RICE UNIVERSITY

Bilevel Clique Interdiction and Related Problems

by

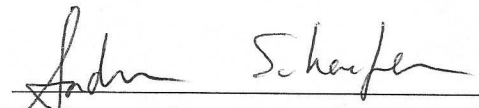
Timothy Joseph Becker

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

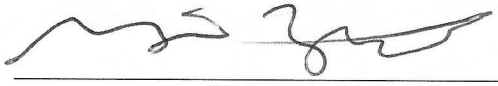
Doctor of Philosophy

Approved, Thesis Committee:

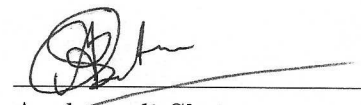
Illya V. Hicks, Chair
Professor of Computational and Applied Mathematics



Andrew J. Schaefer
Noah Harding Chair and Professor of Computational and Applied Mathematics



Yin Zhang
Professor of Computational and Applied Mathematics



Anshumali Shrivastava
Assistant Professor of Computer Science

Houston, Texas

April, 2017

ABSTRACT

Bilevel Clique Interdiction and Related Problems

by

Timothy Joseph Becker

I introduce a formulation of the bilevel clique interdiction problem. Interdiction, a military term, describes the removal of enemy resources. The single level clique interdiction problem describes the attempt of an attacker to interdict a maximum number of cliques. The bilevel form of the problem introduces a defender who attempts to minimize the number of cliques interdicted by the attacker. An algorithm and formulation for the bilevel clique interdiction problem has not previously been investigated. I start by introducing a formulation and a column-generation algorithm to solve the problem of bilevel interdiction of a minimum clique transversal and move forward to the creation of a delayed row-and-column generation algorithm for bilevel clique interdiction.

Next, I introduce a formulation and algorithm to solve the bilevel interdiction of a maximum stable set problem. Bilevel interdiction of a maximum stable set is choosing a maximum stable set, but with a defender who is attempting to minimize the maximum stable set that can be chosen by the interdictor. I introduce a deterministic formulation and a delayed column generation algorithm. Additionally, I introduce a stochastic formulation of the problem. I solve this problem using a cross-decomposition method that involves L-shaped cuts into a master problem as well as new "clique" cuts for the inner problem.

Lastly, I define new classes of valid inequalities and facets for the clique transversal polytope. The valid inequalities come from two graph structures who have a closed form for their vertex cover number, which we use as a specific case for finding a minimum clique transversal. The first class of facets are just the maximal clique constraints of the clique transversal polytope. The next class contains an odd hole with distinct cliques on each edge of the hole. Another similar class contains an odd clique with distinct maximal cliques on the edges of one of its spanning cycles. The fourth class contains a clique with distinct maximal cliques on every edge of the initial clique, while the last class is a prism graph with distinct maximal cliques on every edge of the prism.

Acknowledgments

I like to thank my advisor, Dr. Illya Hicks, for all of his help and patience over the last five years as he introduced me to the world of graph theory. I will be forever grateful for the opportunities provided to me that came from working with Dr. Hicks.

I would also like to thank Dr. Andrew Schaefer for his help in understanding stochastic methods, as well as for the time he took to have multiple conversations with me about all sorts of things in life.

I'd like to thank my other committee members, Dr. Yin Zhang and Dr. Anshumali Shrivastava for their helpful comments and ideas as I was going through the process of writing and defending this work.

I want to especially thank my wife, Elizabeth, for being by my side during this whole process even while getting her M.D. and balancing residency hours. I am a better person and graduate student because of her.

Finally, I want to thank my parents, brother, and sister for their support and constant communication over the last five years. I'm sure the multiple phone calls describing my work in detail were not as pleasant for them as they were for me, but I appreciate the never-ending willingness to talk at any time.

This research was supported by the National Science Foundation under grants CMMI-1404864, CMMI-1300477, and CMMI-1634550.

Contents

Illustrations

Tables

Chapter 1

Definitions

1.1 Definitions

I begin by defining many of the terms that will appear throughout this thesis. I start with definitions of graphs and graph structures and then move into definitions in integer programming.

1.1.1 Graph Theory Definitions

I first begin by defining a graph, since every problem in this thesis is described on a graph.

Definition 1.1 A graph G=(V,E) is a set of vertices (or nodes), V, and a set of edges, E, connecting vertices.

Next, I introduce subsets of vertices in a graph with specific structure that I will use often throughout this work. I begin with a clique. Figure 1.1 shows an example of a clique of size four.

Definition 1.2 A clique is a subset of vertices in a graph in which any two vertices are adjacent to each other.

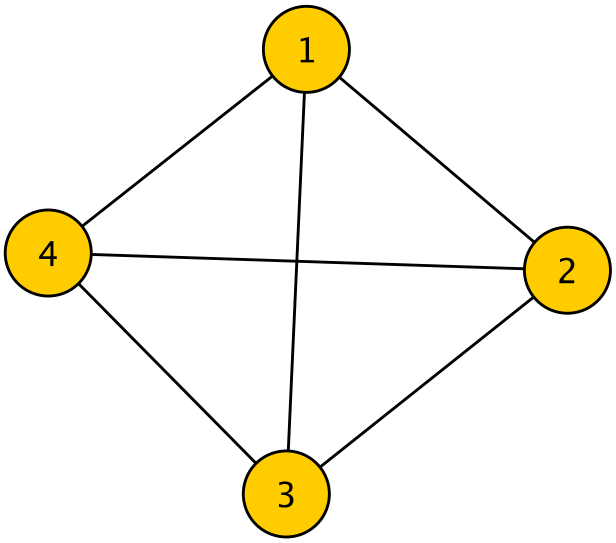


Figure 1.1 : A clique of size four

Since any vertex itself is a clique, I often focus on maximal cliques to emphasize the specific cohesive subgroups, rather than putting importance on every vertex and edge.

Definition 1.3 A maximal clique is a clique that cannot be extended by the addition of any other vertex.

A maximal clique can be thought of as a clique that is not contained in a bigger clique. Figure 1.2 shows a graph with three maximal cliques.

Now that I have the clique definitions, I can define other subsets of vertices based on their relationships to the maximal cliques in a graph.

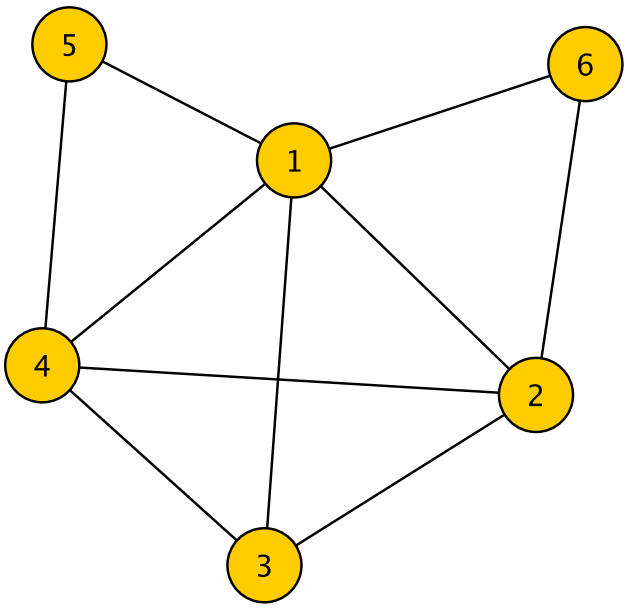


Figure 1.2 : Cliques $\{1, 2, 3, 4\}, \{1, 4, 5\}$, and $\{1, 2, 6\}$ are maximal cliques

Definition 1.4 A clique transversal is a subset of vertices such that every maximal clique contains at least one vertex in the subset.

A clique transversal can be thought of as a way of hitting every single clique in the graph. In Figure 1.2, it is trivial to find the minimum clique transversal. If node 1 is chosen, then every maximal clique in the graph is hit. Finding a minimum clique transversal is a classic problem in the literature and one that will be of further focus in Chapter 3 and 5.

Definition 1.5 A stable set is a subset of vertices such that every maximal clique contains at most one vertex from the set.

A stable set in a graph is a clique in the complement graph. Stable sets are also often called independent sets in the literature, but I will call them stable sets throughout this work. Based on the definition, one can see that the constraints of a stable set are the same as that of a clique transversal except for flipping the inequality. Based on this, I will have similar algorithms for solving problems involving both clique transversals and stable sets in Chapter 3 and 5. In Figure 1.2, one can find a maximum stable set by choosing nodes $\{3, 5, 6\}$.

I define a vertex cover, which is important in Chapter 5 for finding valid inequalities and facets for the clique transversal polytope.

Definition 1.6 A vertex cover is a subset of vertices such that every edge is incident to at least one vertex of the subset.

Next, I give the definition of a path. I need this definition to then define cycles and holes that are used in Chapter 5.

Definition 1.7 A path is a sequence of edges connecting a sequence of vertices that are distinct from one another.

With this definition of a path, I can define a cycle. Figure 1.3 has a cycle of size seven around the outer edges.

Definition 1.8 A cycle is a path of edges and vertices such that every vertex is reachable from itself.

Definition 1.9 A spanning cycle is a cycle that contains all of the vertices of the graph.

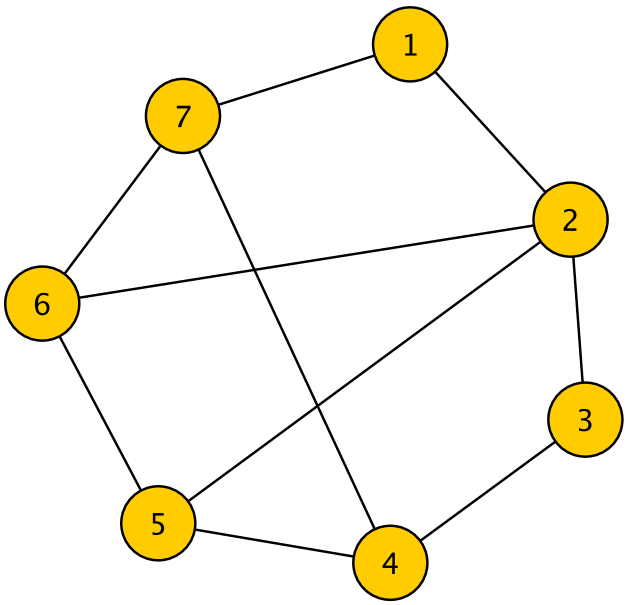


Figure 1.3 : Nodes $\{1, 2, 3, 4, 5, 6, 7, 1\}$ make up a cycle

A cycle can also be considered as a closed path. A hole is a specific type of cycle, but before I define a hole, I must define a chord of a graph.

Definition 1.10 A chord is an edge that joins two non-adjacent vertices in a cycle.

Now I can define a hole in a graph. Figure 1.4 is an example of a hole, since the chords have been removed.

Definition 1.11 A hole is a chordless cycle of length at least 4.

Lastly, I define certain types of graphs based on their structure and compositions. Firstly, I define a chordal graph.

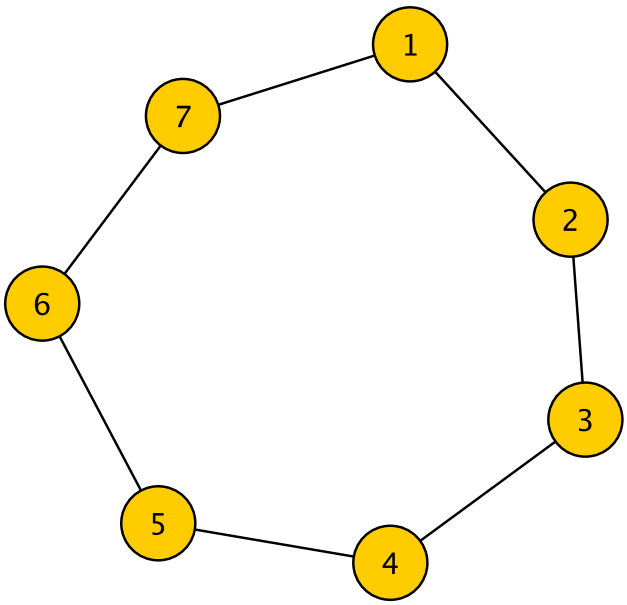


Figure 1.4 : Nodes $\{1, 2, 3, 4, 5, 6, 7, 1\}$ make up a hole, since there are no chords in the graph

Definition 1.12 A chordal graph is a graph within which every cycle of size 4 *or greater contains a chord.*

A chordal graph is one that does not contain a hole. Chordal graphs have special properties with respect to the number of cliques they contain so they will be used as test cases for some algorithms.

Lastly, I define a triangle-free graph.

Definition 1.13 A triangle-free graph is a graph within which no three vertices form a triangle.

Triangle-free graphs are useful for finding valid inequalities in Chapter 5.

1.1.2 Integer Programming Definitions

For the integer programming definitions, I am building each definition off of the previous definition to finally get to the definition of a facet at the end, which are the important pieces of Chapter 5. I begin with a polyhedron.

Definition 1.14 A polyhedron is the intersection of finite halfspaces.

The constraint space of a linear or integer program is called a polyhedron. This is the space over which one optimizes.

Definition 1.15 A polytope is a bounded polyhedron.

Many famous problems in integer programming have a constraint space that is a polytope rather than a polyhedron. In Chapter 5, I will consider the polytope of the clique transversal problem.

Next, I consider valid inequalities. Valid inequalities are useful for solving linear and integer programs because they can help to define the constraint space for a problem.

Definition 1.16 An inequality $w^T x \leq t$ is valid for a polyhedron, P, if $P \subseteq \{x : w^T x \leq t\}$.

If I consider where a valid inequality is satisfied with equality, then I have a hyperplane, which is the next definition.

Definition 1.17 A hyperplane is the solution set of an equation $w^T x = t$.

A specific type of hyperplane is called a supporting hyperplane. Supporting hyperplanes are always defined with respect to a given polyhedron, P.

Definition 1.18 A hyperplane is a supporting hyperplane with respect to P if $w^T x \leq t$ is valid for P and $P \cap \{x : w^T x = t\} \neq \emptyset$.

With the definition of supporting hyperplane, I can finally get to a face.

Definition 1.19 The intersection of a polyhedron with one of its supporting hyperplanes is called a face.

For a two-dimensional polyhedron, the faces are the vertices of the polyhedron and the edges between the vertices, along with the entire polyhedron and the empty set. The faces define the constraint space for the integer program. However, the most useful faces are facets.

Definition 1.20 A facet is a maximal proper face of a polyhedron.

The facets of the aforementioned two-dimensional polyhedron would be the edges, since the vertices are contained within the edges. Finding facets is an important problem in solving large integer programs. Adding in facets helps to shrink the solution space so that the problem can be solver faster numerically. I describe new facets for the clique transversal polytope in Chapter 5.

With all of these definitions in hand, I can move forward to the literature review in Chapter 2.

Chapter 2

Literature Review

2.1 Interdiction

Because most of this work is concerned with interdiction models, I begin with an overview of interdiction in the literature. Interdiction is the removal of an enemy's resources in a military sense. For graph theory, I consider interdiction to be choosing/removing a subset of nodes from a given graph. For applications involving dark networks, the nodes chosen are optimized with respect to the cost of removing a person from such a dark network. Figure 2.1 shows a sample dark network. For other applications like marketing, one can optimize the choice of nodes with respect to the cost of advertising towards members of the network. Interdiction models can be applied to homeland security, military operations, drug smuggling, hospital safety, and marketing, amongst many other applications [37] [51] [79] [4].

In graph theory, interdiction is mostly used to study problems involving the optimization of a disruption of the connectivity of a given network. This is an example of single level interdiction. However, my focus is on bilevel interdiction. Bilevel interdiction introduces a defender for the network whose job is it to optimize the protection strategy of the network. This is a common model form for applications involving

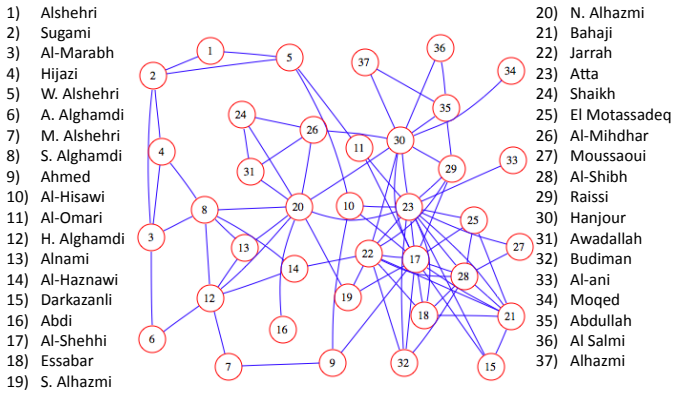


Figure 2.1 : 9-11 Terrorist Graph. Adapted by Dr. Illya Hicks from Krebs [42]

military operations. I will go through the instances of single level interdiction and bilevel interdiction in the following sections.

2.1.1 Single Level Interdiction

There are different methods to determine the type of interdiction used in a given scenario. Two of these types in this section include network interdiction and clique interdiction. Network interdiction models are used to describe various applications including many military situations [80]. Another application of network interdiction is infection control within hospitals [4].

Network interdiction is the removal of vertices in a graph to adversely affect the flow across the network. Network interdiction was first introduced in the literature by researchers at the Naval Postgraduate School in 1970s [51]. In 1971, Helmbold [37] solved a restricted case of network interdiction. Golden [34] formulated network

interdiction as a linear program in 1978 and solved a special case by using a minimum-cost flow technique. Wood [79] created the first linear programming formulation of deterministic network interdiction in 1993.

Shortest-path network interdiction and multi-commodity network interdiction also have been studied recently [40] [44]. Stochastic network interdiction has been developed for a variety of problems [54] [26] [63] [6]. Stochastic network interdiction minimizes the expected maximum flow through a network but takes into account some randomness in the problem. Some stochastic network interdiction problems have been shown to be able to be solved faster with parallel computing [41]. These aforementioned problems are examples of single level interdiction, where the interdictor chooses nodes to remove without any form of defense. Network interdiction is just one methodology for choosing which nodes to interdict. This thesis focuses on interdiction of specific graph subsets, which will be discussed in further sections. First, I must consider the problem of bilevel interdiction, which introduces a defender into the formulation.

2.1.2 Bilevel Interdiction

In bilevel interdiction, both sides are represented in the formulation. I have an interdictor who is trying to break up the network, as well as a defender who is trying to protect the network. These types of problems are more realistic in real world applications. It is easy to see how such a problem occurs with respect to homeland

security or drug networks. However, these problems also occur in applications such as marketing, where a consumer can attempt to protect themselves from advertisement by unsubscribing from emails, for example.

I develop a bilevel formulation for all of the interdiction problems within this thesis. Within these bilevel formulations, both the interdictor and the defender maximize their own objectives. Bilevel formulations have been used for network interdiction problems [68]. A further network interdiction formulation for zero sum game was introduced by Washburn and Wood [76]. Beyond just focusing on the formulations, bilevel network interdiction algorithms have been developed over the past decades. Wood [80] used Bender's Decomposition to solve these problems. Shen [68] developed bilevel formulations for other interdiction problems that optimized the number of components in the graph, the size of the largest component, or the minimum cost to rebuild the graph.

The next sections describe the three graph structures that are used for determining which nodes to interdict throughout this thesis. I begin with a clique, then a clique transversal, and, lastly, a stable set.

2.2 Clique

"Clique" was used by Luce and Perry [45] in 1949, making it the first instance of using the term in a graph theoretical sense. Examples of problems that find cliques in a given graph include finding a maximum cardinality clique, finding a maximum

weighted clique, enumerating all maximal cliques, and solving the decision problem to determine whether a clique larger than a given size exists. The maximum number of cliques in a graph with n vertices was determined to be $3^{\frac{n}{3}}$ for a graph with n vertices by Moon and Moser [53]. Bron and Kerbosch [18] subsequently developed an algorithm to enumerate all of the maximal cliques in a graph. They used a backtracking algorithm that is still used widely today. Further work on the backtracking algorithm has been done by Tomita et al [70] in 2006, in which they studied the worst time complexity for finding all maximal cliques in a graph. Furthermore, it was shown by Cazals and Karande [20] that Tomita's work was just a simple modification of the Bron and Kerbosch algorithm, using a piece of insight developed by Koch. Additionally, two years earlier, Makino and Uno [47] proposed two new algorithms for enumerating all of the maximal cliques.

Cliques are used in social network analysis as a measure of cohesiveness. In a social network, a group of people is said to be a "cohesive" group if it exhibits familiarity, reachability, and robustness [77]. Cliques demonstrate familiarity since an edge exists between any two nodes in the graph. Cliques have the reachability property for the same reason, since any node can be reached from another node by using the edge connecting the two nodes. Cliques are also robust, since the removal of one vertex creates a new clique with a cardinality one less than before. Hence, cliques are good representers of such subgroups.

For clique interdiction, the vertices are chosen to maximize the number of cliques

in the graph with at least one vertex chosen. Pajouh et al. [46] proposed a problem with a similar formulation to the clique interdiction problem. The similarities between clique interdiction and their problem is in removing vertices that attempt to break up cliques in the graph. Both formulations focus on removing a subset of nodes who are contained in maximal cliques. In contrast, they attempt to minimize the subset of nodes removed such that the maximum weight of a remaining clique is bounded above by a given integer, while I maximize the number of cliques interdicted while staying under a budget for node removal [46]. Figure 2.2 shows a small example of clique interdiction.

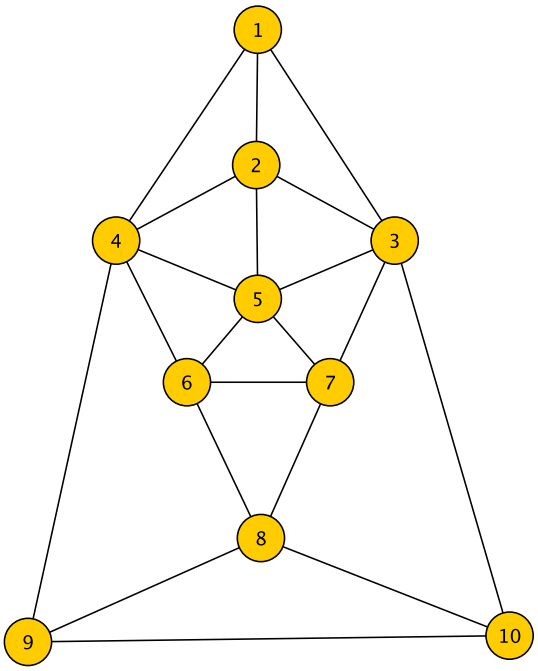


Figure 2.2 : Clique Interdiction Example: The maximal cliques of this graph are triangles. Choosing just vertices 2, 5, and 8 makes it such that every maximal clique is interdicted.

Maximum network disconnection partitions the network into as many pieces as possible, which is similar to clique interdiction. Walteros [75] proposed a formulation for the maximum network disconnection problem in 2012. Additionally, van Bevern et al [72] describe finding a minimum set of nodes whose removal splits the graph into k-plexes. These k-plexes are a relaxation of cliques, in which each node must have at most $k-1$ non-neighbors in the subgraph [50]. The problem in the literature that is closest to clique interdiction is the minimum clique-transversal problem, in which one attempts to find the minimum set of vertices that intersects every clique in a graph. However, clique interdiction includes a budget, so not every clique is necessarily interdicted based on their costs. In the minimum clique-transversal problem, every clique must have at least on of its members chosen.

2.3 Clique Transversal

Clique transversals were first introduced by Tuza [71] and have since been extensively studied. In 1992, the minimum clique transversal problem was determined to be NP-hard [30]. I consider the problem of bilevel interdiction of a minimum clique transversal.

The minimum clique transversal problem can be solved with basic integer and linear programming techniques. One of the most widely studied avenues of research with respect to clique transversals is that of finding what is called the clique transversal number. That is, what is the cardinality of the minimum clique transversal of a given

graph? Shan et al [67] described bounds on the clique transversal number of regular graphs. Andreae [2] examined the clique transversal number on chordal graphs. Andreae et al [3] also considered clique transversal sets on line graphs and complements of line graphs.

Clique transversals are used to define clique-perfect graphs. I have to go through a few definitions here that are not used throughout the rest of the thesis. These terms are necessary to understand the results for clique-perfect graphs in the next paragraph. A *clique-independent set* is a collection of pairwise node-disjoint cliques. A graph, G is *clique-perfect* if the cardinality of a maximum clique-independent set of G' is equal to the cardinality of a minimum clique transversal of G', for every induced subgraph G' of G [16]. An *antihole* is the complement of a hole. A *line graph*, $L(G)$ is a graph whose vertices represent the edges of G and whose edges represent adjacencies between edges of G. Line graphs are always defined with respect to a given graph, G. An *r-sun*, $r \geq 3$ is a chordal graph with $2r$ nodes such that the node set can be split into two distinct sets V, W such that W is a stable set and w_j is adjacent to u_i if and only if $i = j$ or $i \equiv j + 1(\text{mod r})$ [14]. A *claw-free* graph is one that does not contain $K_{1,3}$ as an induced subgraph. Lastly, a group of sets S has the *Helly property of order* k if any minimal subgroup whose intersection is empty has k of fewer sets in it. If all of the cliques in a given graph, G, satisfy the Helly property, then G is *clique-Helly* and G is *hereditary clique-Helly (HCH)* is H is clique-Helly for all induced subgraphs H of G [14].

Bonomo worked on characterizations of clique-perfect graphs, and Bonomo et al [16] showed in 2006 that odd generalized suns are not clique-perfect. Additionally, they showed that if G is a chordal graph, then G is clique-perfect if and only if G does not contain any odd suns [14]. Furthermore, if G is a line graph, then G is clique-perfect if and only if no induced subgraph of G is an odd hole or a 3-sun [14]. Along the same vein, Bonomo et al [15] showed that if G is an HCH claw-free graph, then G is clique-perfect if and only if no induced subgraph of G is an odd hole or an antihole of length 7.

Besides its use in defining clique-perfect graphs, finding a minimum clique transversal is a fundamental problem in graph theory and is comparable to clique interdiction. It, however, does not take into account the budget constraints necessary in clique interdiction. Additionally, clique interdiction maximizes the number of cliques interdicted, whereas minimum clique-transversal minimizes the number of vertices selected but must intersect all of the cliques. I consider the problem of bilevel interdiction of a minimum clique transversal. The differences between this and bilevel clique interdiction is that my interdictor is finding a minimum clique transversal and the defender is attempting to maximize the size of that minimum clique transversal.

Bilevel interdiction of a minimum clique transversal is important for applications in the case where it is necessary for every clique to be interdicted. While clique interdiction just maximizes the number of cliques interdiction, interdiction of a minimum clique transversal guarantees that every maximal clique has one of its vertices chosen

for interdiction. Therefore, if it is necessary for a marketing company to guarantee that its advertisement will reach all of its potential customers, this problem can guarantee it while minimizing the cost.

Considering the applications for interdiction, choosing a minimum clique transversal to interdict guarantees that the interdictor can break up every single maximal clique in the graph, as well as potentially learn information about every maximal clique in the graph. Thus, in terms of homeland security or drug-smuggling prevention, being able to interdict a minimum clique transversal can be of great importance.

2.4 Stable Set

Finding a maximum stable set in a graph is a widely known problem in graph theory. Since a stable set in a graph is equivalent to a clique in the complement graph, then the problem of finding a maximum stable set is complementary to that of finding a maximum clique. The cardinality of the maximum stable set in a graph is called the independence number, where the name is derived from the fact that a stable set is also known as an independent set.

Gavril [32] [33] produced an algorithm to find the maximum stable set on chordal graphs and another algorithm for circle graphs. Tarjan and Trojanowski [69] created an algorithm to find the maximum stable set on n-vertex graphs in $0(2^{n/3})$ time. There are specific graphs on which finding the maximum stable set is possible in polynomial time. Mosca [55] showed that one can find the maximum stable set in polynomial time

on graphs who do not have a path of 5 vertices as an induced subgraphs. Minty [52] created an algorithm to find a maximum stable set in polynomial time on claw-free graphs that was subsequently revised by Nakamura and Tamura [57]. At a similar time to Minty, Sbihi [65] independently found that the maximum stable set problem could be solved in polynomial time on claw-free graphs.

The key difference between clique interdiction and interdiction of a maximum stable set is that the choice of a maximum stable set ensures that no edge has both of its ends interdicted. Therefore, the interdictor does not "waste" any choice of nodes. Every single node that is chosen is useful for its own information. Clique Interdiction attempts to interdict as many cliques as possible in a graph. The optimal set of nodes for interdiction will be a subset that is close to a stable set, but it does not necessarily have to be. Interdicting a maximum stable set makes it such that the interdictor can get as much information as possible without overlapping on information that is already available due to the interdiction of another node. Therefore, I allow the interdictor to have no budget constraint, as compared to clique interdiction. The interdictor attempts to choose the maximum stable set possible, but has to deal with the defender protecting nodes in the network.

In a real world interdiction scenario, a deterministic model may not the best way to model the uncertainty inherent in the interactions. Stochastic integer programs have been used to model airline crew scheduling [66], telecommunications [43], and finance [29]. For this problem, I attempt to model a more realistic scenario by introducing

uncertainty into the defender's choice of nodes to protect. In a realistic scenario, the defender may attempt to protect specific nodes, but they do not know if their protection will actually work. For example, a person may be hidden away, but there is no guarantee that someone will not find that person, regardless of how well the defender thinks they are hidden. Therefore, I have a random aspect to whether or not the protection truly works in the problem. This models the fact that the defender has to take into account that the protection may not work and attempt to optimize their goals taking this uncertainty into account.

The way I do this is to use a stochastic formulation of the problem. Stochastic interdiction problems have been studied widely [6][26][41][61][63]. Two-stage stochastic problems have been used to study horticulture [28], disaster response [10], and many more applications [48] [21].

Our problem is a two-stage stochastic integer program, so it has a deterministic equivalent program (DEP) with a guaranteed convex objective function [13]. Thus, I can use the L-Shaped method described by Van Slyke and Wets [74]. Branch-and-bound is another method that has been used on two-stage stochastic integer programs [1], but I will focus on an adaptation of the L-Shaped method for my algorithm.

The last section details valid inequalities and facets and how they have been previously used for a multitude of problems in integer programming.

2.5 Facets for Clique Transversal Polytope

The clique transversal problem is to find a minimum cardinality set of nodes that intersects every maximal clique in the graph. The problem is a specific subset of the general set covering problem, where I choose the sets to be the maximal cliques in the graph.

The set covering problem has been widely studied and many algorithms have been developed to solve it, including a greedy algorithm by Chvátal [24]. Beasley and Chu [12] developed a genetic algorithm for the set covering problem. Furthermore, Beasley [11] and Caprara et al [19] created Lagrangian heuristics for solving the set covering problem.

Balas and Ng [7] provided a class of facets with coefficients in $\{0, 1, 2\}$, while Cornuéjols and Sassano [27] provided the $\{0, 1\}$ facets of the set covering polytope. Furthermore, lifting procedures have been utilized to find even more facets of the set covering polytope. Nobili [58] and Sassano [64] provided a generalized lifting procedure to find facets, while Balas and Ng [8] used lifting on their facets with coefficients in $\{0, 1, 2\}$ to extend the inequalities to an entire graph.

Valid inequalities and facets are common tools to help solve linear programs and integer programs. Padberg [60] identified facets for the set packing problem as well as working with Balas [9] to find facets for the set partitioning problem. Coll et al [25] found facets for the graph coloring polytope, while Hicks [38] discovered new facets for the planar subgraph polytope. Grötschel [35] and Padberg [36] described

inequalities and facets for the traveling salesman problem.

Moving forward, Chapter 3 details bilevel interdiction of a minimum clique transversal before moving into bilevel clique interdiction. Chapter 4 describes both deterministic and stochastic formulations for bilevel interdiction of a maximum stable set and Chapter 5 discusses new valid inequalities and facets for the clique transversal polytope. Chapter 6 concludes the thesis and briefly touches on potential future work.

Chapter 3

Bilevel Clique Interdiction

3.1 Bilevel Interdiction of a Minimum Clique Transversal

The problem of bilevel interdiction of a minimum clique transversal is as follows: given a budget, Q, for the defender, protection costs d_i on every node i, and a node goal, B, find a set, S, of nodes for the defender to protect such that $\sum_{i \in S} d_i \leq Q$ and the cardinality of the minimum clique transversal in $V \backslash S$ is greater than B. The interdictor will choose the minimum clique transversal possible in the graph among the nodes that are unprotected.

To begin the discussion of my formulation, I have the interdictor find a minimum clique transversal using the standard integer program formulation of the problem:

$$\begin{array}{lrcll} min & 1^T x & & & \\ \text{subject to} & Ax & \geq & 1, & (3.1) \\ & x_i & \in & \{0,1\}, \quad i \in V. & \end{array}$$

$$x_i = \begin{cases} 1 & \text{if node i is interdicted} \\ 0 & \text{otherwise.} \end{cases}$$

$$A_{ij} = \begin{cases} 1 & \text{if node j is in clique i} \\ 0 & \text{otherwise.} \end{cases}$$

This is just the minimum clique transversal integer program. Specific to my formulation, I consider this to be single level interdiction of a minimum clique transversal. I am interested in the bilevel case though, so I must add a defender into the integer program. Thus, I move on to the bilevel formulation:

$$\begin{array}{lll} max & f(z) + 1^T z & \\ \text{subject to} & d^T z \leq Q, & (3.2) \\ & z_i \in \{0,1\}, \quad i \in V. & \end{array}$$

$$f(z) = \begin{cases} min & 1^T x & \\ \text{subject to} & Ax \geq 1, & \\ & x + z \leq 1, & \\ & x_i \in \{0,1\}, \quad i \in V. & \end{cases} \qquad (3.3)$$

$$z_i = \begin{cases} 1 & \text{if node i is protected from interdiction} \\ 0 & \text{otherwise.} \end{cases}$$

Now, (3.2) and (3.3) describe bilevel interdiction of a minimum clique transversal. The subproblem is the interdictor's problem, while the outer problem is the

defender's problem. The defender attempts to maximize the size of the minimum clique transversal found by the interdictor.

Theorem 3.1 The problem of bilevel interdiction of a minimum clique transversal is NP-hard.

Proof 3.1 Finding a minimum clique transversal is NP-hard [30], so setting the defender's budget to 0 *and solving the problem is itself NP-hard. Therefore, bilevel interdiction of a minimum clique transversal is at least NP-hard since finding a minimum clique transversal is a specific case of bilevel interdiction of a minimum clique transversal.*

I relax the integral constraints for the inner problem so that I can take the dual and combine it with the outer problem to formulate a mixed integer program (MIP) formulation. The dual of the relaxed linear program in the inner problem is:

$$\begin{array}{lll} max & 1^T b + z^T b + 1^T a & \\ \text{subject to} & -b + A^T a \leq 1, & \\ & a \geq 0, \quad \forall \text{ maximal cliques k}, & \\ & b \geq 0, \quad i \in V. & \end{array} \tag{3.4}$$

where the dual variables are:

$$b : x + z \leq 1,$$

$$a : Ax \geq 1$$

In order to remove the non-linear term from the objective function, let: $s_i = z_i b_i$ for $i \in V$:

$$s_i \leq z_i,$$

$$s_i \leq b_i,$$

$$s_i - z_i - b_i \geq -1,$$

$$s_i \geq 0.$$

I combine these new constraints with the initial constraints from (3.1) to have the final formulation of bilevel interdiction of a minimum clique transversal as the following mixed integer program

$$\begin{array}{lrcll}
max & 1^T b + 1^T s + 1^T a + 1^T z & & & \\
\text{subject to} & -b + A^T a & \leq & 1, & \\
& s_i & \leq & z_i, & i \in V, \\
& s_i & \leq & b_i, & i \in V, \\
& s_i - z_i - b_i & \geq & -1, & i \in V, \\
& d^T z & \leq & Q, & \\
& z_i & \in & \{0,1\}, & i \in V, \\
& a_k & \geq & 0, & \forall \text{ maximal cliques k}, \\
& s_i, b_i & \geq & 0, & i \in V.
\end{array} \tag{3.5}$$

3.1.1 Algorithm Details & Computational Results

I consider (3.5) and first relax z to be linear variables as opposed to binary, so that I can solve as a linear program and then branch on the z variables to find the optimal mixed integral solution.

I let t be the dual variable associated with the primal constraint $-b + A^T a \leq 1$. I remove all a variables from the formulation and look to add them back in only if their reduced cost is non-negative. Thus, I look for cliques, k, such that $A_k t < 1$. These are the new columns that are added into the formulation.

To find these cliques, I enumerate all of the maximal cliques in the graph using the Bron and Kerbosch algorithm, specifically from the adaptation of Wood and Hicks [18] [78]. I search through all of the maximal cliques and add in all of those whose reduced cost is non-negative. Due to this, I will generally add a large number of cliques in initially, which cuts down on computational time of iterating over the list of maximal cliques and adding in one clique at a time.

Finally, when I have added in all of the necessary cliques, I branch on the non-integral z variables to achieve the optimal solution with binary z values.

Tables 3.1 and 3.2 show the averaged results on random graphs. The algorithm is exponential due to the addition of the maximal cliques, so I see the expected trajectory with respect to time as the number of nodes and the edge density grows. For the sparse graphs, I have to add in almost every maximal clique to reach the optimal solution. However, for graphs around 0.5 edge density, I do not have to add

in as many maximal cliques to reach the optimal solution. Therefore, this algorithm is more effective on graphs around 0.5 edge density, which are the graphs that are generally the hardest for clique problems.

For dense graphs, Table 3.1 shows that I go above the time limit of one hour around $50 - 70$ nodes. However, for sparse graphs, Table 3.2 shows that I can solve the problem in under an hour for up to 500 nodes. Since social networks are sparse, this algorithm will be effective on solving this problem on real world social networks. Table 3.3 gives an example of the algorithm on a small social networks.

Table 3.3 shows the results on a specific dark network from Everton [31]. This network has 79 people and the different rows describe different levels of connections between the members. Each graph is sparse and can be solved within seconds, so the addition of almost every maximal clique to the formulation is expected.

Tables 3.4, 3.5, 3.6, and 3.7 show the results of the algorithm on planar graphs. These graphs have a bounded number of cliques [62], so the algorithm can solve these instances in under an hour for up to 2400 nodes. The results show the same behavior as with random graphs with respect to the number of cliques necessary to reach the optimal solution. The sparsity of these planar graphs makes it such that I need almost all of the maximal cliques added into the formulation to solve the problem. However, due to their structure, this does not take the amount of time that it does for random graphs. Thus, the algorithm is effective on planar graphs up to 2400 nodes, as opposed to just $50 - 100$ for random graphs and 500 for sparse random graphs.

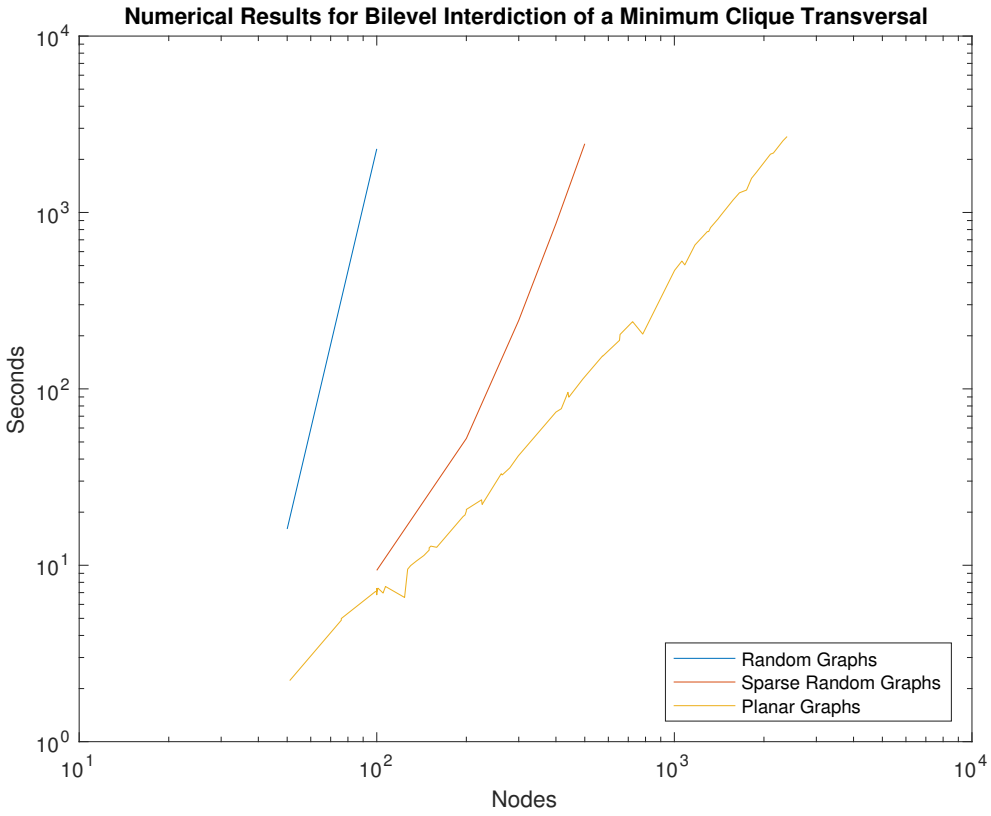


Figure 3.1 : Numerical Results for Bilevel Interdiction of a Minimum Clique Transversal

Figure 3.1 shows the comparative seconds to solve bilevel interdiction of a minimum clique transversal on random graphs, sparse random graphs, and planar graphs.

Table 3.1 : Numerical Results for Bilevel Interdiction of a Minimum Clique Transversal on Random Graphs (Averaged)

Nodes	Edge Density	Seconds	Total Cliques	Cliques Added	Solution
50	0.1	1.96	94.0	92.8	119.56
50	0.2	3.05	161.4	157.5	111.21
50	0.3	4.64	265.9	255.6	104.60
50	0.4	8.21	499.8	473.6	101.10
50	0.5	16.06	924.2	824.5	95.38
50	0.6	53.78	2047.9	1799.1	91.68
50	0.7	347.83	5442.0	4486.0	85.50
50	0.8	>3600	N/A	N/A	N/A
100	0.1	9.36	333.6	331.3	225.20
100	0.2	19.21	823.0	820.2	202.56
100	0.3	57.62	1970.6	1892.7	197.41
100	0.4	300.17	5214.7	4954.2	179.94
100	0.5	2294.02	15292.4	13734.5	177.54
100	0.6	>3600	N/A	N/A	N/A

Table 3.2 : Numerical Results on Sparse Random Graphs for Bilevel Interdiction of a Minimum Clique Transversal (Averaged)

Nodes	Edge Density	Seconds	Total Cliques	Cliques Added	Solution
100	0.1	9.36	333.6	331.3	225.20
200	0.1	52.34	1427.7	1417.5	428.61
300	0.1	244.67	3877.4	3866.7	610.45
400	0.1	863.39	8354.1	8346.0	780.40
500	0.1	2451.80	15196.2	15183.6	943.00
600	0.1	>3600	N/A	N/A	N/A

Table 3.3 : Numerical Results on Dark Networks

Name	Edges	Seconds	Total Cliques	Cliques Added	Solution
Classmates	175	3.50	61	61	197.98
Friendship	90	3.54	78	75	182.99
Kinship	16	3.46	67	67	232.40
Soulmates	11	3.62	74	74	247.40
Agg. Trust	258	3.93	87	84	159.98
Logistics	29	3.57	72	72	237.78
Meetings	64	3.70	68	68	218.60
Operations	268	3.32	50	50	189.18
Training	147	3.58	52	52	194.80
Agg. Operational	438	3.20	48	41	141.39
Communication	200	4.03	95	90	152.17
Business & Financial	15	3.36	71	71	244.18
Overall Aggregate	623	4.62	139	110	125.18

Table 3.4 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
a280	35.67	507	507	510.67
bier127	9.50	236	233	245.67
ch130	9.96	240	240	252.00
ch150	12.19	277	275	282.00
d1291	779.48	2555	2555	2248.00
d1655	1292.79	3236	3225	2867.00
d198	19.29	372	372	363.83
d2103	2140.10	4182	4182	3619.67
d493	113.91	965	965	890.33
d657	203.46	1290	1290	1167.00
eil101	7.41	190	190	200.33
eil51	2.22	90	89	99.00
eil76	4.88	140	139	150.67
fl1400	915.94	2719	2719	2448.67
fl1577	1179.44	3052	3052	2709.40

Table 3.5 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
fl417	77.18	753	752	739.67
gil262	33.05	504	504	478.00
kroA100	7.07	180	180	192.67
kroA150	12.24	273	272	280.67
kroA200	20.30	383	383	376.00
kroB100	7.39	185	185	188.67
kroB150	12.56	281	281	288.00
kroB200	20.72	377	377	360.00
kroC100	7.00	181	180	190.67
kroE100	6.81	176	176	190.00
lin105	6.96	184	179	199.67
nrw1379	889.84	2717	2717	2391.67
p654	188.40	1151	1145	1152.71
pcb1173	654.43	2314	2314	2036.33
pcb442	89.71	843	843	799.96

Table 3.6 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
pr1002	469.92	1971	1971	1756.00
pr107	7.58	177	176	203.00
pr124	6.57	195	191	255.20
pr136	10.58	242	238	260.67
pr144	11.36	245	239	260.50
pr152	12.84	270	270	282.67
pr226	22.12	357	341	372.00
pr2392	2693.79	4677	4677	4087.33
pr264	32.52	509	509	482.00
pr299	41.70	551	549	549.00
pr439	95.58	851	851	799.00
pr76	4.99	139	139	151.33
rat195	18.89	364	364	369.67
rat575	153.43	1121	1121	1033.00
rat783	204.44	1532	1532	1386.30
rat99	7.10	179	179	184.33

Table 3.7 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
rd100	7.14	182	182	205.33
rd400	73.87	780	776	725.33
rl1304	779.48	2545	2481	2261.19
rl1323	820.77	2599	2549	2256.90
rl1889	1693.52	3682	3647	3202.97
tsp225	23.50	398	398	416.17
u1060	530.37	2082	2082	1858.67
u1432	962.69	2773	2773	2484.67
u159	12.64	269	268	286.53
u1817	1562.04	3570	3570	3139.67
u2152	2173.23	4157	4157	3695.33
u2319	2553.18	4551	4551	4019.00
u574	153.81	1118	1116	1025.33
u724	240.49	1376	1374	1284.00
vm1084	504.43	1778	1771	1766.28
vm1748	1340.09	3015	3015	2768.67

3.1.2 Conclusion

I have developed an MIP formulation for bilevel interdiction of a minimum clique transversal and demonstrated the efficacy of the column generation algorithm on solving the problem. For sparse random graphs, I can solve the problem up to 500 nodes within one hour. Since social networks are sparse, I expect this algorithm to be usable on real world social networks up to, and perhaps beyond, 500 nodes. The algorithm solves the problem within seconds for the dark network example that I provide results for in this paper. For planar graphs, since they have a bounded number of maximal cliques, I can solve the problem within one hour for up to 2400 nodes.

With respect to the number of maximal cliques I need to add into the formulation, I add in almost every maximal clique for sparse graphs, but for graphs around 0.5 edge density, I do not need to add in as many maximal cliques. Thus, this algorithm can be most effective at shrinking the size of the integer problem for such graphs.

3.2 Bilevel Clique Interdiction

Bilevel clique interdiction is a similar problem to that of bilevel interdiction of a minimum clique transversal. The main difference is that the interdictor does not have to remove a node from every clique in bilevel clique interdiction; they attempt to interdict as many cliques as possible while remaining under a new budget constraint. Therefore, the defender is trying to minimize the number of cliques interdicted by the

interdictor.

Complexity

First, I consider the single level problem. I will show that the problem is NP-hard. The decision version of the Single Level Clique Interdiction problem is the following:

Instance: A graph G with node weights w; budget constraint B (positive integer); clique goal K (positive integer)

Question: Does there exists a subset S of nodes such that the total weight of S is at most B and the total number of maximal cliques transversed by S is at least K.

Theorem 3.2 The Single Level Clique Interdiction Decision Problem is NP-complete for split graphs, complement of bipartite graphs, planar graphs, and line graphs.

Proof 3.2 To determine whether clique interdiction is in NP, I can consider the clique transversal problem, since the question for each decision problem concerns counting the number of cliques that a subset of nodes intersects. In general, clique interdiction is not in NP, because finding the number of cliques that a given number of nodes interdicts is not known to be done in polynomial time for general graphs. In contrast, the clique transversal problem is in NP for certain graphs: split graphs (graphs whose vertices can be partitioned into a clique and independent set), complement of bipartite graphs, planar graphs, and line graphs [49]. Therefore, for these graphs, clique interdiction is also in NP, since determining if a subset of nodes is a clique transversal is

solvable in polynomial time for these graphs.

I reduce from the knapsack problem. For the knapsack problem, I have:

Instance: A finite set U; a size s (positive integer) and value v (positive integer) for each element of U; a size constraint B (positive integer); and a value goal K (positive integer)

Question: Is there a subset $\hat{U}$ of U such that the total size of $\hat{U}$ is at most B and the total value of $\hat{U}$ is at least K?

To reduce down to the single level clique interdiction problem, I construct a new graph. For each of the items in U, draw a star graph with v_i neighbors from the center node. Let the weight of each central node be $w_i = v_i$ for the associated item from the knapsack and let the weight of each auxiliary node on the star be $M >> B$. I have constructed a graph that is the disjoint union of star graphs. Additionally, I just let the cost of each central node be $c_i = s_i$. Now, I have the budget constraint $\sum_{i \in \hat{U}} c_i \leq B$. Solving the knapsack problem is now equivalent to solving the clique interdiction problem on this graph. Therefore, the Clique Interdiction decision problem is NP-complete for split graphs, complement of bipartite graphs, planar graphs, and line graphs.

Since the Single Level Clique Interdiction Decision Problem is NP-complete for these graphs, but not known to be in NP for general graphs, then I conclude that the Single Level Clique Interdiction Problem is NP-hard.

Now, the decision version of the Bilevel Clique Interdiction problem is the follow-

ing:

Instance: A graph G with node weights w and z; budget constraint B and D (positive integers); clique and protection goal K (positive integer)

Question: Does there exists a subset S of nodes such that the total weight of S is at most B and a subset of nodes S' such that the total weight of S' is at most D and the sum of the total number of maximal cliques transversed by S and the total number of nodes in S' is at least K.

Theorem 3.3 The Bilevel Clique Interdiction Problem is NP-hard.

Proof 3.3 If I take $D = 0$, then the defender has no budget, so S' is empty. This is just the single level clique interdiction decision problem. Since the single level clique interdiction problem is a specific case of the bilevel clique interdiction decision problem, then I can conclude that the bilevel clique interdiction problem is NP-hard.

Having shown that the problem is NP-hard, I detail the integer program formulation of the bilevel clique interdiction problem in the next section.

3.2.1 Integer Program Formulation

Given a graph $G = (V, E)$, I define the clique interdiction problem as finding the minimum set of nodes that intersects the maximum number of cliques in the graph, while staying below a given budget. As such, I define c to be node weights, and R to be the pre-determined budget. Additionally, x is the binary vector over nodes for

deciding which nodes to target, while y is the binary vector over all cliques detailing which cliques are interdicted by the choice of x. Thus,

$$x_i = \begin{cases} 1 & \text{if node i is interdicted} \\ 0 & \text{otherwise.} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if clique k is interdicted} \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, I define A to be the clique incidence matrix with

$$a_{ij} = \begin{cases} 1 & \text{if node j is in clique i} \\ 0 & \text{otherwise.} \end{cases}$$

With these variables, and the notation I_k as the index set of all nodes in a clique k, I introduce the following integer program formulation of the Single-Level Clique Interdiction Problem:

$$\begin{aligned} max \quad & 1^T y \\ \text{subject to} \quad & Ax \geq y, \\ & c^T x \leq R, \\ & y_k \in \{0,1\}, \quad \forall \text{ maximal cliques k}, \\ & x_i \in \{0,1\}, \quad i \in V. \end{aligned} \tag{3.6}$$

The objective function maximizes the cardinality of the cliques interdicted. The first constraint makes sure that if a clique k is interdicted, then at least one of its nodes is interdicted. The second constraint restricts the interdictor to stay within the given

budget. Lastly, it is a binary integer program; each of the variable are constrained to binary. This formulation is a basic starting point for the clique interdiction problem, but if one wishes to model the problem from both the attacker and defender's position, this requires a bilevel formulation.

Now, I generalize the bilevel interdiction problem with a defender who attempts to minimize the maximum number of cliques that the attacker can interdict.

I define c, d to be node weights, and R, Q to be the pre-determined budget for the interdictor and the defender, respectively. Additionally, x and z are the binary vectors over nodes for deciding which nodes to target or protect, respectively, while y is the binary vector over all cliques detailing which cliques are interdicted by the choice of x. Thus,

$$x_i = \begin{cases} 1 & \text{if node i is interdicted} \\ 0 & \text{otherwise.} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if clique k is interdicted} \\ 0 & \text{otherwise.} \end{cases}$$

Thus,

$$z_i = \begin{cases} 1 & \text{if node i is protected} \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, I_k is the index set of all nodes i within a given clique $k \in K$. After the variables have been defined, I have the following two stage integer program formulation

of the bilevel clique interdiction problem.

$$\begin{aligned} min \quad & f(z) - 1^T z \\ \text{subject to} \quad & d^T z \leq Q, \\ & z_i \in \{0,1\}, \quad i \in V. \end{aligned} \tag{3.7}$$

$$f(z) = \begin{cases} max & 1^T y \\ \text{subject to} & Ax \geq y, \\ & c^T x \leq R, \\ & x + z \leq 1, \\ & y_k \in \{0,1\}, \quad \forall \text{ maximal cliques k}, \\ & x_i \in \{0,1\}, \quad i \in V. \end{cases} \tag{3.8}$$

The only constraint in the outer problem is a simple budget constraint. The defender here is minimizing the maximum number of cliques interdicted ($f(z)$) and also attempting to do so by protecting as many of their own nodes as possible, hence the z term in the objective function. $f(z)$ is the optimal value attained by the inner integer program.

The first constraint in the inner integer program says that a clique cannot be interdicted unless at least one node within that clique is interdicted. The second constraint is a budget constraint, and the third constraint is the bilevel constraint that shows that a node cannot be both protected and interdicted at the same time.

This is a common form of a two-stage optimization problem. I have a minimax

problem where each player attempts to maximize their objective within the constraints of their integer program.

I relax the inner problem into a linear program and take the dual of it. The relaxed linear program for (3.8) is:

$$
f(z) = \begin{array}{lrcll}
max & 1^T y & & & \\
\text{subject to} & Ax & \geq & y, & \\
& c^T x & \leq & R, & \\
& x + z & \leq & 1, & \\
& y & \leq & 1, & \forall \text{ maximal cliques k}, \\
& y & \geq & 0, & \forall \text{ maximal cliques k}, \\
& x & \leq & 1, & i \in V, \\
& x & \geq & 0, & i \in V.
\end{array} \tag{3.9}
$$

The dual of this linear program is as follows, with the dual variables shown explicitly in Table 3.8.

$$
\begin{array}{lrcll}
min & \alpha R + 1^T b - z^T b + 1^T u + 1^T p & & & \\
\text{subject to} & \alpha c + b + u - A^T t & \geq & 0, & \\
& t_k + p_k & \geq & 1, & \forall \text{ maximal cliques k}, \\
& t_k, p_k & \geq & 0, & \forall \text{ maximal cliques k}, \\
& b_i, u_i & \geq & 0, & i \in V, \\
& \alpha & \geq & 0. &
\end{array} \tag{3.10}
$$

I linearize $z^T b$ by introducing a new variable, s, where $s_i = z_i b_i$ for $i \in V$, and by

Table 3.8 : Dual Variables for (3.9) appearing in (3.10)

Dual Variables	Primal Constraints
α	$c^T x \leq R$
b	$x + z \leq 1$
t	$Ax \geq y$
u	$x \leq 1$
p	$y \leq 1$

adding in the following constraints (since I know that z is binary valued):

$$s_i \leq z_i,$$

$$s_i \leq b_i,$$

$$s_i \geq z_i + b_i - 1,$$

$$s_i \geq 0.$$

Replacing s_i for $z_i b_i$ in (3.10):

$$
\begin{aligned}
min \qquad & \alpha R + 1^T b - 1^T s + 1^T u + 1^T p & & \\
\text{subject to} \qquad & \alpha c + b + u - A^T t \;\geq\; 0, & & \\
& t_k + p_k \;\geq\; 1, & & \forall \text{ maximal cliques k}, \\
& s_i \;\leq\; z_i, & & i \in V, \\
& s_i \;\leq\; b_i, & & i \in V, \\
& s_i \;\geq\; z_i + b_i - 1, & & i \in V, \\
& t_k, p_k \;\geq\; 0, & & \forall \text{ maximal cliques k}, \\
& b_i, u_i, s_i \;\geq\; 0, & & i \in V, \\
& \alpha \;\geq\; 0. & &
\end{aligned}
\tag{3.11}
$$

Replacing $f(z)$ in (3.7) with this gives a mixed integer program for the bilevel clique interdiction problem:

$$
\begin{aligned}
min \qquad & \alpha R + 1^T b - 1^T s + 1^T u + 1^T p - 1^T z & & \\
\text{subject to} \qquad & d^T z & \leq & Q, \\
& \alpha c + b + u - A^T t & \geq & 0, \\
& t_k + p_k & \geq & 1, \qquad \forall \text{ maximal cliques k}, \\
& s_i & \leq & z_i, \qquad i \in V, \\
& s_i & \leq & b_i, \qquad i \in V, \\
& s_i & \geq & z_i + b_i - 1, \quad i \in V, \\
& t_k, p_k & \geq & 0, \qquad \forall \text{ maximal cliques k}, \\
& b_i, u_i, s_i & \geq & 0, \qquad i \in V, \\
& \alpha & \geq & 0, \\
& z_i & \in & \{0, 1\}, \qquad i \in V.
\end{aligned}
\tag{3.12}
$$

This is the final formulation of the problem. With this formulation, I move forward into developing an algorithm to solve the problem. In the following sections, I will describe the algorithm and the motivation for the methods used in the algorithm.

3.2.2 Delayed Column-and-Row Generation Algorithm

I introduce an algorithm to solve the Bilevel Clique Interdiction Problem. In the mixed integer formulation of the problem, there is a constraint on all of the cliques in the graph. This is problematic, as the number of cliques in a graph grows exponentially with the number of nodes [18]. Accordingly, the formulation has an exponential number of these clique constraints. It is not feasible to solve a problem of such size.

Thus, the first step is to remove all of the cliques from the problem formulation. This is a relaxation of the previous mixed integer program since it removes all of the aforementioned clique constraints. However, since it also removes the clique variables, it also has restricted the problem as well. Once a solution is found to the new problem, a clique is found and the associated constraint is added back into the program to produce a tighter relaxation. These are the steps in the cutting plane process. Therefore, I remove the cliques and put the following linear relaxation formulation into Gurobi to find an initial solution. This is where the initial step is taken for the defender to maximize the number of nodes protected before the attacker comes into play.

Each time that I look to add a clique into the formulation, I find a clique whose reduced cost is positive. Therefore, I bring this clique into the solution and resolve the problem. Choosing which clique to include requires examining the dual of the linear program.

I look at the dual of the formulation to look for a way to bound the reduced cost of a clique. First, I relax the integrality constraints on the z variables from (3.12). I call this LP formulation denoted in (3.13) as BCI_{LP}.

$$
\begin{aligned}
min \qquad & \alpha R + 1^T b - 1^T s + 1^T u + 1^T p - 1^T z \\
\text{subject to} \qquad & d^T z \leq Q, \\
& \alpha c + b + u - A^T t \geq 0, \\
& t_k + p_k \geq 1, \qquad \forall \text{ maximal cliques k}, \\
& s_i \leq z_i, \qquad i \in V, \\
& s_i \leq b_i, \qquad i \in V, \\
& s_i \geq z_i + b_i - 1, \quad i \in V, \\
& t_k, p_k \geq 0, \qquad \forall \text{ maximal cliques k}, \\
& b_i, u_i, s_i \geq 0, \qquad i \in V, \\
& \alpha \geq 0, \\
& z \geq 0, \qquad i \in V, \\
& z \leq 1, \qquad i \in V.
\end{aligned}
\tag{3.13}
$$

Now, I take the dual of (3.13), where the dual variables are shown in Table 3.9:

$$
\begin{aligned}
min \qquad & \phi Q + 1^T\omega - 1^T\sigma + 1^T\chi \\
\text{subject to} \qquad & d\phi_i - \lambda_i - \sigma_i + \chi_i \leq -1, \quad i \in V, \\
& c^T\psi \leq R, \\
& \psi_i - \upsilon_i - \sigma_i \leq 1, \quad i \in V, \\
& \psi_i \leq 1, \quad i \in V, \\
& \lambda_i + \upsilon_i + \sigma_i \leq -1, \quad i \in V, \\
& \omega - A\psi \leq 0, \\
& \omega_k \leq 1, \quad \forall \text{ maximal cliques k}, \\
& \omega_k \geq 0, \quad \forall \text{ maximal cliques k}, \\
& \chi, \phi, \lambda, \upsilon \leq 0, \quad i \in V, \\
& \psi, \sigma \geq 0, \quad i \in V,
\end{aligned}
\tag{3.14}
$$

I have removed all cliques from the formulation, so I have removed the variables over cliques, as well as the clique constraint containing these two variables:

$$t_k, \quad p_k,$$

$$t_k + p_k \geq 1$$

I call this new formulation $rBCI_{LP}$, for the restricted LP. Since I have removed rows and columns, I need to add in both variables and constraints back into the problem. This is an example of delayed column-and-row generation. This is different than standard delayed column generation, since I have to contend with the addition of rows as well. Since both constraints and variables are being added, then I have to make sure the optimality conditions are clear with respect to the primal and dual

Table 3.9 : Dual Variables for (3.13) appearing in (3.14)

Dual Variables	Primal Constraints
ϕ	$d^T z \leq Q$
ψ	$\alpha c + b + u - A^T t \geq 0$
ω	$t_k + p_k \geq 1$
λ	$s_i \leq z_i$
υ	$s_i \leq b_i$
σ	$s_i \geq z_i + b_i - 1$
χ	$z \leq 1$

problems, similar to that of Avella et. al [5]. Muter et. al describe a general procedure for solving large scale problems with column-dependent rows by using a column-and-row generation algorithm [56], but I will follow the technique of Avella et. al [5].

Let $\hat{t_k}(k \in K)$ and $\hat{p_k}(k \in K)$ be the reduced costs for the variables t and p respectively. Then

$$\hat{t} = A\psi - \omega,$$

$$\hat{p} = \omega - 1.$$

I need to define some notation before I get to the theorem. First, I let A_k be the row of A corresponding to clique k. I let $\bar{K}$ be the set of cliques that have been added into the formulation, so $K \backslash \bar{K}$ are the remaining candidate cliques that I check to see if they need to be added into the formulation at each step. Lastly,

let $(\alpha, b, u, s, \bar{t}, \bar{p})$ be an optimal solution for $rBCI_{LP}$ and let $(\alpha, b, u, s, t^A, p^A)$ be its extension to BCI_{LP} where $t^A = (\bar{t}, 0)$ and $p^A = (\bar{p}, 0)$. I will define sufficient conditions to show where $(\alpha, b, u, s, t^A, p^A)$ is an optimal solution of BCI_{LP}. When these conditions are satisfied, $rBCI_{LP}$ is "optimal" for BCI_{LP}.

Theorem 3.4 $rBCI_{LP}$ *is "optimal" for* BCI_{LP} *if* $A_k\psi = 1$ *for every clique* $k \in K\backslash\bar{K}$.

Proof 3.4 Let $(\phi, \psi, \bar{\omega}, \lambda, \upsilon, \sigma, \chi)$ *be the optimal dual solution for* $rBCI_{LP}$. *I extend this solution to* $(\phi, \psi, \omega, \lambda, \upsilon, \sigma, \chi)$ *by defining* ω *as follows:*

$$\omega = \begin{cases} \omega_k = \bar{\omega_k}, k \in \bar{K} \\ \\ \omega_k = A_k\psi = \omega_k = 1, k \in K\backslash\bar{K}, \end{cases}$$

where $\bar{K} \subseteq K$ *is the set of cliques already added into the formulation, has the same objective value as the primal solution* (α, b, u, s, t, p). *It remains to check that this solution is dual feasible. If* $(\phi, \psi, \omega, \lambda, \upsilon, \sigma, \chi)$ *is dual feasible, then* (α, b, u, s, t, p) *is optimal for* BCI_{LP}. *Thus, the current solution for* $rBCI_{LP}$ *is optimal for the original problem* BCI_{LP}.

I know that the solution is dual feasible for the cliques $\bar{K}$ *that have already been added into the formulation. Therefore, I just need to check dual feasibility for the remaining* $K\backslash\bar{K}$ *cliques.*

I first check dual feasibility for $\omega_k = A_k\psi$. *From* (3.14)*:*

$$\omega_k - A_k\psi = A_k\psi - A_k\psi = 0 \leq 0,$$

$$\omega_k = A_k\psi \leq 1.$$

Next, I check dual feasibility for $\omega_k = 1$. *From* (3.14)*:*

$$\omega_k - A_k\psi = 1 - A_k\psi \leq 0,$$

$$\omega_k = 1 \leq 1.$$

From these, $A_k\psi \leq 1$ *and* $A_k\psi \geq 1$ *for* $k \in K\backslash\bar{K}$. *Therefore, the solution is dual feasible if and only if* $A_k\psi = 1$ *for all* $k \in K\backslash\bar{K}$. *Thus, the theorem is proved.*

I search for cliques k such that $A_k\psi \neq 1$, to bring into the formulation. If such a clique is found, then I add back in the t_k and w_k variables and the clique constraint $t_k + p_k \geq 1$. I use enumeration of all of the maximal cliques in the graph to find the cuts. This is done by using the Bron & Kerbosch recursive algorithm [18]. I add cliques in waves, by adding every clique such that $\sum_{i \in I_k} \psi_i \neq 1$. Therefore, I throw in a lot of cliques to begin, but still make sure that I only add as many are necessary for the formulation to reach optimality.

Lastly, I branch on the z variables to ensure that I retain integrality of these binary variables. I tested the code with strong branching, which chooses a node to branch on that will benefit the objective function the most, and most infeasible branching, which chooses a node whose value is closest to 0.5. I found that most infeasible branching outperformed strong branching with respect to time, so that is the branching decision that is used for the results shown in the next section.

3.2.3 Numerical Results

The algorithm is written in C++, with calls to GUROBI from the C++ interface. The numerical experiments were run on a Linux Desktop Computer with a 3:10 GHz Intel Xeon Processor E3-1220 v2. For the random graphs, I ran 100 instances and averaged the time in seconds taken for each run, using Bernoulli random graphs generated with a given edge density probability. The cost of both interdiction and protection of a given node was set to be a random rational number between 0 and $|V|$. For the dark networks, I used a data set from Everton [31]. For the planar graphs, I used a group of graphs created from Delaunay triangulations.

Random Graphs

I ran 100 instances and averaged the time in seconds taken for each run, using Bernoulli random graphs generated with a given edge density probability. Table 3.10 shows the results on graphs of 50 and 100 nodes. Table 3.11 shows the results on sparse graphs up to 600 nodes.

Table 3.10 shows that I can go up to an edge density of 0.7 for random graphs with 50 nodes before I go over the hour time limit, while I can only reach 0.4 for random graphs with 100 nodes before the time limit is reached. The real world applications for this problem generally would be used on sparse graphs, however, so these results were anticipated. Thus, I ran my algorithm on sparse random graphs and Table 3.11 shows that I can reach at least 500 nodes for an edge density of 0.1 within one hour.

Since I am adding in the cliques to the formulation, I wanted to see just how many cliques I was having to add back into the problem. Based on the time results, I am still having exponential growth with respect to the nodes in the graph, so I was not surprised to find that I have to add in almost all of the cliques to get the optimal solution. For 50 nodes and edge density 0.7, I am able to cut down the average number of cliques by 16 cliques out of 5506 cliques.

Table 3.10 : Numerical Results for Clique Interdiction on Random Graphs (Averaged)

Nodes	Edge Density	Seconds	Total Cliques	Cliques Added	Solution
50	0.1	0.98	93.9	93.5	19.35
50	0.2	1.90	158.3	157.7	86.96
50	0.3	3.74	274.6	273.0	205.52
50	0.4	7.76	493.9	493.5	428.70
50	0.5	20.75	950.5	950.0	887.60
50	0.6	82.88	2060.0	2032.1	1971.35
50	0.7	552.75	5506.9	5488.1	5030.40
50	0.8	>3600	N/A	N/A	N/A
50	0.9	>3600	N/A	N/A	N/A
100	0.1	5.53	338.0	337.0	188.65
100	0.2	17.45	852.5	846.5	709.07
100	0.3	68.78	1944.6	1939.4	1806.14
100	0.4	457.77	5250.2	5243.1	5115.95
100	0.5	>3600	N/A	N/A	N/A

Table 3.11 : Numerical Results on Sparse Random Graphs for Clique Interdiction (Averaged)

Nodes	Edge Density	Seconds	Total Cliques	Cliques Added	Solution
100	0.1	5.53	338.0	337.0	188.65
200	0.1	45.73	1413.6	1412.9	1118.66
300	0.1	266.76	3900.0	3893.7	3462.39
400	0.1	1085.23	8355.5	8337.4	7778.40
500	0.1	3312.00	15216.5	15188.3	14502.60
600	0.1	>3600	N/A	N/A	N/A

Dark Network

Table 3.12 shows the results on a specific dark network taken from Everton [31]. Dark networks are social networks who attempt to conceal their members and organizational strategies. In general, dark networks have 0.08 as their edge density, so they are very sparse graphs. For these terrorist networks consisting of 79 vertices and $11 - 623$ edges, the algorithm solved to optimality within 2 seconds. This is what I expected to see with real world data, because social networks are generally sparse. I also see that, in most cases, I have to add in all of the maximal cliques. However, this is not a problem due to the size of the graph. These sparse, small graphs lead to linear programming formulations that are easy to solve, even with adding in all of the maximal cliques.

In Table 3.12, rows $1-4$ are the "Trust" relationships. Row 5 aggregates all of the edges for these relationships. Rows $6-9$ are the "Operational" relationships and row 10 aggregates all of the edges for these specific relationships. Lastly, row 13 aggregates all of the edges in "Trust," "Operational," and the last two rows (Communication, Business & Financial) as well.

Table 3.12 : Numerical Results on Dark Networks

Name	Edges	Seconds	Total Cliques	Cliques Added	Solution
Classmates	175	0.88	61	61	-65
Friendship	90	1.07	78	75	-42
Kinship	16	0.95	67	67	-77
Soulmates	11	1.04	74	74	-77
Agg. Trust	258	1.24	87	85	-22
Logistics	29	1.02	72	72	-74
Meetings	64	0.97	68	68	-69
Operations	268	0.73	50	50	-71
Training	147	0.72	52	52	-72
Agg. Operational	438	0.64	48	44	-53
Communication	200	1.37	95	94	-9
Business & Financial	15	1.00	71	71	-78
Overall Aggregate	623	2.11	139	139	50

Planar Graphs

The results on planar graphs show the efficacy of the algorithm on graphs with a fewer number of cliques. These planar graphs have a bounded number of maximal cliques [62], so Tables 3.13, 3.14, 3.15, and 3.16 show that the algorithm still grows exponentially in time with respect to number of nodes on planar graphs, but grows much slower than for random graphs. An instance of a planar graph with 2400 vertices can still be solved in under an hour, while such a planar graph with 150 vertices can be solved in under 5 seconds.

Similar to the random graphs, I see that I have to add in numerous, but not all, maximal cliques in order to reach the optimal solutions. The largest instance has 2392 vertices and 4677 maximal cliques. I had to add in 4676 maximal cliques to reach the optimal solution. However, I am still able to solve all of these instances in under an hour with this algorithm.

Figure 3.2 shows the comparative seconds to solve bilevel clique interdiction on random graphs, sparse random graphs, and planar graphs.

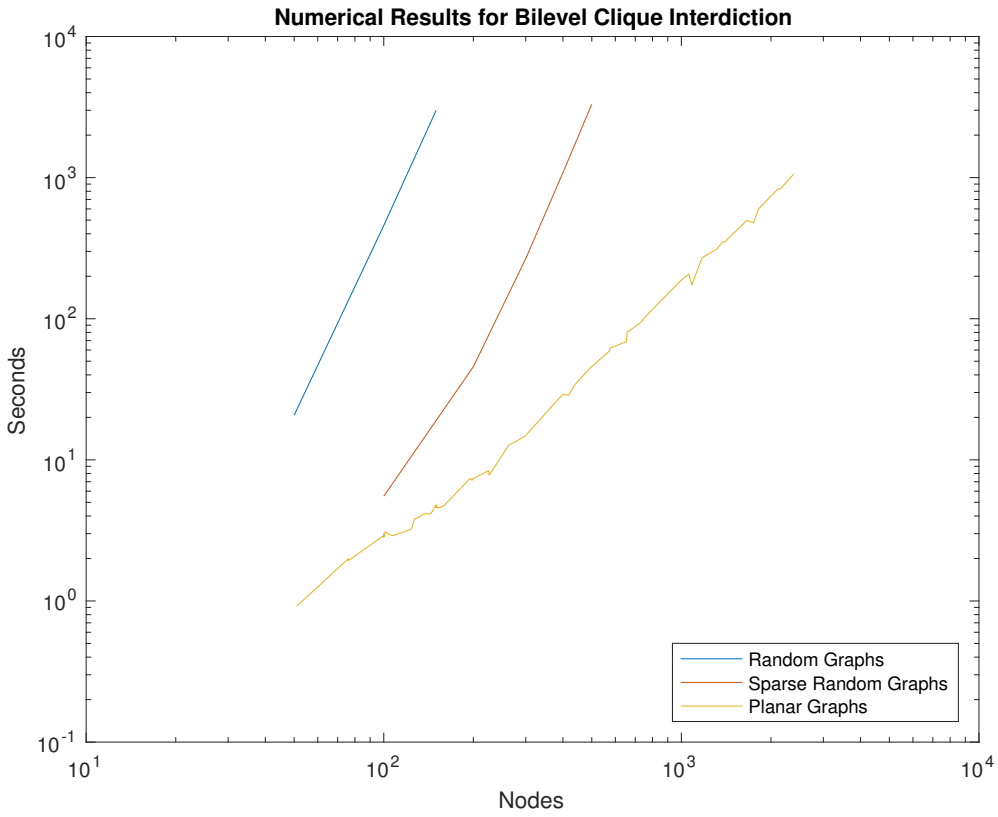


Figure 3.2 : Numerical Results for Bilevel Clique Interdiction

Table 3.13 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
a280	13.61	507	504	134.67
bier127	3.83	236	235	66.67
ch130	3.89	240	238	66.00
ch150	4.79	277	274	75.00
d1291	304.81	2555	2553	832.50
d1655	496.99	3236	3236	1030.00
d198	7.18	372	370	107.08
d2103	826.03	4182	4181	1377.67
d493	44.60	965	965	309.33
d657	80.43	1290	1288	414.00
eil101	3.09	190	190	55.33
eil51	0.92	90	88	20.00
eil76	1.99	140	138	36.67
fl1400	349.68	2719	2708	844.67
fl1577	446.79	3052	3045	956.80

Table 3.14 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
fl417	28.67	753	750	195.67
gil262	12.55	504	504	156.00
kroA100	2.83	180	178	45.67
kroA150	4.59	273	272	73.67
kroA200	7.42	383	382	116.00
kroB100	2.86	185	182	48.67
kroB150	4.78	281	281	82.00
kroB200	7.38	377	376	110.00
kroC100	2.98	181	181	48.67
kroE100	2.90	176	176	44.00
lin105	2.94	184	182	43.67
nrw1379	352.26	2717	2715	879.67
p654	68.88	1151	1151	280.64
pcb1173	270.57	2314	2314	752.33
pcb442	34.70	843	840	252.02

Table 3.15 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
pr1002	188.98	1971	1970	634.00
pr107	2.90	177	176	34.00
pr124	3.23	195	195	37.25
pr136	4.12	242	242	60.67
pr144	4.16	245	244	57.75
pr152	4.56	270	268	66.67
pr226	7.78	357	357	80.00
pr2392	1067.24	4677	4676	1496.33
pr264	12.78	509	509	157.00
pr299	14.80	551	542	146.00
pr439	34.14	851	847	263.00
pr76	1.93	139	139	38.33
rat195	7.39	364	364	104.67
rat575	61.76	1121	1120	354.00
rat783	111.33	1532	1529	486.33
rat99	2.87	179	179	47.33

Table 3.16 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
rd100	2.85	182	182	49.33
rd400	29.22	780	779	246.33
rl1304	307.89	2545	2541	816.41
rl1323	316.93	2599	2591	844.05
rl1889	652.68	3682	3667	1185.01
tsp225	8.39	398	397	97.42
u1060	207.30	2082	2081	669.67
u1432	368.42	2773	2769	859.67
u159	4.71	269	269	62.24
u1817	601.13	3570	3556	1133.67
u2152	834.68	4157	4151	1283.33
u2319	992.36	4551	4542	1450.00
u574	59.28	1118	1118	355.33
u724	92.76	1376	1373	411.00
vm1084	173.22	1778	1777	398.86
vm1748	478.95	3015	3008	813.67

3.2.4 Conclusion

I have described an integer program for the Bilevel Clique Interdiction Problem, which contains an interdictor attempting to maximize the number of cliques interdicted, and a defender who attempts to minimize that maximum number of cliques interdicted. The problem has been shown to be NP-hard and I have created a delayed column-and-row generation algorithm to solve the problem.

Numerical results show that the algorithm performs well on sparse random graphs at least up to 500 nodes for clique interdiction. For denser random graphs, the algorithm is solvable in under an hour up to $50 - 100$ nodes. Additionally, this algorithm performs well up to, and beyond, 2400 nodes for planar graphs in terms of elapsed time. This was to be expected as the subroutine of finding all maximal cliques dominated the runtime and this subroutine is exponential with respect to the nodes in the graph.

Chapter 4

Deterministic and Stochastic Bilevel Interdiction of a Maximum Stable Set

4.1 Deterministic Interdiction

I introduce the problem of interdicting a maximum stable set and a defender who attempts to minimize the size of the maximum stable set that is interdicted. The defender has a given budget to work under, while the interdictor is only burdened by the defender's efforts.

The problem of bilevel interdiction of a maximum stable set is: given a budget, R, for the defender, protection costs d_i on every node i, and a node goal, B, find a set, S, of nodes for the defender to protect such that $\sum_{i \in S} d_i \leq R$ and the cardinality of the maximum stable set in $V \backslash S$ is less than B. The interdictor will choose the maximum stable set possible in the graph among the nodes that are unprotected.

4.1.1 Complexity

Theorem 4.1 The problem of bilevel interdiction of a maximum stable set is strongly NP-hard.

The decision problem for finding a stable set is as follows: Given a graph $G =$

(V, E), does there exist a subset $S \subseteq V$ having k nodes, such that no pair of nodes in S are neighbors? This decision problem is NP-complete in the strong sense [68]. Therefore, the problem of finding a maximum stable set is strongly NP-hard. Since this is a specific case of bilevel interdiction of a maximum stable set (where the defender has no budget to protect nodes), then bilevel interdiction of a maximum stable set is strongly NP-hard.

4.1.2 Integer Program Formulation

I define d to be node weights, and R to be the pre-determined budget for the defender. Additionally, x, z is the binary vector over nodes for deciding which nodes to target/protect, respectively. Thus,

$$x_i = \begin{cases} 1 & \text{if node i is interdicted} \\ 0 & \text{otherwise.} \end{cases}$$

$$z_i = \begin{cases} 1 & \text{if node i is protected} \\ 0 & \text{otherwise.} \end{cases}$$

Define A to be the clique incidence matrix with

$$a_{ij} = \begin{cases} 1 & \text{if node j is in clique i} \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, I_k is the index set of all nodes i within a given clique $k \in K$. After the variables have been defined, I have the following two stage integer program formulation

of the bilevel maximum stable set interdiction problem.

$$\begin{array}{lll} min & f(z) - 1^T z & \\ \text{subject to} & d^T z \leq R, & (4.1) \\ & z_i \in \{0,1\}, \quad i \in V. & \end{array}$$

The only constraint is a simple budget constraint. The defender here is minimizing the maximum stable set interdicted ($f(z)$) and also attempting to do so by protecting as many of their own nodes as possible, hence the z term in the objective function. $f(z)$ is the optimal value attained by the following integer program.

$$f(z) = \left\{ \begin{array}{lll} max & 1^T x & \\ \text{subject to} & Ax \leq 1, & \\ & x + z \leq 1, & \\ & x_i \in \{0,1\}, \quad i \in V. & \end{array} \right. \qquad (4.2)$$

The first constraint says that a clique cannot have more than one node interdicted. The second constraint is the bilevel constraint that shows that a node cannot be both protected and interdicted at the same time.

This is a common form of a two-stage optimization problem. I have a minimax problem where each player attempts to maximize their objective within the constraints of their integer program.

One common method of dealing with bilevel integer programs is to take the linear relaxation of the inner problem and take the dual of it. The relaxed linear program for (4.2) follows, where I drop the $x \leq 1$ constraint because it is redundant:

Table 4.1 : Dual Variables for (4.3) appearing in (4.4)

Dual Variables	Primal Constraints
b	$x + z \leq 1$
a	$Ax \leq 1$

$$f(z) = \begin{array}{lrcl} max & 1^T x & & \\ \text{subject to} & Ax & \leq & 1, \\ & x + z & \leq & 1, \\ & x & \geq & 0, \quad i \in V. \end{array} \tag{4.3}$$

The dual of this linear program is as follows, with the dual variables shown explicity in Table (4.1).

$$\begin{array}{lrcl} min & 1^T b - z^T b + 1^T a & & \\ \text{subject to} & b + A^T a & \geq & 1, \\ & a_k & \geq & 0, \quad \forall \text{ maximal cliques k}, \\ & b_i & \geq & 0, \quad i \in V. \end{array} \tag{4.4}$$

I linearize $z^T b$ by introducing a new variable, s, where $s_i = z_i b_i$ for $i \in V$, and by adding in the following constraints (since I know that z is binary valued):

$$s_i \leq z_i,$$

$$s_i \leq b_i,$$

$$s_i \geq z_i + b_i - 1,$$

$$s_i \geq 0.$$

Replacing s_i for $z_i b_i$ in (4.4), I have:

$$
\begin{array}{lrcll}
min & 1^T b - 1^T s + 1^T a & & & \\
\text{subject to} & b + A^T a & \geq & 1, & \\
& s_i & \leq & z_i, & i \in V, \\
& s_i & \leq & b_i, & i \in V, \\
& s_i & \geq & z_i + b_i - 1, & i \in V, \\
& a_k & \geq & 0, & \forall \text{ maximal cliques k}, \\
& b_i, s_i & \geq & 0, & i \in V.
\end{array}
\tag{4.5}
$$

Replacing $f(z)$ in (4.1) with this gives a mixed integer program for the bilevel maximum stable set interdiction problem:

$$
\begin{array}{lrcll}
min & 1^T b - 1^T s + 1^T a - 1^T z & & & \\
\text{subject to} & d'z & \leq & R, & \\
& b + A^T a & \geq & 1, & \\
& s_i & \leq & z_i, & i \in V, \\
& s_i & \leq & b_i, & i \in V, \\
& s_i & \geq & z_i + b_i - 1, & i \in V, \\
& a_k & \geq & 0, & \forall \text{ maximal cliques k}, \\
& b_i, s_i & \geq & 0, & i \in V, \\
& z_i & \in & \{0, 1\}, & i \in V.
\end{array}
\tag{4.6}
$$

4.1.3 Column Generation

First, I relax the linearity constraint on the z variables from (4.6).

$$
\begin{aligned}
min \qquad & 1^T b - 1^T s + 1^T a - 1^T z \\
\text{subject to} \qquad & d'z \leq R, \\
& b + A^T a \geq 1, \\
& s_i \leq z_i, \qquad i \in V, \\
& s_i \leq b_i, \qquad i \in V, \\
& s_i \geq z_i + b_i - 1, \quad i \in V, \\
& a_k \geq 0, \qquad \forall \text{ maximal cliques k}, \\
& b_i, s_i \geq 0, \qquad i \in V, \\
& z \geq 0, \qquad i \in V, \\
& z \leq 1, \qquad i \in V.
\end{aligned} \tag{4.7}
$$

I remove all cliques from the formulation and then add them back in individually. To determine which cliques to add into the formulation, I look for cliques that break one of the clique constraints in the dual. These cliques are breaking dual feasibility, so I know that they affect primal optimality and further optimize the objective function if they are added into the formulation.

I take t to be the dual variable associated with the constraint $b + A^T a \geq 1$. Then, I look for cliques k such that $A_k t > 1$. If I find such a clique, k, then I add in a_k as a new column into the primal.

To find such cliques, I, again, enumerate all of the maximal cliques in the graph using Bron & Kerbosch [18]. I add in every clique that satisfies this constraint into the problem, so the columns are added in waves. Therefore, around 90% of the columns

are added in initially so we cut down on the computational time of enumerating over all of the maximal cliques and only adding in one clique at each time.

4.1.4 Computational Results

This algorithm is written in C++, with calls to GUROBI from the C++ interface. The experiments were run on a Linux Desktop Computer with a 3:10 GHz Intel Xeon Processor E3-1220 v2. With respect to the random graphs, I ran 100 instances and averaged the time in seconds taken for each run, using Bernoulli random graphs generated with a given edge density probability. I let the interdiction cost and the protection cost for any given node be a random rational number between 0 and the number of nodes in the graph. The dark networks come from Everton, and the planar graphs are Delaunay triangulations [31].

Random Graphs

Table 4.2 shows the results of the algorithm on random graphs of 50 and 100 nodes, with edge densities varying from 0.1 to 0.9. If the algorithm did not finish in under one hour, then there is "N/A" in all of the columns. Table 4.3 shows the results on sparse random graphs (i.e. having edge density of 0.1) for 100 to 600 nodes.

From Table 4.2, the algorithm runs in under an hour for 50 nodes up to, and including, an edge density of 0.7. For 100 nodes, however, the algorithm runs in under an hour only up to an edge density of 0.5. However, for sparse graphs (again, those here with edge density of 0.1), Table 4.3 shows that the algorithm runs in under

an hour for up to 500 nodes. Since social networks are generally sparse, this algorithm is able to solve problems on social networks up to around 500 nodes, even if it is not able to solver denser graphs with that many nodes. Figures **??** and **??** show the runtime of the algorithm plotted vs the number of nodes

I notice that I have to add in almost all of the cliques into the problem to reach the optimal solution. For every instance, I had to add in at least 90% of the total cliques in the graph. For large graphs though, this leads to a larger decrease in the size of the linear program. The algorithm is still exponential in runtime with respect to the nodes in the graph, so I do not gain any complexity results because I still have to add in so many cliques, but for every instance I have reduced the overall number of cliques necessary to add into the formulation to reach the optimal solution.

Table 4.2 : Numerical Results for Stable Set (Averaged)

Nodes	Edge Density	Seconds	Total Cliques	Cliques Added	Solution
50	0.1	1.94	96.5	87.1	9.11
50	0.2	3.30	158.6	145.7	4.43
50	0.3	4.56	275.3	248.6	1.48
50	0.4	8.17	505.1	450.6	0.06
50	0.5	19.30	993.6	943.6	-2.01
50	0.6	82.35	2292.9	2244.7	-3.67
50	0.7	449.95	5313.9	5227.9	-5.26
100	0.1	9.56	332.8	303.3	18.10
100	0.2	20.82	853.0	793.2	8.65
100	0.3	63.69	2036.0	1947.7	5.41
100	0.4	326.96	5268.2	5122.8	0.95
100	0.5	3066.60	15988.4	15814.6	-3.43

Table 4.3 : Numerical Results on Sparse Random Graphs for Stable Set

Nodes	Edge Density	Seconds	Total Cliques	Cliques Added	Solution
100	0.1	9.56	332.8	303.3	18.10
200	0.1	57.46	1405.3	1317.38	37.23
300	0.1	232.72	3831.6	3584.7	51.45
400	0.1	919.66	8347.9	8136.2	70.11
500	0.1	2410.30	14958.6	14658.1	84.66
600	0.1	>3600	N/A	N/A	N/A

Dark Networks

A dark network is an example of a social network whose members attempt to conceal their identity and the structure and plans of the network. As specific social networks, dark networks are generally very sparse graphs. Table 4.4 shows the results of running the algorithm multiple times on a specific dark network [31]. This terrorist network has 79 vertices and each row depicts a different network between them based on the relationship described in the title, ranging from containing 11 edges up to 623 edges. Each one could be solved in under 5 seconds on average. Additionally, I do not need to add in as many cliques to reach the optimal solution. The percentage of maximal cliques necessary is smaller for these dark networks than for random graphs or planar graphs. This could be based on the number of singletons in some of these graphs. It might not necessary to add in those singletons as maximal cliques, depending on their costs, to reach the optimal solution.

In Table 4.4, the"Trust" relationships are outlined in rows $1-4$. Row 5 aggregates all of the edges for these relationships into one network. The "Operational" relationships are shown in rows $6-9$ and row 10 aggregates all of the edges for these relationships into one network. Rows 11 and 12 are individual networks, based on "Communication" and "Business & Financial" relationships. Row 13 combines all of the previous edges in the networks into one larger network depicting all of the relationships between members of the terrorist network.

Table 4.4 : Numerical Results on Dark Networks

Name	Edges	Seconds	Total Cliques	Cliques Added	Solution
Classmates	175	3.25	61	40.1	31.4
Friendship	90	3.58	78	58.6	31.6
Kinship	16	3.00	67	42.9	45.5
Soulmates	11	3.11	74	39.7	49.0
Agg. Trust	258	3.63	87	64.5	18.9
Logistics	29	3.28	72	40.0	47.6
Meetings	64	3.01	68	40.0	42.1
Operations	268	3.25	50	39.1	25.7
Training	147	3.20	52	39.0	29.8
Agg. Operational	438	3.15	48	39.8	11.5
Communication	200	3.82	95	76.2	22.6
Business & Financial	15	3.06	71	39.0	45.9
Overall Aggregate	623	4.78	139	110.8	7.5

Planar Graphs

The results on planar graphs demonstrate the algorithm's effectiveness on larger graphs with a bounded number of maximal cliques [62]. The runtime still grows exponentially with respect to the number of nodes in the graph, but it grows at a much slower rate than the runtime for random graphs. I can solve an instance of a planar graph with 2400 nodes in under an hour, while I cannot get above 600 nodes for even a sparse random graph.

Tables 4.5, 4.6, 4.7, and 4.8 show the results for a variety of planar graphs. I have to add in almost every single clique to solve the problem on these graphs. However, I am able to do so efficiently due to the reduced number of cliques in these graphs compared to random graphs.

Table 4.5 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
a280	36.08	507	507	64.52
bier127	10.22	236	233	22.00
ch130	10.05	240	240	24.67
ch150	13.74	277	274	32.50
d1291	931.27	2555	2555	361.50
d1655	1703.45	3236	3236	477.00
d198	19.02	372	368	43.00
d2103	3002.12	4182	4182	616.67
d493	130.70	965	965	120.45
d657	209.34	1290	1290	169.33
eil101	7.63	190	190	14.00
eil51	2.16	90	90	4.00
eil76	4.83	140	135	10.00
fl1400	1258.06	2719	2719	404.55
fl1577	1213.49	3052	3052	464.08

Table 4.6 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
fl417	79.59	753	753	104.30
gil262	34.24	504	504	59.33
kroA100	6.78	180	170	11.50
kroA150	11.96	273	264	31.00
kroA200	22.23	383	383	39.31
kroB100	6.66	185	179	13.00
kroB150	12.35	281	278	30.00
kroB200	20.61	377	377	43.00
kroC100	6.86	181	174	14.00
kroE100	9.74	176	175	19.00
lin105	7.45	184	183	18.00
nrw1379	1172.65	2717	2717	387.10
p654	193.88	1151	1151	180.50
pcb1173	726.83	2314	2314	333.79
pcb442	96.76	843	843	111.31

Table 4.7 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
pr1002	536.06	1971	1971	276.29
pr107	7.67	177	176	22.00
pr124	8.99	195	189	21.00
pr136	10.56	242	240	24.00
pr144	10.87	245	236	32.00
pr152	12.34	270	265	30.00
pr226	21.46	357	356	60.00
pr2392	3510.34	4677	4677	709.90
pr264	34.78	509	509	59.69
pr299	42.16	551	549	71.33
pr439	104.29	851	851	104.15
pr76	4.93	139	136	8.00
rat195	19.80	364	364	39.03
rat575	160.79	1121	1121	151.67
rat783	305.36	1532	1532	206.17
rat99	7.42	179	178	12.00

Table 4.8 : Numerical Results on Planar Graphs

Name	Seconds	Total Cliques	Cliques Added	Solution
rd100	7.86	182	182	20.00
rd400	109.86	780	780	100.66
rl1304	801.15	2545	2545	372.19
rl1323	833.79	2599	2599	382.21
rl1889	2913.14	3682	3682	583.67
tsp225	25.61	398	398	49.58
u1060	771.67	2082	2082	293.92
u1432	1154.56	2773	2773	398.67
u159	12.93	269	269	30.50
u1817	2408.10	3570	3570	530.75
u2152	3266.87	4157	4157	627.67
u2319	3424.91	4551	4551	672.33
u574	165.94	1118	1118	142.67
u724	255.16	1376	1376	188.86
vm1084	508.34	1778	1776	342.02
vm1748	1987.34	3015	3015	580.74

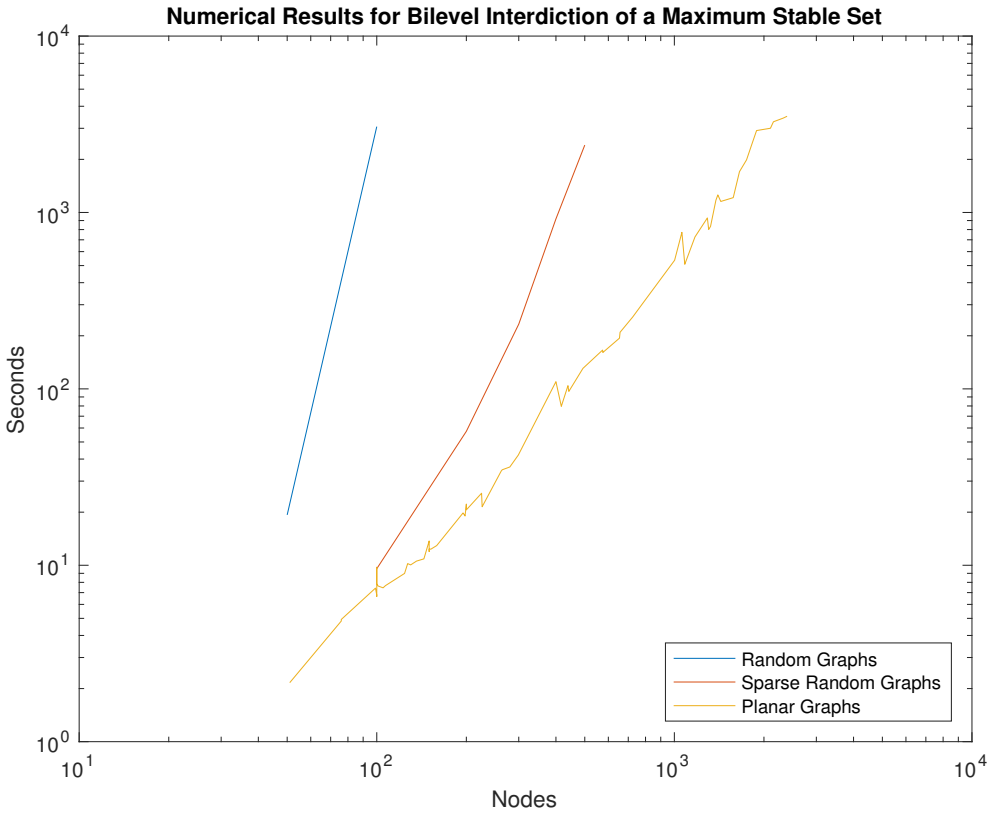


Figure 4.1 : Numerical Results for Bilevel Interdiction of a Maximum Stable Set

Figure 4.1 shows the nodes plotted against the runtime of the algorithm for random graphs, sparse random graphs, and planar graphs.

The algorithm runs in exponential time, but is solvable on sparse graphs up to 100 nodes within a couple of minutes. Dark networks in particular have an average edge density probability around 8%, so the algorithm can solve instances on such networks for > 100 nodes in a few minutes. In general, social networks are sparse as well, so it

is an algorithm that is exponential in runtime, but it is very practical for moderate sized social networks. In the next section, I will discuss a stochastic extension of the same problem.

4.2 Stochastic Interdiction

I introduce a stochastic variable into the defender's protection to mimic the realistic scenario of not knowing whether the protection of a given node is completely successful or not.

Here, the interdictor chooses a maximum stable set to remove, while the defender is given a budget, R, and a probability vector ω, where ω_i is the probability of successfully protecting node i from interdiction. Thus, the problem is: Given a budget, R, the defender chooses a set of nodes, S, to protect such that $\sum_{i \in S} d_i \leq R$ and minimizes the size of the expected maximum stable set, $Q(\hat{z})$, that the interdictor can remove by choosing nodes to attempt to protect. In the following formulation, $\hat{z}_i = 1$ with probability ω_i if node i is chosen for protection.

The following is the bilevel stochastic formulation of the problem.

4.2.1 Integer Program Formulation

$$
\begin{array}{lll}
min & \mathbf{E}_\omega Q(\hat{z}) - 1^T z & \\
\text{subject to} & d^T z \leq R, & (4.8) \\
& z_i \in \{0,1\}, \quad i \in V. &
\end{array}
$$

$$Q(\hat{z}) = \begin{cases} min & -1^T x \\ \text{subject to} & Ax \leq 1, \\ & x \leq 1 - \hat{z}, \\ & x_i \in \{0,1\}, \quad i \in V, \end{cases} \tag{4.9}$$

Let A be the clique matrix, with cliques as rows and nodes as columns, where $a_{ij} = 1$ if node j is in clique i. I restrict my attention to perfect graphs because, for perfect graphs and for A the clique matrix, the convex hull of of $\{x \in \{0,1\}^n | Ax \leq 1\}$ is just $\{x \in \mathbf{R}^n | Ax \leq 1, 0 \leq x \leq 1\}$. This is because the clique matrix of a perfect graph is perfect, so A is a perfect matrix [23].

Thus, I rewrite the previous formulation to relax the binary constraints on the x variables in the inner problem since I am focusing on perfect graphs:

$$\begin{array}{lll} min & \mathbf{E}_\omega Q(\hat{z}) - 1^T z & \\ \text{subject to} & d^T z \leq R, & \\ & z_i \in \{0,1\}, \quad i \in V. & \end{array} \tag{4.10}$$

$$Q(\hat{z}) = \begin{cases} max & 1^T x \\ \text{subject to} & Ax \leq 1, \\ & x \leq 1 - \hat{z}, \\ & x_i \geq 0. \end{cases} \tag{4.11}$$

4.2.2 Algorithm

I use the L-Shaped Method [74], but also have to account for the exponentially many constraints in the lower problem. Therefore, I use a method that will switch between adding L-Shaped cuts, both feasibility and optimality, and adding clique constraints into the lower problem. This method is similar to that of cross decomposition, which is a primal-dual iterative algorithm that adds in L-Shaped cuts and Lagrangian multipliers [73] [39] [59]. However, I must make sure here that the L-Shaped cuts that I add stay valid when more cliques are added to the lower problem, since I am adding in clique constraints to the master problem after the addition of L-Shaped cuts.

To begin, I take some subset, κ, of the set of maximal cliques, K, of the graph. I put these constraints in the lower problem, but no more. Thus, I have an LP with n columns and $\kappa + 2 * n$ rows.

L-Shaped Cuts

1. *Master Problem.* Set $v = v + 1$. Solve the restricted master problem below:

$$
\begin{aligned}
min \qquad & (-1^T \hat{z})^T \lambda + \theta \\
\text{subject to} \qquad & A\lambda \leq 1, \\
& D_\ell \lambda \geq d_\ell, \\
& E_\ell \lambda + \theta \geq e_\ell, \\
& \lambda_i \geq 0.
\end{aligned} \tag{4.12}
$$

where the constraint set are the L-Shaped feasibility and optimality cuts, respectively. Let (λ_v, θ_v) denote the optimal solution.

2. *Feasibility cut generation.* Set the scenario index $k = 1$

(a) *Feasibility subproblem.* Solve the following linear program:

$$\begin{aligned} w' = min \quad & 1^T v^+ + 1^T v^- \\ \text{subject to} \quad & Wy + I^+ v^+ - I^- v^- \geq -1 - T_k \lambda_v, \\ & y \geq 0. \end{aligned} \tag{4.13}$$

where $W = -I$, and $T_k = -\omega_k I$. Let δ_k be the optimal dual for the constraint. If $w' = 0$ and $k = K$, go to step 3. If $w' = 0$ and $k < K$, let $k = k + 1$ and go to step $2a$. If $w' > 0$, go to step $2b$.

(*b*) *Feasibility cut insertion.* Calculate the cut coefficients

$$D = (\delta_k^v)^T T_k,$$

$$d = (\delta_k^v)^T (-1),$$

and add a feasibility cut to the restricted master problem. Go to step 1.

3. *Optimality cut generation.* Set the scenario index $k = 1$.

(a) *Optimality subproblem.* Solve the following linear program:

$$\begin{aligned} Q(\lambda_v, \omega_k) = min \quad & -1^T x \\ \text{subject to} \quad & Wx \geq -1 - T_k \lambda_v, \\ & x \geq 0. \end{aligned} \tag{4.14}$$

Let δ_k be the optimal dual vector for the constraint. If $Q(\lambda_v, \omega_k)$ is unbounded, then the original was unbounded too, so the algorithm terminates. If $k = K$, go to step $3b$. Otherwise, set $k = k + 1$ and go to step $3a$.

(b) *Optimality cut insertion.* Calculate the cut coefficients

$$E = \sum_{k=1}^{K} p^k (\delta_k^v)^T T_k,$$

$$e = \sum_{k=1}^{K} p^k (\delta_k^v)^T (-1).$$

If $e - E\lambda^v > \theta^v$, add the corresponding optimality cut. Else, the restricted master problem is optimal for λ.

Cutting Plane for Lower problem

The lower problem, 4.9, has exponentially many constraints, so I do not want to solve the problem with all of these initially. I wish to devise a scheme where I add in a subset of the constraints and then add in clique constraints one at a time as cuts. To this end, I wish to add in any cliques,k, such that $Ax > 1$. I enumerate all of the maximal cliques and iterate through them and add in every single clique that satisfies this constraint. Therefore, we add in waves of cliques at a time and we only have to iterate through all of the maximal cliques once.

4.2.3 Proof of Validity for L-Shaped Cuts with Subset of Lower Problem Constraints

It is important to show that the L-Shaped cuts that are found before more cliques are added into the lower problem stay valid for the restricted master problem. Thus, I can switch between adding L-Shaped cuts and adding new cliques until I reach an optimal solution.

The dual of the lower problem is:

$$\begin{aligned} min \quad & 1^T\alpha + (1-\omega\hat{z})^T\beta \\ \text{subject to} \quad & A^T\alpha + \beta \geq 1, \\ & \alpha_k \geq 0 \quad \forall k \in K, \\ & \beta_i \geq 0 \quad \forall i \in V. \end{aligned} \tag{4.15}$$

where α is the dual variable associated with the clique constraint in the primal, and β is the dual variable associated with the bilevel constraint in the primal.

If I have a subset of cliques as constraints in the primal, then I do not have as many α_k variables in the dual. Therefore, if I have an optimal dual vector $[\alpha, \beta](= \delta_k^v$ for simplicity) for some set of cliques $\tilde{K} \subset K$, then this dual vector remains feasible when more cliques are added in, since $A^T\alpha + \beta \geq 1$ will still hold when more α variables are added to the formulation, as those variables are just set to 0 and I still have a feasible dual vector that was previously optimal.

Therefore, any optimal dual vector that I find at some stage for the L-Shaped cuts will remain a feasible dual vector no matter how many cliques are added into the lower problem afterwards.

Proposition 4.1 L-Shaped optimality cuts will remain valid for the restricted master problem.

Proof 4.1 It suffices to show that the L-Shaped optimality cuts hold for any feasible dual vector, rather than specifically for an optimal dual vector. From weak duality,

$$Q(\hat{z}^v) \geq (\delta_k^v)^T(-1 - T(\omega_k)z^v).$$

By convexity of $Q(\hat{z}^v)$, I use the subgradient inequality to see

$$Q(\hat{z}) \geq (\delta_k^v)^T(-1) - (\delta_k^v)^T T(\omega_k)z.$$

I take the expectation of both sides to obtain

$$Q(z^v) \geq \mathbf{E}(\delta^v)^T(-1 - T(\omega)z^v) = \sum_{k=1}^{K} p_k(\delta_k^v)^T(-1 - T(\omega_k)z^v),$$

and thus

$$Q(z) \geq \mathbf{E}(\delta^v)^T(-1 - T(\omega)z) = \sum_{k=1}^{K} p_k(\delta_k^v)^T(-1) - \left(\sum_{k=1}^{K} p_k(\delta_k^v)^T T(\omega_k)\right) z.$$

Since $\theta \geq Q(z)$, (z, θ) is feasible only if $\theta \geq \mathbf{E}(\delta^v)^T(-1 - T(\omega)z)$. These are exactly the optimality cuts of the L-Shaped method. Therefore, all of these cuts are supporting linear functions of $Q(z)$.

Clearly, the cuts may not be as tight as they would be for an optimal dual vector, but they will remain valid even with the addition of extra cliques that make the dual vectors solely feasible rather than being optimal.

For feasibility cuts, I first emphasize that λ being feasible is to say

$$\lambda \in \{\lambda | \textit{for } k = 1, \ldots, K, \exists y \geq 0 s.t.\ Wy = -1 - \Gamma_k \lambda\}.$$

This is equivalent to saying that

$$-1 - \Gamma^k \lambda \in \textit{range}(W), \quad k = 1, \ldots, K.$$

Thus, if λ is not feasible, then there exists a separating hyperplane between *range*(W) and $-1 - \Gamma_k \lambda$. The simplex multipliers δ^v give this separating hyperplane for the

general case. From 4.13, if $w' > 0$, then $(\delta^v)^T(-1-\Gamma_k\lambda^v) > 0$ by duality. Additionally, since δ^v is an optimal simplex multiplier, then $(\delta^v)^T W \leq 0$ because the reduced costs for y must be non-negative at optimality. Therefore, $(\delta^v)^T(-1-\Gamma_k\lambda^v) \leq 0$ is necessary for any λ^v to be feasible.

For this specific problem, this proof still holds even though I do not have all of the constraints in the lower problem yet. Adding more constraints to the lower problem will just add columns to the dual. Therefore, I just consider column generation for the dual.

Proposition 4.2 $(\delta^v)^T(-1-\Gamma_k\lambda^v) \leq 0$ *is still necessary for any* λ^v *to be feasible after the addition of clique constraints.*

Proof 4.2 I know that $(\delta^v)^T(-1-\Gamma_k\lambda^v) > 0$ *and* $(\delta^v)^T W \leq 0$ *before the addition of any more cliques. If* $\tilde{K}$ *is the number of cliques added into the formulation, then call* $\hat{P}$ *the projection on to the new constraint space with an extra* $\tilde{K}$ *columns. Then,* $([\delta^v \;\; 0^{\tilde{K}}])^T(-\hat{1}-\hat{\Gamma}_k\lambda^v) > 0$ *and* $([\delta^v \;\; 0^{\tilde{K}}])^T\hat{W} \leq 0$ *is clearly true, where* $0^{\tilde{K}}$ *is just a vector of zeros of size* $\tilde{K}$ *and* $\hat{1}$, $\hat{\Gamma}$, *and* $\hat{W}$ *are the vectors and matrices with the new rows corresponding to the new cliques. The projection of the current solution to the new constraint space by setting the new dual variables equal to* 0 *will clearly keep the inequalities satisfied in the new space.*

Therefore, I know that $(\delta^v)^T(-1-\Gamma_k\lambda^v) \leq 0$ *is still a valid inequality for the master problem because the addition of more dual columns trivially corresponds to adding more zeros to the left hand side of each inequality, so the inequality stays*

valid.

Therefore, I can switch between adding L-Shaped cuts into the master problem and adding more clique constraints into the lower problem.

4.2.4 Computational Results

The cliques first algorithm finds the necessary cliques to add into the formulation at constraints by enumerating the maximal cliques as described before. After this, the L-Shaped method is used to solve the master problem. The switching algorithm begins without any clique constraints and adds L-Shaped cuts. When no more L-Shaped cuts are necessary, clique constraints are used to remove non-feasible solutions before moving back to adding more L-Shaped cuts. This process continues until no L-Shaped cuts or clique constraints are necessary to add.

First, I have the results of the algorithm where I add the clique constraints first and then the L-Shaped cuts. Table 4.9 shows the results for adding the clique constraints first for $50 - 1000$ nodes, while Table 4.10 shows the results for $1100 - 2000$ nodes. Table 4.11 shows the results for switching between the clique constraints and the L-Shaped cuts for $50 - 1000$ nodes, and Table 4.12 shows the results for $1100 - 2000$ nodes. Figure 4.2 shows the differences in runtime between the algorithms visually.

The algorithm was run on instances of chordal graphs with the given number of nodes, since chordal graphs are a subset of perfect graphs. It is clear to see that the algorithm that added the clique constraints first was significantly faster than the

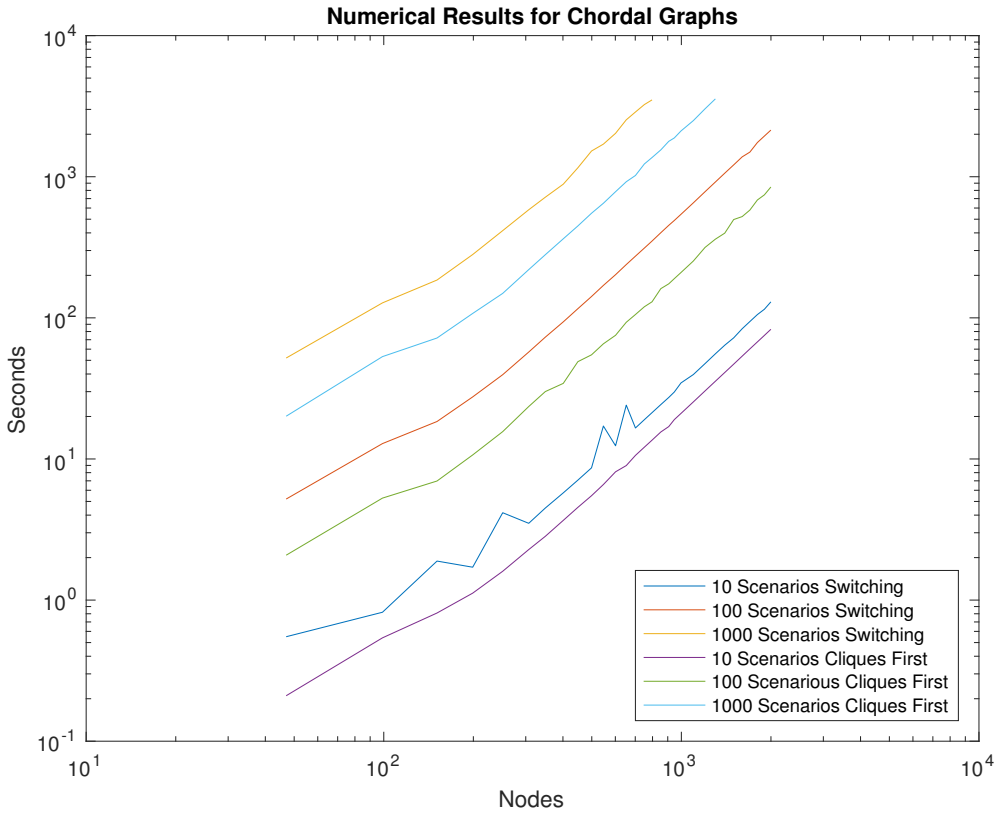


Figure 4.2 : Numerical Results on Chordal Graphs (Tables 4.9, 4.10, 4.11, and 4.12)

switching algorithm. This was not the expected result, since the switching algorithm should reduce the solution space faster to find the optimal solution. However, when adding in the clique constraints, we must consider all of the maximal cliques in the graph initially, and then iterate through the rest of the cliques that have not been added in at each step. This could be causing some of the delay when switching back and forth. Additionally, the problem could be that the L-Shaped cuts, while staying valid after the addition of clique constraints, are redundant and are not as helpful as those L-Shaped cuts that are added after all of the clique constraints in the other algorithm. The majority of the runtime of the algorithms are in the building of the model, as opposed to solving the model, so future work would be to optimize the process of finding cliques and L-Shaped cuts.

Table 4.9 : Numerical Results (Seconds) on Chordal Graphs for Cliques First and $50 - 1000$ nodes

Nodes	10 Scenarios	100 Scenarios	1000 Scenarios
47	0.21	2.03	20.09
99	0.54	5.28	53.12
151	0.81	6.98	71.98
199	1.12	10.68	107.46
251	1.60	15.62	149.04
307	2.28	23.53	219.67
349	2.83	30.03	280.12
401	3.67	34.32	363.10
449	4.54	48.92	447.90
499	5.49	54.70	551.12
547	6.59	65.47	648.58
601	8.11	75.23	783.22
653	8.98	93.05	921.57
701	10.60	105.58	1020.45
751	12.12	119.82	1231.43
797	13.59	129.93	1367.93
853	15.53	161.03	1549.21
907	16.96	174.18	1781.31
947	19.01	189.62	1883.31
997	21.02	209.47	2109.57

Table 4.10 : Numerical Results (Seconds) on Chordal Graphs for Cliques First and $1100 - 2000$ nodes

Nodes	10 Scenarios	100 Scenarios	1000 Scenarios
1097	25.36	252.67	2497.56
1201	30.35	315.46	3028.60
1301	35.54	361.03	3560.64
1399	40.99	398.32	>3600
1499	46.97	497.21	>3600
1601	53.54	521.92	>3600
1699	60.24	580.34	>3600
1801	67.53	684.35	>3600
1901	75.09	743.98	>3600
1999	82.96	845.74	>3600

Table 4.11 : Numerical Results (Seconds) on Chordal Graphs for Switching and $50 - 1000$ nodes

Nodes	10 Scenarios	100 Scenarios	1000 Scenarios
47	0.55	5.20	51.91
99	0.82	12.85	127.66
151	1.89	18.43	185.36
199	1.71	27.53	281.03
251	4.16	39.56	415.01
307	3.50	57.24	583.29
349	4.49	72.97	715.84
401	5.75	93.63	885.46
449	7.08	116.12	1157.13
499	8.65	141.66	1521.12
547	17.12	169.77	1702.47
601	12.40	202.23	2030.64
653	24.07	238.63	2531.23
701	16.58	273.58	2875.05
751	18.98	312.16	3245.39
797	21.29	350.08	3503.55
853	24.28	401.09	>3600
907	27.28	452.38	>3600
947	29.82	489.96	>3600
997	34.57	541.52	>3600

Table 4.12 : Numerical Results (Seconds) on Chordal Graphs for Switching and $1100 - 2000$ nodes

Nodes	10 Scenarios	100 Scenarios	1000 Scenarios
1097	39.64	652.88	>3600
1201	47.39	782.96	>3600
1301	55.60	917.35	>3600
1399	64.00	1059.73	>3600
1499	72.14	1212.92	>3600
1601	83.73	1383.02	>3600
1699	94.13	1494.42	>3600
1801	105.50	1747.32	>3600
1901	115.02	1941.70	>3600
1999	129.81	2140.79	>3600

4.3 Conclusion

I have described a row generation algorithm for solving deterministic bilevel interdiction of a maximum stable set and showed numerical results to demonstrate the efficacy of the algorithm on test instances. The problem is NP-hard due to the complexity of finding a maximum stable set in a graph.

I also have created two algorithms for solving stochastic bilevel interdiction of a maximum stable set on perfect graphs. The first consists of a row generation algorithm to add cliques to the subproblem and then adds L-shaped cuts to the master problem. The second is a cross-decomposition algorithm that allows for switching between the two subsections rather than doing them sequentially.

Future work is to consider the comparative results between the two algorithms for the stochastic problem and determine exactly what is causing the slow down for the switching algorithm.

Chapter 5

New Valid Inequalities and Facets for the Clique Transversal Polytope

5.1 Clique Transversal Polytope

The clique transversal problem is to find a minimum cardinality set of nodes that intersects every maximal clique in the graph. It is a specific subproblem of the set covering problem. Therefore, I start with the general set covering problem. The set covering polytope is as follows:

$$\min\{1'x|Ax \geq 1, x \in \{0,1\}^n\},$$

where A is an $m \times n$ matrix with $a_{ij} \in \{0,1\}, \forall i,j$, and 1 is the m-vector of ones.

The clique transversal polytope is:

$$\min\{1'x|Ax \geq 1, x \in \{0,1\}^n\},$$

where A is an $m \times n$ matrix with $a_{ij} \in \{0,1\}, \forall i,j$, and 1 is the m-vector of ones. A is the clique-node incidence matrix given by:

$$a_{ij} = \begin{cases} 1 & \text{if node j is in clique i} \\ 0 & \text{otherwise.} \end{cases}$$

This is a specific example of the set covering problem, where the sets are maximal cliques. Thus, A from the set covering polytope is taken as the clique-node incidence

matrix for the clique transversal polytope. Following the notation from Balas and Ng [7], I let M be the row index set of A, and N be the column index set of A. Also, let $N^i = \{j \in N | a_{ij} = 1\}$.

Define $C(A) := \{x \in \{0,1\}^n | Ax \geq 1\}$, and let us refer to the convex hull of $C(A)$ as the clique transversal polytope. Theorems 5.1 and 5.2 follow directly from Balas and Ng's more general result for set covering [7].

Theorem 5.1 The clique transversal polytope is full dimensional ($dim(C(A)) = n$) if and only if every maximal clique contains at least two vertices (i.e. there are no singletons in the graph: $|N^i| \geq 2 \ \ \forall i \in M$).

Proof 5.1 Comes directly from results in [7] since the clique transversal polytope is a specific case of the set covering polytope.

Theorem 5.2 Assume that the clique transversal polytope is full dimensional.

1. The inequality $x_j \geq 0$ defines a facet of the clique transversal polytope if and only if the cardinality of each maximal clique containing j is greater than 3 (i.e. $|N^i \backslash \{j\}| \geq 2 \ \ \forall i \in M$).

2. All inequalities $x_j \leq 1$ define facets for the clique transversal polytope.

Proof 5.2 Comes directly from results in [7] since the clique transversal polytope is a specific case of the set covering polytope.

Moving forward, I can assume that the clique transversal polytope is full dimensional. If the clique transversal polytope is not full dimensional, then there exists

some maximal cliques that contain only one node. Setting these subsequent x values to be 1 and removing these satisfied inequalities from the polyhedron give a clique transversal polytope that is full dimensional . Thus, I can assume that the clique transversal polytope is full dimensional without loss of generality.

Section 5.2 describes valid inequalities for the clique transversal polytope. Section 5.3 describes the classes of facets of the clique transversal polytope, while Section 5.4 briefly sums up the findings of the paper.

5.2 Valid Inequalities for the Convex Hull of C(A)

To present this class of valid inequalities for triangle-free graphs, I present two more definitions before moving to the theorems. A *vertex cover* is a set of vertices such that every edge is incident to at least one vertex in the set. The *vertex cover number* of a graph is the size of the minimum vertex cover of a graph.

Theorem 5.3 Let the clique transversal polytope be full dimensional, G be a triangle-free graph, and $\tau(G)$ be the vertex cover number of G. Then $\sum_{i \in V} x_i \geq \tau(G)$ is a valid inequality of the clique transversal polytope.

Proof 5.3 Take some $x \in$ the convex hull of $C(A)$. I know that x must cover all of the maximal cliques that are contained in V. Since G is triangle-free, I know that x must cover every edge of G, since each edge of G is a distinct maximal clique. To cover the edges of G, I need at least $\tau(G)$ nodes chosen for a vertex cover. Therefore, any

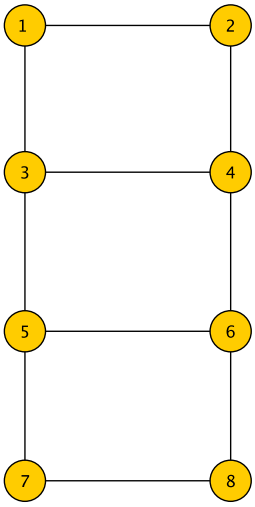


Figure 5.1 : Ladder Graph: $\sum_{i \in S} x_i \geq \frac{|L|}{2}$.

$x \in$ the clique transversal polytope must have at least this many nodes in V chosen, so $\sum_{i \in V} x_i \geq \tau(G)$ is a valid inequality of the clique transversal polytope.

I look at two examples. First, I have a ladder graph, depicted in Figure 5.1. I see that $\tau(L) = \frac{|L|}{2}$, where L is the ladder graph. Next, let $K_{p,q}$ be the complete bipartite graph with sets of size p and q. Figure 5.2 shows the complete bipartite graph for $p = 3$ and $q = 4$. Here, $\tau(K_{p,q}) = |K_p|$ for $p < q$ WLOG.

5.3 Facets for the Convex Hull of $C(A)$

Before moving on to some facets of the clique transversal polytope, I present some notation. Let J to be the set of nodes with coefficient 0 for the given inequality. Additionally, for each $v \in J$, define $T(v)$ as the number of maximal cliques that v is

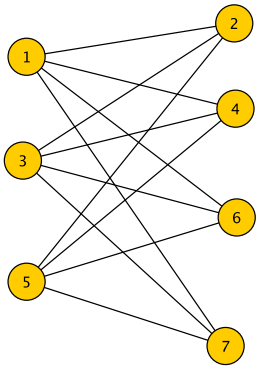


Figure 5.2 : Complete Bipartite Graph with $p = 3$ and $q = 4$: $\sum_{i \in S} x_i \geq |K_p|$, for $p < q$ WLOG

contained in that do not contain any other members of J. I define $G[H]$ to be the subgraph of G defined by the set H. If H is a set of vertices, then $G[H]$ is the graph made up of these vertices and the edges of G that are incident to two nodes in H. If H is a set of edges, then $G[H]$ is the graph made up of these edges and all vertices of G that are an endpoint of at least one edge in H.

5.3.1 Maximal Clique Inequalities

The first class of facets for the clique transversal polytope are just the individual maximal clique constraints

$$\sum_{i \in I_K} x_i \geq 1, \ \forall \text{ cliques } K.$$

It is trivial to see that these are valid inequalities.

Theorem 5.4 Let the clique transversal polytope be full dimensional and K be a maximal clique. Then $\sum_{i \in K} x_i \geq 1$ defines a facet of the clique transversal polytope.

Proof 5.4 I construct $|V|$ affinely independent vectors that belong to the clique transversal polytope and satisfy the inequality with equality. For a given maximal clique K, I begin by finding $|K|$ vectors, x^v, for each $v \in K$. Let:

$$x_i^v = \begin{cases} 1 & i = v \\ 0 & i \in K \backslash \{v\} \\ 1 & i \in V \backslash K \end{cases}$$

It is very easy to verify that each vector x^v defined above is a feasible clique transversal set. Thus, I have $|K|$ vectors that belong to the clique transversal polytope and satisfy the inequality with equality. Now, I construct the remaining $|V \backslash K|$ vectors, x^v, for each $v \in V \backslash K$. Let:

$$x_i^v = \begin{cases} 0 & i = v \\ 1 & i \in V \backslash K \end{cases}$$

To obtain a feasible clique transversal set, I have to be careful with the node I select from inside the clique, K, to be set to one. If $N_G(v) \cap K = \emptyset$, (i.e. $T(v) = 0$), then I can arbitrarily select any vertex from K to be set to one. Otherwise, $T(v) = 1$ and I should select a vertex inside $N_G(v) \cap K$ to be set to one. To show this, suppose there exists a maximal clique that has not been touched by any of the chosen vertices (vertices set to be one). Then, that maximal clique should contain vertex v. Additionally, since there are no isolated vertices, this maximal clique should also contain all elements of $N_G(v) \cap K \neq \emptyset$. Since it has not been touched, then the vertex chosen from K to be set to one is not a member of $N_G(v) \cap K$. So, if I select a

member of $N_G(v) \cap K$ (whenever this set is not empty) as the vertex in K to be set to one, then I will touch all maximal cliques. This results in an additional $|V \backslash K|$ vectors that belong to the convex hull of C(A) and satisfy the inequality with equality. It is easy to verify that these $|V|$ constructed vectors are affinely independent, and this completes the proof.

Next, I consider specific classes of facets. Balas and Ng [7] described a class of facets for the set covering problem of the form $\alpha x \geq 2$, where $\alpha_j = 0$, 1, or 2, for $j \in N$. Their procedure to find the inequalities is a specific version of Chvátal's [22] general procedure. I used their more general method of finding facets and proving they are facets with regard to my initial findings, similar to how Hicks [38] found facets for the planar subgraph problem. I further generalize the classes of facets to specific graph structures with coefficients 1:

$$\sum_{i \in S} x_i \geq g(S),$$

where S the union of nodes in the subgraph and $g(S)$ is a positive integer depending on the structure of the subgraph. I prove them by finding the n affinely independent vectors that each satisfy the inequality with equality.

5.3.2 Odd Hole Cover Inequalities

I present the first class of specific facets. This class of facets consists of an odd hole and distinct maximal cliques on each edge of the hole.

Figure 5.3 shows the graph structure that will make up the first facet. S consists

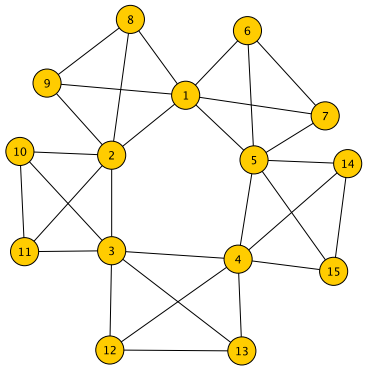


Figure 5.3 : Odd hole with distinct maximal cliques on each edge: $\sum_{i \in S} x_i \geq \frac{|H|+1}{2}$. Nodes $\{1, 2, 3, 4, 5\}$ are the odd hole. The outer maximal cliques are: $\{1, 5, 6, 7\}$, $\{4, 5, 14, 15\}$, $\{3, 4, 12, 13\}$, $\{2, 3, 10, 11\}$, and $\{1, 2, 9, 10\}$,.

of all of the nodes in the union of the distinct cliques that each share a distinct edge with the odd hole, H. This facet can be thought of as a generalization of the complement of the odd-hole facets for the Stable Set Polytope [60]

Theorem 5.5 Let the clique transversal polytope be full dimensional, H be the interior odd hole, and S the set of all nodes in the union of the outer maximal cliques. Then, $\sum_{i \in S} x_i \geq \frac{|H|+1}{2}$ is a valid inequality of the clique transversal polytope.

Proof 5.5 Suppose $\sum_{i \in S} x_i^0 < \frac{|H|+1}{2}$, where x^0 is the incidence vector of a clique transversal T^0 in graph G. Considering one of the distinct maximal cliques C^e for an edge e on the hole H, let u_e and v_e denote two vertices on the endpoints of e. If $T^0 \cap \{u_e, v_e\} = \emptyset$, then there exists a vertex $w^e \in C^e \setminus \{u_e, v_e\}$ such that $w^e \in T^0$. Remove w^e from T^0 and add one of the vertices u_e or v_e to T^0. Repeat this step for all distinct maximal cliques on all edges e of hole H for which $T^0 \cap \{u_e, v_e\} = \emptyset$. At

termination of this procedure, cardinality of T^0 *does not change and set* $T^0 \setminus (V \setminus H)$ *will form a vertex cover for* $G[H]$*. Notice that* $|T^0 \setminus (V \setminus H)| <= |T^0 \setminus (V \setminus S)| < \frac{|H|+1}{2}$*. Thus,* $T^0 \setminus (V \setminus H)$ *is a vertex cover of cardinality strictly smaller than* $\frac{|H|+1}{2}$ *for* $G[H]$*, which is a contradiction.*

Theorem 5.6 Let the clique transversal polytope be full dimensional, H *be the interior odd hole, and* S *the set of all nodes in the union of the outer maximal cliques. Also, let* $T(v) \leq 1$ *for all* $v \in J$*. Then* $\sum_{i \in S} x_i \geq \frac{|H|+1}{2}$ *defines a facet of the clique transversal polytope.*

Proof 5.6 I must construct $|V|$ *affinely independent points that belong to the clique transversal polytope and satisfy the* $\sum_{i \in S} x_i = \frac{|H|+1}{2}$*. The first* $|S \backslash H|$ *vectors,* x^v*, for each* $v \in S \backslash H$ *are defined by:*

$$x_i^v = \begin{cases} 1 & i \in J \\ 1 & i = v \\ 0 & i \in S \backslash (H \cup \{v\}) \end{cases}$$

The remaining nodes that are not specified by the above definition are the nodes on the interior odd hole, H*. For these nodes, let the nodes contained in the maximal clique with* v *be set to* 0*. Then the remaining nodes form a path of* $|H| - 3$ *edges. Cover all of these edges by alternating between* 0 *and* 1 *(starting from* 1*) for these nodes for a simple vertex cover over the path. This finishes* x^v *for each* $v \in S \backslash H$*. Thus, I have* $|S \backslash H|$ *vectors,* x^v*, such that* $\sum_{i \in S} x_i^v = \frac{|H|+1}{2}$*. It is easy to verify that each* x^v *constructed is a feasible clique transversal set, and hence belongs to the clique*

transversal polytope.

The next $|H|$ *vectors,* x^v*, for each* $v \in H$ *are defined by:*

$$x_i^v = \begin{cases} 1 & i \in J \\ 1 & i = v \\ 0 & i \in S \backslash H \end{cases}$$

Now, the remaining nodes that are not specified by the above definition are the nodes on the path, $H \backslash \{v\}$*. Choose one node adjacent to* v *on* H *to be* 1 *and alternate between* 0 *and* 1 *around the rest of* H *(starting from* 0*). This finishes* x^v *for each* $v \in H$*. Thus, I have* $|H|$ *vectors,* x^v*, such that* $\sum_{i \in S} x_i^v = \frac{|H|+1}{2}$*. It is easy to verify that each* x^v *constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.*

The last $|J|$ *vectors,* x^v*, for each* $v \in J$ *are defined by:*

$$x_i^v = \begin{cases} 1 & i \in J \backslash \{v\} \\ 0 & i = v \end{cases}$$

Now, if $T(v) = 0$*, then choose one of the previous* $|S|$ *vectors already defined, and use the values of elements of* S *in the chosen vector as the values for the elements of* S *in the new vector. The two vectors will be very similar and the only difference is that* $v \in J$ *is now* 0 *in the new vector as opposed to* 1 *in the chosen vector. If* $T(v) = 1$*, then choose one of the previous* $|S|$ *vectors already defined such that at least one node from* S *that is contained in the maximal clique with* v *is set to be* 1*. I know this exists because each node is equal to* 1 *for at least one of the* $|S|$ *vectors. Similarly, use the*

values of elements of S in the chosen vector as the values for the elements of S in the new vector. Therefore, this finishes x^v for each $v \in J$ with specifications for $T(v) = 0$ or $T(v) = 1$. I have $|J|$ vectors, x^v, such that $\sum_{i \in S} x_i^v = \frac{|H|+1}{2}$, since each vector here is a previous vector with only a change to x_v for each $v \in J$. It is easy to verify that each x^v constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.

I can see that the matrix X that is made up of rows which are the n vectors x^v, $v \in S \cup J$, is nonsingular. Therefore, $\sum_{i \in S} x_i \geq \frac{|H|+1}{2}$ defines a facet of the clique transversal polytope.

5.3.3 Odd Clique Cover Inequalities

The graph structure determining the second class of specific facets is an odd clique with distinct maximal cliques off of every edge of one of its spanning cycles.

Figure 5.4 depicts the graph structure for the second class of specific facets. I have one inner odd clique, a spanning cycle within the inner clique, and outer distinct maximal cliques that share one edge with the spanning cycle in the inner clique. I have S as the union of all of the nodes in all of the maximal cliques, K as the inner odd clique, and C as the spanning cycle in K.

Theorem 5.7 Let the clique transversal polytope be full dimensional, K be the inner odd clique, and S the set of all nodes in the union of the outer maximal cliques on the distinct edges of the spanning cycle H. Then, $\sum_{i \in S} x_i \geq \frac{|K|+1}{2}$ is a valid inequality

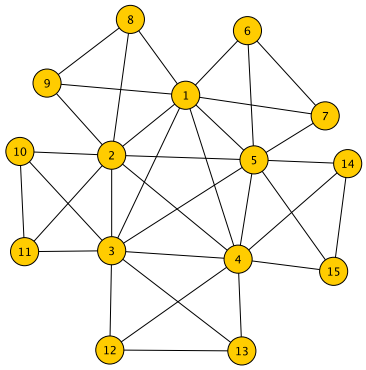


Figure 5.4 : An inner odd clique with distinct maximal cliques on each edge of one of its spanning cycles: $\sum_{i \in S} x_i \geq \left\lfloor \frac{|K|+1}{2} \right\rfloor$. Nodes $\{1, 2, 3, 4, 5\}$ are the inner clique. Edges $\{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 1\}\}$ are the cycle. The outer maximal cliques are: $\{1, 2, 8, 9\}$, $\{1, 5, 6, 7\}$, $\{4, 5, 14, 15\}$, $\{3, 4, 12, 13\}$, and $\{2, 3, 10, 11\}$.

for the clique transversal polytope.

Proof 5.7 Suppose $\sum_{i \in S} x_i^0 < \frac{|K|+1}{2}$, *where* x^0 *is the incidence vector of a clique transversal* T^0 *in graph* G. *Considering one of the distinct maximal cliques* C^e *for an edge* e *on the spanning cycle* H, *let* u_e *and* v_e *denote two vertices on the endpoints of* e. *Let* E_H *be the set of edges in the spanning cycle* H. *If* $T^0 \cap \{u_e, v_e\} = \emptyset$, *then there exists a vertex* $w^e \in C^e \setminus \{u_e, v_e\}$ *such that* $w^e \in T^0$. *Remove* w^e *from* T^0 *and add one of the vertices* u_e *or* v_e *to* T^0. *Repeat this step for all distinct maximal cliques on all edges* e *of spanning cycle* H *for which* $T^0 \cap \{u_e, v_e\} = \emptyset$. *At termination of this procedure, cardinality of* T^0 *does not change and set* $T^0 \setminus (V \setminus H)$ *will form a vertex cover for* $G[E_H]$. *Notice that* $|T^0 \setminus (V \setminus H)| <= |T^0 \setminus (V \setminus S)| < \frac{|K|+1}{2}$ *and* $|K| = |H|$. *Thus,* $T^0 \setminus (V \setminus H)$ *is a vertex cover of cardinality strictly smaller than* $\frac{|K|+1}{2}$ *for* $G[E_H]$, *which is a contradiction.*

Theorem 5.8 Let the clique transversal polytope be full dimensional, K be the inner odd clique, and S the set of all nodes in the union of the outer maximal cliques on the distinct edges of the spanning cycle H. Also, let $T(v) \leq 1$ for all $v \in J$. Then $\sum_{i \in S} x_i \geq \frac{|K|+1}{2}$ defines a facet of the clique transversal polytope.

Proof 5.8 I must construct $|V|$ affinely independent points that belong to the clique transversal polytope and satisfy the $\sum_{i \in S} x_i = \frac{|K|+1}{2}$. The first $|S \backslash K|$ vectors, x^v, for each $v \in S \backslash K$ are defined by:

$$x_i^v = \begin{cases} 1 & i \in J \\ 1 & i = v \\ 0 & i \in S \backslash (K \cup \{v\}) \end{cases}$$

The remaining nodes that are not specified by the above definition are the nodes in the inner clique, K. Let the two nodes that are contained in the outer maximal clique that contains v be set equal to 0. Then, alternate between 1 and 0 on the other nodes around H (starting from 1). This finishes x^v for each $v \in S \backslash K$. Thus, I have $|S \backslash K|$ vectors, x^v, such that $\sum_{i \in S} x_i^v = \frac{|K|+1}{2}$. It is easy to verify that each x^v constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.

The next $|K|$ vectors, x^v, for each $v \in K$ are defined by:

$$x_i^v = \begin{cases} 1 & i \in J \\ 1 & i = v \\ 0 & i \in S \backslash K \end{cases}$$

Now, the remaining nodes that are not specified by the above definition are the

nodes in $K\backslash\{v\}$*. Choose one of the nodes adjacent to* v *in* H *to be* 1 *(starting from* 0*). Alternate around the rest of* C *between* 0 *and* 1*. This finishes* x^v *for each* $v \in K$*. Thus, I have* $|K|$ *vectors,* x^v*, such that* $\sum_{i \in S} x_i^v = \frac{|K|+1}{2}$*. It is easy to verify that each* x^v *constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.*

The last $|J|$ *vectors,* x^v*, for each* $v \in J$ *are defined by:*

$$x_i^v = \begin{cases} 1 & i \in J\backslash\{v\} \\ \\ 0 & i = v \end{cases}$$

Now, if $T(v) = 0$*, then choose one of the previous* $|S|$ *vectors already defined, and use the values of elements of* S *in the chosen vector as the values for the elements of* S *in the new vector. The two vectors will be very similar and the only difference is that* $v \in J$ *is now* 0 *in the new vector as opposed to* 1 *in the chosen vector. If* $T(v) = 1$*, then choose one of the previous* $|S|$ *vectors already defined such that at least one node from* S *that is contained in the maximal clique with* v *is set to be* 1*. I know this exists because each node is equal to* 1 *for at least one of the* $|S|$ *vectors. Similarly, use the values of elements of* S *in the chosen vector as the values for the elements of* S *in the new vector. This finishes* x^v *for each* $v \in J$*. Therefore, I have* $|J|$ *vectors,* x^v*, such that* $\sum_{i \in S} x_i^v = \frac{|K|+1}{2}$*, since each vector here is a previous vector with only a change to* x_v *for each* $v \in J$*. It is easy to verify that each* x^v *constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.*

Again, I can see that the matrix X *that is made up of rows which are the* n *vectors*

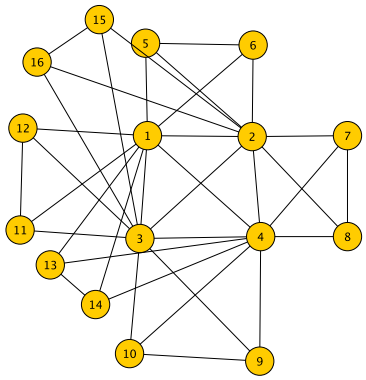


Figure 5.5 : Clique with distinct maximal cliques on every edge: $\sum_{i \in S} x_i \geq |K| - 1$. Nodes $\{1, 2, 3, 4\}$ are the inner clique. The outer maximal cliques are: $\{1, 2, 5, 6\}$, $\{2, 4, 7, 8\}$, $\{3, 4, 9, 10\}$, $\{1, 3, 11, 12\}$, $\{1, 4, 13, 14\}$, and $\{2, 3, 15, 16\}$.

x^v, $v \in S \cup J$, is nonsingular. Therefore, $\sum_{i \in S} x_i \geq \frac{|K|+1}{2}$ defines a facet of the clique transversal polytope.

5.3.4 Clique Cover Inequalities

Another specific class of facets also comes from a graph structure with a clique at the center. However, this class contains distinct maximal cliques off of every edge of the inner maximal clique.

Figure 5.5 depicts the final class of facets for the clique transversal polytope that I present. I have an inner clique K, and I have distinct outer maximal cliques that each share one edge with K.

Theorem 5.9 Let the clique transversal polytope be full dimensional, K be the inner clique, and S the set of all nodes in the union of the outer maximal cliques. Then, $\sum_{i \in S} x_i \geq |K| - 1$ is a valid inequality of the clique transversal polytope.

Proof 5.9 Suppose $\sum_{i \in S} x_i^0 < |K| - 1$, *where* x^0 *is the incidence vector of a clique transversal* T^0 *in graph* G. *Considering one of the distinct maximal cliques* C^e *for an edge* e *on the clique* K, *let* u_e *and* v_e *denote two vertices on the endpoints of* e. *If* $T^0 \cap \{u_e, v_e\} = \emptyset$, *then there exists a vertex* $w^e \in C^e \setminus \{u_e, v_e\}$ *such that* $w^e \in T^0$. *Remove* w^e *from* T^0 *and add one of the vertices* u_e *or* v_e *to* T^0. *Repeat this step for all distinct maximal cliques on all edges* e *of clique* K *for which* $T^0 \cap \{u_e, v_e\} = \emptyset$. *At termination of this procedure, cardinality of* T^0 *does not change and set* $T^0 \setminus (V \setminus K)$ *will form a vertex cover for* $G[K]$. *Notice that* $|T^0 \setminus (V \setminus K)| <= |T^0 \setminus (V \setminus S)| < |K| - 1$. *Thus,* $T^0 \setminus (V \setminus K)$ *is a vertex cover of cardinality strictly smaller than* $|K| - 1$ *for* $G[K]$, *which is a contradiction.*

Theorem 5.10 Let the clique transversal polytope be full dimensional, K *be the inner clique, and* S *the set of all nodes in the union of the outer maximal cliques. Also, let* $T(v) \leq 1$ *for all* $v \in J$. *Then* $\sum_{i \in S} x_i \geq |K| - 1$ *defines a facet of the clique transversal polytope.*

Proof 5.10 I must construct $|V|$ *affinely independent points that belong to the clique transversal polytope and satisfy the* $\sum_{i \in S} x_i = |K| - 1$. *The first* $|S \setminus K|$ *vectors,* x^v, *for each* $v \in S \setminus K$ *are defined by:*

$$x_i^v = \begin{cases} 1 & i \in J \\ 1 & i = v \\ 0 & i \in S \setminus (K \cup \{v\}) \end{cases}$$

The remaining nodes that are not specified by the above definition are the nodes in the inner clique, K. Let the two nodes that are contained in the outer maximal clique that contains v be set equal to 0. *Then, choose every other node in K to be* 1. *This finishes x^v for each $v \in S\backslash K$. Thus, I have $|S\backslash K|$ vectors, x^v, such that $\sum_{i\in S} x_i^v = |K| - 1$. It is easy to verify that each x^v constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.*

The next $|K|$ vectors, x^v, for each $v \in K$ are defined by:

$$x_i^v = \begin{cases} 1 & i \in J \\ 0 & i = v \\ 0 & i \in S\backslash K \\ 1 & i \in K\backslash\{v\} \end{cases}$$

This finishes x^v for each $v \in K$. Thus, I have $|K|$ vectors, x^v, such that $\sum_{i\in S} x_i^v = |K| - 1$. It is easy to verify that each x^v constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.

The last $|J|$ vectors, x^v, for each $v \in J$ are defined by:

$$x_i^v = \begin{cases} 1 & i \in J\backslash\{v\} \\ 0 & i = v \end{cases}$$

Now, if $T(v) = 0$, then choose one of the previous $|S|$ vectors already defined, and use the values of elements of S in the chosen vector as the values for the elements of S in the new vector. The two vectors will be very similar and the only difference is that $v \in J$ is now 0 *in the new vector as opposed to* 1 *in the chosen vector. If $T(v) = 1$,*

then choose one of the previous $|S|$ vectors already defined such that at least one node from S that is contained in the maximal clique with v is set to be 1. *I know this exists because each node is equal to* 1 *for at least one of the $|S|$ vectors. Similarly, use the values of elements of S in the chosen vector as the values for the elements of S in the new vector. This finishes x^v for each $v \in J$. Therefore, I have $|J|$ vectors, x^v, such that $\sum_{i \in S} x_i^v = |K| - 1$, since each vector here is a previous vector with only a change to x_v for each $v \in J$. It is easy to verify that each x^v constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.*

Again, I can see that the matrix X that is made up of rows which are the n vectors x^v, $v \in S \cup J$, is nonsingular. Therefore, $\sum_{i \in S} x_i \geq |K| - 1$ defines a facet of the clique transversal polytope.

5.3.5 Prism Cover Inequalities

My last class of specific facets comes from a prism graph. A prism graph consists of an inner cycle and an outer cycle of the same size, with edges connecting corresponding nodes as shown in Figure 5.6. This class of facets comes from a graph structure that has a prism at the center but also has distinct maximal cliques off of every edge of the prism.

Let $\hat{P}$ denote the number of nodes in the interior of the prism (e.g. nodes $\{1, 2, 3, 4, 5\}$ in Figure 5.6). I see that $\hat{P} = \frac{|P|}{2}$, where P is the set of nodes in the entire prism. Figure 5.7 depicts a 5-prism with distinct maximal cliques of size 3

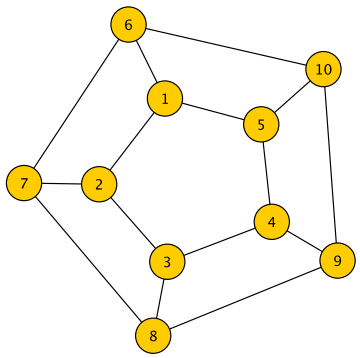


Figure 5.6 : Prism Graph: $\sum_{i \in S} x_i \geq \hat{P} + \hat{P}(mod\ 2)$.

on every edge of the prism.

Theorem 5.11 Let the clique transversal polytope be full dimensional, P be the prism, and S the set of all nodes in the union of the outer maximal cliques. Then, $\sum_{i \in S} x_i \geq \hat{P} + \hat{P}(mod\ 2)$ is a valid inequality of the clique transversal polytope.

Proof 5.11 Suppose $\sum_{i \in S} x_i^0 < \hat{P} + \hat{P}(mod\ 2)$, where x^0 is the incidence vector of a clique transversal T^0 in graph G. Considering one of the distinct maximal cliques C^e for an edge e on the prism P, let u_e and v_e denote two vertices on the endpoints of e. If $T^0 \cap \{u_e, v_e\} = \emptyset$, then there exists a vertex $w^e \in C^e \setminus \{u_e, v_e\}$ such that $w^e \in T^0$. Remove w^e from T^0 and add one of the vertices u_e or v_e to T^0. Repeat this step for all distinct maximal cliques on all edges e of prism P for which $T^0 \cap \{u_e, v_e\} = \emptyset$. At termination of this procedure, cardinality of T^0 does not change and set $T^0 \setminus (V \setminus P)$ will form a vertex cover for $G[P]$. Notice that $|T^0 \setminus (V \setminus P)| <= |T^0 \setminus (V \setminus S)| < \hat{P} + \hat{P}(mod\ 2)$. Thus, $T^0 \setminus (V \setminus P)$ is a vertex cover of cardinality strictly smaller

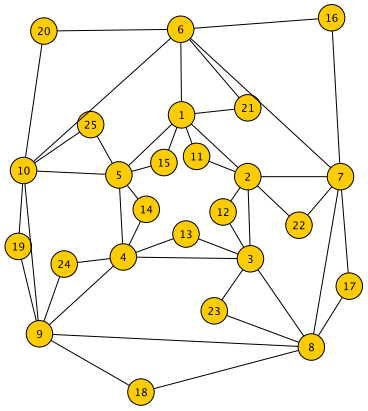


Figure 5.7 : Prism graph with distinct maximal cliques on every edge: $\sum_{i \in S} x_i \geq \hat{P} + \hat{P}(mod\ 2)$. Nodes $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ are the inner prism. The distinct maximal cliques are: $\{1, 2, 11\}$, $\{2, 3, 12\}$, $\{3, 4, 13\}$, $\{4, 5, 14\}$, $\{1, 5, 15\}$, $\{6, 7, 16\}$, $\{7, 8, 17\}$, $\{8, 9, 18\}$, $\{9, 10, 19\}$, $\{6, 10, 20\}$, $\{1, 6, 21\}$, $\{2, 7, 22\}$, $\{3, 8, 23\}$, $\{4, 9, 24\}$, and $\{5, 10, 25\}$.

than $\hat{P} + \hat{P}(mod\ 2)$ *for* $G[P]$*, which is a contradiction.*

Theorem 5.12 Let the clique transversal polytope be full dimensional, P *be an odd prism with* $\hat{P} \geq 5$*, and* S *the set of all nodes in the union of the outer maximal cliques. Also, let* $T(v) \leq 1$ *for all* $v \in J$*. Then* $\sum_{i \in S} x_i \geq \hat{P} + 1$ *defines a facet of the clique transversal polytope.*

Proof 5.12 Let I_P *be the indicator set of nodes in the prism,* P*. I must construct* $|V|$ *affinely independent points that belong to the convex hull of C(A) and satisfy the* $\sum_{i \in S} x_i = \hat{P} + 1$*, since this only holds for odd prisms. The first* $|S \backslash I_P|$ *vectors,* x^v*, for each* $v \in S \backslash I_P$ *are defined by:*

$$x_i^v = \begin{cases} 1 & i \in J \\ 1 & i = v \\ 0 & i \in S \backslash (I_P \cup \{v\}) \end{cases}$$

The remaining nodes that are not specified by the above definition are the nodes of the prism, P*. I have three cases. First, if the edge of the prism that is contained in the maximal clique containing* v *is on the inner cycle, then set the two nodes contained in this edge to be* 0*. Alternate between* 0 *and* 1 *on the remaining nodes of the inner cycle (starting from* 1*). Set the two nodes of the outer cycle that are adjacent to the two inner nodes that are contained in the maximal clique containing* v *to be* 1*. Alternate between* 0 *and* 1 *on the remaining nodes of the outer cycle (starting from* 0*). This is* $\hat{P} + 1$ *nodes chosen for the clique transversal. The second case is If the edge of the prism that is contained in the maximal clique containing* v *is on the outer cycle. Then, set the two nodes contained in this edge to be* 0*. Alternate between* 0 *and* 1 *on the remaining nodes of the outer cycle (starting from* 1*). Set the two nodes of the inner cycle that are adjacent to the two outer nodes that are contained in the maximal clique containing* v *to be* 1*. Alternate between* 0 *and* 1 *on the remaining nodes of the inner cycle (starting from* 0*). This is* $\hat{P} + 1$ *nodes chosen for the clique transversal. Lastly, if the edge of the prism that is contained in the maximal clique containing* v *is not on the inner or outer cycle, set the two nodes contained in this edge to be* 0*. Alternate between* 0 *and* 1 *around the remaining nodes in the outer cycle (starting from* 1*) and alternate between* 0 *and* 1 *around the remaining nodes in the inner cycle*

(starting from 0). This is $\hat{P}+1$ nodes chosen for the clique transversal.

This finishes x^v for each $v \in S \backslash I_P$. Thus, I have $|S \backslash I_P|$ vectors, x^v, such that $\sum_{i \in S} x_i^v = \hat{P}+1$. It is easy to verify that each x^v constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.

Now, for $v \in I_P$ and v is in the interior of the prism, I find two affinely independent vectors for each v. Thus, I have my next $|P|$ vectors defined by:

$$x_i^v = \begin{cases} 1 & i \in J \\ 0 & i = v \\ 0 & i \in S \backslash I_P \end{cases}$$

Now, I need to choose for the remaining nodes in the prism. Choose the corresponding node on the exterior of the prism that is adjacent to v. Call this node $\hat{v}$. Choose nodes v and $\hat{v}$. Now, I have a ladder left, rather than a prism. There are two different ways to do a vertex cover of the remaining ladder with $\hat{P}-1$ nodes.

This finishes x^v for each $v \in I_P$ and v is in the interior of the prism. Thus, I have two distinct vectors here that satisfy with equality. Therefore, for every vertex on the interior of the prism, I have two affinely independent vectors that satisfy with equality. Thus, I have $|P|$ vectors, x^v, such that $\sum_{i \in S} x_i^v = \hat{P}+1$. It is easy to verify that each x^v constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.

The last $|J|$ vectors, x^v, for each $v \in J$ are defined by:

$$x_i^v = \begin{cases} 1 & i \in J \backslash \{v\} \\ 0 & i = v \end{cases}$$

Now, if $T(v) = 0$*, then choose one of the previous* $|S|$ *vectors already defined for the remaining nodes in* S*, and use the values of elements of* S *in the chosen vector as the values for the elements of* S *in the new vector. The two vectors will be very similar and the only difference is that* $v \in J$ *is now* 0 *in the new vector as opposed to* 1 *in the chosen vector. If* $T(v) = 1$*, then choose one of the previous* $|S|$ *vectors already defined such that at least one node from* S *that is contained in the maximal clique with* v *is set to be* 1*. I know this exists because each node is equal to* 1 *for at least one of the* $|S|$ *vectors. Similarly, use the values of elements of* S *in the chosen vector as the values for the elements of* S *in the new vector. This finishes* x^v *for each* $v \in J$*. Therefore, I have* $|J|$ *vectors,* x^v*, such that* $\sum_{i \in S} x_i^v = \hat{P} + 1$*, since each vector here is a previous vector with only a change to* x_v *for each* $v \in J$*. It is easy to verify that each* x^v *constructed is a feasible clique transversal set, and hence belongs to the clique transversal polytope.*

Again, I can see that the matrix X *that is made up of rows which are the* n *vectors* x^v*,* $v \in S \cup J$*, is nonsingular. Therefore,* $\sum_{i \in S} x_i \geq \hat{P} + 1$ *defines a facet of the clique transversal polytope where* P *is an odd prism with* $\hat{P} \geq 5$*.*

5.4 Conclusion

I have found new valid inequalities and facets for the clique transversal polytope. The valid inequalities consist of triangle-free graphs or triangle-free subgraphs in other graphs. The ladder graph and the complete bipartite graph are examples of such triangle-free graphs. In each of these graphs, a vertex cover is needed for these structures to be valid for the clique transversal polytope.

I have shown that the maximal clique inequalities that define the clique transversal linear program are facets of the clique transversal polytope. Additionally, I have shown four new classes of facets for the clique transversal polytope similar to those of the set covering polytope. The first consists of an odd hole, H, whose edges are in distinct maximal cliques, where S is the union of all of the nodes in the maximal cliques:

$$\sum_{i \in S} x_i \geq \frac{|H|+1}{2}.$$

The second class consists of an odd clique K, whose outer edges are contained in distinct maximal cliques. Set S again is the union of all of the nodes in the maximal cliques:

$$\sum_{i \in S} x_i \geq \frac{|K|+1}{2}.$$

Next, I have a clique K where each edge of the clique is contained in a distinct maximal clique:

$$\sum_{i \in S} x_i \geq |K| - 1.$$

The last class of facets come from odd prism graphs and have the form:

$$\sum_{i \in S} x_i \geq \hat{P} + \hat{P}(mod\ 2),$$

where $\hat{P}$ is the number of the nodes in the interior of the prism. These inequalities are facets for odd prisms with $\hat{P} \geq 5$, but are just valid inequalities for $\hat{P} = 3$ and even prisms. Each of these classes of facets is generalized from facets with coefficients in $\{0, 1, 2\}$ that were obtained by using Balas and Ng's [7] method for the facets of the set covering polytope.

An open problem is to create a separation algorithm for each of these classes of facets and show computational results that demonstrate the efficacy of using these inequalities to shrink the clique transversal polytope.

Chapter 6

Conclusion

6.1 Conclusion

I have described a formulation for bilevel interdiction of a minimum clique transversal and a subsequent column generation algorithm for solving the integer program. This problem is NP-hard due to the known complexity of finding a minimum clique transversal. Computational results show the algorithm to be effective on sparse random graphs up to 500 nodes. For planar graphs, the algorithm can solve the problem in under an hour for up to 2400 nodes. This formulation and algorithm is also used as an introduction into bilevel clique interdiction since it introduces an interdiction problem with an exponential number of constraints, but not the exponential number of variables that is necessary for bilevel clique interdiction.

I have constructed a formulation for bilevel clique interdiction and a delayed column-and-row generation algorithm. I showed that the problem is NP-hard by a reduction to the knapsack problem. I proved the correctness of the algorithm and stopping criterion showing that I do not necessarily have to add in every clique to the formulation as a row and a column. Computational results show that the algorithm solves the problem on dense random graphs in under an hour for up to $50-100$ nodes.

For sparse random graphs, I can use this algorithm for graphs with up to 500 nodes and still expect a solution in under an hour. Additionally, this algorithm can solve bilevel clique interdiction in under an hour for planar graphs with up to 2400 nodes, similar to the algorithm for bilevel interdiction of a minimum clique transversal.

I created a deterministic formulation for bilevel interdiction of a maximum stable set and have a column generation algorithm. This problem is NP-hard due to previous work showing that finding a maximum stable set in a graph is NP-hard. Similar to the other interdiction algorithms, this algorithm can solve bilevel interdiction of a maximum stable set in under an hour for sparse random graphs up to 500 nodes and planar graphs up to 2400 nodes.

I developed a stochastic formulation for bilevel interdiction of a maximum stable set and have a sequential method with L-shaped cuts and a cross-decomposition method using cutting plane and L-shaped cuts. I proved the validity of switching between adding clique cuts and L-shaped cuts. That is to say, the L-shaped cuts that are added into the restricted master problem stay feasible even after more clique cuts are added into the lower problem. The sequential algorithm outperforms the cross-decomposition method. For 1000 scenarios, the sequential algorithm can solve the problem in under an hour for chordal graphs up to 1400 nodes, while the cross-decomposition algorithm can only reach 800 nodes. However, for both 10 and 100 scenarios, both algorithms can solve the problem in under an hour for chordal graphs with up to 2000 nodes.

I have discovered multiple new valid inequalities and facets for the clique transversal polytope. The valid inequalities come from triangle-free subgraphs with distinct cliques on the edges of the subgraph, where the right hand side is the vertex cover number. The IP constraints for the clique transversal polytope were shown to be facets, as well as four new classes of facets. The first consists of an odd hole with distinct maximal cliques on each edge. The second is an odd clique with distinct maximal cliques on each edge of one of its spanning cycles. The third is a clique with distinct maximal cliques on each edge of the entire clique. Lastly, there is a prism with distinct maximal cliques on each edge.

Future work should focus on the cross-decomposition algorithm for stochastic bilevel interdiction of a maximum stable set. The cross-decomposition algorithm is outperformed by the sequential algorithm with clique cuts and L-shaped cuts where it should be competitive at least. Additionally, future work should be done on finding a separation algorithm for the facets for the clique transversal polytope. Finding the subgraphs to be used in the facets can be NP-hard so further study of specific types of graphs and graph structures could lead to separation algorithms in certain graphs.

Bibliography

[1] Shabbir Ahmed, Mohit Tawarmalani, and Nikolaos V Sahinidis. A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377, 2004.

[2] Thomas Andreae. On the clique-transversal number of chordal graphs. *Discrete Mathematics*, 191(1-3):3–11, 1998.

[3] Thomas Andreae, Martin Schughart, and Zsolt Tuza. Clique-transversal sets of line graphs and complements of line graphs. *Discrete Mathematics*, 88(1):11–20, 1991.

[4] Nikitas Assimakopoulos. A network interdiction model for hospital infection control. *Computers in biology and medicine*, 17(6):413–422, 1987.

[5] Pasquale Avella, Antonio Sassano, and Igor Vasil'ev. Computational study of large-scale p-median problems. *Mathematical Programming*, 109(1):89–114, 2007.

[6] Matthew D Bailey, Steven M Shechter, and Andrew J Schaefer. Spar: stochastic programming with adversarial recourse. *Operations Research Letters*, 34(3):307–315, 2006.

[7] Egon Balas and Shu Ming Ng. On the set covering polytope: {I}. all the facets with coefficients in {0, 1, 2}. *Mathematical Programming*, 43(1-3):57–69, 1989.

[8] Egon Balas and Shu Ming Ng. On the set covering polytope: {II}. lifting the facets with coefficients in {0, 1, 2}. *Mathematical Programming*, 45(1-3):1–20, 1989.

[9] Egon Balas and Manfred W Padberg. Set partitioning: A survey. *SIAM review*, 18(4):710–760, 1976.

[10] Gulay Barbarosoğlu and Yasemin Arda. A two-stage stochastic programming framework for transportation planning in disaster response. *Journal of the operational research society*, 55(1):43–53, 2004.

[11] John E Beasley. A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):151–164, 1990.

[12] John E Beasley and Paul C Chu. A genetic algorithm for the set covering problem. *European journal of operational research*, 94(2):392–404, 1996.

[13] John R Birge and Francois V Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, 1988.

[14] Flavia Bonomo, Maria Chudnovsky, and Guillermo Durán. Partial characterizations of clique-perfect graphs {I}: Subclasses of claw-free graphs. *Discrete Applied Mathematics*, 156(7):1058–1082, 2008.

[15] Flavia Bonomo, Maria Chudnovsky, and Guillermo Durán. Partial characterizations of clique-perfect graphs {II}: Diamond-free and helly circular-arc graphs. *Discrete Mathematics*, 309(11):3485–3499, 2009.

[16] Flavio Bonomo, Guillermo Durán, Marina Groshaus, and Jayme L Szwarcfiter. On clique-perfect and k-perfect graphs. *Ars Combinatoria*, 80:97–112, 2006.

[17] Nicolas Bourgeois, Bruno Escoffier, Vangelis Paschos, and Johan van Rooij. A bottom-up method and fast algorithms for max independent set. In *Algorithm Theory-SWAT 2010*, pages 62–73. Springer, 2010.

[18] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[19] Alberto Caprara, Matteo Fischetti, and Paolo Toth. A heuristic method for the set covering problem. *Operations research*, 47(5):730–743, 1999.

[20] Frédéric Cazals and Chinmay Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1):564–568, 2008.

[21] Raymond K Cheung and Chuen-Yih Chen. A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem. *Transportation science*, 32(2):142–162, 1998.

[22] Vašek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete mathematics*, 4(4):305–337, 1973.

[23] Vašek Chvátal. On certain polytopes associated with graphs. *Journal of Combinatorial Theory, Series B*, 18(2):138–154, 1975.

[24] Vašek Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[25] Pablo Coll, Javier Marenco, Isabel Méndez Díaz, and Paula Zabala. Facets of the graph coloring polytope. *Annals of Operations Research*, 116(1-4):79–90, 2002.

[26] Kelly J Cormican, David P Morton, and R Kevin Wood. Stochastic network interdiction. *Operations Research*, 46(2):184–197, 1998.

[27] Gérard Cornuéjols and Antonio Sassano. On the 0, 1 facets of the set covering polytope. *Mathematical Programming*, 43(1-3):45–55, 1989.

[28] Ken Darby-Dowman, Simon Barker, Eric Audsley, and David Parsons. A two-stage stochastic programming with recourse model for determining robust planting plans in horticulture. *Journal of the Operational Research Society*, 51(1):83–89, 2000.

[29] Cees Dert. Asset liability management for pension funds: a multistage chance constrained programming approach. 1995.

[30] Paul Erdös, Tibor Gallai, and Zsolt Tuza. Covering the cliques of a graph with vertices. *Discrete mathematics*, 108(1-3):279–289, 1992.

[31] Sean F Everton. *Disrupting dark networks*, volume 34. Cambridge University Press, 2012.

[32] Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

[33] Fănică Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973.

[34] Bruce Golden. A problem in network interdiction. *Naval Research Logistics Quarterly*, 25(4):711–713, 1978.

[35] Martin Grötschel and Manfred W Padberg. On the symmetric travelling salesman problem {I}: inequalities. *Mathematical Programming*, 16(1):265–280, 1979.

[36] Martin Grötschel and Manfred W Padberg. On the symmetric travelling salesman problem {II}: Lifting theorems and facets. *Mathematical Programming*, 16(1):281–302, 1979.

[37] Robert L Helmbold. A countercapacity network interdiction model. 1971.

[38] Illya V Hicks. New facets for the planar subgraph polytope. *Networks*, 51(2):120–132, 2008.

[39] Kaj Holmberg. On the convergence of cross decomposition. *Mathematical Programming*, 47(1):269–296, 1990.

[40] Eitan Israeli and R Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

[41] Udom Janjarassuk and Jeff Linderoth. Reformulation and sampling to solve a stochastic network interdiction problem. *Networks*, 52(3):120–132, 2008.

[42] Valdis E Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2002.

[43] BJ Lageweg, Jan Karel Lenstra, AHG Rinnooy Kan, and Leen Stougie. Stochastic integer programming by dynamic programming. *Statistica Neerlandica*, 39(2):97–113, 1985.

[44] Churlzu Lim and J Cole Smith. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*, 39(1):15–26, 2007.

[45] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[46] Foad Mahdavi Pajouh, Vladimir Boginski, and Eduardo L Pasiliao. Minimum vertex blocker clique problem. *Networks*, 64(1):48–64, 2014.

[47] Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Algorithm Theory-SWAT 2004*, pages 260–272. Springer, 2004.

[48] Imran Maqsood, Guo H Huang, and Julian Scott Yeomans. An interval-parameter fuzzy two-stage stochastic program for water resources management under uncertainty. *European Journal of Operational Research*, 167(1):208–225, 2005.

[49] Chang Maw-Shang, Chen Yi-Hua, Gerard J Chang, and Yan Jing-Ho. Algorithmic aspects of the generalized clique-transversal problem on chordal graphs. *Discrete Applied Mathematics*, 66(3):189–203, 1996.

[50] Benjamin McClosky and Illya V Hicks. Combinatorial algorithms for the maximum k-plex problem. *Journal of combinatorial optimization*, 23(1):29–49, 2012.

[51] Alan W McMasters and Thomas M Mustin. Optimal interdiction of a supply network. *Naval Research Logistics Quarterly*, 17(3):261–268, 1970.

[52] George J Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304, 1980.

[53] John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

[54] David P Morton, Feng Pan, and Kevin J Saeger. Models for nuclear smuggling interdiction. *IIE Transactions*, 39(1):3–14, 2007.

[55] Raffaele Mosca. Polynomial algorithms for the maximum stable set problem on particular classes of p5-free graphs. *Information Processing Letters*, 61(3):137–143, 1997.

[56] Ibrahim Muter, Ş İlker Birbil, and Kerem Bülbül. Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. *Mathematical Programming*, 142(1-2):47–82, 2013.

[57] Daishin Nakamura and Akihisa Tamura. A revision of minty's algorithm for finding a maximum weight stable set of a claw-free graph. *Journal of the Operations Research Society of Japan*, 44(2):194–204, 2001.

[58] Paolo Nobili and Antonio Sassano. Facets and lifting procedures for the set covering polytope. *Mathematical Programming*, 45(1-3):111–137, 1989.

[59] Emmanuel Ogbe and Xiang Li. A new cross decomposition method for stochastic mixed-integer linear programming. *European Journal of Operational Research*, 256(2):487–499, 2017.

[60] Manfred W Padberg. On the facial structure of set packing polyhedra. *Mathematical programming*, 5(1):199–215, 1973.

[61] Feng Pan, William S Charlton, and David P Morton. A stochastic program for interdicting smuggled nuclear material. In *Network interdiction and stochastic integer programming*, pages 1–19. Springer, 2003.

[62] Erich Prisner. Graphs with few cliques. *Graph theory, combinatorics, and algorithms*, 1:2, 1995.

[63] José Emmanuel Ramirez-Marquez and Claudio M Rocco S. Stochastic network interdiction optimization via capacitated network reliability modeling and probabilistic solution discovery. *Reliability Engineering & System Safety*, 94(5):913–921, 2009.

[64] Antonio Sassano. On the facial structure of the set covering polytope. *Mathematical Programming*, 44(1-3):181–202, 1989.

[65] Najiba Sbihi. Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. *Discrete Mathematics*, 29(1):53–76, 1980.

[66] Andrew J Schaefer, Ellis L Johnson, Anton J Kleywegt, and George L Nemhauser. Airline crew scheduling under uncertainty. *Transportation science*, 39(3):340–348, 2005.

[67] ErFang Shan, TCE Cheng, and LiYing Kang. Bounds on the clique-transversal number of regular graphs. *Science in China Series A: Mathematics*, 51(5):851–863, 2008.

[68] Siqian Shen, J Cole Smith, and Roshan Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization*, 2012.

[69] Robert Endre Tarjan and Anthony E Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

[70] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.

[71] Zsolt Tuza. Covering all cliques of a graph. *Discrete Mathematics*, 86(1-3):117–126, 1990.

[72] René van Bevern, Hannes Moser, and Rolf Niedermeier. Approximation and tidying-a problem kernel for s-plex cluster vertex deletion. *Algorithmica*, 62(3-4):930–950, 2012.

[73] Tony J Van Roy. Cross decomposition for mixed integer programming. *Mathematical programming*, 25(1):46–63, 1983.

[74] Richard M Van Slyke and Roger Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

[75] Jose L Walteros and Panos M Pardalos. A decomposition approach for solving critical clique detection problems. In *Experimental Algorithms*, pages 393–404. Springer, 2012.

[76] Alan Washburn and Kevin Wood. Two-person zero-sum games for network interdiction. *Operations Research*, 43(2):243–251, 1995.

[77] Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[78] Cynthia I Wood and Illya V Hicks. The minimal k-core problem for modeling k-assemblies. *The Journal of Mathematical Neuroscience (JMN)*, 5(1):1–19, 2015.

[79] R Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[80] R Kevin Wood. Bilevel network interdiction models: Formulations and solutions. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.