_{t-1}}$. Since G'_C is a union of at most d cliques we have $\alpha(G'_C) \leq d$. But G'_C and the bottleneck graph $G_C(w_{e_{t-1}})$ have the same vertex set and $E(G'_C)$ is contained in the edges of $G_C(w_{e_{t-1}})$. Hence we have $\alpha(G'_C) \geq \alpha(G_C(w_{e_{t-1}})) \geq |I_{t-1}| > d$. This contradiction establishes our claim. Since k^* has to be an edge weight, we have $k^* \geq w_{e_t}$. Now $G_C[V_1], \ldots, G_C[V_p]$ is a partition of G_C into p cliques and any edge in any clique $G_C[V_l]$ has weight at most $2w_{e_t}$. This is because any edge in the bottleneck graph $G_C(w_{e_t})$ has weight at most w_{e_t} and the weights obey triangle inequality since the shortest path distances in G satisfy triangle inequality. Thus we have the objective value of the solution produced by the bottleneck approach is at most $2w_{e_t} \leq 2k^*$. It is easy to see that, the algorithm would be optimal if a maximum (instead of maximal) independent set is found. But the approximation algorithm runs in polynomial time if we find MIS.

Type I and Type II k-club clustering problems (covering/partitioning) can be defined similar to the k-clique problems. However, the standard approaches are not easily extended to these problems because of the special properties of k-clubs. The fact that a subset of a k-club need not necessarily be a k-club and the difficulty of finding maximal k-clubs have a lot of impact on the theoretical and algorithmic aspects of these problems as they did in the case of the maximum k-club problem. However, the Type I problem of partitioning an arbitrary graph into the minimum number of k-clubs is studied in [82] (here the problem is called minimum k-clustering). The authors of [82] show that the minimum k-club partitioning problem is NP-hard for every fixed k and it is NP-hard to approximate within a factor of n^{ϵ}. Furthermore,

in [3] minimum k-club partitioning is shown to be NP-hard on bipartite graphs for every fixed positive integer $k > 1$ and for $k = 2$ on chordal graphs (graphs with no chordless cycles). Note the contrast with $k = 1$ case, both these graph classes are perfect and minimum clique partitioning (graph coloring) is polynomial-time solvable on perfect graphs. A shortest path in G of length $diam(G)$ whose vertices form a dominating set is called a *dominating diametral path*. Polynomial-time approximation algorithms for the problem on the class of graphs containing a *dominating diametral path* is provided in [82].

IX.4.2. *k*-Plex Clustering

Minimum k-plex partitioning/covering problem is to partition/cover an arbitrary graph G using minimum number of k-plexes and it is a Type I clustering problem. A Type II *min-max k-plex d-clustering problem* can also be defined analogous to k-cliques, partition (cover) an arbitrary graph G using no more than d subsets such that each subset is a k-plex with the minimum possible k. In this section we will discuss some concepts related to the Type I minimum k-plex partitioning problem.

First, we present the notion of *co-k-plex coloring* generalizing the standard graph coloring problem. Thereby generalizing the equivalence between clique partitioning and coloring into k-plex partitioning and co-k-plex coloring.

Definition 29. A *proper co-k-plex coloring* of a graph is one in which every vertex is colored such that at most $k - 1$ of its neighbors have the same color.

A graph is said to be *t-colorable* if it admits a proper co-k-plex coloring with t colors. Vertices of the same color are referred to as a *color class* and they induce a co-k-plex. The *k-chromatic number* of the graph, denoted by $\chi_k(G)$ is the minimum number of colors that admit a proper co-k-plex coloring in G.

Recall that for any graph G, $\omega_k(G)$ is the size of a maximum k-plex, say S^*. Clearly, $\chi_k(G) \geq \chi_k(G[S^*])$. Any proper co-$k$-plex coloring of G also partitions the maximum k-plex S^* in to co-k-plexes. The number of such co-k-plexes inside S^* and hence the number of colors required will be a minimum when the size of each co-k-plex is a maximum. Based on Lemmas 1 and 2 and the complementary relationship, the size of the maximum co-k-plex inside any k-plex is at most $\rho_k = 2k - 1 - \frac{1+(-1)^k}{2}$. Thus we have the following inequality.

$$\omega_k(G) \leq \rho_k \chi_k(G) \tag{9.1}$$

Note that this inequality holds at equality on graphs G_k that illustrate the sharpness of Lemmas 1 and 2 in Section VI.2. In order to solve the minimum k-plex partitioning problem on G, we can solve the minimum co-k-plex coloring problem on $\bar{G}$ where the color classes form the required cluster partition.

Co-k-plex coloring problem has been well studied in literature as *defective coloring* especially in topological graph theory (see [79] and references therein). Co-k-plex coloring using c colors has been shown to be NP-complete for any $c \geq 3$ and any fixed $k \geq 1$ in [79]. Further, for a fixed k there exists an $\epsilon > 0$ such that unless P=NP there does not exist a polynomial time algorithm that can approximate $\chi_k(G)$ within a factor of n^ϵ [79]. It is also shown that $\chi_k(G) \leq \lceil \frac{\Delta(G)+1}{k} \rceil$ [79] which generalizes the well known result that $\chi(G) \leq \Delta(G)+1$ which follows from a simple greedy heuristic for coloring [188] and it can be easily extended to co-k-plex coloring.

IX.5. Network Design Problem

The properties of k-plex such as relaxed familiarity, robustness and reachability make it an attractive structure that can be used in designing a network. A natural question

to ask is the following: Given a *complete* graph $G_C = (V, K)$, and a cost c_e $\forall e \in K$, select a subset E of all possible edges K to construct $G = (V, E)$ such that the sum of the edge costs of edges in E is a minimum and G is a k-plex for a given k.

Note that we are constructing a k-plex G on n vertices with the objective that sum of edge costs of edges included in the k-plex is a minimum. Recall from Theorem 7 that if $k < \frac{n+2}{2}$ then $diam(G) \leq 2$ and $\kappa(G) \geq n - 2k + 2$. By definition every vertex in G can have at most $k - 1$ non-neighbors. Clearly, if z_k^* denotes the optimum of the above problem $z_{k+1}^* \leq z_k^*$ as every k-plex on n vertices is also a $k + 1$-plex on n vertices. Thus one must choose the largest $k < \frac{n+2}{2}$ based on required familiarity and robustness. Before studying this problem in detail, we should note the following definition from social network literature. Seidman [172] introduced the concept of a *k-core*, which is a graph with minimum degree at least k.

Definition 30. Given $G = (V, E)$, a set $S \subseteq V$ is a *k-core* if $|N(v) \cap S| \geq k \quad \forall\, v \in S$.

The k-core model in SNA was also introduced in the study of social cohesion. However, k-cores were noted to only indicate dense regions of the graph and not necessarily identify a cohesive subgroup [172, 185]. As suggested by Seidman, this approach was only to produce global measures that captured the cohesive subgroups as well as regions surrounding them. We will now describe a simple greedy algorithm that finds the largest k-core in a graph in polynomial time. Pick a vertex v of minimum degree $\delta(G)$, if $\delta(G) \geq k$ then we have a k-core. If $\delta(G) < k$, then that vertex cannot be in a k-core. Hence, delete the corresponding vertex and continue recursively until a maximum k-core or the empty set is found. Note that even though these structures are easy to find, they only point out dense regions of the graph where interesting subgroups may be found.

The connection to our network design model is as follows. For a given n and k,

from among all graphs on n vertices we wish to find a graph G of minimum degree $n-k$ that minimizes a certain objective. Stated in this format, the network design problem resembles *extremal graph theoretic problems* [39] rather than the graph optimization problems we studied so far, wherein we look for optimal structures/subsets in a given graph G. Henceforth we refer to our design problem as the *k-core network design problem* to emphasize that the degree lower-bound is fixed.

The following is a binary IP formulation of the k-core network design problem. Assume that $k = n - k'$ where $k' < \frac{n+2}{2}$ is appropriately chosen based on required familiarity and robustness. The binary variable x_e is one if and only if edge e is included in E. Recall that K is the set of all possible edges and let K_v denote all possible edges incident at v for each $v \in V$.

$$z_k^* = \min \sum_{e \in K} c_e x_e \tag{9.2}$$

subject to:

$$\sum_{e \in K_v} x_e \geq k \quad \forall\, v \in V \tag{9.3}$$

$$x_e \in \{0,1\} \quad \forall\, e \in K \tag{9.4}$$

It turns out that the k-core network design problem can be shown to be polynomial-time solvable by reduction to *the maximum weighted b-matching problem*. Given a graph $G = (V, E)$ (not necessarily complete) and a vector $b \in \mathbb{Z}_+^{|V|}$, a *b-matching* is a subset of edges M such that every vertex $v \in V$ is incident with at most b_v edges in M. The *maximum weight b-matching problem* defined on G with edge weights c_e $\forall e \in E$ is to find a b-matching M such that sum of the edge weights of edges in M is a maximum. When every component of b equals one, we have the classical definition of a matching. When every component of c equals one, the problems are simply called the *maximum matching problem* or the *maximum b-matching problem*.

The classical paper of Edmonds [86] established the polynomial-time solvability of this problem by a "blossom" algorithm. This concept was used by Edmonds in obtaining the complete linear description of the associated matching polytope in the paper [85] considered a cornerstone of polyhedral combinatorics. The minimal defining system for the convex hull of b-matchings was identified in [73]. A pseudo-polynomial algorithm for solving the maximum weighted b-matching was provided by Edmonds and Pulleyblank [164, 85]. A strongly polynomial-time algorithm was provided by Anstee [14]. The readers are referred to [139, 170] for results on matchings, b-matchings and extensions.

The reduction of k-core network design to maximum b-matching problem follows from the observation that by solving the latter on $G_C = (V, K)$ for a particular choice of b identifies those edges that must be *excluded* in the k-core network design problem. To illustrate, we first formulate the latter into the following integer program. Let $G_C = (V, K)$, K_v be defined as before and let y be the incidence vector of b-matchings in G_C.

$$z_b^* = \max \sum_{e \in K} c_e y_e \tag{9.5}$$

subject to:

$$\sum_{e \in K_v} y_e \leq b_v \quad \forall\, v \in V \tag{9.6}$$

$$y_e \in \{0, 1\} \quad \forall\, e \in K \tag{9.7}$$

The reduction is clear when we make the observation that k-core network design on G_C can be reduced to this problem with the substitution that $x_e = 1 - y_e$. Then we have,

$$z_k^* = \min \left\{ C - \sum_{e \in K} c_e y_e : n - 1 - \sum_{e \in K_v} y_e \geq k \;\; \forall\, v \in V, y_e \text{ binary } \forall e \in K \right\},$$

$$z_k^* = C - \max\left\{\sum_{e\in K} c_e y_e : \sum_{e\in K_v} y_e \leq n-1-k \;\; \forall\, v \in V, y_e \text{ binary } \forall e \in K\right\},$$

where $C = \sum_{e\in K} c_e$ and $k \leq n-1$. Thus minimum k-core network design problem can be solved in polynomial time by reducing to maximum weighted b-matching on $G_C = (V, K)$ where each component of b equals $n-1-k$. Clearly, the general maximum weighted b-matching algorithm can be drastically improved given our special conditions. This is however a topic for future research.

Summary. This chapter was exploratory in nature where we introduced the readers to classical network clustering approaches that utilize cliques. Sensitivity of cliques to missing edges arising from experimental errors given the nature of the real-life networks was the motivation for employing clique relaxations in clustering. Type I and II approaches for clustering using clique relaxations are described. The concept of graph coloring was also generalized. A network design extension was introduced and its polynomial-time solvability was established in this chapter. The next chapter concludes this dissertation by summarizing our contributions and by restating open problems identified in the previous chapters. We also outline directions for future research with the optimization problems dealt with in this chapter.

CHAPTER X

CONCLUSION AND FUTURE WORK

This dissertation considered graph theoretic generalizations of the classical maximum clique problem. Models that were originally proposed in SNA literature, are investigated from a mathematical programming perspective for the first time. The k-clique, k-club and k-plex models are compared based on structural properties expected of a cohesive subgroup that are guaranteed by their definitions, and the associated optimization problems are formally defined. Emphasis was on the maximum k-plex problem in this dissertation. Our bias is justified by the structural guarantees of a k-plex that make it a systematic and realistic relaxation of cliques. We will now summarize our specific contributions and outline possible directions for future research.

Complexity. Computational complexity of k-clique, k-club and k-plex models are the first significant contributions of this work. Although, the problems (decision versions) are trivially NP-complete for arbitrary k, their complexity when k was a fixed integer was unknown. We establish that all three problems are NP-complete for every *fixed* positive integer k on arbitrary graphs. Further, the k-clique and k-club problems were shown to be NP-complete for fixed k on graphs of diameter d for all $d > k$. As a special case, all three optimization problems are shown to be polynomial-time solvable when $k = n - t$, for every fixed positive integer t.

Complexity issues that need to be addressed in the future are as follows. Analogous to our complexity results on bounded diameter graphs for k-cliques and k-clubs, it would be interesting to establish the complexity of finding a maximum k-plex in a graph that is a $(k + 1)$-plex. Some special situations can be immediately resolved. Finding, a maximum clique in a 2-plex is equivalent to finding a maximum indepen-

dent set in a co-2-plex. Since every co-2-plex is also *claw-free* (does not contain $K_{1,3}$ as an induced subgraph), this problem can be solved in polynomial time [146]. By the same argument maximum clique in a 3-plex can also be found in polynomial time. However, polynomial solvability may not last for long since the maximum independent set is NP-hard to solve on planar cubic graphs [98].

Complexity of maximum k-plex, k-club and k-clique problems on restricted graph classes such as planar and perfect graphs is important, even more so on graph classes that have practical applicability such as *unit disk graphs*. Disk graphs are intersection graphs of circles on a plane and are used to model connectivity information in wireless communication. An interesting fact is that on unit disk graphs, the maximum clique problem is polynomial-time solvable while the maximum independent set problem is NP-hard [70].

Another important problem that needs to be addressed in the future is with regards to finding inclusion-wise maximal k-clubs. The lack of hierarchical property makes this problem non-trivial, however its complexity is still open. Resolving this problem will have a great impact on algorithmic approaches, exact and heuristic for solving the maximum k-club problem.

Polyhedra. The k-plex polytope and the maximum k-plex problem received the most attention in this dissertation. The notion of a co-k-plex is introduced in this dissertation that complements a k-plex and generalizes an independent set in a graph. Sharp bounds are established for the size of a k-plex inside a co-k-plex, inside a hole and inside an independent set. Based on these results, valid inequalities for the k-plex polytope are developed. The co-k-plex inequalities are especially interesting since they generalize MIS inequalities that play an important role in describing the clique polytope as well as their role in polyhedral characterization of graph perfection. We find that co-2-plex inequalities induce facets of the 2-plex polytope while the result

is not true in general for $k \geq 3$.

In the future, it would be interesting to identify graph classes for which co-k-plex inequalities are facet inducing. This could be facilitated by redefining them as *rank inequalities* [69] instead of using the general upper-bound ρ_k for the size of a maximum k-plex inside a co-k-plex, *i.e.*,

$$\sum_{i \in J} x_i \leq \omega_k(G[J])$$

where J is a maximal co-k-plex. Note that $\omega_k(G[J]) \leq \rho_k$. It would also be possible to generalize antiweb inequalities introduced by Trotter [182] for the clique polytope to the k-plex polytope. Identifying other subgraphs for generating facets and valid inequalities, as well as lifting techniques to make them globally valid is also an important problem for future research. In fact, one of the earliest applications of this approach was in lifting odd hole inequalities that are facet defining for the independent set polytope of a hole H, to the graph containing H [158].

The binary IP formulation for the maximum k-plex problem when $k = 1$ led to a novel formulation for the maximum clique problem which is possibly the most compact IP formulation for the problem. This formulation is shown theoretically to have a weaker LP relaxation compared to the popular edge formulation for the problem, but appears to be more effective experimentally. It would be interesting to devise a method to "mix" the two formulations to yield one that is computationally more effective in a BC setting.

The foundation for polyhedral study of the k-clique and k-club models were also laid in this dissertation. Since the k-clique polytope is exactly the clique polytope of the power graph, the polyhedral results known for the clique polytope can be extended. The maximum k-club problem is formulated as a binary integer program. However, it is especially difficult to address using traditional polyhedral methods since

it is not hierarchical in nature. Despite this difficulty, a family of valid inequalities called the *maximal k-independent set inequalities* are identified. A compact formulation is possible for the problem when restricted to the special case $k = 2$. Interesting properties of the 2-club polytope are also revealed in this dissertation and maximal 2-independent set inequalities are found to be facet-inducing for the 2-club polytope.

The absence of hierarchical property in 2-clubs also manifests itself in the following way. Traditional polyhedral approaches would entail finding combinatorial valid inequalities for the 2-club polytope $W_2(G)$ by first finding a specific induced subgraph G' and facets of $W_2(G')$. Facets of $W_2(G')$ are not necessarily facet-defining for the higher-dimensional polytope $W_2(G)$, but are usually expected to be *valid* for $W_2(G)$. This however is not true in general for 2-clubs. For instance, let G denote the star graph in Fig. 16, and let G' denote the edgeless graph obtained by deleting vertex 6. Now the inequality $x_1 + x_2 + x_3 + x_4 + x_5 \leq 1$ is facet defining for $W_2(G')$ but it is not even valid for $W_2(G)$ as it cuts off a feasible integer point $[1, 1, 1, 1, 1, 1]$ in $W_2(G)$. But lifting x_6 would yield $-4x_6 + x_1 + x_2 + x_3 + x_4 + x_5 \leq 1$ which is a facet of $W_2(G)$. Note that lifting here was necessary to generate a *valid* inequality of $W_2(G)$ from one that was *invalid*. Focussing on this issue for the 2-club polytope is important as it addresses some fundamental issues for the case of general k as well as contribute to the theory of combinatorial optimization for such non-hierarchical problems.

Algorithms. Given the intractability of all the optimization problems, worst case exponential algorithms are inevitable unless $P = NP$. BC techniques facilitated by the polyhedral study of the problems is the approach taken by this dissertation. Variable fixing procedures are identified for the maximum k-plex problem based on its domination property. A peeling procedure is also developed to remove vertices from the graph based on the size of a known k-plex. These procedures are incorporated in a BC framework for the problem. The algorithms developed are implemented and

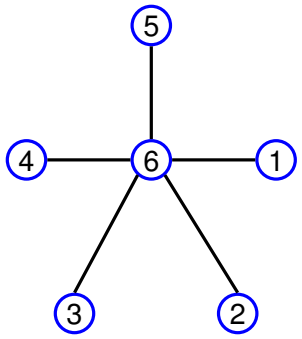


Fig. 16 A star graph

computationally tested using MIS cuts and co-k-plex cuts. Computational experiments indicate that the preprocessing approaches enable optimal resolution of large sparse graphs, such as real-life graphs based on social and biological data. Similar approaches are also developed for the maximum 2-club problem and a BC is implemented using maximal 2-independent set facets.

Although the current implementations are working extremely well given the hardness of the problems, there are still aspects of the implementation that can be improved. For instance, we have observed from our computational experiments that faster separation heuristics are needed to make co-k-plex cuts effective in practice. An experimental study to determine a good combination of cuts would be beneficial. It appears in the current setting that few cuts added at each node is reasonably effective. However, the number and frequency of cuts can be made dynamic. Frequent addition of larger number of cuts when the cuts are found to be "effective" at some nodes, and vice versa could also improve the implementation. Effectiveness of cuts could also be measured as the *depth* of the cut (Euclidean distance of the cut from LP optimum cut off) instead of constraint violation as in the current implementation. Ordering the cuts in the cut pool based on other quality measures (such as depth) and selecting the top few could also be considered. This approach needs to be traded

off against the addition of "similar" cuts. Cuts that are close and almost parallel could all have high measure of effectiveness and we might choose not to add all such parallel cuts in order to keep the size of the linear system small. In general, other aspects of managing the cut pool such as number of cuts generated and applied, purging cuts when they cease to be effective and size of the cut pool itself are important considerations. The decision to lift local cuts brings an additional dimension to the implementation issue. However, dynamic cut generation, addition and cut pool management are extremely difficult to address by using the commercial software currently available. But one can expect this situation to change rapidly.

Finally, it is necessary to develop meta-heuristic approaches for both maximum k-plex and k-club problems given their intractability and the massive size of real-life data. It should be noted however, that traditional approaches for neighborhood definition and local search may be ineffective in the case of the maximum k-club problem due to its non-hierarchical nature. Heuristics for maximum k-plex and k-club problems could also benefit greatly by adopting the preprocessing techniques developed in this dissertation.

Continuous Formulations. We presented in this dissertation a new continuous fractional formulation for the classical maximum independent set problem. A deterministic proof of correctness of the formulation which maximizes a continuous function over the unit hypercube is presented. Its local maxima are proven to be binary vectors and characterized in terms of structures they represent in the graph. The formulation is suitably modified to induce one-to-one correspondence between MIS and local maxima of the continuous formulation.

A problem for study in the immediate future is to use the continuous fractional program for independence number, Formulation (8.1) in a global optimization algorithm. This is encouraged by the fact that maximizing *sum of affine ratios fractional*

programs subject to linear constraints can be reformulated such that the Lagrangean dual is a linear program [142].

Our work is an attempt to strengthen the link between discrete combinatorial optimization and continuous global optimization that many researchers have forged over the past decades with the hope that any breakthrough in one field would also benefit the other. Continuous approaches to combinatorial optimization problems have led to several effective heuristics and exact algorithms for the problems. Commercial packages are also available that are capable of handling large instances of several types of global optimization formulations. Furthermore, continuous approaches present interesting and distinct perspectives on the combinatorial problems. Developing effective continuous approaches to the clique relaxations would be extremely beneficial for these reasons.

Modeling Extensions. In this dissertation, we have presented exploratory work on clustering using clique relaxations. After surveying some existing approaches that use clique-based clustering, we have argued that clique relaxations are more practical when clustering data from real life. To this end, two types of clustering optimization problems are defined for each model, based on the objective function used. Further, each type of clustering problem on each model could be defined as a covering or partitioning version. We briefly review existing literature related to minimum clique partitioning and covering that is applicable to minimum k-clique partitioning and covering problems. We then present a polynomial-time factor-two approximation algorithm for the min-max k-clique d-clustering problem based on a bottleneck approach. Existing results for k-club partitioning are also surveyed. We introduce the notion of co-k-plex coloring which is equivalent to k-plex partitioning in the complement graph.

A novel model is proposed to design low-diameter networks with desired minimum degree and connectivity. This model, called the k-core network design problem

is shown to be polynomial-time solvable using algorithms for the b-matching problem.

Exact and heuristic approaches to solve these problems and a detailed computational study on real data is extremely important and of great practical value. In particular, developing column generation approaches for the minimum co-k-plex coloring problem along similar lines as those for the classical coloring problem proposed in [144] would be useful. It would also be beneficial to develop other practical modeling extensions of the clique relaxations that exploit their desirable properties. Innovative modeling of real-life problems is as important as creativity in theory and solution, and these clique relaxations have tremendous potential in this regard.

REFERENCES

1. Aardal, K., van Hoesel, S.: Polyhedral techniques in combinatorial optimization I: Theory. Statistica Neerlandica **50**(3), 3–26 (1996)

2. Aardal, K.I., van Hoesel, C.P.M.: Polyhedral techniques in combinatorial optimization II: Computations and applications. Statistica Neerlandica **53**, 129–178 (1999)

3. Abbas, N., Stewart, L.: Clustering bipartite and chordal graphs: Complexity, sequential and parallel algorithm. Discrete Applied Mathematics **91**(1-3), 1–23 (1999)

4. Abello, J., Butenko, S., Pardalos, P., Resende, M.: Finding independent sets in a graph using continuous multivariable polynomial formulations. Journal of Global Optimization **21**, 111–137 (2001)

5. Abello, J., Pardalos, P., Resende, M.: On maximum clique problems in very large graphs. In: J. Abello, J. Vitter (eds.) External Memory Algorithms and Visualization, pp. 119–130. American Mathematical Society, Boston (1999)

6. Abello, J., Pardalos, P., Resende, M. (eds.): Handbook of Massive Data Sets. Kluwer Academic Publishers, Dordrecht, The Netherlands (2002)

7. Abello, J., Resende, M., Sudarsky, S.: Massive quasi-clique detection. In: S. Rajsbaum (ed.) LATIN 2002: Theoretical Informatics, pp. 598–612. Springer-Verlag, London (2002)

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

9. Albert, R., Jeong, H., Barabási, A.L.: Error and attack tolerance of complex networks. Nature **406**, 378–382 (2000)

10. Almaas, E., Barabási, A.L.: Power laws in biological networks. In: E. Koonin, Y.I. Wolf, G.P. Karev (eds.) Power Laws, Scale-Free Networks and Genome Biology, pp. 1–11. Springer Science + Business Media, New York (2006)

11. Alon, N., Spencer, J.H., Erdös, P.: The Probabilistic Method. Wiley, New York (1991)

12. Anderberg, M.R.: Cluster Analysis for Applications. Academic Press, New York (1973)

13. de Angelis, P.L., Bomze, I.M., Toraldo, G.: Ellipsoidal approach to box-constrained quadratic problems. Journal of Global Optimization **28**, 1–15 (2004)

14. Anstee, R.P.: A polynomial algorithm for b-matchings: An alternative approach. Information Processing Letters **24**(3), 153–157 (1987)

15. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: On the solution of traveling salesman problems. Documenta Mathematica **ICM III**, 645–656 (1998)

16. Asahiro, Y., Iwama, K., Tamaki, H., Tokuyama, T.: Greedily finding dense subgraphs. Journal of Algorithms **34**, 203–221 (2000)

17. Bader, J.S., Chaudhuri, A., Rothberg, J.M., Chant, J.: Gaining confidence in high-throughput protein interaction networks. Nature Biotechnology **22**(1), 78–85 (2004)

18. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Mathematical Programming **58**, 295–324 (1993)

19. Balas, E., Ceria, S., Cornuéjols, G.: Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. Management Science **42**, 1229–1246 (1996)

20. Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. Operations Research Letters **19**, 1–9 (1996)

21. Balas, E., Ceria, S., Cornuejols, G., Pataki, G.: Polyhedral methods for the maximum clique problem. In: D.S. Johnson, M.A. Trick (eds.) Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, pp. 11–28. American Mathematical Society, Providence, RI (1996)

22. Balas, E., Xue, J.: Weighted and unweighted maximum clique algorithms with upper bounds from factional coloring. Algorithmica **15**, 397–412 (1996)

23. Balas, E., Yu, C.: Finding a maximum clique in an arbitrary graph. SIAM Journal of Computing **15**, 1054–1068 (1986)

24. Balasundaram, B., Butenko, S.: Constructing test functions for global optimization using continuous formulations of graph problems. Journal of Optimization Methods and Software **20**(4-5), 439–452 (2005)

25. Balasundaram, B., Butenko, S.: Graph domination, coloring and cliques in telecommunications. In: M.G.C. Resende, P.M. Pardalos (eds.) Handbook of Optimization in Telecommunications, pp. 865–890. Springer Science + Business Media, New York (2006)

26. Balasundaram, B., Butenko, S.: On a polynomial fractional formulation for independence number of a graph. Journal of Global Optimization **35**(3), 405–421 (2006)

27. Balasundaram, B., Butenko, S.: Network clustering. In: B.H. Junker, F. Schreiber (eds.) Analysis of Biological Networks. Wiley, New York (2007). To Appear.

28. Balasundaram, B., Butenko, S.: Optimization problems in unit-disk graphs. In: C.A. Floudas, P.M. Pardalos (eds.) Encyclopedia of Optimization. Springer Science + Business Media, New York (2008). To Appear.

29. Balasundaram, B., Butenko, S., Hicks, I.V., Sachdeva, S.: Clique relaxations in social network analysis: The maximum k-plex problem (2007). Submitted.

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

31. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)

32. Barabási, A.L., Albert, R., Jeong, H.: Scale-free characteristics of random networks: The topology of the World Wide Web. Physica A **281**, 69–77 (2000)

33. Batagelj, V., Mrvar, A.: Pajek datasets (2006). Online: http://vlado.fmf.uni−lj.si/pub/networks/data/. Accessed June 2006

34. Berge, C.: Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind (zusammenfassung). Wissenschaftliche Zeitschrift, Martin Luther Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche **10**, 114–115 (1961)

35. Berge, C., Ramírez-Alfonsín, J.L.: Origins and genesis. In: J.L. Ramírez-Alfonsín, B.A. Reed (eds.) Perfect Graphs, pp. 1–12. Wiley, New York (2001)

36. Berry, N., Ko, T., Moy, T., Smrcka, J., Turnley, J., Wu, B.: Emergent clique formation in terrorist recruitment. The AAAI-04 Workshop on Agent Organizations: Theory and Practice, July 25, 2004, San Jose, California (2004). Online: http://www.cs.uu.nl/ virginia/aotp/papers.htm

37. Boginski, V., Butenko, S., Pardalos, P.M.: Modeling and optimization in massive graphs. In: P.M. Pardalos, H. Wolkowicz (eds.) Novel Approaches to Hard Discrete Optimization, pp. 17–39. American Mathematical Society, Providence, RI (2003)

38. Boginski, V., Butenko, S., Pardalos, P.M.: On structural properties of the market graph. In: A. Nagurney (ed.) Innovation in Financial and Economic Networks. Edward Elgar Publishers, London (2003)

39. Bollobás, B.: Extremal Graph Theory. Academic Press, New York (1978)

40. Bollobás, B., Erdös, P.: Cliques in random graphs. Math. Proc. Camb. Phil. Soc. **80**, 419–427 (1976)

41. Bomze, I.M.: Evolution towards the maximum clique. Journal of Global Optimization **10**, 143–164 (1997)

42. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In: D.Z. Du, P.M. Pardalos (eds.) Handbook of Combinatorial Optimization, pp. 1–74. Kluwer Academic Publishers, Dordrecht, The Netherlands (1999)

43. Boppana, R., Halldórsson, M.M.: Approximating maximum independent sets by excluding subgraphs. BIT **32**(2), 180–196 (1992)

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

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

46. Brélaz, D.: New methods to color the vertices of a graph. Communications of the ACM **22**(4), 251–256 (1979)

47. Biomolecular relations in information transmission and expression. Generalized protein interactions (2005). Online: http://www.genome.jp/brite/generalized_interactions.html. Accessed March 2005

48. Broido, A., Claffy, K.C.: Internet topology: connectivity of IP graphs. In: S. Fahmy, K. Park (eds.) Scalability and Traffic Control in IP Networks, pp. 172–187. SPIE Publications, Bellingham, WA (2001)

49. Burer, S., Monteiro, R.D.C., Zhang, Y.: Maximum stable set formulations and heuristics based on continuous optimization. Mathematical Programming **94**, 137–166 (2002)

50. Busygin, S., Butenko, S., Pardalos, P.M.: A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. Journal of Combinatorial Optimization **6**, 287–297 (2002)

51. Butenko, S., Festa, P., Pardalos, P.M.: On the chromatic number of graphs. J. Optim. Theory Appl. **109**, 51–67 (2001)

52. Butenko, S., Pardalos, P.M., Sergienko, I.V., Shylo, V., Stetsyuk, P.: Finding maximum independent sets in graphs arising from coding theory. In: Proceedings of the Seventeenth ACM Symposium on Applied Computing, pp. 542–546. ACM Press, New York (2002)

53. Butenko, S., Wilhelm, W.: Clique-detection models in computational biochemistry and genomics. European Journal of Operational Research **173**, 1–17 (2006)

54. Cánovas, L., Landete, M., Marín, A.: Facet obtaining procedures for set packing problems. SIAM Journal of Discrete Mathematics **16**(1), 127155 (2003)

55. Cardoso, D.M.: Convex quadratic programming approach to the maximum matching problem. Journal of Global Optimization **21**, 91–106 (2001)

56. Caro, Y., Tuza, Z.: Improved lower bounds on k-independence. J. Graph Theory **15**, 99–107 (1991)

57. Carraghan, R., Pardalos, P.: An exact algorithm for the maximum clique problem. Operations Research Letters **9**, 375–382 (1990)

58. Ceria, S., Cordier, C., Marchand, H., Wolsey, L.A.: Cutting plane algorithms for integer programs with general integer variables. Mathematical Programming **81**, 201–214 (1998)

59. Chang, G.J., Nemhauser, G.L.: The k-domination and k-stability problems on sun-free chordal graphs. SIAM Journal on Algebraic and Discrete Methods **5**, 332–345 (1984)

60. Chen, H., Chung, W., Xu, J.J., Wang, G., Qin, Y., Chau, M.: Crime data mining: A general framework and some examples. Computer **37**(4), 50–56 (2004)

61. Chen, H., Zeng, D., Atabakhsh, H., Wyzga, W., Schroeder, J.: COPLINK: Managing law enforcement data and knowledge. Communications of the ACM **46**(1), 28–34 (2003)

62. Chen, Y.P., Liestman, A.L., Liu, J.: Clustering algorithms for ad hoc wireless networks. In: Y. Pan, Y. Xiao (eds.) Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing, pp. 145–164. Nova Science Publishers, New York (2005)

63. Cheng, E., Cunningham, W.H.: Wheel inequalities for stable set polytopes. Mathematical Programming **77**, 389421 (1997)

64. Cheng, E., de Vries, S.: Antiweb-wheel inequalities and their separation problems over the stable set polytopes. Mathematical Programming **92**, 153175 (2002)

65. Cheng, E., de Vries, S.: On the facet-inducing antiweb-wheel inequalities for stable set polytopes. SIAM Journal of Discrete Mathematics **15**(4), 470–487 (2002)

66. Chesler, E.J., Langston, M.A.: Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data. Tech. Rep. ut-cs-06-575, CS Technical Reports, University of Tennessee, Knoxville (2006)

67. Chudnovsky, M., Robertson, N., Seymour, P.D., Thomas, R.: The strong perfect graph theorem (manuscript) (2002). Online: http://www.math.gatech.edu/∼thomas/spgc.html. Accessed February 2006

68. Chung, F., Lu, L.: Complex Graphs and Networks. CBMS Lecture Series. American Mathematical Soceity, Providence, RI (2006)

69. Chvátal, V.: On certain polytopes associated with graphs. Journal of Combinatorial Theory (B) **18**, 138–154 (1975)

70. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. Discrete Mathematics **86**, 165–177 (1990)

71. Cook, D.J., Holder, L.B.: Graph-based data mining. IEEE Intelligent Systems **15**(2), 32–41 (2000)

72. Cook, W., Cunningham, W., Pulleyblank, W., Schrijver, A.: Combinatorial Optimization. John Wiley and Sons, New York (1998)

73. Cook, W., Pulleyblank, W.R.: Linear systems for constrained matching problems. Mathematics of Operations Research **12**(1), 97–120 (1987)

74. Cordier, C., Marchand, H., Laundy, R., Wolsey, L.A.: bc-opt: A branch-and-cut code for mixed integer programs. Mathematical Programming **86**(2), 335–354 (1999)

75. Corman, S., Dooley, K., McPhee, R.: LOCKS: Analysis of media coverage of the terrorist attacks (2006). Online: http://locks.asu.edu/terror/. Accessed June 2006

76. Corman, S., Kuhn, T., McPhee, R., Dooley, K.: Studying complex discursive systems: Centering resonance analysis of organizational communication. Human Communication Research **28**(2), 157–206 (2002)

77. Corneil, D., Perl, Y.: Clustering and domination in perfect graphs. Discrete Applied Mathematics **9**, 27–39 (1984)

78. Cornuéjols, G.: Combinatorial Optimization: Packing and Covering. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia (2001)

79. Cowen, L., Goddard, W., Jesurum, C.E.: Defective coloring revisited. Journal of Graph Theory **24**(3), 205–219 (1997)

80. Davis, R.H.: Social network analysis: An aid in conspiracy investigations. FBI Law Enforcement Bulletin **50**(12), 11–19 (1981)

81. Dekker, A.H.: Social network analysis in military headquarters using CAVALIER. pp. 24–26. Canberra, Australia (2000)

82. Deogun, J., Kratsch, D., Steiner, G.: An approximation algorithm for clustering graphs with dominating diametral path. Information Processing Letters **61**, 121–127 (1997)

83. Diestel, R.: Graph Theory. Springer-Verlag, Berlin (1997)

84. DIMACS: Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge (1995). Online: http://dimacs.rutgers.edu/Challenges/. Accessed March 2007

85. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. Journal of Research of the National Bureau of Standards - B **69B**, 125–130 (1965)

86. Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics **17**, 449–467 (1965)

87. Erdös, P., Rényi, A.: On random graphs. Publicationes Mathematicae **6**, 290–297 (1959)

88. Erdös, P., Rényi, A.: On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci. **5**, 17–61 (1960)

89. Euler, R., Jünger, M., Reinelt, G.: Generalizations of cliques, odd cycles and anticycles and their relation to independence system polyhedra. Mathematics of Operations Research **12**, 451–462 (1987)

90. Feige, U., Kortsarz, G., Peleg, D.: The dense k-subgraph problem. Algorithmica **29**, 410–421 (2001)

91. Fischer, I., Meinl, T.: Graph based molecular data mining - an overview. In: Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, pp. 4578–4582. IEEE, Piscataway, NJ (2004)

92. The MathWorks MATLAB® Optimization Toolbox - `fmincon`. Online: http://www.mathworks.com/access/helpdesk/help/toolbox/optim/fmincon.html. Accessed August 2004

93. Freeman, L.C.: The sociological concept of "group": An empirical test of two models. American Journal of Sociology **98**, 152–166 (1992)

94. Fulkerson, D.R.: Blocking and anti-blocking pairs of polyhedra. Mathematical Programming **1**, 168–194 (1971)

95. Gagneur, J., Krause, R., Bouwmeester, T., Casari, G.: Modular decomposition of protein-protein interaction networks. Genome Biology **5**(8), R57.1–R57.12 (2004)

96. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM Journal on Computing **18**(1), 30–55 (1989)

97. GAMS. Online: http://www.gams.com/. Accessed March 2007

98. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York (1979)

99. Gibbons, L.E., Hearn, D.W., Pardalos, P.M.: A continuous based heuristic for the maximum clique problem. In: D.S. Johnson, M.A. Trick (eds.) Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, pp. 103–124. American Mathematical Society, Providence, RI (1996)

100. Gibbons, L.E., Hearn, D.W., Pardalos, P.M., Ramana, M.V.: Continuous characterizations of the maximum clique problem. Mathematics of Operations Research **22**, 754–768 (1997)

101. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences **99**(12), 7821–7826 (2002)

102. Glover, F.: Tutorial on surrogate constraint approaches for optimization in graphs. Journal of Heuristics **9**, 175–227 (2003)

103. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38**(2-3), 293–306 (1985)

104. GRAPHVIZ: Graph visualization software. Online: http://www.graphviz.org/About.php. Accessed March 2007

105. Grossman, J., Ion, P., Castro, R.D.: The Erdös Number Project (1995). Online: http://www.oakland.edu/enp/. Accessed March 2007

106. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, 2nd edn. Springer-Verlag, Berlin (1993)

107. Harant, J.: A lower bound on the independence number of a graph. Discrete Mathematics **188**, 239–243 (1998)

108. Harant, J.: Some news about the independence number of a graph. Discussiones Mathematicae Graph Theory **20**, 71–79 (2000)

109. Harant, J., Pruchnewski, A., Voigt, M.: On dominating sets and independent sets of graphs. Combinatorics, Probability and Computing **8**, 547–553 (1999)

110. Harary, F., Ross, I.C.: A procedure for clique detection using the group matrix. Sociometry **20**, 205–215 (1957)

111. Hartigan, J.A.: Clustering Algorithms. John Wiley and Sons, New York (1975)

112. Hasselberg, J., Pardalos, P.M., Vairaktarakis, G.: Test case generators and computational results for the maximum clique problem. Journal of Global Optimization **3**, 463–482 (1993)

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

114. Hochbaum, D.S.: Approximation Algorithms for NP-hard Problems. PWS Publishing Company, Boston (1997)

115. Hochbaum, D.S., Shmoys, D.B.: A unified approach to approximation algorithms for bottleneck problems. Journal of the ACM **33**(3), 533–550 (1986)

116. Holton, D.A., Sheehan, J.: The Petersen Graph. Cambridge University Press, Cambridge, England (1993)

117. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization, 2nd edn. Kluwer Academic Publishers, Dordrecht, The Netherlands (2000)

118. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search. Journal of Global Optimization **14**, 331–355 (1999)

119. Ilog Cplex. Online: http://www.ilog.com/products/cplex/. Accessed March 2007

120. Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., Sakaki, Y.: A comprehensive two-hybrid analysis to explore the yeast protein interactome. Proceedings of the National Academy of Sciences of the USA **98**(8), 4569–4574 (2001)

121. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Upper Saddle River, NJ (1988)

122. Jambu, M., Lebeaux, M.O.: Cluster Analysis and Data Analysis. North-Holland, New York (1983)

123. Jeong, H., Mason, S.P., Barabási, A.L., Oltvai, Z.N.: Centrality and lethality of protein networks. Nature **411**, 41–42 (2001). Online: http://www.nd.edu/~networks/database/index.html. Accessed March 2005

124. Jiang, D., Tang, C., Zhang, A.: Cluster analysis for gene expression data: A survey. IEEE Transactions on Knowledge and Data Engineering **16**(11), 1370–1386 (2004)

125. Johnson, D., Trick, M. (eds.): Cliques, Coloring, and Satisfiablility: Second DIMACS Implementation Challenge. American Mathematical Society, Providence, RI (1996)

126. Junker, B.H.: Biological networks. In: B.H. Junker, F. Schreiber (eds.) Analysis of Biological Networks. Wiley, New York (2007)

127. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining, pp. 137–146. ACM Press, New York (2003)

128. Khachian, L.G.: A polynomial algorithm in linear programming. Soviet Mathematics Doklady **20**, 1093–1096 (1979)

129. Korman, S.M.: The graph-colouring problem. In: N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.) Combinatorial Optimization, pp. 211–235. Wiley, New York (1979)

130. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms, 2 edn. Springer-Verlag, Berlin (2002)

131. Kortsarz, G., Peleg., D.: On choosing a dense subgraph. In: Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, pp. 692–701. IEEE, Piscataway, NJ (1993)

132. Krishna, P., Vaidya, N.H., Chatterjee, M., Pradhan, D.K.: A cluster-based approach for routing in dynamic networks. ACM SIGCOMM Computer Communication Review **27**(2), 49–64 (1997)

133. Kubale, M., Jackowski, B.: A generalized implicit enumeration algorithm for graph coloring. Communications of the ACM **28**(4), 412–418 (1985)

134. Laguna, M., Martí, R.: A GRASP for coloring sparse graphs. Computational Optimization and Applications **19**(2), 165–178 (2001)

135. Laurent, M.: A generalization of antiwebs to independence systems and their canonical facets. Mathematical Programming **45**, 97108 (1989)

136. Lawler, E.: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart, and Winston, New York (1976)

137. Lovàsz, L.: A characterization of perfect graphs. Journal of Combinatorial Theory **13**, 95–98 (1972)

138. Lovàsz, L.: Normal hypergraphs and the perfect graph conjecture. Discrete Mathematics **2**, 253–267 (1972)

139. Lovàsz, L., Plummer, M.: Matching Theory. Elsevier, New York (1986)

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

141. Luce, R., Perry, A.: A method of matrix analysis of group structure. Psychometrika **14**, 95–116 (1949)

142. M. Dür, R.H., Thoai, N.: Solving sum-of-ratios fractional programs using efficient points. Optimization **49**, 447466 (2001)

143. McAndrew, D.: The structural analysis of criminal networks. In: D. Canter, L. Alison (eds.) The Social Psychology of Crime: Groups, Teams, and Networks, Offender Profiling Series, III. Dartmouth, Aldershot, UK (1999)

144. Mehrotra, A., Trick, M.A.: A column generation approach for graph coloring. INFORMS Journal on Computing **8**(4), 344–354 (1996)

145. Milgram, S.: The small world problem. Psychology Today **1**, 61–67 (1967)

146. Minty, G.: On maximal independent sets of vertices in claw-free graphs. Journal of Combinatorial Theory, Series B **28**, 284–304 (1980)

147. Mitchell, J.: Branch-and-cut algorithms for combinatorial optimization problems. In: P.M. Pardalos, M.G.C. Resende (eds.) Handbook of Applied Optimization, pp. 65–77. Oxford University Press, New York (2002)

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

149. Moon, J.W., Moser, L.: On cliques in graphs. Israel Journal of Mathematics **3**, 23–28 (1965)

150. Motzkin, T.S., Straus, E.G.: Maxima for graphs and a new proof of a theorem of Turán. Canad. J. Math. **17**, 533–540 (1965)

151. Mukherjee, M., Holder, L.B.: Graph-based data mining on social networks. In: Workshop on Link Analysis and Group Detection. ACM Press, New York (2004)

152. Nemhauser, G.L., Trotter, L.E.: Properties of vertex packings and independence system. Mathematical Programming **6**, 48–61 (1974)

153. Nemhauser, G.L., Trotter, L.E.: Vertex packing: structural properties and algorithms. Mathematical Programming **8**, 232–248 (1975)

154. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1999)

155. Neumaier, A.: Global optimization website. Online: http://www.mat.univie.ac.at/∼neum/glopt.html. Accessed November 2004

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

157. Padberg, M.W.: On the facial structure of set packing polyhedra. Mathematical Programming **5**, 199–215 (1973)

158. Padberg, M.W., Rinaldi, G.: A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problems. SIAM Review **33**, 60–100 (1991)

159. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading, MA (1994)

160. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications, Inc., Mineola, NY (1988)

161. Pardalos, P.M., Rodgers, G.P.: A branch and bound algorithm for the maximum clique problem. Computers and Operations Research **19**, 363–375 (1992)

162. Pardalos, P.M., Xue, J.: The maximum clique problem. Journal of Global Optimization **4**, 301–328 (1994)

163. Peng, X., Langston, M.A., Saxton, A.M., Baldwin, N.E., Snoddy, J.R.: Detecting network motifs in gene co-expression networks through integration of protein domain information. In: P. McConnell, S.M. Lin, P. Hurban (eds.) Methods of Microarray Data Analysis V, pp. 89–102. Springer, New York (2007)

164. Pulleyblank, W.R.: Faces of matching polyhedra. Ph.D. thesis, University of Waterloo, Canada (1973)

165. Rain, J.C., Selig, L., Reuse, H.D., Battaglia, V., Reverdy, C., Simon, S., Lenzen, G., Petel, F., Wojcik, J., Schachter, V., Chemama, Y., Labigne, A., Legrain, P.: The protein-protein interaction map of helicobacter pylori. Nature **409**(6817), 211–215 (2004). Erratum in: Nature **409**(6820):553 and **409**(6821):743, 2001.

166. Ravi, S.S., Rosenkrantz, D., Tayi, G.K.: Heuristics and special case algorithms for dispersion problems. Operations Research **42**, 299–310 (1994)

167. Sageman, M.: Understanding Terrorist Networks. University of Pennsylvania Press, Phladelphia, PA (2004)

168. Sanchis, L.A., Jagota, A.: Some experimental and theoretical results on test case generators for the maximum clique problem. INFORMS Journal on Computing **8**(2), 103–117 (1996)

169. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, New York (1986)

170. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, New York (2003)

171. Scott, J.: Social Network Analysis: A Handbook, 2 edn. Sage Publications, London (2000)

172. Seidman, S.B.: Network structure and minimum degree. Social Networks **5**, 269–287 (1983)

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

174. Selkow, S.M.: A probabilistic lower bound on the independence number of graphs. Discrete Mathematics **132**, 363–365 (1994)

175. Sherali, H.D., Smith, J.C.: A polyhedral study of the generalized vertex packing problem. Mathematical Programming **107**(3), 367 – 390 (2006)

176. Shor, N.Z.: Dual quadratic estimates in polynomial and Boolean programming. Annals of Operations Research **25**, 163–168 (1990)

177. Sloane, N.: Challenge problems: Independent sets in graphs (2000). Online: http://www.research.att.com/~njas/doc/graphs.html. Accessed July 2003

178. Sparrow, M.K.: The application of network analysis to criminal intelligence: An assessment of the prospects. Social Networks **13**, 251–274 (1991)

179. Spath, H.: Cluster Analysis Algorithms. Ellis Horwood, Chichester, West Sussex (1980)

180. Spirin, V., Mirny, L.A.: Protein complexes and functional modules in molecular networks. Proceedings of the National Academy of Sciences **100**(21), 12,123–12,128 (2003)

181. Terveen, L., Hill, W., Amento, B.: Constructing, organizing, and visualizing collections of topically related web resources. ACM Transactions on Computer-Human Interaction **6**, 67–94 (1999)

182. Trotter Jr., L.: Solution characteristics and algorithms for the vertex packing problem. Tech. Rep. 168, Dept. of Operations Research, Cornell University, Ithaca, NY (1973)

183. U.S. Congress, O.o.T.A.: Technologies for detecting money laundering. In: Information Technologies for the Control of Money Laundering, pp. 51–74. Washington DC: U.S. Government Printing Office (1995). Online: http://www.wws.princeton.edu/ota/disk1/1995/9529/9529.PDF. Accessed May 2006

184. Washio, T., Motoda, H.: State of the art of graph-based data mining. SIGKDD Explor. Newsl. **5**(1), 59–68 (2003)

185. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge University Press, New York (1994)

186. Watts, D., Strogatz, S.: Collective dynamics of "small-world" networks. Nature **393**, 440–442 (1998)

187. Wei, V.K.: A lower bound on the stability number of a simple graph. Tech. Rep. TM 81-11217-9, Bell Laboratories, Murray Hill, NJ (1981)

188. West, D.: Introduction to Graph Theory. Prentice-Hall, Upper Saddle River, NJ (2001)

189. Wood, D.R.: An algorithm for finding a maximum clique in a graph. Operations Research Letters **21**(5), 211–217 (1997)

APPENDIX A

RESULTS OF BC ALGORITHMS ON SANCHIS INSTANCES AND IPBC ALGORITHM ON GROUP II INSTANCES

For Sanchis-log and Sanchis-linear instances, running time, best integer solution found, best upper-bound found, number of BC nodes enumerated and number of cuts generated are provided for BC implementation ($k = 1$), BC-MIS implementation ($k = 2$) and BC-C2PLX implementation ($k = 2$). In all the tables reporting numerical results on Sanchis instances, the superscript † indicates non-optimal termination when time limit set by parameter *TiLim* was reached; $[l, u]$ represent lower and upper bounds on $\omega_k(G)$ in case of non-optimal termination; "-" indicates instances not attempted; $*$ indicates that the maximum k-plex was found even though termination was not optimal. The clique numbers of Sanchis-log instances is provided in Table 24 for the sake of clarity. Running time reported is the duration CPLEX® was engaged in solution process which includes formulation, presolving, BC and primal heuristics and it does not include time to read data or write output.

For Group II instances, the following details are provided. Tables 49, 50 and 51 list the authors belonging to a maximum k-plex identified for $k = 1, 2, 3$ in Erdös Collaboration Networks. Tables 52 and 53 list the author names (verbatim) included in a maximum k-plex identified for $k = 1, 2, 3$ on computational geometers collaboration networks for threshold $t = 0, 1, 2$. Words included in the maximum k-plexes identified from the text mining network DAYS-t.PAJ for $k = 1, 2, 3$ and $t = 3, 4, 5$ are listed in Tables 54 and 55.

Table 24 Clique numbers of Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	11	14	19	26	42	88
200	12	16	21	30	48	101
300	13	17	23	32	52	109
400	14	18	24	34	54	114
500	14	18	25	35	56	118
600	14	19	26	36	58	122
700	15	19	26	37	59	125
800	15	20	27	38	60	127
900	15	20	27	39	61	130
1000	16	20	28	39	62	132

Table 25 Running time (secs) of BC, $k = 1$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	0.703	0.797	0.578	0.75	0.125	0.015
200	7.891	6.86	15.469	25.455	22.033	0.063
300	97.6	127.007	317.924	877.426	1027.51	45.456
400	410.055	780.826	1591.47	6041.69	$10799.6^{\dagger}$	$10800.2^{\dagger}$
500	1593.31	4458.98	$10801^{\dagger}$	$10800.5^{\dagger}$	-	-
600	3378.54	$10800.4^{\dagger}$	-	-	-	-
700	5613.22	$10800.4^{\dagger}$	-	-	-	-
800	$10800.5^{\dagger}$	-	-	-	-	-

Table 26 1-plex numbers found by BC on Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	11	14	19	26	42	88
200	12	16	21	30	48	101
300	13	17	23	32	52	109
400	14	18	24	34	$[51, 71]$	$[114, 140]^{*}$
500	14	18	$[19, 29]$	$[24, 43]$	-	-
600	14	$[14, 20]$	-	-	-	-
700	15	$[14, 25]$	-	-	-	-
800	$[11, 19]$	-	-	-	-	-

Table 27 Number of nodes enumerated by BC, $k = 1$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	77	239	151	285	0	0
200	759	672	2713	4557	5131	0
300	6734	15119	46134	128503	119899	6847
400	19889	67740	149180	500128	732940†	1121575†
500	107722	308110	637257†	605248†	-	-
600	155407	546396†	-	-	-	-
700	106706	256836†	-	-	-	-
800	129248†	-	-	-	-	-

Table 28 Number of cuts generated by BC, $k = 1$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	122	345	279	370	0	0
200	1230	611	3127	9585	13552	0
300	7255	18996	66994	175884	265001	5584
400	17446	58532	165833	812310	1227376†	1928377†
500	93265	195996	403970†	335075†	-	-
600	92287	246706†	-	-	-	-
700	44977	83787†	-	-	-	-
800	43786†	-	-	-	-	-

Table 29 Running time (secs) of BC, $k = 1$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	0.391	0.485	0.609	1.36	15.658	4.36
200	7.376	5.625	5.001	14.205	64.458	10800.6†
300	43.611	85.977	28.774	82.007	448.903	10800.5†
400	300.747	129.476	236.213	337.202	1954.35	10800.6†
500	435.315	325.688	252.136	652.093	5792.38	-
600	812.927	650.591	578.729	1998.4	10800.4†	-
700	1781.46	1216.49	1153.37	8806.92	10800.4†	-
800	2854.67	1931.55	2183.15	10431.8	10800.3†	-
900	4501.52	3541.75	10800.4†	10800.4†	-	-
1000	7620.85	5534.83	6695.89	10801†	-	-

Table 30 1-plex numbers found by BC on Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	20	20	20	20	20	20
200	40	40	40	40	40	$[38, 55]$
300	60	60	60	60	60	$[56, 101]$
400	80	80	80	80	80	$[77, 146]$
500	100	100	100	100	100	-
600	120	120	120	120	$[120, 200]^*$	-
700	140	140	140	140	$[140, 261]^*$	-
800	160	160	160	160	$[160, 306]^*$	-
900	180	180	$[90, 383]$	$[180, 314]^*$	-	-
1000	200	200	200	$[200, 356]^*$	-	-

Table 31 Number of nodes enumerated by BC, $k = 1$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	24	69	147	776	12583	3882
200	300	206	296	2291	16092	$1274754^\dagger$
300	572	5909	793	5699	47742	$1118366^\dagger$
400	10440	1065	8049	11694	94501	$1000516^\dagger$
500	858	1731	1790	8352	141947	-
600	878	1669	2576	16948	$163915^\dagger$	-
700	1308	1841	3383	82505	$103549^\dagger$	-
800	1320	1595	4877	35896	$87403^\dagger$	-
900	1643	2805	$51665^\dagger$	$33976^\dagger$	-	-
1000	1774	2576	8200	$24270^\dagger$	-	-

Table 32 Number of cuts generated by BC, $k = 1$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	22	104	338	1264	15063	1264
200	209	407	709	4304	39726	1439053†
300	737	6539	2380	10427	147825	1227792†
400	12414	3221	11478	23862	303508	1187218 †
500	1673	4467	6752	33034	567800	-
600	1677	5074	11128	62069	657015†	-
700	2667	5561	15236	164728	378961†	-
800	2773	5459	17438	154844	250647†	-
900	3900	9571	59648†	102024†	-	-
1000	4207	8580	29295	74542†	-	-

Table 33 Running time (secs) of BC-MIS, $k = 2$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	11.922	29.126	29.235	19.36	1.672	0.015
200	684.529	2467.22	7920.65	10800.2†	10800.2†	1.091
300	9050.57	10800.4†	10800.3†	-	-	10800.7†
400	10800.4†	-	-	-	-	10800.2†
500	10800.3†	-	-	-	-	-

Table 34 2-plex numbers found by BC-MIS on Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	12	14	20	26	42	88
200	13	17	23	[31, 39]	[48, 55]	101
300	15	[19, 33]	[25, 45]	-	-	[109, 125]
400	[15, 28]	-	-	-	-	[115, 178]
500	[15, 28]	-	-	-	-	-

Table 35 Number of nodes enumerated by BC-MIS, $k = 2$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	4687	17291	16482	11356	978	0
200	139080	562367	1432228	1389938†	1691054†	0
300	829225	583982†	612239†	-	-	915382†
400	348089†	-	-	-	-	736123†
500	112203†	-	-	-	-	-

Table 36 Number of cuts generated by BC-MIS, $k = 2$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	1740	4903	6462	6575	544	0
200	38297	181394	536488	859530†	1265270†	0
300	181916	139325†	199689†	-	-	631101†
400	348089†	-	-	-	-	736123†
500	112203†	-	-	-	-	-

Table 37 Running time (secs) of BC-MIS, $k = 2$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	2.016	6.312	21.907	325.18	10800.3†	291.586
200	16.751	92.909	179.192	2025.52	10800.5†	10800.5†
300	84.266	539.763	1115.4	10800.3†	-	-
400	298.601	2106.16	4392.91	10800.4†	-	-
500	855.319	5675.32	10800.5†	-	-	-
600	1761.98	10800.3†	10800.4†	-	-	-
700	3495.51	10801.2†	10801†	-	-	-
800	6491.4	-	-	-	-	-
900	10627	-	-	-	-	-
1000	10800.6†	-	-	-	-	-

Table 38 2-plex numbers found by BC-MIS on Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	20	20	20	20	[24, 27]	38
200	40	40	40	40	[40, 57]	[49, 80]
300	60	60	60	[60, 62]	-	-
400	80	80	80	[80, 140]	-	-
500	100	100	[100, 132]	-	-	-
600	120	[120, 180]	[120, 210]	-	-	-
700	140	[140, 231]	[140258]	-	-	-
800	160	-	-	-	-	-
900	180	-	-	-	-	-
1000	[200, 282]	-	-	-	-	-

Table 39 Number of nodes enumerated by BC-MIS, $k = 2$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	407	2139	12609	224245	$1223136^{\dagger}$	178159
200	678	4574	21258	404025	$1225284^{\dagger}$	$958143^{\dagger}$
300	1169	8478	41907	$729325^{\dagger}$	-	-
400	2875	13952	74848	$196993^{\dagger}$	-	-
500	3160	18131	$69593^{\dagger}$	-	-	-
600	5433	$13957^{\dagger}$	$28097^{\dagger}$	-	-	-
700	8288	$7483^{\dagger}$	$16323^{\dagger}$	-	-	-
800	7627	-	-	-	-	-
900	9383	-	-	-	-	-
1000	$6919^{\dagger}$	-	-	-	-	-

Table 40 Number of cuts generated by BC-MIS, $k = 2$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	414	1320	5089	53805	318711†	66803
200	877	5115	16249	260290	1037180†	666585†
300	1320	10853	32841	552828†	-	-
400	2040	18363	52769	226690†	-	-
500	2549	26500	67133†	-	-	-
600	3445	27701†	30599†	-	-	-
700	4512	15112†	18884†	-	-	-
800	4888	-	-	-	-	-
900	5745	-	-	-	-	-
1000	4445†	-	-	-	-	-

Table 41 Running time (secs) of BC-C2PLX, $k = 2$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	124.925	307.086	478.996	505.872	56.47	0.016
200	10800.4†	10801.2†	10802.4†	10800.8†	10800.8†	1.077
300	-	10813.5†	-	-	-	10814.5†

Table 42 2-plex numbers found by BC-C2PLX on Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	12	14	20	26	42	88
200	[13, 22]	[16, 62]	[22, 71]	[30, 80]	[48, 88]	101
300	-	[18, 101]	-	-	-	[109, 150]

Table 43 Number of nodes enumerated by BC-C2PLX, $k = 2$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	2490	7839	6536	4340	304	0
200	25612†	8110†	5480†	5213†	4853†	0
300	-	2243†	-	-	-	708†

Table 44 Number of cuts generated by BC-C2PLX, $k = 2$, Sanchis-log instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	1559	4842	5862	5448	362	0
200	13666	4432	3994	4365	4508	0
300	-	812	-	-	-	637

Table 45 Running time (secs) of BC-C2PLX, $k = 2$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	18.204	107.096	426.464	4109.14	$10800.3^{\dagger}$	$10800.6^{\dagger}$
200	667.423	5418.7	$10802.1^{\dagger}$	$10800.8^{\dagger}$	-	-
300	7453.67	$10821^{\dagger}$	-	-	-	-
400	$10852.4^{\dagger}$	$10849.5^{\dagger}$	-	-	-	-

Table 46 2-plex numbers found by BC-C2PLX on Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	20	20	20	20	$[24, 29]$	$[37, 39]$
200	40	40	$[40, 72]$	$[40, 80]$	-	-
300	60	$[60, 100]$	-	-	-	-
400	$[80, 115]$	$[80, 136]$	-	-	-	-

Table 47 Number of nodes enumerated by BC-C2PLX, $k = 2$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	273	876	5437	101207	$126558^{\dagger}$	$56697^{\dagger}$
200	724	2200	$2760^{\dagger}$	$4315^{\dagger}$	-	-
300	1483	$625^{\dagger}$	-	-	-	-
400	$433^{\dagger}$	$487^{\dagger}$	-	-	-	-

Table 48 Number of cuts generated by BC-C2PLX, $k = 2$, Sanchis-linear instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	456	1194	5225	81143	109238†	37941†
200	1091	4142	3724†	5096†	-	-
300	2227	1403†	-	-	-	-
400	643†	626†	-	-	-	-

Table 49 Authors belonging to a maximum k-plex identified for $k = 1, 2, 3$ in ERDOS-97-1.NET and ERDOS-97-2.NET

k	ERDOS-97-1.NET	ERDOS-97-2.NET
1	Stefan Andrus Burr Ralph J. Faudree Ronald J. Gould Andras Gyarfas Michael S. Jacobson Cecil Clyde Rousseau Richard H. Schelp	Stefan Andrus Burr Ralph J. Faudree Ronald J. Gould Andras Gyarfas Michael S. Jacobson Cecil Clyde Rousseau Richard H. Schelp
2	Noga Alon Fan Rong K. Chung Zoltan Furedi Andras Gyarfas Vojtech Rodl Endre Szemeredi William T. Trotter Zsolt Tuza	Gary Chartrand Frank Harary Michael S. Jacobson Ewa Marie Kubicka Grzegorz Kubicki Jeno Lehel Ortrud R. Oellermann Farrokh Saba
3	Stefan Andrus Burr Ralph J. Faudree Frank Harary Michael S. Jacobson Jeno Lehel Jaroslav Nesetril Vojtech Rodl Richard H. Schelp Zsolt Tuza	Gary Chartrand Ronald J. Gould Frank Harary Michael S. Jacobson Ewa Marie Kubicka Grzegorz Kubicki Jeno Lehel Ortrud R. Oellermann Farrokh Saba

Table 50 Authors belonging to a maximum k-plex identified for $k = 1, 2, 3$ in ERDOS-98-1.NET and ERDOS-98-2.NET

k	ERDOS-98-1.NET	ERDOS-98-2.NET
1	Stefan Andrus Burr Ralph J. Faudree Ronald J. Gould Andras Gyarfas Michael S. Jacobson Cecil Clyde Rousseau Richard H. Schelp	Guantao Chen Ralph J. Faudree Ronald J. Gould Andras Gyarfas Michael S. Jacobson Richard H. Schelp Linda M. Lesniak
2	Noga Alon Fan Rong K. Chung Zoltan Furedi Andras Gyarfas Vojtech Rodl Endre Szemeredi William T. Trotter Zsolt Tuza	Gary Chartrand Frank Harary Michael S. Jacobson Ewa Marie Kubicka Grzegorz Kubicki Jeno Lehel Ortrud R. Oellermann Farrokh Saba
3	Noga Alon Fan Rong K. Chung Peter Frankl Zoltan Furedi Ronald L. Graham Laszlo Lovasz Vojtech Rodl Joel H. Spencer William T. Trotter	Noga Alon Vaclav Chvatal Ronald L. Graham Frank Harary Jaroslav Nesetril Vojtech Rodl Endre Szemeredi William T. Trotter Zsolt Tuza

Table 51 Authors belonging to a maximum k-plex identified for $k = 1, 2, 3$ in ERDOS-99-1.NET and ERDOS-99-2.NET

k	ERDOS-99-1.NET	ERDOS-99-2.NET
1	Guantao Chen Ralph J. Faudree Ronald J. Gould Andras Gyarfas Michael S. Jacobson Cecil Clyde Rousseau Richard H. Schelp	Guantao Chen Ralph J. Faudree Ronald J. Gould Andras Gyarfas Michael S. Jacobson Richard H. Schelp Linda M. Lesniak
2	Noga Alon Fan Rong K. Chung Zoltan Furedi Andras Gyarfas Vojtech Rodl Endre Szemeredi William T. Trotter Zsolt Tuza	Gary Chartrand Frank Harary Michael S. Jacobson Ewa Marie Kubicka Grzegorz Kubicki Jeno Lehel Ortrud R. Oellermann Farrokh Saba
3	Noga Alon Fan Rong K. Chung Peter Frankl Zoltan Furedi Ronald L. Graham Laszlo Lovasz Vojtech Rodl Joel H. Spencer William T. Trotter	Stefan Andrus Burr Ralph J. Faudree Andras Gyarfas Frank Harary Michael S. Jacobson Jaroslav Nesetril Vojtech Rodl Richard H. Schelp Zsolt Tuza

Table 52 Authors belonging to a maximum k-plex identified for $k = 1, 2, 3$ in COMP-GEOM-0.PAJ (same 22 authors listed for all k)

A. Hicks	J. Weeks	Joel Hass
Leonidas J. Guibas	Marshall W. Bern	David Letscher
L. Paul Chew	Pankaj K. Agarwal	G. Lerman
Jack Scott Snoeyink	Herbert Edelsbrunner	Tamal K. Dey
David P. Dobkin	Paul Plassmann	Nina Amenta
Chee-Keng Yap	C. Grimm	C. K. Johnson
J. Harer	D. Zorin	E. Sedgwick
David Eppstein		

Table 53 Authors belonging to a maximum k-plex identified for $k = 1, 2, 3$ in COMP-GEOM-t.PAJ

	$k = 1$	$k = 2$	$k = 3$
$t = 1$	Mark H. Overmars Sue H. Whitesides Erik D. Demaine Therese C. Biedl Godfried T. Toussaint Anna Lubiw Martin L. Demaine Steve M. Robbins Ileana Streinu Joseph O'Rourke	Leonidas J. Guibas Micha Sharir Bernard Chazelle Jack Scott Snoeyink John E. Hershberger Richard Pollack Herbert Edelsbrunner Emo Welzl Janos Pach Raimund Seidel	Leonidas J. Guibas Micha Sharir Bernard Chazelle Jack Scott Snoeyink John E. Hershberger Chee-Keng Yap Richard Pollack Herbert Edelsbrunner Emo Welzl Janos Pach Raimund Seidel
	$k = 1$	$k = 2$	$k = 3$
$t = 2$	Mark H. Overmars Sue H. Whitesides Erik D. Demaine Therese C. Biedl Anna Lubiw Martin L. Demaine Steve M. Robbins Joseph O'Rourke	Leonidas J. Guibas Micha Sharir Bernard Chazelle Jack Scott Snoeyink John E. Hershberger Richard Pollack Herbert Edelsbrunner Raimund Seidel	Leonidas J. Guibas Micha Sharir Bernard Chazelle John E. Hershberger Richard Pollack Pankaj K. Agarwal Herbert Edelsbrunner Emo Welzl Janos Pach Raimund Seidel

Table 54 Words belonging to a maximum k-plex identified for $k = 1, 2, 3$ and $t = 3, 4$ in DAYS-t.PAJ

	$k = 1$	$k = 2$	$k = 3$
$t = 3$	attack official pentagon people pres_bush tuesday united_states washington	attack new_york official pentagon people plane pres_bush tuesday united_states washington	attack city new_york official pentagon people pres_bush thousand tuesday united_states washington
	$k = 1$	$k = 2$	$k = 3$
$t = 4$	landmark new_york plane tower tuesday twin world_trade_ctr	attack pentagon people plane tower tuesday united_states world_trade_ctr	attack city new_york pentagon people plane tuesday united_states washington

Table 55 Words belonging to a maximum k-plex identified for $k = 1, 2, 3$ and in DAYS-5.PAJ

	$k = 1$	$k = 2$	$k = 3$
$t = 5$	attack new_york pentagon plane tuesday world_trade_ctr	attack new_york pentagon plane tower tuesday world_trade_ctr	attack new_york pentagon plane tower tuesday twin world_trade_ctr

VITA

Balabhaskar "Baski" Balasundaram received his Bachelor of Technology degree in Mechanical Engineering from the Indian Institute of Technology – Madras, India in 2002. He joined the Industrial Engineering doctoral program at Texas A&M University in September 2002 and received his Doctor of Philosophy degree in August 2007. His research interests are in combinatorial optimization, with clustering and data-mining applications in social, biological and wireless networks. He will join the faculty of School of Industrial Engineering and Management at Oklahoma State University, Stillwater in August 2007.

Mr. Balasundaram may be reached at his work address:

School of Industrial Engineering and Management
322 Engineering North
Oklahoma State University
Stillwater, OK 74078

This document was typeset in LaTeX by the author.