Clique Relaxations in Social Network Analysis: The Maximum k-plex Problem

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

Sergiy Butenko
Department of Industrial and Systems Engineering,
Texas A&M University, College Station, Texas 77843, USA.
butenko@tamu.edu

Illya V. Hicks
Computational and Applied Mathematics Department,
Rice University, Houston, TX 77005, USA.
ivhicks@rice.edu

Abstract

This paper introduces and studies the *maximum k-plex problem*, which arises in social network analysis and has wider applicability in several important areas employing graph-based data mining. After establishing NP-completeness of the decision version of the problem on arbitrary graphs, an integer programming formulation is presented followed by a polyhedral study to identify combinatorial valid inequalities and facets. A branch-and-cut algorithm is implemented and tested on proposed benchmark instances. An algorithmic approach is developed exploiting the graph-theoretic properties of a k-plex, that is effective in solving the problem to optimality on very large, sparse graphs such as the *power law graphs* frequently encountered in the applications of interest.

Keywords. maximum k-plex problem; maximum clique problem; social network analysis; clique relaxations; cohesive subgroups; scale free graphs; power law graphs

1 Introduction

1.1 Graphs and Complex Systems

Network analysis has garnered significant attention from practitioners in diverse fields as an effective approach to study complex natural and engineered systems [Alderson, 2008, Cook and Holder, 2000, Washio and Motoda, 2003, Fischer and Meinl, 2004, Nagurney, 2003]. Novel network models of data arising from applications in internet analytics, systems biology, social network analysis, computational finance and telecommunication have led to many interesting insights. Cases in point are the recent construction of internet topology maps using trace-route probes [Broido and Claffy, 2001], construction of gene co-expression networks based on data from micro-array experiments [Peng et al., 2007], protein interaction networks based on data from two-hybrid assays [Ito et al., 2001], social interaction data from internet communities and data bases [Grossman et al., 1995, Chung and Lu, 2006], stock market networks in finance [Boginski et al., 2006], and call data in telecommunication [Abello et al., 1999]. Graph models have become indispensable tools for representing massive data sets arising from complex, multi-scale systems due to several unique advantages. Possibly the most important is the ability of graph models

to capture a snapshot of system-level properties represented by the data, starting with component-level pairwise interconnections.

To illustrate this idea, consider the *protein interaction networks* (PINs) that are built from pairwise interactions of proteins in an organism. Vertices of a PIN represent proteins in an organism, and an edge between two vertices indicates that the corresponding proteins are known to interact. Pairwise interaction information is collected from high-throughput biological experiments such as two-hybrid assays [Ito et al., 2001]. Identifying clusters in PINs helps identify *protein complexes* and *functional modules* that influence cellular functions [Spirin and Mirny, 2003]. Protein complexes are groups of proteins that interact at the same time and at the same place in the cell, acting as a macro-molecular machine to carry out specific cellular tasks. Functional modules represent groups of proteins that interact during different phases of a cellular process and in different parts of the cell to carry out specific cellular functions [Spirin and Mirny, 2003]. Thus, a PIN is able to capture multi-scale information from a complex system, and mining a PIN yields important insights into the cellular processes of the organism.

1.2 Graph-Theoretic Clique Relaxations and Social Network Analysis

A *clique* in a graph is a set of pairwise adjacent vertices, that is the graph induced by a clique is *complete* with all possible edges. Clique has long been considered the standard graph-theoretic cluster model, and early algorithms for identifying large cliques were motivated by sociological applications [Luce and Perry, 1949, Harary and Ross, 1957]. *Social network analysis* (SNA) aims to study sociological connections using graph-theoretic concepts. The notion of a *cohesive subgroup* which is a "tightly knit" subgroup in a social network (analogous to clusters in graph-based data mining) is often used to explain and develop sociological theories [Wasserman and Faust, 1994].

From sociological and data mining perspectives, the following properties are desirable in a cohesive subgroup: *(i) familiarity* among members (few strangers); *(ii) reachability* among members (quick communication); and *(iii) robustness* of the subgroup (it is not easily destroyed by removing members). These properties are conveniently modeled using the graph theoretic terms of vertex degrees, pairwise distances/diameter, and vertex connectivity, respectively. Clearly, a clique is ideal with respect to all of the corresponding requirements since it induces a subgraph in which *(i)* each vertex has the maximum possible degree; *(ii)* any pair of members has the minimum possible distance between them; and *(iii)* the vertex connectivity is maximum possible.

Cliques were, hence, the earliest graph models for cohesive subgroups in SNA [Scott, 2000]. However, requiring the existence of all possible edges between a group of vertices for cohesiveness by the "clique standard" was soon found to be overly restrictive and impractical [Alba, 1973, Freeman, 1992, Seidman and Foster, 1978], primarily because real-life cohesive subgroups (or clusters) need not meet the "ideal" notion of cliques and could be missing a few edges. Furthermore, in a wider application context, large real-life data sets are prone to errors that could lead to missing edges. The clique model, while being robust against edges included in error, becomes very sensitive to edges missed in error and is no longer practical.

The need for *clique relaxations* was apparent not only in SNA, but also in operations research and computer science communities. Approaches developed to address this issue employed the notion of *high density subgraphs*, which requires the group of vertices to have at least a threshold number of edges [Corneil and Perl, 1984, Ravi et al., 1994, Abello et al., 1999, 2002]. Although this is a natural relaxation of the clique requirement to have all possible edges, unlike the clique model, dense subgraph models are unable to provide guarantees about the structural properties of the resulting cluster. Structural relaxations of cliques such as *k-cliques* [Luce, 1950], *k-clubs* [Alba, 1973, Mokken, 1979] and *k-plexes* [Seidman and Foster, 1978] were introduced in SNA by relaxing the clique requirements for pairwise distances, diameter and degrees that are desirable from a social perspective. These models are suitable, practical alternatives to the clique model, as the structural properties they guarantee are meaningful and often necessary in most graph-based data mining applications. Furthermore, the cohesive subgroup models from SNA are parameterized (with k), they relax different structural aspects of a clique (when $k > 1$), and include clique as a special case (when $k = 1$). Hence, they provide a systematic sequence of relaxations of a clique (for each positive integer k) with an appropriate structural characterization of the resulting cluster that aids the critical process of *interpreting a cluster*. The focus of this paper is on the degree based model called *k-plex*, which was originally introduced by Seidman and Foster [1978]. We describe its key advantages in the next section after stating a formal definition.

2 Background

Let $G = (V, E)$ be a simple undirected graph representing a social network, $d_G(u, v)$ denote the length of a shortest path between vertices u and v in G, and $diam(G) = \max_{u,v \in V} d_G(u, v)$ be the diameter of G. Denote by $G[S] = (S, E \cap (S \times S))$ the subgraph induced by $S \subseteq V$. Let $N(u)$ and $deg_G(u)$ denote the set of neighbors $\{v : (u, v) \in E\}$ of a vertex $u \in V$ and the number of neighbors of u in G, respectively. Let $N[u]$ denote the closed neighborhood of a vertex u, that is $N[u] = \{u\} \cup N(u)$.

Seidman [1983] introduced the concept of a *k-core*, which is designed to capture the cohesive subgroups as well as regions surrounding them, and is defined as a subgraph with minimum degree at least k. 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 v cannot be in a k-core. Hence, delete v from G and continue recursively until $\delta(G) \geq k$ (in which case the vertex set of G is the maximum k-core) or G is empty (no k-core).

Definition 1 *$S \subseteq V$ is a k-plex in $G = (V, E)$ if $deg_{G[S]}(v) = |N(v) \cap S| \geq |S| - k \; \forall \; v \in S$.*

That is, a subset of vertices S is said to be a k-plex if the degree of every vertex in the induced subgraph $G[S]$ is at least $|S| - k$. Thus, a k-plex corresponds to a clique for $k = 1$ and relaxes the clique requirement for $k > 1$. Note that in contrast to a k-core, the minimum degree requirement *varies with* S. A k-plex is said to be *maximal* if it is not strictly contained in any other k-plex. We call the cardinality of a largest k-plex in the graph as *the k-plex number* and denote it by $\omega_k(G)$. The *maximum k-plex problem* is to find a largest k-plex of the given graph.

Seidman and Foster (1978) proposed an equivalent characterization of k-plexes. Namely, they have shown that G is a k-plex if and only if for any k-element subset of vertices $\{v_1, \ldots, v_k\} \subseteq V$, $V = \bigcup_{i=1}^{k} N[v_i]$. In other words, G is a k-plex if and only if *any* k vertices form a *dominating set* in G [Harary, 1988]. Some basic graph theoretic properties of a k-plex are stated next. Note that the vertex connectivity $\kappa(G)$ is defined as the minimum number of vertices whose removal results in a disconnected or trivial graph [Harary, 1988].

Theorem 1 (Seidman and Foster, 1978) *Let graph G be a k-plex on n vertices. Then, (1) Any vertex-induced subgraph of G is a k-plex; (2) If $k < \frac{(n+2)}{2}$, then $diam(G) \leq 2$; (3) $\kappa(G) \geq n - 2k + 2$.*

Members of a k-plex S can have at most $k - 1$ non-neighbors inside S. Hence, k-plexes with low k values ($k = 2, 3$) provide good relaxations of a clique that closely resemble the cohesive subgroups that can be found in real-life social networks. The k-plex also retains other desirable properties of a clique such as low diameter (reachability) and high connectivity (robustness) for low values of k.

The maximum clique problem is closely related to the well known *maximum independent set problem*. An independent set (or stable set) is a subset of pairwise nonadjacent vertices. A subset of vertices forms a clique in $G = (V, E)$ if and only if it forms an independent set in the complement graph $\bar{G} = (V, \bar{E})$. We relate k-plex to a similar complementary structure in the following manner.

Definition 2 *$S \subseteq V$ is a* co-k-plex *in $G = (V, E)$ if $deg_{G[S]}(v) = |N(v) \cap S| \leq k - 1 \; \forall \; v \in S$.*

In other words, the induced subgraph $G[S]$ has a maximum degree of $k - 1$ or less. It should be noted that S is a co-k-plex in G if and only if S is a k-plex in the complement graph $\bar{G}$. In particular, 1-plex is a clique and a co-1-plex is an independent set. Thus, k-plexes and co-k-plexes provide parameterized relaxations of two classical combinatorial optimization problems. The k-plex model is extremely practical as it provides a realistic alternative to the popular, but idealistic clique model in graph-based data mining applications. By varying k, one can balance the sensitivity of the model to edges missed in error and the reliability of the detected cluster under edges included in error. Detailed discussions of the clique relaxations from SNA and their applications in diverse fields, such as criminal/terrorist network analysis, systems biology, telecommunication, finance, and organizational management among others can be found in [Balasundaram, 2007].

Our contributions. We first demonstrate the NP-completeness result for the decision version of the maximum k-plex problem as this is the first formal study of this problem. A polyhedral study is then carried out in Sec. 3 focusing on extending two well known classes of valid inequalities for the maximum clique problem into three distinct families of valid inequalities for the maximum k-plex problem. Facet results are obtained for specialized support graphs, and whenever possible, for arbitrary graphs. Two of the families of inequalities are subject to extensive computational testing in a branch-and-cut framework (Sec. 4), following which we develop an iterative scheme with the branch-and-cut as its core subroutine,

that exploits graph theoretic properties of a k-plex through decomposition and preprocessing (Sec. 5). This algorithm is designed to solve the maximum k-plex problem on very large, sparse graphs. Our interest in such graphs stems from the ubiquitous presence of power law degree distribution [Chung and Lu, 2006] in natural and man-made networks including social and biological networks, well documented in the past decade [Barabási and Albert, 1999, Barabási et al., 2000b, Almaas and Barabási, 2006, Albert et al., 2000, Barabási et al., 2000a]. This approach is used to solve the maximum k-plex problem to optimality on real-life networks that exhibit a power law degree distribution with 400–13000 vertices and 0.0025%–1.18% edge density for $k = 1, \ldots, 5$. These instances cannot be solved directly using branch-and-cut due to extremely dense integer programs that result from extremely low edge density.

2.1 Computational Complexity

Consider the decision version of the the maximum k-plex problem, k-PLEX: Given a simple undirected graph $G = (V, E)$ and positive integers c, k, does there exist a k-plex of size c in G?

Theorem 2 *k-PLEX is NP-complete for any fixed positive integer k.*

PROOF. See Appendix A.

This result shows that the maximum k-plex problem is hard not only as a generalization of the maximum clique problem, but for any fixed k it is a hard problem in its own respect.

3 The k-plex Polytope

Given a graph $G = (V, E)$ with $|V| = n$, let $\bar{d}_i = |V \setminus N[i]|$ denote the degree of vertex i in the complement graph $\bar{G} = (V, \bar{E})$. We will further assume that $k > 1$ since $k = 1$ yields the well known maximum clique problem. The k-plex polytope $P_k(G)$ is given by

$$P_k(G) = conv(\{x \in \{0,1\}^n | \sum_{j \in V \setminus N[i]} x_j \leq (k-1)x_i + \bar{d}_i(1 - x_i) \; \forall \; i \in V\}).$$

Then, $\omega_k(G) = \max\{\sum_{i \in V} x_i | x \in P_k(G)\}$. The following theorem establishes the basic properties.

Theorem 3 *Let $P_k(G)$ denote the k-plex polytope of a given graph $G = (V, E)$, where $k > 1$. Then, (1) $dim(P_k(G)) = n$; (2) $x_i \geq 0$, and $x_i \leq 1$ induce facets of $P_k(G)$ for every $i \in V$.*

PROOF. See Appendix A.

3.1 Valid Inequalities

We introduce three types of valid inequalities for the k-plex polytope: *independent set inequalities, co-k-plex inequalities, and hole inequalities*. The first two generalize the well known stable set inequalities for the clique polytope, and the third generalizes the hole inequality for the clique polytope [Padberg, 1973]. We also show that the maximal independent set inequalities and hole inequalities (under some conditions) induce facets of the k-plex polytope for the support graph on which they are based. These could be lifted to yield facets for the graph containing the support graph as an induced subgraph. Furthermore, we show that maximal co-k-plex inequalities induce facets of the k-plex polytope when $k = 2$ and discuss the case when $k \geq 3$.

3.1.1 Independent Set Inequalities.

Note that $k + 1$ or more independent vertices do not form a k-plex and cannot be contained in one. Let I denote a maximal independent set (MIS) of size $k + 1$ or more in G. The inequality $\sum_{i \in I} x_i \leq k$ is valid for $P_k(G)$.

Theorem 4 *Let $G = (V, \emptyset)$ with $|V| \geq k + 1$. The inequality $\sum_{i \in V} x_i \leq k$ induces a facet of $P_k(G)$.*

PROOF. Let $F = \{x \in P_k(G) : \sum_{i \in V} x_i = k\}$ denote the face induced. Suppose there exists a valid inequality $ax \leq b$ such that $F \subseteq \{x : ax = b\}$. Let $S \subseteq V$ such that $|S| = k$, and let x_S denote the incidence vector of S. Since $x_S \in F$ we have $ax_S = b$. Consider $v \in S$ and $w \in V \setminus S$. Let $T = S \cup \{w\} \setminus \{v\}$, with x_T defined as before we have $ax_T = b$ implying $a_v = a_w$. Since S, v and w were arbitrary, $a_i = \lambda, i \in V$ and $b = k\lambda$ for some scalar λ. Since F is a maximal face, it is a facet. □

Lifting MIS Inequalities. Let I be an MIS in G, $\sum_{i\in I} x_i \leq k$ induces a facet of $P_k(G[I])$. Let $j \in V \setminus I$ (then $|N(j)\cap I| \geq 1$) and let $a_j = k - \max\{\sum_{i\in I} x_i : x \in P_k(G[I\cup\{j\}]), x_j = 1\}$. Then $a_j x_j + \sum_{i\in I} x_i \leq k$ induces a facet of $P_k(G[I \cup \{j\}])$. Further assume $|N(j) \cap I| \geq k$, then we can pick k neighbors of j along with j to form a k-plex. Hence, the maximum in that expression is k and $a_j = 0$. By sequentially lifting every j outside I such that $|N(j) \cap I| \geq k$, we see that $\sum_{i\in I} x_i \leq k$ induces a facet of $P_k(G[I \cup \tilde{I}])$, where $\tilde{I} = \{j \in V \setminus I : |N(j) \cap I| \geq k\}$. Consider $v \in V \setminus (I \cup \tilde{I})$ with $|N(v) \cap I| \leq k-1$ and let $a_v = k - \max\{\sum_{i\in I} x_i : x \in P_k(G[I \cup \tilde{I} \cup \{v\}]), x_v = 1\}$. To form a k-plex, we cannot pick k vertices from I in addition to v as at least one of the k vertices (one not adjacent to v) has zero degree. However, we can pick $k-1$ vertices from I and hence, $a_v = 1$. Now consider $u \in V \setminus (I \cup \tilde{I} \cup \{v\})$ with $|N(u)\cap I| \leq k-1$ and let $a_u = k - \max\{x_v + \sum_{i\in I} x_i : x \in P_k(G[I \cup \tilde{I} \cup \{v, u\}]), x_u = 1\}$. Firstly, u along with k vertices from I does not form a k-plex (since, u has at most $k-1$ neighbors in I). Clearly, we can pick $k-1$ vertices from I along with u to form a k-plex. If $|(N(u)\cap I)\cup(N(v)\cap I)| \geq k-1$, then we can pick $k-1$ vertices from this union of neighbors in I, each of which is adjacent to at least one of u and v. Hence, these $k-1$ vertices along with u and v form a k-plex. Otherwise, in every $(k-1)$-tuple from I, there exists at least one vertex that is not adjacent to both u and v. Hence, we have, $a_u = 1$ if $|(N(u)\cap I)\cup(N(v)\cap I)| < k-1$, and 0 otherwise. Clearly, for lifting further combinatorial enumeration becomes tedious and will not be pursued. But note that for a subset $I \subseteq V$, maximally lifting any inequality of the form $\sum_{i\in I} x_i \leq k$ that is valid for $P_k(G[I])$ will result in valid inequalities with 0,1 coefficients for $P_k(G)$.

3.1.2 Co-k-plex Inequalities.

Lemma 1 *The maximum size of a k-plex in a co-k-plex is at most $r_k = 2k - 2 + (k \mod 2)$.*

PROOF. Let k be even and 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 $|N(i) \cap S| \geq 2k-1-k = k-1 \ \forall i \in S$. Since G is co-k-plex, we have $|N(i) \cap S| \leq |N(i)| \leq k-1 \ \forall i \in S$. The two conditions then imply that the induced graph $G[S]$ is $(k-1)$-regular of order $2k-1$. But $k-1$ is odd and we cannot have an odd number of vertices of odd degree. This contradiction establishes that S does not exist. Now let k be odd and 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 $|N(i) \cap S| \geq 2k - k = k \ \forall i \in S$. Since G is co-k-plex, we have $|N(i) \cap S| \leq |N(i)| \leq k-1 \ \forall i \in S$. This contradiction establishes that S does not exist. □

This bound is sharp since the union of complete graphs $G_k = K_k \cup K_{k-1}$ for each even k forms a co-k-plex of size $2k-1$ that contains $K_{k-1} \cup K_{k-1}$, a k-plex of size $2k-2$. For odd k, 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 $V = V' \cup \{2k\}$, $V' = \{1, \ldots, 2k-1\}$, and $E = \{(i,j) : i \in V'$ and $j = i+1, \ldots, \left(i + \frac{k-1}{2}\right) \mod (2k-1)\}$. Maximum degree in G_k is $k-1$ and 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. It is also known as an *antiweb* and its complement is known as a *web*. Webs were introduced by Trotter [1975] to generalize odd hole and antihole inequalities developed by Padberg [1973] for the stable set polytope.

Lemma 1 implies that if J is a maximal co-k-plex of size more than r_k in G, the inequality $\sum_{i\in J} x_i \leq r_k$ is valid for $P_k(G)$. If J is a *maximal* co-k-plex, 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 upon inclusion v would have degree k or more in the induced subgraph. The next theorem uses this observation to show that for $k = 2$ the co-2-plex inequalities form facets of the 2-plex polytope.

Theorem 5 *For a subset $J \subseteq V$ such that $|J| \geq 3$, the inequality $\sum_{i\in J} x_i \leq 2$, induces a facet of $P_2(G)$ if and only if J is a maximal co-2-plex.*

PROOF. See Appendix A. □

Some important remarks on the dominance relationship between MIS and co-k-plex inequalities for $k \geq 3$ can be found in Appendix B.

3.1.3 Hole Inequalities.

Let $H \subseteq V$ be a hole (induced chordless cycle). If $|H| \leq k+2$, then H is a k-plex. 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 a hole and this bound is sharp. The inequality $\sum_{i\in H} x_i \leq k+1$ is valid for $P_k(G)$.

Theorem 6 *Let $G=(V,E)$ be a cycle on n vertices with $n \geq k+3$ such that n and $k+1$ are relatively prime. Then, the inequality $\sum_{i\in V} x_i \leq k+1$ induces a facet of $P_k(G)$.*

PROOF. Consider the $n \times n$ circulant matrix $A(n,k+1)=\{a_{ij}\}$, where $a_{ij}=1$ for $i=j,j+1,\ldots,j+k(\mod n)$ and $j=1,\ldots,n$; $a_{ij}=0$ otherwise. The columns of $A(n,k+1)$ correspond to n incidence vectors of k-plexes (paths) of size $k+1$ that satisfy the above valid inequality as equality. By the result of Trotter [1975], $A(n,k+1)$ is invertible since n and $k+1$ are relatively prime. Hence, the n incidence vectors are linearly independent, and the above valid inequality is facet inducing. □

The hole inequalities are not facet inducing in general, when n and $k+1$ are not co-primes. Consider for instance a 6 vertex cycle, $V=\{1,2,\ldots,6\}$ and $E=\{(i,i+1 \mod 6) : i \in V\}$ with $k=2$. Inequality $\sum_{i\in V} x_i \leq 3$ is valid, but not facet inducing. Sets $\{1,2,4,5\},\{2,3,5,6\},\{3,4,6,1\}$ form maximal co-2-plexes in this graph and we can obtain the hole inequality as a positive linear combination of the three facet inducing co-2-plex inequalities. In fact, this example can be generalized for any n, multiple of $k+1$, when $k=2$. However, there are cases where the hole inequalities with n and $k+1$ not coprime induce facets for the support graph.

Theorem 7 *Let $G=(V,E)$ be a cycle on $n=t(k+1)$ vertices with $t \geq 2, k \geq 3$ such that $k+1$ is odd. Then the inequality $\sum_{i\in V} x_i \leq k+1$ induces a facet of $P_k(G)$.*

PROOF. See Appendix A. □

Theorem 8 *Let $G=(V,E)$ be a cycle on $n=t(k+1)$ vertices with $t \geq 2, k \geq 5$ such that $k+1$ is even. Then the inequality $\sum_{i\in V} x_i \leq k+1$ induces a facet of $P_k(G)$.*

PROOF. See Appendix A. □

The hole inequalities are clearly an interesting family of inequalities, given the above results. Anti-webs [Trotter, 1975] can lead to a further class of facet defining inequalities, as they generalize holes, and present an interesting topic for future research.

4 Branch & Cut

In this section, we describe our branch-and-cut (BC) implementation and study the performance of the MIS and co-k-plex inequalities for the maximum k-plex problem when $k=1,2$. Since the separation heuristics for holes are very different from the greedy heuristics we use for MIS and co-k-plex inequalities, we do not attempt them in this paper. Recall that for the cases we consider, $k=1,2$, facet defining co-k-plex inequalities are equivalent to MIS inequalities when $k=1$ and dominate them when $k=2$. However, when $k=2$, our computational experiments illustrate the benefits of using strong MIS cuts generated quickly as opposed to stronger co-2-plex cuts generated by expensive separation heuristics. Also, when $k=2$, we do not consider lifting MIS inequalities as they will only lead to co-2-plex inequalities. The aim of this part of the paper is to judge the effectiveness of the cuts in solving the problem of interest, the order and size of instances that can be solved under this framework, and to provide some benchmark instances.

4.1 General Implementation Details

All numerical experiments were conducted on *Dell Precision PWS690®* computers with 2.66GHz XEON® processor, 3GB RAM and 120GB HDD . The core of all our algorithms is a BC implemented using ILOG CPLEX 10.0® [ILOG]. The 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. *Local cuts* that are valid at the node in which they are generated and for the sub-tree rooted at that node are implemented using the CPLEX *goals* feature. Cuts were generated every SKIPFACTOR number of nodes in the BC tree. We ensure that the *round of cuts* added are distinct, violated by the LP optimum, and at most MAXLOCALCUTSPERNODE many *most violated* cuts are added to the system. CPLEX re-solves the problem at that node and handles the cut management from that point onwards.

Greedy Separation Heuristics. For generating local cuts, a graph G is obtained from the original graph after deleting vertices that are fixed to zero in the BC node where cuts are generated. To generate a round of cuts, we initialize I with every vertex from this graph and call one of the following greedy algorithms. For finding an MIS in G, we find a vertex v of minimum degree in $G[V \setminus N[I]]$. Vertex v is added to I, and we repeat the previous step until $V \setminus N[I]$ is empty. Note that $N[I]$ is the union of vertices in I and their neighbors. For finding a co-2-plex in G, we start with an MIS I found using the greedy algorithm described above. I is deleted from G, and a vertex v of minimum degree in the residual G is found. If $I \cup \{v\}$ is a co-2-plex, then v is added to I. The vertex v is deleted from G and the process is repeated until no more vertices are left.

Non-dominated Variable Fixing. In the BC tree, when k variables are fixed to one for the first time at a BC node, every vertex that is not adjacent to any of the k vertices can be fixed to zero for the subtree rooted at that BC node. This is valid since these vertices cannot belong to a k-plex containing the k fixed vertices as they are not dominated by them (recall the alternate characterization of k-plexes).

Note that CPLEX generates its own classes of cuts to solve any given MIP. All CPLEX cuts were turned off for experiments evaluating our BC implementation. However, we compare our BC implementation against the CPLEX MIP solver under its default settings. Apart from regular termination of an MIP (optimal or infeasible), CPLEX can 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 (set at 3 hours). By setting CPLEX parameter *NodeFileInd* to 2, CPLEX can be forced to write the BC tree to hard disk without any compression. This enabled CPLEX to proceed without any memory shortage as the BC tree grows exponentially in size, without significant increase in runtime [ILOG].

The test-bed for these experiments consists of graphs of various order and size generated using *Sanchis generators* [Sanchis and Jagota, 1996], used to get a sense of the influence of order and density of graphs on our BC implementation's running times. The Sanchis generator available at [DIMACS, 1995] produces graphs with known maximum clique size with a specified number of vertices, edges and a *construction parameter*, r. In our experiments, the maximum clique size was fixed at $\lceil \frac{n}{5} \rceil$ (where $\lceil a \rceil$ is the smallest integer greater than or equal to a) and the construction parameter r, which has to be an integer from interval $[0, \frac{n}{\omega(G)} - 1]$, was set at $\lfloor 0.75(\frac{n}{\omega(G)} - 1) \rfloor$ (where $\lfloor a \rfloor$ is the largest integer less than or equal to a). 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 $\lfloor \frac{dn(n-1)}{2} \rfloor$. Benchmark clique instances from the Second DIMACS Challenge [DIMACS, 1995, Johnson and Trick, 1996] are also used. Description of these instances can be found in [Hasselberg et al., 1993, Bomze et al., 1999].

4.2 Computational Experience

The BC(MIS) implementation was used for $k = 1$ and $k = 2$, while BC(co2plex) was also used when $k = 2$. The non-dominated variable fixing technique was used in all our implementations. The parameter SKIPFACTOR was set at 64 for edge densities $d = 0.4, 0.5, 0.6$ and SKIPFACTOR was 0 for $d = 0.7, 0.8, 0.9$ (meaning separation heuristics were attempted at every node of the BC tree). In all experiments MAXLOCALCUTSPERNODE was set at $\lceil 0.6n \rceil$. We arrived at these values after preliminary experimentation on Sanchis instances. The largest order up to which optimal resolution was possible on Sanchis instances within the 3-hour time limit, using the specified algorithm, for each density is presented in Table 1. Note that "< 100" indicates that the smallest instance in our test bed with 100 vertices was not solved optimally. Tables 6 and 7 in Appendix C present the total running time (excluding read/write time) and number of BC nodes enumerated for solving maximum 1-plex problem on Sanchis graphs using BC(MIS) implementation. Running times and number of BC nodes enumerated by BC(MIS) for $k = 2$ is presented in Tables 8 and 9 in Appendix C. The size of the largest 2-plex found and an upper bound on the 2-plex numbers obtained from the BC(MIS) implementation is provided in Table 10 in Appendix C. Running times and number of BC nodes enumerated by BC(co2plex) for $k = 2$ is presented in Tables 11 and 12 in Appendix C. Results for the DIMACS benchmarks are provided in Table 13 in Appendix C.

Exponential growth in the number of BC nodes and running time was observed, which is not surprising given the intractability of the problem. We are able to perform better than CPLEX default MIP solver with a basic BC(MIS) implementation. Generally speaking, our ability to optimally solve larger Sanchis instances

Table 1: Summary of results on Sanchis instances

k	Algorithm	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
1	CPLEX default	800	900	900	900	700	200
1	BC(MIS)	1000	1000	900	1000	800	300
2	CPLEX default	1000	600	200	100	< 100	< 100
2	BC(MIS)	1000	900	600	200	< 100	100
2	BC(co2plex)	400	300	200	< 100	< 100	< 100

decreases with increase in edge density when $k = 2$. 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(co2plex) despite the fact that co-2-plex inequalities are theoretically stronger. This observation clearly demands further investigation.

Table 2: BC(MIS) and BC(co2plex) comparison with $k = 2$ on Sanchis graphs of order 100 using global cuts

	Algorithm	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
Running time (secs)	BC(MIS)	1.469	3.703	33.234	1419.25	*Tilim*	140.969
	BC(co2plex)	0.906	2.063	23.172	1283.92	*Tilim*	118.078
No. of BC nodes	BC(MIS)	675	1955	20754	498479	714206†	68253
	BC(co2plex)	335	1019	15842	482644	748666†	68261
2-plex numbers	BC(MIS)	20	20	20	20	[24,30]	38
	BC(co2plex)	20	20	20	20	[24,29]	38

In order to understand this observation, we solve maximum 2-plex problem on Sanchis graphs of order $n = 100$, using BC(MIS) and BC(co2plex) implementations. To ensure a meaningful comparison, we generate and add *global cuts* instead of local cuts. Global cuts (MIS or co-2-plex) are generated using the original input graph, by applying the aforementioned greedy heuristics starting from every vertex. CPLEX applies the cuts that are violated by the LP optimum at a BC node and resolves the LP relaxation. This guarantees that, for every MIS cut generated for the BC(MIS) implementation, there is a dominating co-2-plex cut that is generated for the BC(co2plex) implementation. Note that this cannot be guaranteed in an implementation using local cuts. It can be seen from the results in Table 2 that BC(co2plex) enumerates fewer BC nodes as predicted by theory, requires less running time, and obtains better bound (in the nonoptimal case) compared to BC(MIS). However, in all three respects a local cut based BC(MIS) is significantly better for the more difficult denser instances and comparable for the easier sparser instances. Local cuts are typically more effective since they exploit "local information" at a BC node that is not considered by global cuts generated *a priori*. We believe the reason for poorer performance of BC(co2plex) using local cuts is the fact that the greedy co-2-plex heuristic is computationally much more expensive compared to the greedy MIS heuristic. When a vertex is added to an independent set, we can delete all its neighbors and repeat until we find an MIS. On the other hand, the 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 the current set that cannot be added. 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 separation heuristic is relatively expensive. A local cut implementation that periodically invokes this scheme hence becomes relatively inefficient. This suggests that future BC implementations for this problem should employ global co-k-plex cuts and local MIS cuts. Data structures for co-k-plexes that speed up the steps of the heuristic could also make them competitive as local cuts.

5 Iterative Peel-Branch-and-Cut

Considering the motivation behind the maximum k-plex problem, in this section we develop an exact algorithm that is able to optimally solve the problem on numerous large-scale real-life social and biological networks on which the straightforward BC implementation fails. The collaboration network of authors in computational geometry is available from [Batagelj and Mrvar, 2006], where for every pair of authors, the number of joint works is available. The instances named GEOM-t have vertices representing authors from this area, and two authors are connected by an edge if they have (strictly) more than *threshold* ($t = 0, 1, 2$) joint works. In Erdös collaboration networks, vertices represent mathematicians and an edge indicates that the mathematicians represented by the endpoints have published together. The collaboration networks of this type 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 [Batagelj and Mrvar, 2006, Grossman et al., 1995] in our experiments: ERDOS$-x-y$, 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. Note that in the instances we used, the vertex corresponding to Erdös himself is excluded. Two biological networks, protein interaction networks of *H. Pylori* and *S. Cerevisiae*, were also used in testing. In these graphs, the vertices represent proteins and edges indicate that the pair of proteins forming the end points are known to interact. The text-mining network from [Batagelj and Mrvar, 2006] 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 [Corman et al., 2006, 2002]. The vertices of the network are selected words that appeared in the news. 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 for DAYS-t with edges of weight at least $t + 1$ included, for $t = 3, 4, 5$.

Apart from the fact that these graphs are constructed from real-life data, they are also extremely large and extremely sparse graphs that obey a power law degree distribution. Such graphs are called scale-free graphs studied extensively recently [Barabási and Albert, 1999, Barabási et al., 2000b, Almaas and Barabási, 2006, Albert et al., 2000, Barabási et al., 2000a]. However, since these are extremely large and extremely sparse graphs, the integer program has an extremely dense constraint matrix, even though it is of size $n \times n$. This resulted in memory crashes while CPLEX was building the integer program, well before any solution technique could be attempted. This challenge led to the development of a decomposition and preprocessing scheme that exploits graph theoretic properties of a k-plex and utilizes the developed BC implementation as its core subroutine.

Iterative Peel-Branch-and-Cut (IPBC) Algorithm. The basic idea here is to find a maximum k-plex containing a fixed vertex in each iteration. If we *assume* that the maximum k-plex S^* satisfies $|S^*| > 2k-2$, and the vertex $v \in S^*$ is fixed, then by Theorem 1, $S^* \subseteq N_2[v] = \{i \in V : d_G(v, i) \leq 2\}$. As we iterate over $v \in V$, we only need to consider vertices in $N_2[v]$ assuming that there is a large enough k-plex. Then for each $v \in V$, a *peeling* procedure is called on the graph induced by $N_2[v]$. The peeling procedure, similar to the one used in [Abello et al., 1999] for the maximum clique problem, removes vertices of low degree based on the size of a known k-plex S. Vertices that cannot belong to k-plex of size at least $|S| + 1$ are identified and removed recursively. In our implementation, S is initialized with a greedily found maximal clique, and updated during the iterations. Observe that given a k-plex S, any maximum k-plex in this graph is a part of its maximum $(|S| - k)$-core, so we can use the algorithm described in Section 2 for peeling. BC is used on the graph obtained 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, since from that point we are not interested in k-plexes containing v. 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 the maximum k-plex problem on the original graph with the additional constraint $\sum_{v \in V} x_v \leq 2k - 2$. Finally, the vertices are fixed in non-increasing order of their degrees in an attempt to find a large k-plex early in the algorithm, so that the subsequent preprocessing was effective. Algorithm 1 is the pseudo-code for the IPBC algorithm. The approach taken in the IPBC algorithm, such as simple checks on sizes in

Table 3: Number of vertices, edges, edge density, and k-plex numbers for $k = 1, 2, 3, 4, 5$.

Graph	$\|V\|$	$\|E\|$	d	$\omega_1(G)$	$\omega_2(G)$	$\omega_3(G)$	$\omega_4(G)$	$\omega_5(G)$
H. Pylori	1570	1399	0.001136	3	5	6	7	8
S. Cerevisiae	2112	2203	0.000988	6	6	7	7	8
ERDOS-97-1	472	1314	0.011821	7	8	9	11	12
ERDOS-98-1	485	1381	0.011766	7	8	9	11	12
ERDOS-99-1	492	1417	0.011732	7	8	9	11	12
ERDOS-97-2	5488	8972	0.000596	7	8	9	11	12
ERDOS-98-2	5822	9505	0.000561	7	8	9	11	12
ERDOS-99-2	6100	9939	0.000534	8	8	9	11	12
GEOM-0	7343	11898	0.000441	22	22	22	22	22
GEOM-1	7343	3939	0.000146	10	10	11	12	13
GEOM-2	7343	1976	0.000073	8	8	10	11	11
DAYS-3	13332	5616	0.000063	8	10	11	13	13
DAYS-4	13332	3251	0.000037	7	8	9	11	11
DAYS-5	13332	2179	0.000025	7	7	8	10	11

combination with peeling and the assumption of a large k-plex, are designed to enable us to handle large, sparse instances by decomposing the graph. Note that the diameter-2 assumption is critical to facilitate the iterative scheme, and to focus on smaller graphs induced by the 2-neighborhood in each iteration.

Algorithm 1 Iterative Peel-and-Branch-and-Cut Algorithm: **IPBC(G)**

initialize $V(G) = \{v_1, \ldots, v_n\}$ $deg_G(v_i) \geq deg_G(v_{i+1})$; $S \leftarrow$ greedy maximal clique; $G_{copy} \leftarrow G$;
for $i = 1$ to n **do**
  if $|N_2[v_i]| > |S|$ **then** $H \leftarrow$ PEEL($G[N_2[v_i]], S$); **else** $H \leftarrow G$;
  if $|V(H)| > |S|$ **then** $\tilde{S} \leftarrow$ BRANCH-AND-CUT($H, x_{v_i} = 1$);
  $G \leftarrow G - v_i$; **if** $|\tilde{S}| > |S|$ **then** $S \leftarrow \tilde{S}$;
end for
if $|S| > 2k - 2$ **then return** S; **else return** $S \leftarrow$ BRANCH-AND-CUT($G_{copy}, \sum_{v \in V} x_v \leq 2k - 2$);

We use the BC(MIS) implementation described before, which employs MIS cuts and non-dominated variable fixing, as the core for the IPBC implementation. We set SKIPFACTOR to 0 and MAXLOCAL-CUTSPERNODE is $\lceil 0.6n \rceil$. The 3-hour time limit in the BC(MIS) implementation was removed. Table 3 presents the k-plex numbers, and Table 4 presents detailed results for the text-mining instances to illustrate our observations. These instances took the longest to resolve optimally. Tables 14, 15, 16 and 17 in Appendix C present the complete set of results. The column titled "IPBC Time" presents the total running time (in seconds) of the IPBC algorithm, "BC Time" presents the cumulative time (in seconds) spent on all BC calls to CPLEX, and "#BC Calls" is the number of BC calls made to CPLEX for each instance. It is clear from the results that very few BC calls are made compared to the number of vertices in each instance and the BC runtimes are often relatively low. In the cases where BC running time is significant, it was almost entirely due to the first BC call that received a poor initial solution from the greedy maximal clique initialization procedure. This encourages the use of better heuristics to obtain an initial k-plex to ensure significant speed-up. The burden of runtime shifted from the BC procedure to the preprocessing as we expected. The decomposition approach appears to be effective in reducing the size of the instance solved in each BC call. The IPBC algorithm was successful in optimally resolving these large-scale real-life instances that obey a power law degree distribution.

6 Conclusion & Future Work

This paper introduces and studies the maximum k-plex problem, where a k-plex is a graph-theoretic relaxation of clique originally introduced in the context of social network analysis. We establish the in-

Table 4: Runtime in seconds for the Reuters terror news networks using IPBC algorithm.

k	Graph	IPBC Time	BC Time	#BC Calls
1	DAYS-3	3110.8	0.1	3
	DAYS-4	2940.8	0.2	1
	DAYS-5	2758.0	0.0	0
2	DAYS-3	3367.8	4.1	1
	DAYS-4	2635.7	0.3	1
	DAYS-5	2462.9	0.1	2
3	DAYS-3	3395.4	45.5	1
	DAYS-4	2625.1	4.7	1
	DAYS-5	2445.5	0.2	2
4	DAYS-3	3489.8	203.0	1
	DAYS-4	2642.3	51.4	1
	DAYS-5	2426.3	2.7	1
5	DAYS-3	15336.9	12329.1	1
	DAYS-4	6201.4	3316.8	1
	DAYS-5	2820.8	113.1	1

tractability of this problem for every fixed k. The problem is formulated as a binary integer program and polyhedral results are presented. Classes of valid inequalities and facets are developed for the problem and implemented in a branch-and-cut framework. The results of computational experiments indicate the effectiveness of the cuts and the framework used. Iterative peel-branch-and-cut algorithm is then developed that exploits graph theoretic properties of a k-plex in preprocessing and decomposition that permits optimal resolution of the maximum k-plex problem in large-scale real-life networks that obey a power law degree distribution.

Several research problems and directions have been identified through the course of the paper that need attention. In particular, new facets of the k-plex polytope need to be discovered and the branch-and-cut algorithm may be modified and tuned to be able to solve larger instances to optimality. Development of meta-heuristics capable of finding good solutions in massive networks in a reasonable amount of time would be of practical value. Combinatorial algorithms for finding a maximum k-plex extending the maximum clique algorithms, e.g., by Carraghan and Pardalos [1990] and Östergård [2002] would also be beneficial. While a maximum k-plex size can be viewed as a global measure characterizing the cohesiveness of a network, in practice one may be interested in finding all maximal cohesive subgroups. Designing algorithms for detecting all maximal k-plexes is another issue to address in the future. In addition, related k-plex and co-k-plex partitioning problems seeking to partition the vertices of a graph into a minimum number of k-plexes and co-k-plexes, respectively, could be of interest. These are natural generalizations of the well-known minimum clique partitioning and graph coloring problems.

Acknowledgment. We thank the area/associate editors and the anonymous referees for their comments and suggestions that greatly improved the content and presentation of this paper. Contribution of Sandeep Sachdeva to the proof of Theorem 2 is gratefully acknowledged. We thank Benjamin McClosky for pointing out the simpler proof of Theorem 4, and Hannes Moser for pointing out some typos in the computational results in an earlier version of this manuscript. The research of S. Butenko was partially supported by AFOSR (FA9550-09-1-0154) and NSF (OISE-0553513). The research of I. V. Hicks was partially supported by NSF grant DMI-0521209.

References

J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, *External memory algorithms and visualization*, volume 50 of *DIMACS Series*

on Discrete Mathematics and Theoretical Computer Science, pages 119–130. American Mathematical Society, 1999.

J. Abello, M.G.C. Resende, and S. Sudarsky. Massive quasi-clique detection. In S. Rajsbaum, editor, *LATIN 2002: Theoretical Informatics*, pages 598–612, London, 2002. Springer-Verlag.

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

R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406: 378–382, 2000.

D. L. Alderson. Catching the "network science" bug: Insight and opportunity for the operations researcher. *Operations Research*, 56(5):1047–1065, 2008.

E. Almaas and A.-L. Barabási. Power laws in biological networks. In E. Koonin, Y. I. Wolf, and G. P. Karev, editors, *Power Laws, Scale-Free Networks and Genome Biology*, pages 1–11. Springer Science + Business Media, New York, 2006.

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

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: The topology of the World Wide Web. *Physica A*, 281:69–77, 2000a.

A.-L. Barabási, R. Albert, H. Jeong, and G. Bianconi. Power-law distribution of the World Wide Web. *Science*, 287:2115a, 2000b.

V. Batagelj and A. Mrvar. Pajek datasets: Reuters terror news network, 2006. Online: http://vlado.fmf.uni−lj.si/pub/networks/data/CRA/terror.htm. Accessed March 2008.

V. Boginski, S. Butenko, and P. Pardalos. Mining market data: a network approach. *Computers & Operations Research*, 33:3171–3184, 2006.

I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 1–74, Dordrecht, The Netherlands, 1999. Kluwer Academic Publishers.

A. Broido and K. C. Claffy. Internet topology: connectivity of ip graphs. In S. Fahmy and K. Park, editors, *Scalability and Traffic Control in IP Networks*, pages 172–187, Bellingham, WA, 2001. SPIE Publications.

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

F. Chung and L. Lu. *Complex Graphs and Networks*. CBMS Lecture Series. American Mathematical Society, Providence, RI, 2006.

D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.

S. Corman, T. Kuhn, R. McPhee, and K. Dooley. Studying complex discursive systems: Centering resonance analysis of organizational communication. *Human Communication Research*, 28(2):157–206, 2002.

S. Corman, K. Dooley, and R. McPhee. LOCKS: Analysis of media coverage of the terrorist attacks, 2006. Online: http://locks.asu.edu/terror/. Accessed June 2006.

D. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9:27–39, 1984.

G. Cornuéjols. *Combinatorial Optimization: Packing and Covering*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, 2001.

DIMACS. Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1995. Online: http://dimacs.rutgers.edu/Challenges/. Accessed March 2007.

I. Fischer and T. Meinl. Graph based molecular data mining - an overview. In W. Thissen, P. Wieringa, M. Pantic, and M. Ludema, editors, *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, pages 4578–4582, Piscataway, NJ, 2004. IEEE.

L. C. Freeman. The sociological concept of "group": An empirical test of two models. *American Journal of Sociology*, 98:152–166, 1992.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

J. Grossman, P. Ion, and R. De Castro. The Erdös Number Project, 1995. Online: http://www.oakland.edu/enp/. Accessed March 2007.

F. Harary. *Graph Theory*. Narosa Publishing House, New Delhi, 1988.

F. Harary and I. C. Ross. A procedure for clique detection using the group matrix. *Sociometry*, 20:205–215, 1957.

J. Hasselberg, P. M. Pardalos, and G. Vairaktarakis. Test case generators and computational results for the maximum clique problem. *Journal of Global Optimization*, 3:463–482, 1993.

ILOG. ILOG CPLEX. `http://www.ilog.com/products/cplex/`, 1987-2009. Accessed May 2009.

T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences of the USA*, 98(8): 4569–4574, 2001.

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

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

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

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

A. Nagurney, editor. *Innovation in Financial and Economic Networks*. Edward Elgar Publishers, London, 2003.

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

M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

X. Peng, M. A. Langston, A. M. Saxton, N. E. Baldwin, and J. R. Snoddy. Detecting network motifs in gene co-expression networks through integration of protein domain information. In P. McConnell, S. M. Lin, and P. Hurban, editors, *Methods of Microarray Data Analysis V*, pages 89–102. Springer, New York, 2007.

S. S. Ravi, D.J. Rosenkrantz, and G. K. Tayi. Heuristics and special case algorithms for dispersion problems. *Operations Research*, 42:299–310, 1994.

L. A. Sanchis and A. Jagota. Some experimental and theoretical results on test case generators for the maximum clique problem. *INFORMS Journal on Computing*, 8(2):103–117, 1996.

J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, London, 2 edition, 2000.

S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

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

V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003.

L. E. Trotter. A class of facet producing graphs for vertex packing polyhedra. *Discrete Mathematics*, 12: 373–388, 1975.

T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

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

M. Yannakakis. The effect of a connectivity requirement on the complexity of maximum subgraph problems. *Journal of the ACM*, 26(4):618–630, 1979.

Appendices

A Proofs

Theorem 2 *k-PLEX is NP-complete for any fixed positive integer k.*

PROOF. Since the k-plex definition is *nontrivial*, *interesting* and *hereditary on induced subgraphs* the maximum k-plex problem is NP-hard by the result of Yannakakis [1979]. Here, we provide a more direct proof by reducing CLIQUE [Garey and Johnson, 1979], a well-known NP-complete problem, to k-PLEX. Given an instance $\langle G = (V, E), c\rangle$ of CLIQUE, we construct an instance $\langle G' = (V', E'), c'\rangle$ 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 $V_r \setminus \{u_r\},\ r = 1, \ldots, k-1$. Putting $c' = c + (k-1)n$ completes the reduction. Note that the instance $\langle G' = (V', E'), c'\rangle$ can be constructed in polynomial time.

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, S induces a k-plex of size c'.

We now establish the other direction stating 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, $\forall\, i \in S \cap V : deg_{G[S]}(i) = |N(i) \cap (S \cap R)| + |N(i) \cap (S \cap V)| \geq c + n(k-1) - k, \Rightarrow |N(i) \cap (S \cap V)| \geq c + p - 1 - q_i$. 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 sum 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 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 and $|C| \geq c$. □

Theorem 3 *Let $P_k(G)$ denote the k-plex polytope of a given graph $G = (V, E)$, where $k > 1$. Then, (1) $dim(P_k(G)) = n$; (2) $x_i \geq 0$, and $x_i \leq 1$ induce facets of $P_k(G)$ for every $i \in V$.*

PROOF. Let e_i be the unit vector with i^{th} component 1 and the rest 0; $e_{ij} = e_i + e_j$. The points $\mathbf{0}, e_1, e_2, \ldots, e_n$ are clearly $n+1$ affinely independent points in $P_k(G) \subset \mathbb{R}^n$. Hence, $dim(P_k(G)) = n$.

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 n affinely independent points in F. This shows that $dim(F) = n - 1$ and it is a facet. 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 $1 < k < n$. Then e_i and e_{ij} for all $j \in V \setminus \{i\}$ form n affinely independent points in F', indicating that $dim(F') = n - 1$ and it is a facet. □

Theorem 5 *For a subset $J \subseteq V$ such that $|J| \geq 3$, the inequality $\sum_{i \in J} x_i \leq 2$, induces a facet of $P_2(G)$ if and only if J is a maximal co-2-plex.*

PROOF. Let J be a maximal co-2-plex. 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 (a.i.) 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, a facet. W.l.o.g. assume that $J = \{1, \ldots, r\}$ and $V \setminus J = \{r+1, \ldots, n\}$, where $r \geq 3$. Let $e_i \in \mathbb{R}^n$ denote the unit vector with i-th component one and all others zero. The a.i. vectors are constructed as: $x^v = e_v + e_r, \quad \forall\, v = 1, \ldots, r-1$; $x^r = e_1 + e_2$ (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 particular vertices described before. Clearly, $x^v \in F$ and it is easy to verify that they are a.i.

For the converse, suppose $\sum_{i \in J} x_i \leq 2$ induces a facet of $P_2(G)$. If J is not a co-2-plex, there exists some $v \in J$ with 2 neighbors in J which form a 2-plex. The incidence vector of this 2-plex violates the facet inducing inequality, leading to a contradiction. Hence, J must be a co-2-plex. If J is not maximal, then there exists a valid maximal co-2-plex inequality that dominates the given facet inducing inequality. Hence, J must be a maximal co-2-plex. □

Theorem 7 *Let $G = (V, E)$ be a cycle on $n = t(k+1)$ vertices with $t \geq 2, k \geq 3$ such that $k+1$ is odd. Then the inequality $\sum_{i \in V} x_i \leq k+1$ induces a facet of $P_k(G)$.*

PROOF. Let $V = \{1, 2, \ldots, n\}$ and $E = \{(i, i+1 \mod n) : i \in V\}$. Suppose there exists a valid inequality $ax \leq b$ that contains the face induced by the hole inequality, we show that $a_i = \lambda, i \in V$ and $b = (k+1)\lambda$ for some scalar λ. Note that the union of a path on $k-1$ vertices ($k \geq 3$) and a path on 2 vertices forms a k-plex (this includes one path on $k+1$ vertices) that satisfies the hole inequality at equality. In the following arguments, we first *fix* a path on $k-1$ vertices and consider every 2-vertex path, with two consecutive k-plexes so constructed, differing by one vertex. That is, if S is such a k-plex with (v, u) as the 2-vertex path, the next k-plex is constructed as $T = S \cup \{w\} \setminus \{v\}$ where $v \in S$, $w \in V \setminus S$, (u, w) is the new 2-vertex path, and hence, $a_v = a_w$.

Case I: n is odd. Fix the path $(1, 2, \ldots, k-1)$ in every solution. We consider 2-vertex paths in the order $(k, k+1)$, then $(k+1, k+2)$ and so on. Thus we obtain $a_k = a_{k+2} = \cdots = a_{n-3} = a_{n-1}$ and $a_{k+1} = a_{k+3} = \cdots = a_{n-2} = a_n$. Fix the path $(n-k+2, \ldots, n)$ in the following solutions, and vary the 2-vertex path sequentially starting with $(1, 2)$. Here we obtain $a_1 = a_3 = \cdots = a_{k-1} = a_{k+1}$ and $a_2 = a_4 = \cdots = a_{k-2} = a_k$. Together we have all odd index coefficients to be equal and all even index coefficients to be equal. Consider paths $(1, 2, \ldots, k+1)$ and $(2, 3, \ldots, k+2)$ to obtain $a_1 = a_{k+2}$.

Case II: n is even. The fixed paths are chosen as in Case I to obtain $a_1 = a_3 = \cdots = a_{n-1}$ and $a_2 = a_4 = \cdots = a_n$. Considering paths $(1, 2, \ldots, k+1)$ and $(2, 3, \ldots, k+2)$ we obtain $a_1 = a_{k+2}$. □

Theorem 8 *Let $G = (V, E)$ be a cycle on $n = t(k+1)$ vertices with $t \geq 2, k \geq 5$ such that $k+1$ is even. Then the inequality $\sum_{i \in V} x_i \leq k+1$ induces a facet of $P_k(G)$.*

PROOF. Note that when $k+1$ is even, so is n. The fixed paths are chosen as in Theorem 7 to obtain $a_1 = a_3 = \cdots = a_{n-3} = a_{n-1}$ and $a_2 = a_4 = \cdots = a_{n-2} = a_n$. In order to connect an odd coefficient to an even coefficient, we require $k \geq 5$. Then consider the paths $(1, 2, 3, 4, 5, \ldots, k+1)$ and $(1, 2, 3, 5, \ldots, k+2)$ (4 is deleted and $k+2$ is added). We obtain $a_4 = a_{k+2}$. □

B Some Remarks on Valid Inequalities

B.1 Co-k-plex Inequalities for $k \geq 3$.

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

arbitrary order $n \geq r_k$ as the union of $n - r_k$ clique components of size one and two clique components of size $k - 1 = r_k/2$. Then G is a co-k-plex with the two "large" clique components forming a k-plex of size r_k. Suppose $F = \{x \in P_k(G) : \sum_{i \in V} x_i = r_k\}$ is a facet of $P_k(G)$. Since $P_k(G)$ is an integral polytope, the extreme points of F are also integral. 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 $\sum_{j \in V \setminus N[i]} x^o_j \leq k - 1$. 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 $r_k > k$. 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 $\sum_{i \in V} x_i \leq r_k$ at equality, which is the incidence vector of $K_{k-1} \cup K_{k-1}$. Thus, F is 0-dimensional and not a facet. For odd k, we can have arbitrarily large graphs by adding single vertex components to the antiweb $G_k[V']$ constructed before. By using 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 $\omega_k(G) = r_k$ is only a necessary condition for the co-k-plex inequality to induce a facet of $P_k(G)$. Identifying graph classes for which the co-k-plex inequalities and rank inequalities $\sum_{i \in J} x_i \leq \omega_k(G[J])$ induce facets of the k-plex polytope when $k \geq 3$ is an important problem for future research. 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 maximal independent set inequalities and non-negativity constraints [Cornuéjols, 2001]. Similarly, we could explore *k-plex perfectness* of graphs whose k-plex polytope can be completely described by the co-k-plex inequalities described here and the trivial facets. This is also an interesting topic for future research.

B.2 MIS and Co-k-plex Inequalities.

Both MIS and (maximal) co-k-plex inequalities generalize the MIS inequalities for the clique polytope. For $k = 1$ they are identical, and for $k = 2$, co-2-plex inequalities induce facets and dominate the MIS inequalities as the right-hand side bounds are equal. But based on Theorem 5 and the observation that every lifted MIS inequality has 0 or 1 variable coefficients, every lifted MIS facet of $P_2(G)$ is a co-2-plex facet. For $k \geq 3$ however, there could exist facets of $P_k(G)$ obtained by sequentially lifting an MIS inequality that are different from any inequality co-k-plexes produce. Similarly, although co-k-plex inequalities do not produce facets in general for $k \geq 3$, they could provide stronger inequalities compared to MIS in some cases. As a result, for $k \geq 3$ neither inequality dominates the other in general. Consider the graph G_3 in Figure 1 when $k = 3$. The vertex set is a co-3-plex and the co-3-plex inequality is $\sum_{i=1}^{7} x_i \leq 5$ (note that we could tighten the RHS bound as $\omega_3(G_3) = 4$). The MIS inequality on the other hand is $x_1 + x_3 + x_5 + x_7 \leq 3$, which cuts off the point $[1, 0, 1, 0, 1, 0, 1]^T$ that is not cut-off by the co-3-plex inequality. Lifting the MIS inequality following the sequence (x_2, x_4, x_6) yields $x_1 + x_2 + x_3 + x_5 + x_6 + x_7 \leq 3$, and following the sequence (x_4, x_2, x_6) yields $x_1 + x_3 + x_4 + x_5 + x_7 \leq 3$. Both are facets, as the variables were maximum lifted. Consider the graph G_4 in Figure 1 when $k = 3$. The independence number is 3 and the MIS inequalities are implied by variable bounds, while the co-3-plex inequality $\sum_{i=1}^{6} x_i \leq 5$ is not.

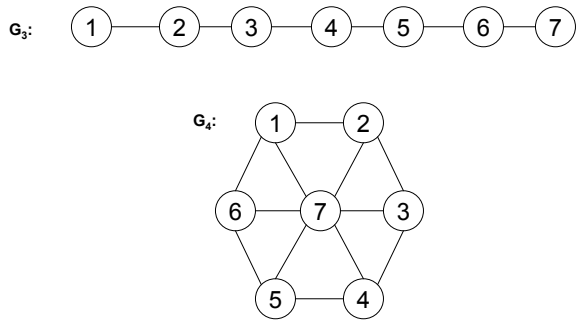

Figure 1: Example graphs for comparing MIS and co-3-plex inequalities.

C Numerical Results: Sanchis Graphs

The largest order up to which optimal resolution was possible on Sanchis instances within the 3-hour time limit, using the specified algorithm, for each density is presented in Table 5. Note that "< 100" indicates that the smallest instance in our test bed with 100 vertices was not solved optimally. Tables 6 and 7 present the total running time (excluding read/write time) and number of BC nodes enumerated for solving maximum 1-plex problem on Sanchis graphs using BC(MIS) implementation. Non-optimal termination is indicated by the dagger symbol (†). Running times and number of BC nodes enumerated by BC(MIS) for $k = 2$ is presented in Tables 8 and 9. The size of the largest 2-plex found and an upper bound on the 2-plex numbers obtained from the BC(MIS) implementation is provided in Table 10. Running times and number of BC nodes enumerated by BC(co2plex) for $k = 2$ is presented in Tables 11 and 12 up to $n = 500$ and $d = 0.6$. Note that none of the other instances were solved optimally.

Table 5: Summary of results on Sanchis instances

k	Algorithm	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
1	CPLEX default	800	900	900	900	700	200
1	BC(MIS)	1000	1000	900	1000	800	300
2	CPLEX default	1000	600	200	100	< 100	< 100
2	BC(MIS)	1000	900	600	200	< 100	100
2	BC(co2plex)	400	300	200	< 100	< 100	< 100

Table 6: Running time (secs) of BC(MIS) for $k = 1$ on Sanchis instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	0.203	0.422	1.812	0.797	0.328	0.14
200	2.172	4.109	10.734	9.86	27.047	22.171
300	8.891	13.922	52.625	52.484	220.422	2.219
400	30.235	47.766	322.235	182.266	681.5	*TiLim*
500	69.86	90.5	807.296	322.953	1402.66	*TiLim*
600	169.171	226.813	2148.77	605.219	3570.42	*TiLim*
700	332.813	453	5594.47	1078.2	6591.23	*TiLim*
800	560.125	2041.39	5623.413252	1750.89	7028.73	*TiLim*
900	1057.34	1269.95	4995.91	3358.47	*TiLim*	*TiLim*
1000	1894.39	2349.67	*TiLim*	4260.28	*TiLim*	*TiLim*

Table 7: Number of nodes enumerated by BC(MIS) for $k = 1$ on Sanchis instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	29	199	1126	86	39	4
200	354	1152	4022	170	1440	4065
300	511	2041	13547	303	3554	5
400	1613	5474	50350	442	4794	250409†
500	1626	4910	94790	373	4575	104917†
600	4199	10788	183446	402	6616	60535†
700	6979	15313	314525	406	7384	38725†
800	6774	83045	423327	458	5377	25228†
900	14221	26694	158301	654	5459†	17790†
1000	21789	40319	276913†	570	3684†	11660†

Table 8: Running time (secs) of BC(MIS) for $k = 2$ on Sanchis instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	1.266	2.562	12.203	320.125	*TiLim*	18.312
200	10.297	17.859	61.797	4075.13	*TiLim*	*TiLim*
300	47.844	76.36	286.235	*TiLim*	*TiLim*	*TiLim*
400	141.61	248.125	999.922	*TiLim*	*TiLim*	*TiLim*
500	377.422	688.25	2495.25	*TiLim*	*TiLim*	*TiLim*
600	820.344	1466.3	5525.61	*TiLim*	*TiLim*	*TiLim*
700	1610.06	2981.74	*TiLim*	*TiLim*	*TiLim*	*TiLim*
800	3083.42	5478.7	*TiLim*	*TiLim*	*TiLim*	*TiLim*
900	6058.16	9204.52	*TiLim*	*TiLim*	*TiLim*	*TiLim*
1000	9926.52	*TiLim*	*TiLim*	*TiLim*	*TiLim*	*TiLim*

Table 9: Number of nodes enumerated by BC-MIS for $k = 2$ on Sanchis instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$	$d = 0.7$	$d = 0.8$	$d = 0.9$
100	523	1467	7946	169319	$1851177^{\dagger}$	10850
200	1574	4762	24859	283774	$753120^{\dagger}$	$699101^{\dagger}$
300	2910	9525	59309	$106882^{\dagger}$	$238681^{\dagger}$	$597044^{\dagger}$
400	3976	15987	119868	$38867^{\dagger}$	$81804^{\dagger}$	$242840^{\dagger}$
500	5249	23613	174602	$16181^{\dagger}$	$34463^{\dagger}$	$101949^{\dagger}$
600	6411	29417	247807	$7115^{\dagger}$	$17901^{\dagger}$	$52399^{\dagger}$
700	8918	41936	$269150^{\dagger}$	$4022^{\dagger}$	$9490^{\dagger}$	$30193^{\dagger}$
800	10095	45481	$141117^{\dagger}$	$2426^{\dagger}$	$5713^{\dagger}$	$19134^{\dagger}$
900	15997	50779	$92591^{\dagger}$	$1622^{\dagger}$	$3693^{\dagger}$	$12578^{\dagger}$
1000	19197	$24291^{\dagger}$	$58027^{\dagger}$	$1109^{\dagger}$	$2528^{\dagger}$	$8579^{\dagger}$

Table 10: 2-plex numbers found by BC(MIS) on Sanchis 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, 25]	38
200	40	40	40	40	[40, 60]	[50, 75]
300	60	60	60	[60, 90]	[60, 103]	[59, 116]
400	80	80	80	[80, 130]	[80, 148]	[76, 159]
500	100	100	100	[100, 176]	[100, 191]	[99, 201]
600	120	120	120	[120, 222]	[120, 231]	[117, 245]
700	140	140	[140, 146]	[140, 266]	[140, 280]	[136, 297]
800	160	160	[160, 236]	[160, 306]	[160, 324]	[155, 339]
900	180	180	[180, 292]	[180, 349]	[180, 370]	[180, 388]
1000	200	[200, 333]	[200, 352]	[200, 386]	[200, 414]	[196, 440]

Table 11: Running time (secs) of BC(co2plex) for $k = 2$ on Sanchis instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$
100	6.781	7.985	35.5
200	189.797	485.625	1667.7
300	1658.7	4966.11	*TiLim*
400	9486.86	*TiLim*	*TiLim*
500	*TiLim*	*TiLim*	*TiLim*

Table 12: Number of nodes enumerated by BC(co2plex) for $k = 2$ on Sanchis instances

n	$d = 0.4$	$d = 0.5$	$d = 0.6$
100	607	1515	8445
200	1779	5780	27896
300	3754	12932	$13899^{\dagger}$
400	7492	$3541^{\dagger}$	$3490^{\dagger}$
500	$1109^{\dagger}$	$1452^{\dagger}$	$762^{\dagger}$

Table 13: DIMACS benchmarks.

Graphs	$\|V\|$	$\|E\|$	d	$\omega(G)$	BC(MIS) (secs)	$\omega_2(G)$	BC(MIS) (secs)
c-fat200-1.clq	200	1534	0.077	12	17.1	12	148.9
c-fat200-2.clq	200	3235	0.163	24	10.4	24	19.1
c-fat200-5.clq	200	8473	0.426	58	2.1	58	2.1
c-fat500-1.clq	500	4459	0.036	14	1334.4	14	1356.1
c-fat500-2.clq	500	9139	0.073	26	535.7	26	605.3
c-fat500-5.clq	500	23191	0.186	64	141.6	64	141.5
c-fat500-10.clq	500	46627	0.374	126	39.3	126	76.5
hamming6-2.clq	64	1824	0.905	32	0.0	32	0.0
hamming6-4.clq	64	704	0.349	4	0.2	6	0.3
hamming8-2.clq	256	31616	0.969	128	0.0	128	189.5
hamming8-4.clq	256	20864	0.639	16	52.2	16	8115.2
hamming10-2.clq	1024	518656	0.990	512	0.8	[512,530]	*TiLim*
hamming10-4.clq	1024	434176	0.829	[36,234]	*TiLim*	[41,153]	*TiLim*
johnson8-2-4.clq	28	210	0.556	4	0.0	5	0.0
johnson8-4-4.clq	70	1855	0.768	14	0.1	14	4.4
MANN_a9.clq	45	918	0.927	16	0.0	26	0.0
MANN_a27.clq	378	70551	0.990	126	430.3	236	79.8
MANN_a45.clq	1035	533115	0.996	[344,347]	*TiLim*	[662,668]	*TiLim*
keller4.clq	171	9435	0.649	11	129.8	15	365.4
brock200_1.clq	200	14834	0.745	[20,31]	*TiLim*	[25,53]	*TiLim*
brock200_2.clq	200	9876	0.496	12	152.5	[13,24]	*TiLim*
brock200_4.clq	200	13089	0.658	17	6617.5	[19,41]	*TiLim*
brock400_2.clq	400	59786	0.749	[24,68]	*TiLim*	[27,133]	*TiLim*
brock400_4.clq	400	59765	0.749	[23,69]	*TiLim*	[27,133]	*TiLim*
brock800_2.clq	800	208166	0.651	[19,116]	*TiLim*	[23,253]	*TiLim*
brock800_4.clq	800	207643	0.650	[19,108]	*TiLim*	[23,252]	*TiLim*
p_hat300-1.clq	300	10933	0.244	8	127.0	[9,66]	*TiLim*
p_hat300-2.clq	300	21928	0.489	[25,51]	*TiLim*	[28,85]	*TiLim*
p_hat300-3.clq	300	33390	0.744	[35,71]	*TiLim*	[43,108]	*TiLim*
p_hat700-1.clq	700	60999	0.249	[11,40]	*TiLim*	[10,291]	*TiLim*
p_hat700-2.clq	700	121728	0.498	[44,208]	*TiLim*	[50,298]	*TiLim*
p_hat700-3.clq	700	183010	0.748	[62,201]	*TiLim*	[73,311]	*TiLim*

Table 14: Runtime in seconds for Erdös networks using IPBC algorithm.

k	Graph	IPBC Time	BC Time	#BC Calls
1	ERDOS-97-1	1.4	0.0	5
	ERDOS-98-1	1.4	0.1	6
	ERDOS-99-1	1.5	0.0	6
	ERDOS-97-2	367.5	0.1	7
	ERDOS-98-2	445.4	0.0	7
	ERDOS-99-2	491.9	0.0	4
2	ERDOS-97-1	1.5	0.2	5
	ERDOS-98-1	1.7	0.2	6
	ERDOS-99-1	1.8	0.3	6
	ERDOS-97-2	392.9	0.5	7
	ERDOS-98-2	464.3	0.6	7
	ERDOS-99-2	526.5	0.4	9
3	ERDOS-97-1	1.8	0.4	5
	ERDOS-98-1	1.8	0.3	6
	ERDOS-99-1	1.8	0.3	6
	ERDOS-97-2	394.1	8.7	7
	ERDOS-98-2	457.1	1.1	7
	ERDOS-99-2	520.0	3.2	9
4	ERDOS-97-1	2.2	1.0	4
	ERDOS-98-1	2.8	1.5	4
	ERDOS-99-1	1.8	0.3	4
	ERDOS-97-2	424.0	39.6	3
	ERDOS-98-2	614.7	159.8	3
	ERDOS-99-2	526.3	10.6	4
5	ERDOS-97-1	5.7	4.5	4
	ERDOS-98-1	7.9	6.6	4
	ERDOS-99-1	9.9	8.5	4
	ERDOS-97-2	1042.8	688.2	3
	ERDOS-98-2	1664.6	1244.6	3
	ERDOS-99-2	653.5	178.1	4

Table 15: Runtime in seconds for biological networks using IPBC algorithm.

k	Graph	IPBC Time	BC Time	#BC Calls
1	H. Pylori	11.5	0.0	13
	S. Cerevisiae	44.1	0.0	0
2	H. Pylori	12.6	1.3	2
	S. Cerevisiae	46.4	0.0	0
3	H. Pylori	37.8	26.6	2
	S. Cerevisiae	45.1	0.0	1
4	H. Pylori	29.3	18.3	2
	S. Cerevisiae	45.0	0.0	1
5	H. Pylori	133.1	123.0	3
	S. Cerevisiae	41.8	0.0	3

Table 16: Runtime in seconds for computational geometers collaboration networks using IPBC algorithm.

k	Graph	IPBC Time	BC Time	#BC Calls
1	GEOM-0	2287.6	0.0	0
	GEOM-1	701.2	0.0	0
	GEOM-2	479.6	0.0	0
2	GEOM-0	2384.4	0.0	0
	GEOM-1	753.2	0.1	2
	GEOM-2	530.6	0.1	4
3	GEOM-0	2387.1	0.0	0
	GEOM-1	747.7	0.1	2
	GEOM-2	524.3	0.0	1
4	GEOM-0	2383.7	0.0	0
	GEOM-1	743.7	0.4	2
	GEOM-2	522.2	0.1	1
5	GEOM-0	2298.1	0.0	0
	GEOM-1	691.6	1.8	2
	GEOM-2	472.6	0.5	4

Table 17: Runtime in seconds for the Reuters terror news networks using IPBC algorithm.

k	Graph	IPBC Time	BC Time	#BC Calls
1	DAYS-3	3110.8	0.1	3
	DAYS-4	2940.8	0.2	1
	DAYS-5	2758.0	0.0	0
2	DAYS-3	3367.8	4.1	1
	DAYS-4	2635.7	0.3	1
	DAYS-5	2462.9	0.1	2
3	DAYS-3	3395.4	45.5	1
	DAYS-4	2625.1	4.7	1
	DAYS-5	2445.5	0.2	2
4	DAYS-3	3489.8	203.0	1
	DAYS-4	2642.3	51.4	1
	DAYS-5	2426.3	2.7	1
5	DAYS-3	15336.9	12329.1	1
	DAYS-4	6201.4	3316.8	1
	DAYS-5	2820.8	113.1	1