The Maximum Clique Interdiction Game

Fabio Furini
PSL, Université Paris-Dauphine, Paris, France, fabio.furini@dauphine.fr

Ivana Ljubić
ESSEC Business School, Cergy-Pontoise, France, ivana.ljubic@essec.edu

Sébastien Martin
LCOMS, Université de Lorraine, Metz, France, sebastien.martin@univ-lorraine.fr

Pablo San Segundo
Universidad de Politécnica de Madrid, Madrid, Spain, pablo.sansegundo@upm.es

We study the two player zero-sum Stackelberg game in which the leader interdicts (removes) a limited number of vertices from the graph, and the follower searches for the maximum clique in the interdicted graph. The goal of the leader is to derive an interdiction policy which will result in the worst possible outcome for the follower. This problem has applications in many areas, such as crime detection, prevention of outbreaks of infectious diseases and surveillance of communication networks. We design an exact solution framework based on a Bilevel Integer Linear Programming model. Thanks to the study of the polytope of the corresponding single-level reformulation, we derive a branch-and-cut algorithm and enhance it by tight combinatorial lower and upper bounds, which also allow for a drastic reduction of the size of the input graph. Our model is based on an exponential family of *Clique-Interdiction Cuts* whose separation requires solving the maximum clique problem. We derive an effective separation procedure based on a newly developed combinatorial algorithm that is tailored for finding maximum cliques in interdicted graphs. We assess the applicability and the limits of our exact framework on publicly available instances, including large-scale social networks with up to one hundred thousand vertices and three million edges. Most of these instances are solved to provable optimality within short computing times. Our code (which will be also publicly available) allows to analyze the resilience of (social) networks with respect to vertex-interdiction attacks, i.e., the decrease of the size of the maximum clique in function of incremental interdiction budget level.

Key words: Network Interdiction, Maximum Clique, Social Network Analysis

1. Introduction

Interdiction games on networks are a special family of two-player zero-sum Stackelberg games, in which the first player (a *leader*) decides which vertices or edges to *interdict* without violating a given interdiction budget, and after that the second player (a *follower*) solves a network optimization problem (e.g., the maximum clique, the shortest-path, the maximum flow) on the remaining network. The goal of the leader is to choose an *interdiction policy* that will guarantee the worst possible outcome for the follower. Interdiction games on networks have been widely studied in the last two decades both in the deterministic and in the stochastic setting, see, e.g., (Washburn and Wood 1995, Cormican et al. 1998, Song and Shen 2016). Some relevant applications include the firefighting problem (Adjiashvili et al. 2017, Baggio 2016, García-Martnez et al. 2015), control of infections in hospitals (Assimakopoulos 1987), allocation of protective resources in shortest-path networks (Cappanera and Scaparra 2011), protection and analysis of supply chain disruptions (Snyder et al. 2016), or detection of drug smuggling (Washburn and Wood 1995). Knapsack interdiction games are studied in (Fischetti et al. 2016, Caprara et al. 2016).

In this article we study the interdiction game of the maximum clique in a network. A clique refers to a subset of vertices that are pairwise connected and it is often used to identify "tightly knit" parts of a network (Wasserman and Faust 1994). In the context of crime detection and prevention, large cliques have been identified as potential origins of catastrophic events such as terrorist or hacker attacks (Berry et al. 2004, Sageman 2004), or sources of outbreaks of sexually transmitted diseases (Rothenberg et al. 1996). In the analysis of terrorist networks (see, e.g., Chen et al. 2004, Sampson and Groves 1989), cliques are used to model communities, due to their ability to encode the dynamics of groups of individuals who are all close friends with one another. Large cliques in a terrorist or crime network are capable of designing sophisticated large-scale operations like those of the 9/11 attacks, the series of coordinated terrorist attacks in Paris in 2014 and 2015, and numerous devastating coordinated attacks throughout Europe, Middle East, Africa and Asia, see (2017 Terrorist Attacks) for a detailed map on recent attacks. Randomly arresting individuals

in such a network has a low impact on its integrity (Barabási and Pósfai 2016): it leaves most of the large cliques intact, and hence it does not significantly impact the damaging potential of the network. In this manuscript we study the problem on how to efficiently reduce the size of the largest clique of a network, given a predefined number of vertices that can be interdicted.

In the context of game theory, this problem can be seen as the two-player *Clique Interdiction Game* (CIG), in which the leader has a limited budget of vertices to interdict in a network and the follower finds the largest clique in the remaining network. Applications of such games can also be found in other areas. For example, in modern telecommunication networks, Service Functions (SFs) such as firewalls, deep packet inspections (DPIs), web proxies, media gateways, etc. are used as middleboxes on the Service Provider Networks. Network Function Virtualization permits to virtualize the SFs so that using Software Defined Networks (SDNs), SFs can be disassociated from the physical elements of the network and installed as a software. This allows for a centralized control of network resources in which a centralized controller has visibility over the entire network, and has a complete view of the network topology (Kim and Feamster 2013). Interdicting a vertex in an SDN network corresponds to an installation of a SF at a vertex that can monitor that vertex' activity. Solution of the CIG in a SDN network returns a limited number of vertices where SFs have to be installed so as to monitor and prevent any suspicious activity within a closed community while keeping the size of the remaining non-monitored cliques as small as possible.

Definition 1 (The Maximum Clique Interdiction Game (CIG)). Given a graph G and an interdiction budget k ($k \geq 1$), the *maximum clique interdiction game* is to find a subset of at most k vertices to delete from G so that the size of the maximum clique in the remaining graph is minimized. The associated set of interdicted vertices is called the *optimal interdiction policy.*

In Figure 1 we present an optimal interdiction policy for a graph G of nine vertices and a budget $k = 3$. In the left part of the figure, we show in red the vertices belonging to the maximum clique of G, i.e., the clique $\tilde{K} = \{v_3, v_4, v_7, v_8, v_9\}$. In the right part of the figure, we show the optimal interdiction policy, i.e., the black vertex set $\{v_4, v_7, v_8\}$. Once these vertices are interdicted, the

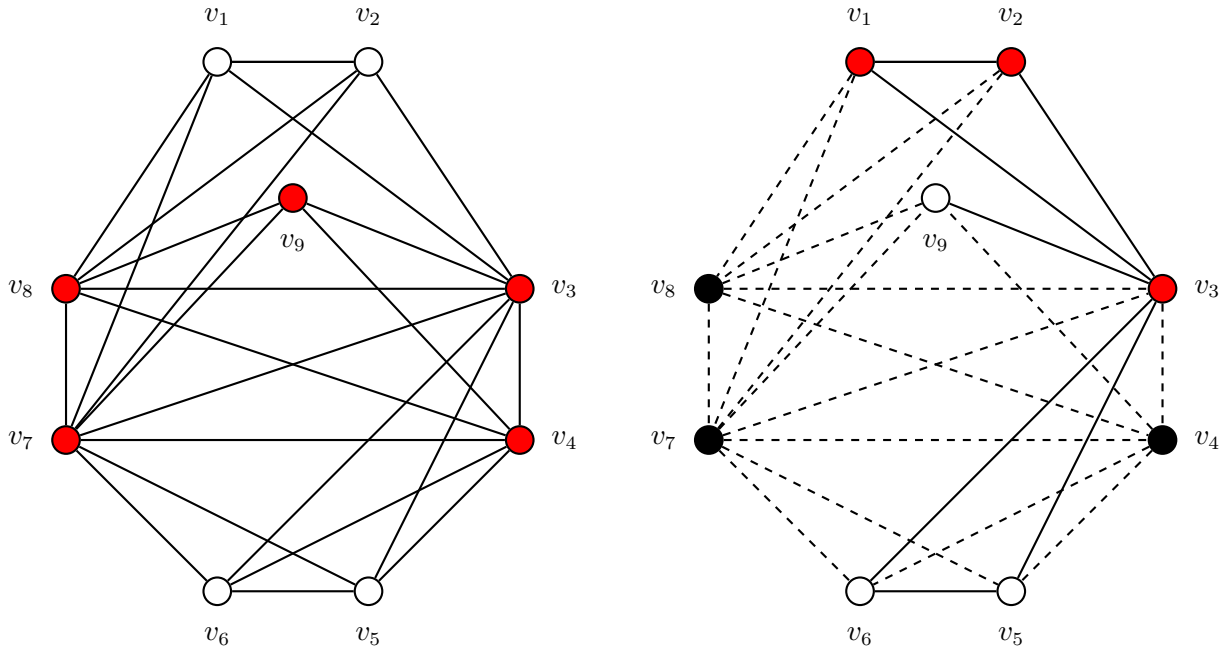

Figure 1 CIG with a budget $k = 3$. Left: The maximum clique in G is of size 5. Right: the optimal interdiction policy is given in black, and a remaining maximum clique of size 3 is shown in red.

remaining maximum clique of the graph is of cardinality 3, i.e., the clique $\hat{K} = \{v_1, v_2, v_3\}$. We draw with dashed lines the edges which are incident to the interdicted vertices, as they cannot be used to build cliques in G.

Literature Overview To the best of our knowledge, despite its relevance in the Stackelberg games on social networks, the CIG is a new problem that has not been studied in the previous literature. There are only two problems directly related to the CIG that we are aware of. The first one is the *Minimum Vertex Blocker Clique Problem*, in which one searches for a subset of vertices of minimum cardinality to be removed from a graph G, so that the maximum (weighted) clique in the remaining graph is bounded from above by a given integer $r \geq 1$. This problem has been studied in Mahdavi Pajouh et al. (2014) where an exact algorithm based on a row-generation has been proposed. In a computational study conducted on uniform randomly generated graphs, the authors report optimal solution values for sparse graphs with 100 and 200 vertices, whereas most of the instances with density 0.5 or above remain unsolved.

The second related problem is the *Edge Interdiction Clique Game* which has been introduced in Tang et al. (2016) and used as a case study for a generic exact approach for (the more general class of) interdiction games. In the edge interdiction clique game, the leader interdicts edges instead of vertices, and therefore majority of results derived in this article are not directly applicable to this problem. Nevertheless it is worth mentioning that the largest graphs considered in the computational study of Tang et al. (2016) contain 15 vertices, and most of them remained unsolved within an hour of computing time. Optimal solutions for these instances have been recently provided by the state-of-the-art exact solver for bilevel mixed integer programs of Fischetti et al. (2017).

CIG also belongs to a larger family of Interdiction Games under Monotonicity, which has been recently addressed in Fischetti et al. (2016). In this setting, the leader has a limited interdiction budget for deleting a subset of items, whereas the follower solves a maximization problem using the remaining items. The follower's subproblem is assumed to satisfy a monotonicity (or hereditary) property, which is then exploited for deriving a single-level integer linear programming reformulation. The authors also study several strengthening inequalities which are of particular relevance for the knapsack interdiction games. A computational study conducted on (multidimensional) knapsack interdiction games shows that this new approach significantly outperforms the previous state-of-the-art algorithm from (Caprara et al. 2016).

Our Contribution A significant portion of the Graph Theory and Operations Research literature is devoted to the study of the maximum clique problem in a graph, as its solution describes one of the most essential graph properties known as the *clique number* of a graph. To the best of our knowledge, this is the first study on how to find the *most vital k vertices of a graph*, so as to reduce its clique number.

The decision version of the CIG is Σ_2^P-complete, but for a special class of graphs, we show that the CIG can be solved in polynomial time. The latter result is also exploited for deriving a tight combinatorial lower bound which is efficiently employed for reducing the number of vertices of the input graph. Starting from a Bilevel Integer Programming model, we then derive a single-level

reformulation based on an exponential family of *Clique-Interdiction Cuts*. We provide necessary and sufficient conditions under which these cuts are facet defining and we propose a fast lifting procedure. Based on the latter reformulation, we propose a branch-and-cut algorithm (B&C) which incorporates an efficient separation procedure of clique-interdiction cuts. This is a specially tailored combinatorial algorithm derived from one of the state-of-the-art approaches for finding maximum cliques in interdicted graphs. We derive a battery of heuristic algorithms as well, to obtain high-quality incumbent solutions.

In an extensive computational study, we demonstrate the effectiveness of all the components of our exact solution framework. The study is based on publicly available benchmark instances from various sources, including large-scale graphs representing real-world (social) networks with up to one hundred thousand vertices and three million edges. Most of these instances are solved to provable optimality within a short computing time, thus significantly outperforming the state-of-the-art generic bilevel mixed integer programming solver of Fischetti et al. (2017).

Our implementation, which will be publicly available, provides a powerful tool to decision makers for analyzing and assessing the graph resilience against vertex-deletion attacks, i.e., the capacity of the graph of preserving a large clique. For different classes of input graphs, we analyze the evolution of the clique number in function of the interdiction budget k. Our analysis reveals that social networks are "vulnerable" to vertex-deletion attacks as their clique number can be easily reduced by a large factor using a relatively low interdiction budget.

Finally, we point out that our exact solution framework can be extended to a more generic interdiction game in which the follower solves a *relaxed* clique problem with hereditary property, such as s-plex, s-bundle, or s-defective clique.

The remainder of the paper is organized as follows: Section 2 provides the formal problem definition, the bilevel integer programming formulation, and complexity results. In Section 3, we provide theory and methodology behind our combinatorial lower bound and the associated reduction procedure. The single-level reformulation along with the polyhedral study is given in Section 4. The separation procedure is described in Section 5. Computational results are provided in Section 6, and concluding remarks are given in Section 7.

2. Bilevel formulation and problem complexity

In this section we first introduce the notation used in this article, we then provide a Bilevel Integer Linear Programming (ILP) formulation, after which we study the problem complexity.

2.1. Notation and definitions

Let $G=(V,E)$ be a simple undirected graph with $|V|=n$ and $|E|=m$. The complement of G is denoted $\overline{G}=(V,\overline{E})$, so $\overline{E}=\{uv : uv \notin E\}$. Given a vertex set S, the subgraph of G induced by S is denoted by $G[S]$. We say that u and v are *neighbours* if there is an edge $uv \in E$. Neighbors of a vertex v are denoted by $N(v)$. A subset $S \subseteq V$ of vertices is a *clique of* G, if any two vertices of S are neighbours, and it is a *stable set* of G if it is a clique in $\overline{G}$. The cardinality of the largest clique of G is denoted by $\omega(G)$, and the cardinality of the largest stable set by $\alpha(G)$. The *maximum clique problem* (MCP) in G is to find a clique of maximum size.

Given a vertex $v \in V$, the *clique number of* v, denoted by $\omega_G(v)$ in the following, is the size of the largest clique v is contained in. The *κ-core* of a graph G is a maximal subgraph in which all vertices have degree at least κ. The *coreness-number* of a vertex v, denoted by coreness(v) in the following, is equal to κ if v belongs to a κ-core but not to any $(\kappa+1)$-core. Obviously, the following inequality holds:

$$\omega_G(v) \leq \text{coreness}(v)+1 \leq |N(v)|+1 \qquad v \in V. \tag{1}$$

For sparse graphs, the coreness-number of a vertex is often used as a rough approximation of its clique number, as it can be computed in $O(|E|)$ time for all vertices in G (for further details, see Batagelj and Zaversnik 2011, San Segundo et al. 2016a). The maximum core number of a graph, denoted by core(G) in the following, is the maximum of the core numbers of the vertices of G. So, we have

$$\omega(G) \leq \text{core}(G)+1 \leq \max_{v \in V} \deg(v)+1. \tag{2}$$

A *coloring* C of G is a partition of V into p non-empty stable sets: $C=\{V_1,\ldots,V_p\}$, where all vertices belonging to V_i are colored with the same color i $(i=1,\ldots,p)$. A graph property is called

hereditary on vertex induced subgraphs if for any $S \subseteq V$ we have that if the property holds on $G[S]$, then it also holds on $G[S']$, for any $S' \subset S, S' \neq \emptyset$.

Let $\tilde{S}$ be the maximal stable set in G, i.e., $\alpha(G) = |\tilde{S}|$. For a given interdiction budget k such that $k \geq |V| - \alpha(G)$, the optimal CIG solution value is one. Hence, without loss of generality, we will assume that $k < |V| - \alpha(G)$.

2.2. Bilevel ILP formulation

The following binary decision variables are needed to model the problem as a bilevel integer linear program:

$$w_u = \begin{cases} 1, & \text{if vertex } u \text{ is interdicted by the leader,} \\ 0, & \text{otherwise} \end{cases} \qquad u \in V$$

$$x_u = \begin{cases} 1, & \text{if vertex } u \text{ is used in the maximum clique of the follower,} \\ 0, & \text{otherwise} \end{cases} \qquad u \in V$$

Let $\mathcal{W}$ be the set of all feasible interdiction policies of the leader, i.e.:

$$\mathcal{W} = \left\{ w \in \{0,1\}^n : \sum_{u \in V} w_u \leq k \right\}. \tag{3}$$

Similarly, let $\mathcal{K}$ represent the set of incidence vectors of all cliques in the graph G, i.e.:

$$\mathcal{K} = \left\{ x \in \{0,1\}^n : x_u + x_v \leq 1, uv \in \overline{E} \right\}, \tag{4}$$

where the constraints $x_u + x_v \leq 1$ ensure that two vertices cannot be part of a clique if there is no edge connecting them. With a slight abuse of notation, we will use both notations $K \in \mathcal{K}$ and $x \in \mathcal{K}$, where x is the incident vector of K. Given an interdiction policy $w^* \in \mathcal{W}$, let V_{w^*} be the associated set of interdicted (deleted) vertices.

Property 1. The CIG can be restated as follows:

$$\min_{w \in \mathcal{W}} \max_{K \in \mathcal{K}} \left\{ |K| - \sum_{u \in K} w_u \right\}. \tag{5}$$

Proof In the objective function of (5) we express the min-max nature of the problem: the inner maximization problem (that we will refer as the *follower's subproblem*) counts the number of vertices in the solution, whereas the leader controls (with the variables w) the outer minimization problem. Observe that for the follower, the size of each clique K in G reduces by the number of interdicted vertices from K, which follows from the hereditary property of the clique (i.e., every vertex-induced subgraph of K is a clique itself). Therefore, the value $|K| - \sum_{u \in K} w_u$ denotes the size of the clique K after applying the interdiction policy defined by $w \in \mathcal{W}$. Hence, the formulation (5) states that the leader chooses a set of vertices to interdict, so that among all possible cliques $K \in \mathcal{K}$, the size of the maximum remaining clique is the smallest possible. □

The CIG can be equivalently stated as:

$$\min_{w \in \mathcal{W}} \max_{x \in \mathcal{K}} \left\{ \sum_{u \in V} (1 - w_u) x_u \right\} \tag{6}$$

Indeed, we can rewrite the objective function of the follower in (6) as $\sum_{u \in V}(1 - w_u)x_u = |K| - \sum_{u \in K} w_u x_u$. The latter sum corresponds to the objective function in (5) if and only if $w_u x_u = w_u$ if $u \in K$ and $w_u x_u = 0$, otherwise. The latter is always true, as the vector x encodes the clique K, and hence $x_u = 1$ if $u \in K$, and $x_u = 0$, otherwise. Finally, the CIG can also be reformulated as the following bilevel integer linear program:

$$\min_{w \in \mathcal{W}} \max_{x \in \mathcal{K}} \left\{ \sum_{u \in V} x_u \ : \ x_u \leq 1 - w_u, \ u \in V \right\}. \tag{7}$$

Indeed, given an interdiction policy $w^* \in \mathcal{W}$, the optimal solution of the follower's subproblem (5) and the optimal solution of the follower's subproblem (7) are the same. Inequalities $x_u \leq 1 - w_u$ ($u \in V$) are the linking interdiction constraints, making sure that the follower cannot chose a vertex u if it has been interdicted by the leader. They ensure that for a given vector w^*, the follower searches for the maximum clique in the support graph in which the vertices v such that $w^*_v = 1$ have been removed. The non-linear ILP formulation (7) can be linearized using a Benders-like decomposition approach (cf. Section 4).

2.3. Problem complexity

In the following, we address the complexity of the decision version of CIG stated as "*Is there an interdiction policy such that the maximum clique in the interdicted graph is not greater than some given bound?*". More precisely, we consider the following problem:

Decision Version of CIG (d-CIG): Given a graph G and two integers k and ℓ, can we remove (at most) k vertices from G such that the resulting graph does not contain a clique of size ℓ?

Observe that the answer to the decision problem is YES if only if the optimal CIG solution is $\leq \ell - 1$. Obviously, d-CIG is not in NP for the following reason: Given a subset of k vertices to remove and a value of ℓ, to test whether the resulting graph does not contain a clique of size ℓ requires answering the decision problem of the maximum clique problem, which is NP-complete. In fact, as most of the interdiction games in which the follower solves an NP-hard problem, d-CIG belongs to the Σ_2^P complexity class. For d-CIG we even have a stronger result, which has been proven by (Rutenburg 1991, 1994), where d-CIG is called the GENERALIZED NODE DELETION problem.

PROPOSITION 1 **(Rutenburg (1991, 1994))**. *The decision version of CIG is Σ_2^P-complete.*

In the following, we consider a special family of graphs, for which we show that, under certain conditions, the CIG can be solved in polynomial time.

PROPOSITION 2. *Assume that the graph $G = (K_1, \ldots, K_p)$ is a union of vertex-disjoint cliques K_i $(1 \leq i \leq p)$ such that $|K_1| \geq \cdots \geq |K_p|$ is given, then the optimal solution value of the CIG can be found in $O(p)$ time, and the optimal solution can be calculated in $O(p+k)$ time.*

Proof Let $\mathcal{Q}_q = (K_1, ..., K_q)$ be the subgraph induced by the first q cliques of G, $1 \leq q \leq p$. Let $k(\mathcal{Q}_q)$ denote the size of an optimal interdiction policy necessary to reduce the size of all cliques in $\mathcal{Q}_q$ to $|K_q| - 1$. One easily verifies that this number can be computed as:

$$k(\mathcal{Q}_q) = q + \sum_{i=1}^{q-1} i \cdot (|K_i| - |K_{i+1}|).$$

In the following we must consider two cases:

$k(\mathcal{Q}_p) > k$: In order to compute the optimal solution value, it is then sufficient to find the value q^* and the associated subgraph $\mathcal{Q}_{q^*}$ such that $k(\mathcal{Q}_{q^*}) > k$ and $k(\mathcal{Q}_{q^*-1}) \leq k$. In that case, the optimal solution value is obtained as:

$$\text{OPT}(G,k) = \max\left\{|K_{q^*}|, |K_{q^*-1}| - 1 - \left\lfloor \frac{k - k(\mathcal{Q}_{q^*-1})}{q^* - 1} \right\rfloor \right\}. \tag{8}$$

By definition, $k(\mathcal{Q}_{q^*-1})$ is the budget necessary to interdict all cliques in $\mathcal{Q}_{q^*-1}$ of size $|K_{q^*-1}|$ or more, so that the largest clique will be of size $|K_{q^*-1}| - 1$. Then, the value $\left\lfloor \frac{k-k(\mathcal{Q}_{q^*-1})}{q^*-1} \right\rfloor$ provides the number of units by which the size of the largest clique in $\mathcal{Q}_{q^*-1}$ can be further reduced using the remaining budget of $k - k(\mathcal{Q}_{q^*-1})$. Finally, if $|K_{q^*}| > |K_{q^*-1}| - 1 - \left\lfloor \frac{k-k(\mathcal{Q}_{q^*-1})}{q^*-1} \right\rfloor$, the optimal solution value will be $|K_{q^*}|$. Finding the value of q^* can be done in $O(p)$ iterations, and each iteration has a constant time complexity, assuming that the size of each clique is known in advance.

$k(\mathcal{Q}_p) \leq k$: Using the same arguments, we can deduce

$$\text{OPT}(G,k) = |K_p| - 1 - \left\lfloor \frac{k - k(\mathcal{Q}_p)}{p} \right\rfloor. \tag{9}$$

To obtain the optimal solution, one has to (randomly) select $|K_i| - \text{OPT}(G,k)$ vertices from each of the cliques K_i $(1 \leq i \leq q^*)$. This can be done in $O(k+p)$ time. □

In order to see how to compute $\text{OPT}(G,k)$ using the formula (8), consider a graph G composed by four cliques ($p = 4$) of size 10, 8, 5 and 5, respectively. Then, $k(\mathcal{Q}_p) = 12$ and this corresponds to the budget k necessary to interdict all cliques of size 5 or more. Consider now a budget of $k = 7$, so we have $k(\mathcal{Q}_2) = 4$ and $k(\mathcal{Q}_3) = 11$, hence $q^* = 3$. The value of $\text{OPT}(G,7)$ is given as $\max\{5,6\}$, where 6 represents the size of the maximum clique after interdicting 7 vertices in the first two largest cliques. Consider now a budget of $k = 11$, in this case $q^* = 4$. The second term in formula (8) provides us the value of 4, as the size of the largest clique in $\mathcal{Q}_3$ after interdicting 11 vertices from the three largest cliques, whereas the size of the fourth clique provides the optimal solution value of 5 (as there is not sufficient budget to reduce its size).

Proposition 2 provides the optimal CIG solution value for the graphs composed by unions of cliques, but it is particularly important for deriving a tight globally valid lower bound in the most general case, without imposing any restrictions on G (cf. Section 3).

3. Combinatorial bounds, tighter gaps and preprocessing

We propose combinatorial algorithms for calculating tight global lower and upper bounds. Besides being useful for exact branch-and-bound-based approaches, we also demonstrate how these tight bounds help in reducing the size of the input graph. Let in the following $\ell_{\min}$, $\ell_{\max}$ denote the global lower and upper bound on the solution value, and let ℓ_{opt} be the optimal solution value (i.e., the size of the maximum clique after applying an optimal interdiction policy).

3.1. Computing the global lower bound $\ell_{\min}$

With the following result we show how removing edges from G allows us to compute a global lower bound on the CIG value.

PROPOSITION 3. *Given a subgraph $G' = (V, E')$ with $E' \subset E$, the optimal CIG solution on G' provides a valid lower bound for the optimal CIG solution on G.*

Proof Observe that $\omega(G') \leq \omega(G)$. Let $w \in \mathcal{W}$ be a feasible interdiction policy for G and let V_w be the associated subset of interdicted vertices. The optimal CIG value on G is equal to

$$\min_{w \in \mathcal{W}} \omega(G[V \setminus V_w])$$

and we want to prove that

$$\min_{w \in \mathcal{W}} \omega(G'[V \setminus V_w]) \leq \min_{w \in \mathcal{W}} \omega(G[V \setminus V_w]). \tag{10}$$

Let $\bar{w} \in \mathcal{W}$ be the optimal interdiction policy on G. Then we have

$$\omega(G'[V \setminus V_{\bar{w}}]) \leq \omega(G[V \setminus V_{\bar{w}}]) = \min_{w \in \mathcal{W}} \omega(G[V \setminus V_w]).$$

Since the most left expression corresponds to a feasible CIG solution on G', and the optimal one is obtained by minimizing over all $w \in \mathcal{W}$, the inequality (10) follows immediately. □

The result of Proposition 3 is rather counter-intuitive, as it states that by reducing the input graph, instead of obtaining a valid upper bound for a minimization problem, we obtain a valid

lower bound. The key issue here is that we are in fact not reducing the feasibility space of the leader (as the set of vertices in G' remains the same as in G), but only the feasibility space of the follower (which is reduced from all cliques induced by E to all cliques induced by E'). Observe, furthermore, that in terms of the problem complexity, the result of Proposition 3 does not help us much in solving the problem, since we still have to solve the same CIG problem, only on a smaller graph. However, this result can be particularly useful for special classes of graphs on which solving the CIG on G' is easier than on G. This is exploited by the following result, where the proof for correctness of formula (11) follows from the proof of Proposition 2.

COROLLARY 1. *Given a set $\mathcal{Q}_{p+1} = (K_1, \ldots, K_{p+1})$ of vertex-disjoint cliques of G, such that $|K_1| \geq \cdots \geq |K_{p+1}|$, a valid lower bound $\ell_{\min}$ for the CIG can be obtained by computing*

$$\ell_{\min} = \begin{cases} \max\left\{|K_{p+1}|, |K_p| - 1 - \left\lfloor \frac{k-k(\mathcal{Q}_p)}{p} \right\rfloor\right\}, & \text{if } k < k(\mathcal{Q}_{p^*+1}) \\ |K_{p+1}| - 1 - \left\lfloor \frac{k-k(\mathcal{Q}_{p+1})}{p+1} \right\rfloor, & \text{otherwise} \end{cases} \tag{11}$$

The quality of this lower bound depends on the number of cliques $p+1$ and their sizes. In order to obtain a tight lower bound (see Section 6 for the discussion on the empirical quality of the bounds), we compute maximum vertex-disjoint cliques on the fly and stop as soon as we find p^* such that $k(\mathcal{Q}_{p^*+1}) > k$, i.e., as soon as the interdiction budget is exhausted. On the one hand, this procedure minimizes the number of times the NP-hard maximum clique problem is solved. On the other hand, by focusing on maximum (rather than random cliques), the procedure preserves the quality of the bound $\ell_{\min}$ in a greedy fashion. By construction, the generated cliques are sorted in non-increasing order by their size, as required by the formula (11).

3.2. Reducing the input graph

We start by describing another important solution property that is exploited in our study. Given a clique K and an interdiction policy $w^* \in \mathcal{W}$, we say that K is *covered* by V_{w^*} if and only if at least one vertex from K is interdicted by w^*, i.e., iff $K \cap V_{w^*} \neq \emptyset$.

PROPERTY 2. If there exists a feasible interdiction policy $w^* \in \mathcal{W}$, such that all cliques of size $\ell+1$ in G are covered by V_{w^*} and there exists at least one clique K^* in G, $|K^*| = \ell$ which is not covered by V_{w^*}, then ℓ is a valid upper bound on the CIG.

The latter property suggests another way of seeing the CIG: find a feasible interdiction policy that minimizes the value of ℓ while making sure all cliques of size $\ell+1$ are covered.

We say that an interdiction policy $w \in \mathcal{W}$ is *minimal*, if for the associated set of interdicted vertices V_w, we have:

$$\omega(G[V \setminus V_w]) < \omega(G[(V \setminus V_w) \cup \{v\}]), \quad \forall v \in V_w.$$

The following result identifies redundant vertices in the input graph G.

PROPOSITION 4. *Let v be an arbitrary vertex from V. If $\omega_G(v) \leq \ell_{\text{opt}}$, then v cannot be part of a minimal optimal interdiction policy.*

Proof By contradiction, assume that there exists a minimal optimal interdiction policy $w^* \in \mathcal{W}$ such that $w^*_v = 1$. We now construct a new feasible interdiction policy $\bar{w} \in \mathcal{W}$ as follows: $\bar{w}_u := w^*_u$ for all $u \in V, u \neq v$ and $\bar{w}_v := 0$. It remains to show that

$$\omega(G[V \setminus V_{\bar{w}}]) = \omega(G[V \setminus V_{w^*}]) = \ell_{\text{opt}}.$$

Clearly, $\omega(G[V \setminus V_{\bar{w}}]) \geq \ell_{\text{opt}}$, since $\bar{w}$ is a feasible interdiction policy. By Property 2, the optimal solution V_{w^*} must cover all cliques of size $\geq \ell_{\text{opt}} + 1$. Since the largest cliques containing v are of size $\leq \ell_{\text{opt}}$, then also the set $V_{\bar{w}} = V_{w^*} \setminus \{v\}$ covers all the cliques of size $\geq \ell_{\text{opt}} + 1$, which implies that $\omega(G[V \setminus V_{\bar{w}}]) \leq \ell_{\text{opt}}$ and this concludes the proof. □

Hence, by focusing on the minimal interdiction policies, which can be done without loss of generality, one can preprocess the graph G and remove redundant vertices from it. To properly exploit the Proposition 4, instead of using the (unknown) value of ℓ_{opt} for removing the redundant vertices, one can employ a tight lower bound $\ell_{\min}$. The following result gives a connection between the optimal solution value and the solution value found on the preprocessed graph from which redundant vertices are removed.

PROPOSITION 5. *Let* $V_{\text{prep}} = \{v \in V : \omega_G(v) \leq \ell_{\min}\}$ *be the set of vertices* v *satisfying the (weakened) property of Proposition 4. Let* $\tilde{V} = V \setminus V_{\text{prep}}$ *and* $\tilde{G} = G[\tilde{V}]$ *and let* $\tilde{\ell}_{\text{opt}}$ *denote the optimal CIG solution value on* $\tilde{G}$*. Then*

$$\ell_{\text{opt}} = \max\{\tilde{\ell}_{\text{opt}}, \ell_{\min}\}.$$

Furthermore, the optimal interdiction policy $\tilde{w}$ *on* $\tilde{G}$*, is also optimal for* G*.*

Proof Let $\tilde{w}$ be the optimal interdiction policy found on $\tilde{G}$. To show that $\tilde{w}$ is also optimal for G, we distinguish between the following two cases:

1. $\omega(G[\tilde{V} \setminus V_{\tilde{w}}]) < l_{\min}$: In that case, $\tilde{w}$ is a feasible interdiction policy on G, with the follower's optimal response on the remaining graph being equal to $\ell_{\min}$. This is due to the facts that: (i) all vertices from V_{prep} cannot appear in a clique of size $> \ell_{\min}$, (ii) $\ell_{\min}$ is a valid lower bound and thus there exist at least one clique of size $\ell_{\min}$ in the graph $G[V \setminus V_{\tilde{w}}]$. Hence, $\ell_{\min}$ is equal to the optimal solution value ℓ_{opt}.

2. $\omega(G[\tilde{V} \setminus V_{\tilde{w}}]) \geq l_{\min}$: In that case, $\tilde{\ell}_{\text{opt}} = \ell_{\text{opt}}$, due to the fact that interdicting any of the vertices from V_{prep} will not result in a better interdiction policy, because the largest clique that can be covered by interdicting vertices from V_{prep} is of size $\ell_{\min}$. This concludes the proof. □

Observe that the latter result allows us not only to fix the binary decision variables w_v to zero, but also to modify the input graph G by removing its vertices, resulting into a smaller input graph for solving the follower's subproblem. This reduction is safe due to the fact that removal of a vertex $v \in V_{\text{prep}}$ only reduces the size of maximal cliques K such that $|K| \leq \ell_{\min}$, which are never going to constitute an optimal follower's response to an optimal interdiction policy.

In a standard implementation, one would start with an arbitrary vertex, and solve the MCP by fixing this vertex to one. If the size of the obtained clique K is smaller than $\ell_{\min}$, vertex v is removed from G and the process is repeated for the remaining vertices. This procedure can be time consuming, as the maximum clique algorithm has to be called per each vertex. To overcome this drawback, degree of a vertex can be used instead, as a trivial upper bound on the clique number of a vertex v. However, according to inequality (1), a much more accurate bound is the coreness-number of a vertex. Hence, in our implementation, we pre-calculate the coreness-number for all vertices (in the initialization phase), and then remove all vertices v such that coreness$(v) \leq \ell_{\min}$.

3.3. Computing the global upper bound $\ell_{\max}$

We have implemented several construction heuristics in order to calculate a tight upper bound $\ell_{\max}$ and to create a pool of initial feasible solutions that are later given to the MIP solver. In all heuristics, if a vertex has been fixed to zero (e.g., removed by the preprocessing described above), it is not considered as a candidate for being interdicted. Our heuristics work in a greedy fashion, based on four different criteria: vertices are interdicted one-by-one, until the interdiction budget is exhausted. The chosen criteria are as follows:

- Vertex-degree: At the beginning, the vertices are sorted in non-increasing order according to their degrees. We start by interdicting the vertices with the highest degree first, and we stop once the interdiction budged is exhausted.
- Updated vertex-degree: The major difference to the "vertex-degree" heuristic is in the fact that now we recompute the vertex degrees, each time a vertex is interdicted.
- Vertex-coreness-number: at the beginning, the vertices are sorted in non-decreasing order according to their coreness number. Intuition behind this approach is that a vertex with a high coreness value is likely to belong to large cliques and shall be interdicted first. As mentioned above, coreness$(v)+1$ is a rough upper bound on the size of the maximum clique containing this vertex, $\omega_G(v)$.
- Vertex-color-number: This heuristic exploits the well-known result that the size of any feasible coloring gives the upper bound on the clique number of a graph (Balas and Yu 1986). We first apply the greedy sequential coloring heuristic based on independent sets, where color classes are obtained incrementally using bitmasks, see (San Segundo et al. 2011, 2013). This procedure runs in $O(|V|^2)$ in the worst case and returns a feasible coloring. We then assign a label (corresponding to the color number) to each vertex. In this sequential greedy coloring heuristic, the higher the color associated to a vertex, the higher are the chances that this vertex belongs to a large clique. Therefore, we interdict the vertices in the non-increasing order with respect to their color numbers. Notice that the reordering of vertices, based on their color number, once they have been interdicted,

will not influence the color number. So, there is no need to reorder the vertices after partially interdicting them, as long as they are interdicted starting with the highest color number first. As it will be shown in our computational study, vertex-color-number provides a very robust measure that delivers excellent heuristic solutions for sparse social networks.

4. Single-level ILP reformulation

In this section we provide an ILP formulation of the problem in the natural space of leader decision variables w. In addition, we provide a facial study of the underlying polytope, discussing under which conditions the proposed inequalities are facet defining.

PROPOSITION 6. *The following is a valid ILP formulation for CIG:*

$$\min \quad \theta \tag{12}$$

$$\theta + \sum_{u \in K} w_u \geq |K| \qquad K \in \mathcal{K} \tag{13}$$

$$\sum_{u \in V} w_u \leq k \tag{14}$$

$$w_u \in \{0,1\} \qquad u \in V. \tag{15}$$

Proof For every feasible interdiction policy $\bar{w} \in \mathcal{W}$, the follower's problem boils down to the following one, see (6) and (7):

$$\max_{x \in \mathcal{K}} \left\{ \sum_{u \in V} x_u \ : \ x_u \leq 1 - \bar{w}_u, \ u \in V \right\} = \max_{x \in \mathcal{K}} \ \sum_{u \in V} x_u (1 - \bar{w}_u)$$

Hence, the problem can be restated to that the set of feasible solutions of the follower does not depend on the actions of the leader anymore. Consequently, one can enumerate all cliques in G and optimize over the set $\mathcal{K}$. This is why the problem can be equivalently restated as

$$\min_{w \in \mathcal{W}} \left\{ \theta \ : \ \theta \geq \sum_{u \in V} \bar{x}_u (1 - w_u), \bar{x} \in \mathcal{K} \right\},$$

where $\bar{x}$ represents an arbitrary incident vector of a clique in G. This concludes the proof. □

The single-level ILP formulation contains an exponential number of constraints of type (13) that we will refer to as *Clique Interdiction* (CI) cuts. These constraints are NP-hard to separate: for each vector $\bar{w}$ and the associated $\bar{\theta}$ given by the current solution of the formulation (13) and (14) , checking if there exists a violated interdiction cut requires finding a maximum weighted clique on G with vertex-weights $c_u = 1 - \bar{w}_u$ for all $u \in V$.

In a branch-and-cut algorithm applied to the above ILP formulation, it is sufficient to separate integer infeasible points only (fractional points being cut off using standard branching and general cutting plane mechanisms embedded in modern MIP solvers). Whenever $\bar{w}$ is integer, the separation problem consists in solving the MCP in the support graph $G[V \setminus V_{\bar{w}}]$. Let $\bar{K}$ be the maximum clique in $G[V \setminus V_{\bar{w}}]$: if $|\bar{K}| > \bar{\theta}$ then a violated CI cut associated to $\bar{K}$ is found and added to the model. This separation procedure can be a potential bottleneck for using a branch-and-cut procedure, unless an efficient clique solver is used for the separation of CI cuts. We have therefore implemented a tailored separation algorithm based on recent state-of-the-art approaches for the MCP (see Section 5 for further details).

4.1. Valid inequalities and facial study

In the following, we study the polytope of the single-level CIG formulation (12)-(15). We provide necessary and sufficient conditions under which the clique interdiction cuts (13) are facet defining and we discuss heuristic lifting procedures designed to strengthen these inequalities.

Dimension of the Polytope. Given the graph G and the interdiction budget k, let $\mathcal{P}(G,k)$ denote the convex hull of feasible solutions of the CIG formulation (12)-(15), that is,

$$\mathcal{P}(G,k) = \text{conv}\left\{ w \in \{0,1\}^{|V|}, \theta \geq 0 : \theta + \sum_{u \in K} w_u \geq |K|, \sum_{u \in V} w_u \leq k, K \in \mathcal{K} \right\}.$$

Let (V', q) denote a CIG solution where $\theta = q$ and the inderdiction policy is defined by V'.

PROPOSITION 7. *The polytope $\mathcal{P}(G,k)$ is full dimensional.*

Proof We will prove this result by contradiction. Suppose that $\mathcal{P}(G,k)$ is contained in a hyperplane defined by the linear equation $\sum_{u\in V}\alpha_u w_u + \beta\theta = \gamma$, where $\alpha \in \mathbb{R}^{|V|}$, $\beta \in \mathbb{R}$ and $\gamma \in \mathbb{R}$. We show that $\alpha = 0$, $\beta = 0$ and thus $\mathcal{P}(G,k)$ cannot be included in this hyperplane, since it is not empty.

Let $v \in V$ be an arbitrary vertex of the graph G. The three solutions $(V_1 = \emptyset, |V|)$, $(V_v = \{v\}, |V| - 1)$ and $(V_v = \{v\}, |V|)$ are valid for $\mathcal{P}(G,k)$. We deduce that, $\alpha^{V_v} + \beta \cdot |V| = \alpha^{V_v} + \beta \cdot (|V| - 1)$ and thus $\beta = 0$. We also deduce that, $\alpha^{V_1} + \beta \cdot |V| = \alpha^{V_v} + \beta \cdot |V| \quad \Rightarrow \quad \alpha_v = 0$ for all $v \in V$. □

Facet-Defining Inequalities.

PROPOSITION 8. *Let $u \in V$. The trivial inequality $w_u \leq 1$ defines a facet of $\mathcal{P}(G,k)$ if and only if $k \geq 2$.*

Proof Let $u \in V$. If $k = 1$ then the inequality (14) dominates the inequality $w_u \leq 1$. For $k \geq 2$, consider the following $|V| + 1$ feasible solutions that belong to $\mathcal{P}(G,k)$ and satisfy the inequality $w_u \leq 1$ with equality: $(V_0 = \{u\}, |V|)$, $(V_1 = \{u\}, |V| - 1)$ and $(V_v = \{u, v\}, |V|)$ for all $v \in V \setminus \{u\}$. Clearly, these solutions are affinely independent, which concludes the proof. □

PROPOSITION 9. *Let $u \in V$. The trivial inequality $w_u \geq 0$ defines a facet of $\mathcal{P}(G,k)$.*

Proof Let $u \in V$. Consider the following $|V| + 1$ feasible solutions from $\mathcal{P}(G,k)$ that satisfy $w_u \geq 0$ with equality: $(V_0 = \emptyset, |V|)$, $(V_v = \{v\}, |V|)$ for all $v \in V \setminus \{u\}$ and $(V_1 = \{v'\}, |V| - 1)$ for some $v' \neq u$ Clearly, these solutions are affinely independent, so $w_u \geq 0$ defines a facet. □

LEMMA 1. *Let $K \in \mathcal{K}$ be an arbitrary clique in G. If $|K| \leq \ell_{\text{opt}}$, then the associated clique interdiction inequality* (13) *cannot define a facet.*

Proof Observe that, by definition, we have $\theta \geq \ell_{\text{opt}}$. Hence, for $|K| < \ell_{\text{opt}}$ the equality

$$\sum_{u\in K} w_u = |K| - \theta$$

has a negative right-hand side, i.e., there is no feasible solution satisfying this equation. On the other hand, for $|K| = \ell_{\text{opt}}$, the CI cut $\theta + \sum_{u\in K} w_u = |K|$ is dominated by $\theta \geq \ell_{\text{opt}}$. □

Even though the value of ℓ_{opt} is not known in advance, the above result is useful for the separation of clique interdiction cuts. The result states that the more promising inequalities (in terms of improving the quality of lower bounds) are those with $|K| \geq \ell_{\min}$, considering thereby the value of $\ell_{\min}$ as tight as possible. This is also in line with our preprocessing procedure that removes all vertices from G whose clique number is not greater than $\ell_{\min}$. The following result is stated without proof:

LEMMA 2. *Let $K \in \mathcal{K}$ be an arbitrary clique in G. The inequality $\theta + \sum_{u \in K} w_u \geq |K|$ defines a facet only if K is maximal.*

Hence, without loss of generality, in the remainder of this section we focus on maximal cliques only.

LEMMA 3. *Let K be a maximal clique and $v \in K$. If*

$$\omega(G[V \setminus V']) \geq |K| - |V' \cap K| + 1 \quad \forall V' \subseteq V \text{ where } v \in V' \text{ and } |V'| \leq k, \tag{16}$$

then there exists $\alpha_v \leq 0$ such that the associated clique interdiction cut (13) *can be down-lifted to*

$$\theta + \sum_{u \in K \setminus \{v\}} w_u + \alpha_v w_v \geq |K|.$$

Proof Let K be a maximal clique and let $v \in K$. The inequality $\theta + \sum_{u \in K} w_u \geq |K|$ is valid. Using a lifting procedure, see (Nemhauser and Wolsey 1988, p. 261), the coefficient α_v of w_v can be calculated as

$$\alpha_v = |K| - \min \left\{ \theta + \sum_{u \in K \setminus \{v\}} w_u | (w, \theta) \in \mathcal{P}(G, k) \text{ and } w_v = 1 \right\}.$$

Recall that to any feasible interdiction policy w, we can associate the set $V' \subset V$, $|V'| \leq k$ of vertices to be deleted from G. Hence the minimum value of the right hand side is equal to $\min_{V'}\{\omega(G[V \setminus V']) + |K \cap V'| - 1\}$ where the latter minimum is calculated over all feasible interdiction policies V' that contain v. According to our condition (16), the size of the maximum clique in the graph $G[V \setminus V']$ plus the number of interdicted vertices which are in K, including v, is $\geq |K| + 1$. Then, $\min\{\theta + \sum_{u \in K \setminus \{v\}} w_u | (w, \theta) \in \mathcal{P}(G, k) \text{ and } w_v = 1\} \geq |K|$ and therefore $\alpha_v \leq 0$. □

COROLLARY 2. *Let $K \subset V$ be a clique. If there exists $v \in K$ satisfying* (16) *then the inequality* (13) *cannot define a facet.*

Finally, the following Proposition provides necessary and sufficient conditions under which the CI cuts are facet defining. This is the major theoretical result of this section, which allows to characterize the strength of the ILP formulation upon which our solution framework is built on.

PROPOSITION 10. *Let $K \in \mathcal{K}$ be a maximal clique. Inequality* (13) *associated with K defines a facet of $\mathcal{P}(G,k)$ if and only if*

- $|K| \geq \ell_{\text{opt}} + 1$
- *for all $v \in K$, there exists a subset $V' \subseteq V$ such that $v \in V'$, $|V'| \leq k$ and $\omega(G[V \setminus V']) + |V' \cap K| \leq |K|$.*

Proof See Appendix. □

4.2. Heuristic lifting procedure

According to Lemma 3, it is NP-hard to down-lift coefficients of a clique interdiction cut. However, we can apply some heuristic ideas instead, by underestimating the left-hand-side and overestimating the right-hand-side of the condition (16).

Observe that the left-hand-side of this inequality can be calculated as:

$$\ell_{\text{opt}}(v) = \min\{\theta | (w,\theta) \in \mathcal{P}(G,k) \text{ and } w_v = 1\}$$

which is the value of the optimal CIG solution, assuming vertex v is interdicted. The value of $\ell_{\text{opt}}(v)$ can be underestimated by calculating $\ell_{\min}(v)$, which is the lower bound obtained using the formula provided in Corollary 1. However, a slight modification is needed, to ensure this bound is valid for the case vertex v is interdicted. To obtain the value of $\ell_{\min}(v)$, we first remove vertex v from G, and then we compute vertex-disjoint cliques $\mathcal{Q}_p$ on the fly, and stop as soon as we find p^* such that $k(\mathcal{Q}_{p^*+1}) > k-1$ (as the vertex v already consumes one unit of the interdiction budget). Notice that the values $\ell_{\min}(v)$ need to be calculated only once in the initialization phase, and they remain valid through the whole branch-and-cut procedure.

The right-hand-side of the condition (16) is simply overestimated by $|K|$ (knowing that $v \in V' \cap K$). This results in the following heuristic condition for down-lifting the coefficients of the clique-interdiction cuts that can be perfomed in a computationally efficient way.

PROPOSITION 11. *Let K be a maximal clique. Let LB be the lower bound in the current node of the branch-and-bound tree. If* $\max\{\ell_{\min}(v), \lceil LB \rceil - 1\} \geq |K|$ *then the down-lifted clique-interdiction cut*

$$\theta + \sum_{u \in K, u \neq v} w_u \geq |K|$$

is valid.

Proof The proof follows from the condition (16) which ensures that in that case α_v, the coefficient next to v in a valid clique-interdiction constraint is such that $\alpha_v \leq 0$. The left-hand-side of (16) is underestimated by $\max\{\ell_{\min}(v), \lceil LB \rceil - 1\}$. We have to subtract the value of 1 from LB, since we do not know if vertex v has been interdicted or not, when this bound is calculated. The right-hand-side of this condition is overestimated by $|K|$. □

Observe that such down-lifted clique interdiction cuts are only locally valid, and the globally valid ones can be similarly obtained by considering a global lower bound LB_g instead.

5. Solving the separation problem by tailoring the state-of-the-art clique solver

The separation problem requires solving the MCP in a number of induced subgraphs $G[V \setminus V_w]$, where V_w is a feasible interdiction policy. To solve each MCP exactly, we have designed a specific branch-and-bound (B&B) algorithm inspired by the ideas described in Li et al. (2017), San Segundo et al. (2016a). The latter algorithm (and so does ours) uses a compact bitstring representation both for vertex sets and the adjacency matrix of the input graph.

Algorithm 1 shows the outline of our exact clique solver referred to as `IMCQ`. `IMCQ` is a purely combinatorial approach that exploits tight lower and upper bounds for the clique number, along with efficient branching rules. It receives as input the original graph $G = (V, E)$ and the set of interdicted vertices V_w. Steps 1-5 refer to the preprocessing procedure which is described in Section 5.1.

In this preprocessing phase, the input graph is modified (by calling the `Precondition` procedure), and a feasible solution K_0 and a tight upper bound ub are calculated. If it turns out that the two obtained bounds overlap, the algorithm terminates, otherwise it enters the B&B phase (cf. Step 6-11) which is described in Section 5.3.

5.1. Preprocessing

The clique solver works on an isomorphic H that is obtained from G by reordering vertices according to the type of input graph. This ordering is essential for the efficient use of cache memory for bitstring graph representations and for reducing the size of the search tree, see (San Segundo et al. 2011, 2013). We distinguish between two different categories of input graphs: large-scale sparse graphs and small or middle-size ($n < 5,000$) but dense graphs. In the remainder of the paper, these categories will be referred to as *sparse* and *non-sparse*, respectively. Depending on the type of graph, different actions are undertaken concerning branching, computation of the bounds at the root node and preconditioning.

Preconditioning for sparse graphs: Vertices are sorted according to non-decreasing coreness in a degeneracy ordering using the algorithm of Batagelj and Zaversnik (2011) which has complexity $O(|V| + |E|)$ and is specially suited for this type of graphs.

Preconditioning for non-sparse graphs: Vertices are sorted according to their degrees. In particular, let $O = \{v_1 \prec v_2 \prec \cdots \prec v_n\}$ be the original ordering of vertices of G. In the new ordering $O' = \{v'_1 \prec v'_2 \prec \cdots \prec v'_n\}$, a vertex $v_l \in O$ with maximum degree in V is taken to become v'_n in the new ordering, then a second vertex with maximum degree in $V \setminus \{v_l\}$ is placed at v'_{n-1} and so on until all vertices in O are sorted, see, e.g., San Segundo et al. (2016a). Consequently, vertices placed at the beginning in O' are expected to have a high degree, and there exists a number $r \geq 2$ such that the first r vertices $\{v'_1, v'_2, \ldots, v'_r\}$ form a clique. Finally, similar to the sparse case, an isomorphic graph H is built according to O'.

The newly obtained graph H is bit-encoded in memory following (San Segundo et al. 2011) and it becomes the input graph in future steps. Further preprocessing for any induced graph $H[V \setminus V_w]$

is reduced to computing an initial upper bound on its clique number, an initial clique and a branching candidate set B_0. In the case of sparse graphs, we use the upper bound 1 + core(G) for all interdicted graphs, where core(G) is the core number of G. Note that for sparse graphs core(G) has already been obtained during the computation of H (it is the coreness of the first vertex in H); it is not recomputed in subsequent interdicted graphs for efficiency reasons. In the case of non-sparse graphs, the core is recomputed for all subsequent interdicted graphs and the corresponding upper bound obtained as $\min\{\omega(G), 1 + \text{core}(H[V \setminus V_w]\}$.

For computing an initial solution K_0, we use the simple greedy heuristic of taking $v_1 \in H$ as the first vertex of clique K_0 and then, at each iteration, enlarge K_0 with the first possible vertex $v_l \in H$ such that $N(v_l) \supseteq K_0$. Note that, in the case of sparse graphs, K_0 will be reasonably large because vertices are sorted according to non-increasing coreness.

The last preprocessing step is to determine an initial branching set of vertices $B_0 \subset V$, which are the candidate vertices who may belong to a clique larger than $|K_0|$. This is done in the `GetBranchingSet` procedure, which is also called later for every node of the search tree by the recursive search algorithm `MCQ`. This first call to `GetBranchingSet` differs slightly from the rest of calls (in `MCQ`), so it is denoted $\texttt{GetBranchingSet}_0$. Both `GetBranchingSet` and $\texttt{GetBranchingSet}_0$ are described in detail in Section 5.2. If the branching set B_0 is empty, K_0 is returned as the optimal solution; otherwise the recursive search procedure `MCQ` is called to improve $|K_0|$. Procedure `MCQ` is described in Section 5.3.

5.2. The branching procedure

Given an input graph G and an integer q (corresponding to a valid lower bound on MCP), it is possible to partition V into two disjoint sets of vertices P and $B = V \setminus P$ such that $\omega(G[P]) \leq q$, so that the optimal MCP solution can be found by branching on the vertices from B only. Reducing the size of the set B is crucial for an efficient implementation of a B&B-based MCP solver, see e.g., (San Segundo and Tapia 2014, San Segundo et al. 2015, 2016b, Li et al. 2017). Determining the set B is done by the novel `GetBranchingSet` procedure, which is an improvement of the procedure

Algorithm 1: `IMCQ`$(G,\ V_w)$

Input: A graph $G=(V,E)$; V_w a set of vertices to be removed from G

Output: A maximum clique in K_{max}

1 **if** $V_w=\emptyset$ **then** $H \leftarrow$ `Precondition`(G)

2 **else** $H \leftarrow G$

3 $ub \leftarrow$ `ComputeUB` $(H[V \setminus V_w])$

4 $K_0 \leftarrow$ `InitialClique` $(H[V \setminus V_w])$

5 **if** $|K_0|=ub$ **then return** K_0

6 $B_0 \leftarrow$ `GetBranchingSet`$_0$ $(H[V \setminus V_w], |K_0|)$

7 **if** $B_0 \neq \emptyset$ **then**

8 $\quad K_{\max} \leftarrow$ `MCQ` $(\ H[V \setminus V_w], V \setminus V_w, \emptyset, K_0, B_0)$

9 $\quad$ **if** $|K_{\max}| < |K_0|$ **then** $K_{max} \leftarrow K_0$

10 **else** $K_{max}=K_0$

11 **return** K_{max}

from (San Segundo et al. 2015), see Algorithm 2. `GetBranchingSet` receives an input graph and an integer q and computes a set of vertices that cannot contain a clique greater than q. The remaining vertices make up the branching set B outputted by the procedure.

A first set B is derived by the `Color` function (Step 2), which implements the greedy sequential coloring heuristic based on independent sets which returns a κ-coloring (San Segundo et al. 2011, 2013). If κ is strictly greater than q, the branching set is determined by all the vertices in color classes with a color number higher than q; otherwise the procedure returns an empty branching set, since it is clear that in the induced subgraph $G[C_1, C_2, \ldots, C_\kappa]$, $\kappa \leq q$, no clique larger than q can exist.

The reduced size of this first branching set B compared to the original vertex set is already enough to reasonably capture the structure, and thus efficiently prune the search space for the large sparse graphs considered in this work. For the non-sparse graphs, we use a so called *infrachromatic* bounding function to further reduce the cardinality of the branching set B (Steps 6-8). The term infrachromatic was first employed in this context in (San Segundo et al. 2015), and refers

Algorithm 2: `GetBranchingSet` (G, q)

Input: A graph $G = (V, E)$, an integer q

Output: A set B of branching vertices

$B \leftarrow \emptyset$

$C = \{C_1, C_2, \ldots, C_\kappa\} \leftarrow$ `Color`(G)

if $\kappa \leq q$ **then return** B

$B \leftarrow V \setminus \{C_1, C_2, \ldots, C_q\} = \{C_{q+1}, C_{q+2}, \ldots, C_\kappa\}$

if is_sparse(G) **then return** B

$P \leftarrow \{C_1, C_2, \ldots, C_q\}$

forall $b_i \in B = \{b_1, b_2, \ldots, b_{|B|}\}$ *in increasing order* **do**

  if *is_conflicting*$(P \cup \{b_i\})$ **then** $P \leftarrow P \cup \{b_i\}, B \leftarrow B \setminus \{b_i\}$

return B

to any bounding function that can, under certain conditions, compute a bound for the clique number of a graph below its chromatic number. A simple infrachromatic procedure was given in (San Segundo et al. 2015). In it, the triplet composed by two disjoint color sets (C_1, C_2) and a vertex v, considered a singleton third color set and disjoint with the other two, was denoted *conflicting* if the induced graph $G[C_1 \cup C_2 \cup \{v\}]$ contains no triangles. `GetBranchingSet` makes use of a novel and more sophisticated infrachromatic procedure, denoted by `is_conflicting`, which may be called a maximum of $|B|$ times. The initial call takes as input the set $P = V \setminus B$ (made up of the first q color classes provided by `Color`) together with the first vertex b_1 from the branching set $B = \{b_1, b_2, \ldots, b_{|B|}\}$; it returns *true* if the induced subgraph $G[P \cup b_1]$ cannot contain a clique larger than q and, if this is the case, moves the vertex b_1 from B to P. Consequently, the enlarged $P \cup b_1$ set becomes the new reference set for future calls. In this scenario, the next call to `is_conflicting` would attempt to prove that $G[(P \cup \{b_1\}) \cup \{b_2\}]$ cannot contain a clique of size q and so on. The procedure terminates once a vertex b_l is found such that we are unable to prove that $G[P \cup \{b_1, b_2, \ldots, b_l\}]$ satisfies the property, i.e., when `is_conflicting` returns *false*, in which case the `GetBranchingSet` outputs $\{b_l, b_{l+1}, \ldots, b_{|B|}\}$ as branching set. The procedure `is_conflicting`

is an efficient bitstring implementation of the partial MaxSAT bound described, among others, in (Li et al. 2017). It has a worst-case complexity of $O(|V|^3)$, which makes it unsuitable for the category of sparse graphs considered in this paper.

`GetBranchingSet` is called by the recursive search algorithm `MCQ`. During preprocessing, $\texttt{GetBranchingSet}_0$ differs from `GetBranchingSet` as follows: in the case of sparse graphs, where the coreness-number of each vertex is known, $\texttt{GetBranchingSet}_0$ first removes from V, prior to the call to `Color`, any vertex $v \in V$ such that $q \leq 1 + \text{coreness}(v)$. The remaining steps are the same as in `GetBranchingSet`. Note that these vertices can be safely removed since, by definition of coreness, they cannot make part of any clique of size greater than q. Moreover, the set of vertices removed by $\texttt{GetBranchingSet}_0$ are propagated to the recursive search procedure, so that `MCQ` does not take as input $V \setminus V_w$, but the reduced set $(V \setminus V_w) \cap \{u \in V \ : \ 1 + \text{coreness}(u) > q\}$ instead. With respect to non-sparse graphs, $\texttt{GetBranchingSet}_0$ is exactly the same as `GetBranchingSet`.

5.3. The recursive search procedure

Once the starting branching set of vertices B_0 is determined, we launch a recursive branch-and-bound search procedure `MCQ`, which is outlined in Algorithm 3. As an input, the procedure requires: U, which is the set of vertices of the induced subproblem $G[U]$ at the current B&B node; B, which is the branching candidate set ($B \subseteq U$); S, which represents the set of vertices of the clique under construction (vertices which are fixed to one along the B&B path, $S \cap U = \emptyset$); and S_{max} which is the incumbent clique (the largest clique found at any point during search).

The procedure is inspired by the notion of *selective coloring* (also known as *relaxed coloring*, see San Segundo and Tapia (2014)) in which a sequential coloring greedy heuristic is used to produce a partial coloring (instead of a feasible coloring) of a certain size q. The value of q is equal to the gap between an incumbent solution $|S_{max}|$ and the size of the clique S under construction at the current B&B node, i.e., $q = |S_{max}| - |S|$. The algorithm then explores only the vertices that do not belong to this partial coloring. Among these vertices, `MCQ` uses the more sophisticated `GetBranchingSet` function to determine a set of candidate vertices to branch on at the next level of the B&B tree.

In the first call to `MCQ`, S is the empty set, S_{max} is the initial clique K_0, $q = |K_0|$ and B_0 is the branching candidate set as determined by the $\texttt{GetBranchingSet}_0$ procedure. As explained in Section 5.1, the initialization of set U depends on the type of graph: in the case of non-sparse graphs $U = V \setminus V_w$, whereas for sparse graphs $U = (V \setminus V_w) \cap \{u \in V \ : \ 1 + \text{coreness}(u) > q\}$.

For every new call to `MCQ`, the algorithm enlarges S when branching on a vertex $b \in B$ (Step 6), so that $S \leftarrow S \cup \{b\}$, and computes the corresponding child node set U_b (Step 2) using efficient bitmasks, see (San Segundo et al. 2013). $G[U_b]$ is then processed by `GetBranchingSet` to determine a fresh candidate subset B_b for the child node. Note that $B_b \subseteq U_b$ is such that at least one of its vertices must make part of any clique larger than $|S_{max}| - |S|$, i.e. improve the incumbent solution. On backtracking, `MCQ` removes b from U to avoid spurious combinations of subproblems.

Algorithm 3: `MCQ`(G, U, S, $S_{\max}$, B) the recursive search algorithm

Input: A graph $G = (V, E)$; vertices $U \subset V$ to be explored; clique under construction S; incumbent clique $S_{\max}$; branching set $B = \{b_1, b_2, \ldots, b_{|B|}\} \subseteq U$

Output: $S \cup S'$, where S' is a maximum clique in $G[U]$, if $|S \cup S'| > |S_{max}|$; otherwise $S_{\max}$

```
1  for b := b_|B| downto b_1 do
2      U_b ← U ∩ N(b)
3      if U_b ≠ ∅ then
4          B_b ← GetBranchingSet(G[U_b], |S_max| − |S|)
5          if B_b ≠ ∅ then
6              S ← S ∪ {b}
7              S_1 ← MCQ(G, U_b, S, S_max, B_b)
8              if |S_1| > |S_max| then S_max ← S_1
9              S ← S \ {b}
10     U ← U \ {b}
11 return S_max
```

6. Computational results

The purpose of this computational study is threefold: (1) to compare the exact solution framework for CIG presented in this paper with the state-of-the-art method available for general bilevel and interdiction problems; (2) to assess the practical applicability and the limits of our framework on large-scale social networks, and (3) to use this framework to analyze the resilience of (social) networks with respect to vertex-interdiction attacks, i.e., the decrease of the size of the maximum clique in function of incremental interdiction budget levels k.

In the following we first summarize our exact solution framework, followed by the explanation of benchmark instances and a detailed computational study. All the experiments have been performed on a computer with a 3.40 Ghz 8-core Intel Core i7-3770 processor and 16Gb RAM, running a 64-bit Linux operating system. The source codes were compiled with `gcc 4.8.4` and `-O3` optimizations. We used `CPLEX 12.7.0` and the `CALLABLE LIBRARIES` framework to implement our branch-and-cut algorithm. `CPLEX` was run in single-threaded mode and all `CPLEX` parameters were set to their default values.

6.1. The exact solution framework `CLIQUE-INTER`

Our exact solution framework is based on the CIG formulation (12)-(15) of Section 4. It is a branch-and-cut procedure combined with a preprocessing and efficient combinatorial algorithms for the computation of lower and upper bounds, and for the separation procedure. The framework is called `CLIQUE-INTER` in the remainder of the paper. These are its main components:

(i) An effective separation procedure of the CI cuts (13): To this end we apply the specialized combinatorial B&B algorithm (`IMCQ`) described in Section 5. `IMCQ` is an enhancement of one of the state-of-the-art MCP algorithms of San Segundo et al. (2016a) capable of effectively solving the MCP once the vertices of an interdiction policy have been removed from the graph G. In addition, sharper cuts are obtained by *maximal* cliques (see Lemma 1); for this reason, before adding each CI cut, we make the associated clique maximal, i.e., potentially enlarging it with vertices that have been interdicted.

(ii) Tight CIG upper and lower bounds (ℓ_{min} and ℓ_{max}): In order to initialize the lower bound value of the variable θ of formulation (12)-(15), we used the global lower bound ℓ_{min} presented in Section 3.1. We create the subset $\mathcal{Q}_{p+1}$ required by Corollary 1 by iteratively extracting vertex-disjoint maximum cliques using `IMCQ` from G until we find the first p^* such that $k(\mathcal{Q}_{p^*+1}) > k$. The cliques from the set $\mathcal{Q}_{p^*+1}$ are also used to create an initial pool of CI cuts. In order to define a high-quality feasible CIG solution of value ℓ_{max} to initialize formulation (12)-(15), we apply a battery of sequential greedy heuristics presented in Section 3.3. If it turns out that $\ell_{min} = \ell_{max}$, the instance is solved to optimality, before even starting the B&C procedure.

(iii) The graph reduction: For large-scale real-world graphs, formulation (12)-(15) becomes easily intractable, unless the input graph can be safely reduced to a smaller one. The reduction technique presented in Section 3.2, which exploits the tight lower bound $\ell_{\min}$ and preserves the optimality of the solution, is applied before calculating an initial upper bound $\ell_{\max}$ and entering the B&C phase.

As later shown in this computational section, all three components are crucial and necessary to guarantee an efficient computational performance of `CLIQUE-INTER`.

6.2. Test-bed of instances

In order to asses the performance of the newly proposed exact algorithm `CLIQUE-INTER`, we consider three sets of benchmarks instances, called *Set A*, *Set B* and *Set C*:

Set A – Random graphs. We created a set of 55 Erdős-Rényi random $G(n,p)$ graphs of different sizes ($n = |V| \in \{50, 75, 100, 125, 150\}$) and edge densities ($p \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.98\}$). We set k to four different percentages of $|V|$, i.e., $k \in \{\lceil 0.05 \cdot |V| \rceil, \lceil 0.1 \cdot |V| \rceil, \lceil 0.2 \cdot |V| \rceil, \lceil 0.4 \cdot |V| \rceil\}$. In this way, we created the Set A of 220 instances.

Set B – Synthetic graphs. This set contains 16 well known synthetic instances with $|V| = 200$ from the 2nd DIMACS challenge on Maximum Clique, Graph Coloring, and Satisfiability (DIMACS2). Most of them are still employed as test-bed by state-of-the-art exact MCP algorithms.

Set C – Real-world networks. These instances have been taken from the 10th DIMACS Implementation Challenge - Graph Partitioning and Graph Clustering and SNAP database. We consider instances with up to $\approx 100,000$ vertices and $\approx 3,200,000$ edges. The reported 30 networks all fall into one of the following categories:

- Social Networks (with vertices representing individuals) such as: *Interaction networks*, where the edges refer to interaction via message posts, for example e-mail, as in *ia-email-EU*; *Recommendation networks*, where the edges indicate communication in terms of a recommendation or an opinion (e.g., *rec-eachmovie* related to movie opinions); *Collaboration networks*, where edges represent scientific or other type of collaborations (e.g., *astro-ph* refers to preprints in the astrophysics archive, and *cond-mat-*$*$ refer to the condensed matter archive `www.arxiv.com`). In some other social networks, edges refer to relationships of different types, such as friendship (e.g. Facebook, *socfb*), who-trusts-whom network (*Soc-epinioins1*), bloggers-interactions (*soc-BlogCatalog*), etc.
- Technological networks: In this case vertices are routers and edges represent communications between them, as in *tech-internet-as*.
- Scientific computing networks: These networks are meshes that are derived from mathematical analysis in different fields of science, i.e. finite elements (*fe-*$*$), and other types (*sc-*$*$ etc.).

6.3. Computational performance on random graphs of different size and densities

The purpose of this section is to demonstrate the effectiveness of the main components of `CLIQUE-INTER` and to determine the relative impact on the computational difficulty of the three main instance features, i.e., the number of vertices of a graph ($|V|$), the edge density of a graph ($\mu = \frac{2|E|}{|V|\cdot(|V|-1)}$) and the amount of available interdiction budget (k). To this end, we use 220 random instances of Set A and we set a *time limit* of 600 seconds of CPU time for each run.

Recall that `CLIQUE-INTER` is the default setting of our exact framework, fully exploiting all its components (see Section 6.1). We test four additional configurations of `CLIQUE-INTER`, removing some of its components and measuring their impact on the computational performance:

1. `CLIQUE-INTER` (no bounds): in this configuration, all the ingredients are turned on, except for the calculation of combinatorial upper and lower bounds (ℓ_{min} and ℓ_{max}).

2. `CLIQUE-INTER` (no maximality): in this configuration, all the ingredients are turned on, except that we did not make the cliques maximal before adding the corresponding CI cuts.

3. Basic `CLIQUE-INTER` with `IMCQ`: in this configuration all components are removed, except the use of `IMCQ` to separate CI cuts.

4. Basic `CLIQUE-INTER` with `CPLEX`: this configuration corresponds to the basic B&C approach in which CI cuts are separated using `CPLEX` as a black-box clique solver applied to the classical clique ILP formulation with constraints associated to non-neighbour vertices. This configuration corresponds to the general implementation for interdiction games under monotonicity as given in Fischetti et al. (2016).

In order to give a graphical representation of the relative performance of the different configurations of `CLIQUE-INTER`, we report a performance profile in Figure 2. For each instance, we compute a normalized time τ as the ratio of the computing time of the considered configuration over the minimum computing time for solving the instance to optimality. For each value of τ in the horizontal axis, the vertical axis reports the percentage of the instances for which the corresponding configuration spent at most τ times the computing time of the fastest configuration. The curves start from the percentage of instances in which the corresponding configuration is the fastest and at the right end of the chart, we can read the percentage of instances solved by a specific `CLIQUE-INTER` configuration. The best performance are graphically represented by the curves in the upper part of Figure 2 and the horizontal axis is represented in logarithmic scale.

Figure 2 clearly shows that each of the components is necessary to achieve the best computational performance. `CLIQUE-INTER` is the fastest for 40% of the instances and it manages to solve to proven optimality almost 90% of the instances of Set A. From the figure, we can also see that the impact of separating CI cuts using `IMCQ` is crucial. Disabling the improvements based on $\ell_{\min}$ and $\ell_{\max}$ and not inserting "maximal" CI cuts have a remarkable detrimental effect, which is why the

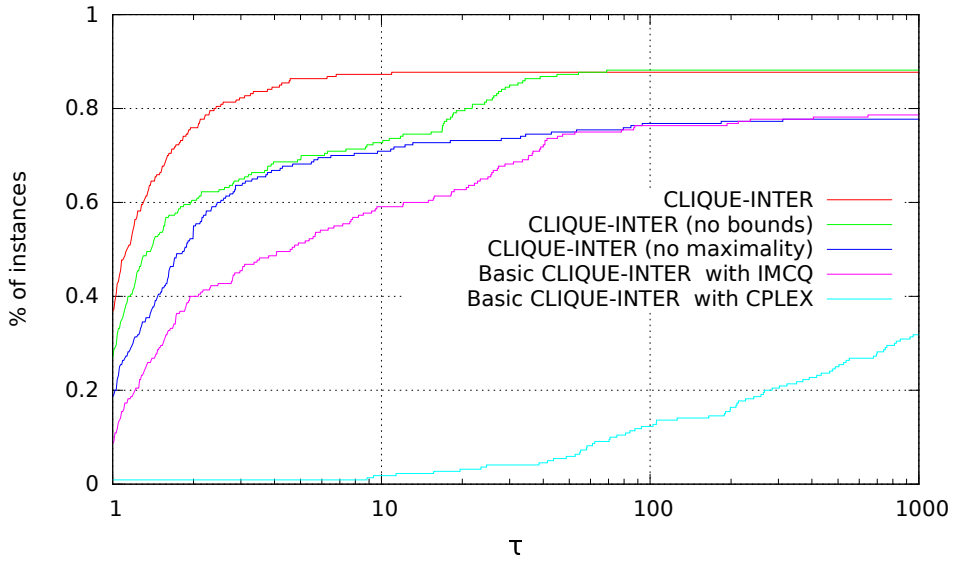

Figure 2 Performance profile of the different configuration of `CLIQUE-INTER`.

basic `CLIQUE-INTER` with `IMCQ` has a very poor computational performance. The computation cost of $\ell_{\min}$ and $\ell_{\max}$ is not negligible. For this reason, for the very small instances which are easily solvable, removing this computation can be somehow beneficial; for the most difficult instances, however, the update of the lower and upper bounds is crucial. The effects of the graph reduction technique for Set A were rather moderate (so we do not report these results separately), however this technique turns out to be inevitable for large-scale real-world networks (cf. Table 3), which is why we have decided to keep it in the default `CLIQUE-INTER` setting.

We now discuss the impact on the CPU time of the instance features using the best configuration of `CLIQUE-INTER`. In Figures 3 and 4, we report the box plots of the computational times grouping the instances by edge density (Figure 3) and by graph size ($|V|$) and by interdiction budget (k) (Figure 4). In these figures, we graphically represent the computing times (in logarithmic scale) through their quartiles; the lines extending vertically from the boxes indicate the variability outside the upper and lower quartiles. Finally the outliers are plotted as individual points. In the upper part of both figures, we report the number of instances solved to proven optimality (#OPT) for each group of instances with similar characteristics and the number of instances solved by preprocessing (#PREP), i.e., when $\ell_{\min} = \ell_{\max}$. In the latter case, an optimal solution is computed without

tackling formulation (12)-(15). The number of instances solved by preprocessing is a proxy of the quality of both the CIG upper and lower bounds proposed in this paper.

In Figure 3, we have 11 groups of 20 instances each. We can observe that the computational time increases with the edge density up to a density value of 0.9 and then it starts to decrease. All instances with up to density 0.3 can be solved to proven optimality by `CLIQUE-INTER`. The hardest instances are the ones with density 0.8 and 0.9, where `CLIQUE-INTER` is able to solve 15 out of 20 instances in each group. Several instances with low density can be solved by preprocessing, e.g., 11 for density 0.1 and 5 for density 0.2. None of the instances of density 0.8 and 0.9 can be solved by preprocessing. All instances with up to density 0.3 can be solved with an average CPU time less than 0.1 seconds. For all the other density classes the average CPU times of the solvable instances do not outmatch 100 seconds, but some instances can take up to several hundreds seconds.

In Figure 4, we have 20 groups of 11 instances each. As expected, we can observe that the computational time increases according to the number of vertices ($|V|$). In addition, the computational time is also directly correlated with the level of interdiction budget (k). All the instances with 50 and 75 vertices can be solved to proven optimality, independently of k. The figure shows that more instances can be solved by preprocessing for low levels of interdiction budget, e.g., for instances with 50 vertices, 7 instances can be solved by preprocessing when $k = \lceil 0.05 \cdot |V| \rceil$ and only 2 when $k = \lceil 0.4 \cdot |V| \rceil$. All the instances with 50 vertices are solved in less than 1 second and all the instances with 75 vertices are solved in less than 100 seconds. All instances with low values of k ($k = \lceil 0.05 \cdot |V| \rceil$ and $k = \lceil 0.1 \cdot |V| \rceil$) can be solved to proven optimality in less than 1 and 100 seconds, respectively.

6.4. Comparison with the state-of-the-art bilevel solver

In Table 1, we compare the results of the state-of-the-art bilevel and interdiction game solver from Fischetti et al. (2017) (called `BILEVEL`) with our new approach `CLIQUE-INTER`. Each row corresponds to 44 instances of Set A grouped by the number of vertices $|V| \in \{50, 75, 100, 125, 150\}$. For this test we used the same time limit of 600 seconds for each run. For each of the two solvers,

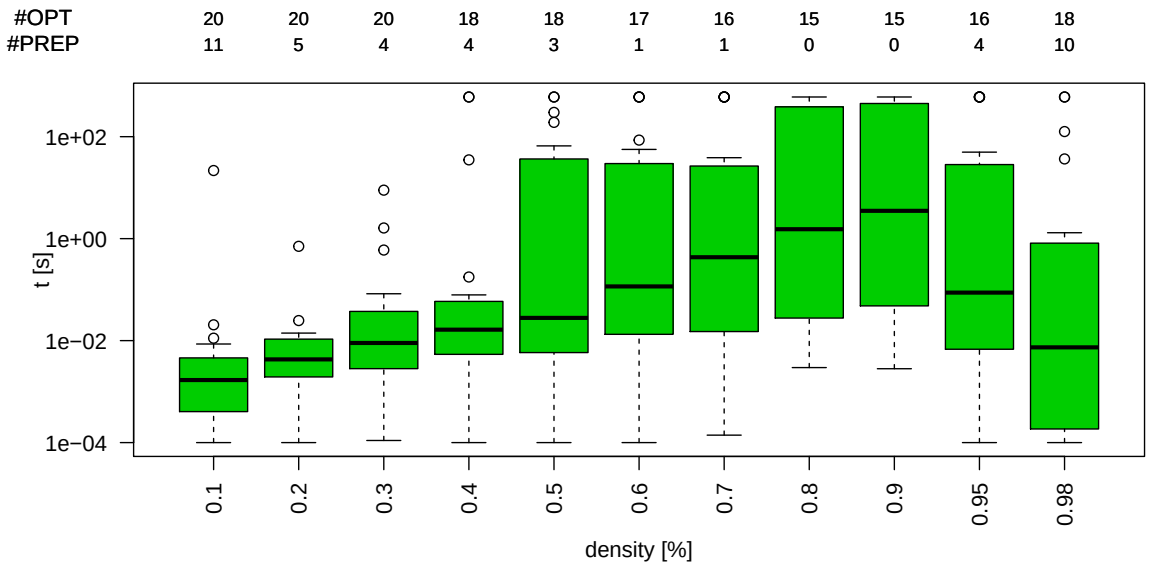

Figure 3 The total CPU times on the random graphs of Set A, grouped by the graph density.

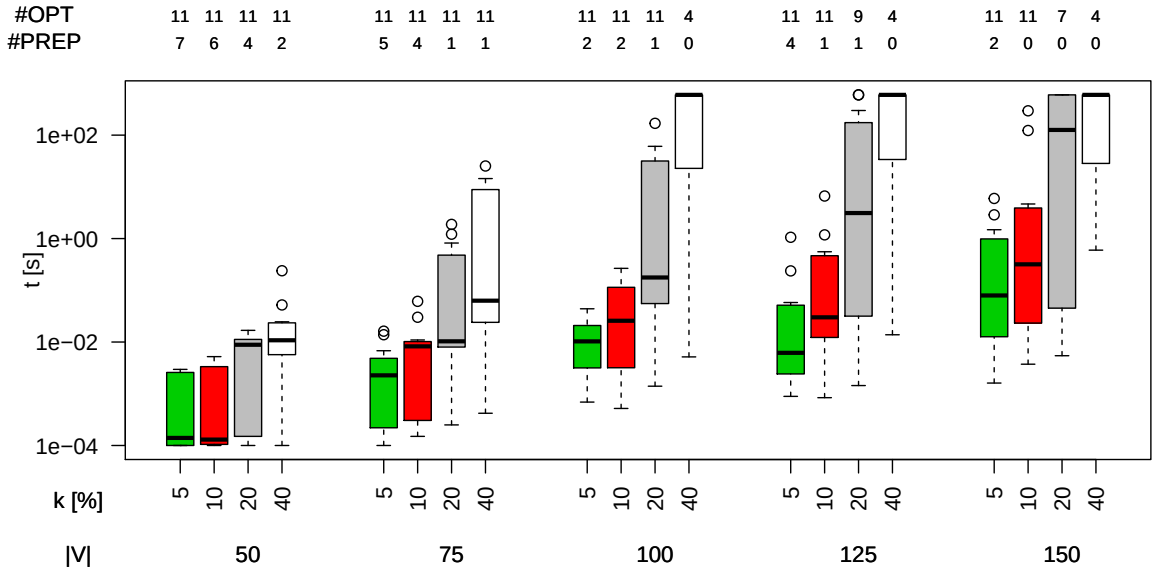

Figure 4 The total CPU times on the random graphs of Set A, grouped by the network size and the interdiction budget.

we report the following values: the number of solved instances per group (#solved), the average computing time in seconds for those instances that were solved to optimality, the average exit gap after the time limit is reached (considering only those instances which were *not* solved to optimality)

		CLIQUE-INTER				BILEVEL			
$\|V\|$	#	# solved	time	exit gap	root gap	# solved	time	exit gap	root gap
50	44	**44**	**0.01**	-	0.16	28	68.58	6.44	8.50
75	44	**44**	**1.45**	-	0.41	14	120.19	9.47	10.91
100	44	**37**	**9.30**	1.00	0.98	7	164.42	12.65	13.11
125	44	**35**	**13.43**	1.33	1.20	2	135.33	13.88	14.73
150	44	**33**	**27.23**	1.91	1.43	1	397.52	16.42	16.39

Table 1 Comparison with state-of-the-art bilevel solver (BILEVEL) from Fischetti et al. (2017) and our approach (CLIQUE-INTER).

and the average root gap (over all instances). The exit gap is calculated as exit gap $=$ UB $-\lceil$LB$\rceil$, where UB refers to the global upper bound computed by the respective method, and LB refers to the global lower bound of the same method. In order to measure the quality of lower bounds at the root node of the B&C tree (denoted by LB_r), we compute them with respect to the best known solution (BKS) as root gap $=$ BKS $-\lceil \text{LB}_r \rceil$.

Table 1 demonstrates that our new approach greatly outperforms the general-purpose bilevel solver of Fischetti et al. (2017) by several orders of magnitude. All instances with 50 and 75 vertices are solved to optimality in less than 2 seconds on average by CLIQUE-INTER, whereas the bilevel solver manages to solve only 2/3, respectively 1/3 of them. Similar behavior can be observed for larger instances, where, for example, for graphs with 150 vertices the bilevel solver manages to solve only a single instance in more than 5 minutes, whereas we solve 70% of them, in less than 1/2 minute on average. These results can be easily explained by the quality of root bounds: whereas the average absolute gaps at the root node for our approach are between 0.16 and 1.4, those of BILEVEL range between 8.5 and 16.4. This has a strong impact on the size of the B&C tree. For example, for 28 instances of size 50 solved by both approaches, CLIQUE-INTER requires an average number of 1.5 B&C nodes, compared to 1960 nodes required by BILEVEL.

6.5. Performance of CLIQUE-INTER on DIMACS2 graphs and large real-world networks

In Table 2, we show the performance of CLIQUE-INTER on the dense graphs of Set B. All these instances have 200 vertices and the table reports their density (μ), the size of maximum cliques

($\omega(G)$) and the CPU time necessary to compute it using `IMCQ` (time$_\omega$). We test two levels of interdiction budget, i.e., $k = 20$ and $k = 40$. For these tests we set a time limit of 3600 seconds and T.L. is reported for the instances not solved to proven optimality. For each value of k, we report the value of the final lower and upper bounds (LB and UB), the total CPU time spent in solving formulation (12)-(15) and the values of the initial lower and upper bounds (ℓ_{min} and ℓ_{max}). In case an instance is solved to proven optimality then LB=UB, in case an instance is solved during the preprocessing $\ell_{min} = \ell_{max}$. In this latter case, we do not report the CPU time since it is not necessary to tackle formulation (12)-(15), and the time necessary to compute ℓ_{min} and ℓ_{max} is negligible. The time taken by `IMCQ` to find $\omega(G)$ for the instances of Set B is often negligible, except for `sanr200_0.9` , where it takes 1.9 seconds. `CLIQUE-INTER` is able to solve 12 and 8 instances out of 14 instances, for $k = 20$ and $k = 40$, respectively. From the table, we can see that the gap between ℓ_{min} and ℓ_{max} is small for all instances except in a few cases, e.g., for `san200_0.9_1` and `san200_0.9_2` and $k = 40$ where the gap is 10 and 8, respectively. In most of the remaining instances the initial gap is less than 3, showing that also for this set of instances ℓ_{min} and ℓ_{max} provide good-quality lower and upper bounds. For $k = 20$, three instances can be solved to proven optimality only by preprocessing and, for $k = 40$, two instances are solved by preprocessing. As already observed on the instance Set A, increasing the level of budget increases the computational difficulty as well.

In Table 3, we show the performance of `CLIQUE-INTER` on the large real-world networks of Set C. As discussed in Section 6.2, this set is composed by 30 sparse graphs with up to 100K vertices and 3200K edges. The table reports for each instance the number of vertices ($|V|$), the number of edges ($|E|$), the size of maximum clique ($\omega(G)$) and the CPU time necessary to compute it using `IMCQ` (time$_\omega$). For these tests, we used a time limit of 3600 seconds for each run. Since these graphs are large, to test realistic levels of interdiction budget, we set $k = \lceil 0.005 \cdot |V| \rceil$ and $k = \lceil 0.01 \cdot |V| \rceil$. For this set of instances the time to compute ℓ_{min} is not always negligible and we report it in the column time$_p$. As explained in Section 3.1, ℓ_{min} is computed starting from a set $\mathcal{Q}_p = (C_1, \ldots, C_p)$

				CLIQUE-INTER $k=20$					CLIQUE-INTER $k=40$				
	μ	$\omega(G)$	time$_\omega$	LB	UB	time	$\ell_{\min}$	$\ell_{\max}$	LB	UB	time	$\ell_{\min}$	$\ell_{\max}$
brock200_1	0.75	21	0.2	18	18	938.2	16	18	15	17	T.L.	13	17
brock200_2	0.50	12	0.0	9	9	0.1	8	10	8	9	T.L.	7	9
brock200_3	0.61	15	0.0	12	12	1.0	11	13	11	11	160.6	9	12
brock200_4	0.66	17	0.0	14	14	2421.8	12	15	12	13	T.L.	10	13
c-fat200-1	0.08	12	0.0	10	10	-	10	10	9	9	-	9	9
c-fat200-2	0.16	24	0.0	20	20	-	20	20	18	18	-	18	18
c-fat200-5	0.43	58	0.0	52	52	0.0	51	52	46	46	0.0	44	46
san200_0.7_1	0.70	30	0.0	17	17	5.4	16	18	15	15	134.4	14	17
san200_0.7_2	0.70	18	0.0	14	14	16.7	13	15	12	12	5.6	11	15
san200_0.9_1	0.90	70	0.0	50	50	-	50	50	40	40	13.3	39	49
san200_0.9_2	0.90	60	0.1	41	41	3.2	41	42	34	34	2266.9	33	41
san200_0.9_3	0.90	44	0.0	33	34	T.L.	32	37	28	31	T.L.	26	34
sanr200_0.7	0.70	18	0.1	15	15	29.2	14	16	13	14	T.L.	11	15
sanr200_0.9	0.90	42	1.9	33	35	T.L.	31	35	28	32	T.L.	25	33
gen200_p0.9_44	0.90	44	0.1	34	34	674.4	32	38	29	31	T.L.	26	36
gen200_p0.9_55	0.90	55	0.1	38	38	62.4	37	41	32	33	T.L.	29	40

Table 2 Computational results obtained by the CLIQUE-INTER on the instances with $|V| = 200$ from the 2nd DIMACS Challenge DIMACS2.

of p vertex-disjoint cliques of G and the minimum p, which provides the best ℓ_{min}, is determined by the level of interdiction budget k. In the table, we only report the CPU time necessary to compute ℓ_{min} for $k = \lceil 0.01 \cdot |V| \rceil$, the time to compute ℓ_{min} for the $k = \lceil 0.001 \cdot |V| \rceil$ is smaller. The table reports for each instance the optimal solution value (OPT^*) and, in case the time limit is reached, we report T.L. and OPT^* is the best known UB. As for Table 2, the total CPU time spent in solving formulation (12)-(15) is reported in the column "time". In the column $|V_{prep}|$, we report the number of vertices that can be removed from the graph, as explained in Section 3.2. We report the value of the initial upper and lower bound ℓ_{min} and ℓ_{max} and the number of CI cuts separated and the number of B&C nodes explored solving formulation (12)-(15). CLIQUE-INTER is able to solve to proven optimality 28 out of 30 instances for both levels of interdiction budget. In the preprocessing phase, 6 and 4 instances are solved for $k = \lceil 0.005 \cdot |V| \rceil$ and $k = \lceil 0.01 \cdot |V| \rceil$, respectively. The gap

between ℓ_{min} and ℓ_{max} is relatively small also for this set of instances, with a few exceptions, e.g, for *socfb-UF* and $k = \lceil 0.01 \cdot |V| \rceil$ the gap is 15 or for *socfb-UIllinois* and $k = \lceil 0.01 \cdot |V| \rceil$ the gap is 11. The graph reduction technique is crucial for these set of large instances. The table shows that huge reductions can be achieved, i.e., if we compare the size of the initial set of vertices $|V|$ and the number of vertices that can be discarded $|V_{prep}|$, we can see that formulation (12)-(15) is actually solved on a small fraction of vertices. This reduction is more effective for small values of interdiction budget k. Finally, the columns "cuts" and "nodes" reveal that the number of CI cuts separated and B&C nodes explored is often small. Only in some cases, several hundreds of cuts are separated using `IMCQ` and the number of explored nodes is typically smaller that 500. In some rare cases the number of explored nodes can be high, as in *Slashdot0902*, which requires 14105 nodes.

6.6. Clique-interdiction curve of a graph

In the following we investigate the ability of a network to survive a clique-interdiction attack. We plot the size of the optimal CIG solution (as a percentage of the size of the maximim clique on the original graph) for different values of the interdiction budget k. Figure 5 depicts this relation for two quite different types of networks: very sparse social networks (taken from the DIMACS10 data set) versus very dense graphs (taken from the DIMACS2 data set). On the x axis we plot the percentage of the total number vertices (representing the interdiction budget k). On the y axis, we provide the relative size of the optimal CIG solution compared to the value of $\omega(G)$. Not surprisingly, the obtained results indicate that significantly higher interdiction budget is needed for dense networks. With the interdiction budget of only 1% of vertices, one can reduce the size of the maximum clique between 40% and 95% in case of social networks (see *astro-ph* and *memplus*, respectively), whereas dense networks are much more resistant to this kind of attacks. With the 1% of interdiction budget, the size of the maximum clique can be reduced only by a few percent in most of the cases shown in Figure 5.

The obtained results, both in terms of CPU times and the size of networks that can be solved to optimality, indicate that our solver can be used as a powerful decision making tool for analyzing and

assessing the network resilience against vertex-deletion attacks. Measuring the clique-interdiction number can provide to decision makers an important information about the structure of the network concerning its densely connected components. Our exact solver can also be used for simulation purposes, to find out which vertices need to be protected (in case of a vertex-deletion attack).

7. Extensions and conclusions

In this article we studied the interdiction game in a network in which the leader chooses up to k vertices to delete, so as to minimize the clique number determined by the follower in the resulting network. We studied the problem complexity for special graphs, and we derived a single-level ILP formulation with an exponential number of constraints called clique-interdiction cuts. In a polyhedral study of the underlying polytope, we provided necessary and sufficient conditions for these cuts to be facet defining and we designed an effective lifting procedures. The separation of these cuts required development of an efficient exact solver for the maximum clique problem, which we tailored for the clique interdiction game. The deep understanding of the underlying problem allowed us to derive tight combinatorial lower and upper bounds, along with an efficient preprocessing phase for drastically reducing the problem size. Our framework is the first one being able to solve to proven optimality real-world large-scale (social) networks from various publicly available resources, thus, drastically outperforming the state-of-the-art general purpose bilevel solver from Fischetti et al. (2017). This shows that significant improvements (in terms of the size of input networks and computing times) can be achieved by tailored exact algorithms for particular classes of interdiction games. The code will be made available online (after the revision process is finished), to help decision makers analyze the resilience of networks to maintain a large clique after a vertex-deletion attack.

						CLIQUE-INTER $k = \lceil 0.005 \cdot \lvert V \rvert \rceil$							CLIQUE-INTER $k = \lceil 0.01 \cdot \lvert V \rvert \rceil$						
	$\lvert V \rvert$	$\lvert E \rvert$	$\omega(G)$	time$_\omega$	time$_p$	OPT*	time	$\lvert V_{prep} \rvert$	$\ell_{\min}$	$\ell_{\max}$	cuts	nodes	OPT*	time	$\lvert V_{prep} \rvert$	$\ell_{\min}$	$\ell_{\max}$	cuts	nodes
PGPgiantcompo	10,680	24,316	25	0.0	0.0	13	0.0	10,409	13	16	11	0	9	0.0	10,047	9	12	31	2
astro-ph	16,706	121,251	57	0.0	0.1	40	0.0	16,206	40	41	0	0	34	0.0	15,872	34	35	4	0
memplus	17,758	54,196	97	0.0	0.1	18	0.0	17,569	18	32	1	0	4	0.0	15,030	4	5	9	0
as-22july06	22,963	48,436	17	0.0	0.7	3	-	-	3	3	-	-	3	-	-	3	3	-	-
cond-mat-2005	40,421	175,691	30	0.0	0.8	14	0.1	38,270	13	15	12	0	11	0.3	36,866	11	13	47	3
Cit-HepPh	34,546	420,877	19	0.0	4.3	11	2.8	15,493	11	14	60	0	9	52.4	12,501	9	13	525	2188
Soc-Epinions1	75,879	405,740	23	0.1	17.6	8	62.7	64,859	8	12	339	138	6	114.2	61,968	6	9	719	126
Slashdot0902	82,168	504,230	27	0.0	37.9	4	921.5	50,736	4	6	2035	14105	4	92.5	50,736	4	5	380	0
socfb-UIllinois	30,795	1,264,421	57	0.5	7.9	33	24.4	10,456	33	42	57	13	27	41.6	8290	27	38	106	44
ia-email-EU	32,430	54,397	12	0.0	0.5	4	0.6	30,375	4	6	178	47	3	0.5	29,212	3	4	165	3
rgg_n_2_15_s0	32,768	160,240	13	0.0	0.7	10	-	-	10	10	-	-	9	0.2	30,848	9	10	35	0
ia-enron-large	33,696	180,811	20	0.0	1.5	9	2.2	27,791	8	12	81	15	7	29.5	26,651	7	10	957	2827
socfb-UF	35,111	1,465,654	55	0.3	7.2	37	17.8	14,264	37	48	74	18	29	87.8	10,708	29	44	272	234
socfb-Texas84	36,364	1,590,651	51	0.3	8.5	30	24.6	10,706	30	43	66	19	25	74.3	8,704	25	36	168	104
tech-internet-as	40,164	85,123	16	0.0	2.8	3	1.4	31,783	3	4	152	4	3	-	-	3	3	-	-
fe-body	45,087	163,734	6	0.1	5.4	4	1.8	2,259	4	5	27	0	4	1.8	2259	4	5	27	0
sc-nasasrb	54,870	1,311,227	24	0.1	7.3	24	-	-	24	24	-	-	23	145.5	1,195	23	24	1249	254
soc-brightkite	56,739	212,945	37	0.0	3.7	7	15.6	47,159	7	11	482	314	6	11.2	44,919	6	8	322	18
soc-loc-brightkite	58,228	214,078	37	0.0	4.0	7	11.9	48,640	7	11	390	169	6	12.6	46,384	6	8	347	18
tech-p2p-gnutella	62,561	147,878	4	0.1	15.4	3	-	-	3	3	-	-	3	-	-	3	3	-	-
delaunay_n16	65,536	196,575	4	0.2	70.9	4	-	-	4	4	-	-	4	-	-	4	4	-	-
rgg_n_2_16_s0	65,536	342,127	14	0.0	5.1	10	0.3	63,637	10	11	26	0	9	1.5	59,534	9	10	57	0
soc-themarker	69,413	1,644,843	22	2.1	68.0	8	T.L.	35,678	7	10	1685	4806	6	T.L.	31,101	5	8	2124	1311
rec-eachmovie	74,424	1,634,743	12	0.7	18.5	3	-	-	3	3	-	-	2	367.3	13669	2	3	3854	569
fe-tooth	78,136	452,591	5	0.5	58.0	4	18.9	7	4	5	30	0	4	19.0	7	4	5	30	0
sc-pkustk11	87,804	2,565,054	36	1.1	54.9	24	70.7	2,712	24	30	62	4	24	57.1	2,712	24	30	51	0
soc-BlogCatalog	88,784	2,093,195	45	11.7	149.8	11	T.L.	51,607	8	12	1327	1376	8	T.L.	46,240	6	9	2281	1637
ia-wiki-Talk	92,117	360,767	15	0.2	16.6	5	49.2	72,678	4	6	345	23	4	87.4	72,678	4	5	711	14
sc-pkustk13	94,893	3,260,967	36	1.3	310.0	35	724.9	2,360	35	36	535	80	34	879.2	2,354	34	36	613	38
fe_rotor	99,617	662,431	5	1.0	182.8	4	200.5	0	4	5	184	0	4	200.2	0	4	5	184	0

Table 3 Computational results obtained by the CLIQUE-INTER on seleceted large-scale real-world networks.

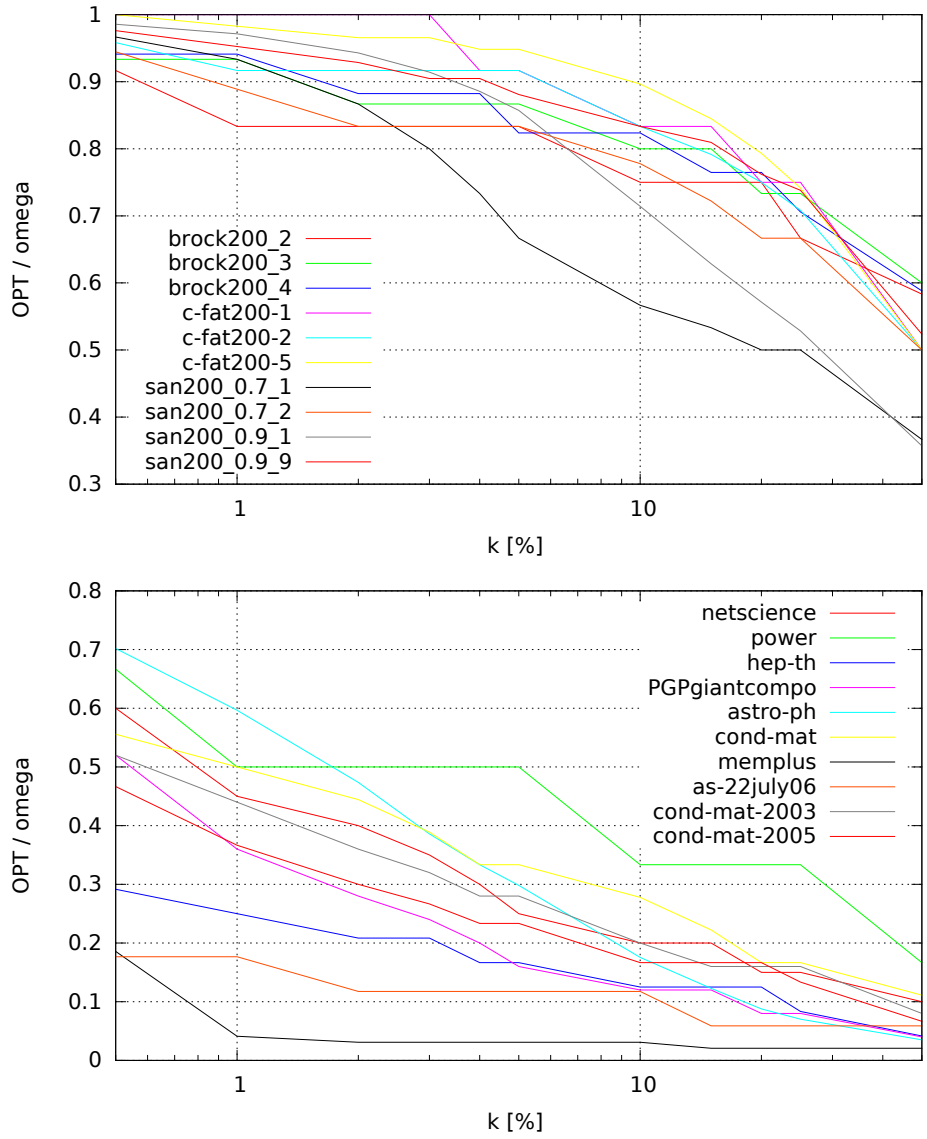

Figure 5 Clique-interdiction curves for social networks (up) and dense graphs (down).

In some applications where the concept of clique may be too restrictive to define a community, *clique-relaxations* can be used instead, see (Balasundaram et al. 2011, Pattillo et al. 2013). Clique-relaxations are obtained by relaxing some of the clique defining properties, and our exact solution framework can be extended to the interdiction games in which the follower solves the maximum relaxed clique problem, as long as the latter one satisfies the hereditary property. Relaxed cliques with hereditary properties are obtained by relaxing: the degree of the vertices (s-plex), the distance between the vertices (s-clique), the density of the edges (s-defective clique) and the connectivity

between the vertices (s-bundle). Precise definition of these problems can be found in e.g., Pattillo et al. (2013). One can show that our single-level reformulation (and accordingly the branch-and-cut algorithm) can be extended for solving this class of interdiction games. The key element of the B&C implementation would be an efficient exact solver for the maximum relaxed clique problem at hand.

We finally point out that the theory and methodology derived in this paper does not directly carry over to solving interdiction games in which the follower maximizes a *weighted* hereditary problem on a network. These problems deserve a special attention and will be subject of our future studies.

Acknowledgments

Ivana Ljubić is partially funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-014. Pablo San Segundo is funded by the Spanish Ministry of Economy and Competitiveness (grants DPI 2014-53525-C3-1-R, DPI2017-86915-C3-3-R).

References

10th DIMACS Implementation Challenge - Graph Partitioning and Graph Clustering (Nov 20, 2017) URL `https://www.cc.gatech.edu/dimacs10/`.

2017 Terrorist Attacks (Nov 20, 2017) URL `http://storymaps.esri.com/stories/terrorist-attacks/`.

Adjiashvili D, Baggio A, Zenklusen R (2017) Firefighting on trees beyond integrality gaps. Klein PN, ed., *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, 2364–2383 (SIAM).

Assimakopoulos N (1987) A network interdiction model for hospital infection control. *Computers in Biology and Medicine* 17(6):413 – 422.

Baggio A (2016) *Towards Optimal Approximations for Firefighting and Related Problems*. Ph.D. thesis, ETH Zürich.

Balas E, Yu CS (1986) Finding a maximum clique in an arbitrary graph. *SIAM J. Comput* 15(4):1054–1068.

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

Barabási AL, Pósfai M (2016) *Network science* (Cambridge: Cambridge University Press).

Batagelj V, Zaversnik M (2011) Fast algorithms for determining (generalized) core groups in social networks. *Adv. Data Analysis and Classification* 5(2):129–145.

Berry N, Ko T, Moy T, Smrcka J, Turnley J, Wu B (2004) Emergent Clique Formation in Terrorist Recruitment. *The AAAI-04 Workshop on Agent Organizations: Theory and Practice, July 25, 2004, San José, California* (National Conference on Artificial Intelligence).

Cappanera P, Scaparra MP (2011) Optimal allocation of protective resources in shortest-path networks. *Transportation Science* 45(1):64–80.

Caprara A, Carvalho M, Lodi A, Woeginger GJ (2016) Bilevel knapsack with interdiction constraints. *INFORMS Journal on Computing* 28(2):319–333.

Chen H, Chung W, Xu JJ, Wang G, Qin Y, Chau M (2004) Crime data mining: A general framework and some examples. *Computer* 37(4):50–56.

Cormican K, Morton D, Wood K (1998) Stochastic network interdiction. *Operations Research* 46(2):184 – 197.

DIMACS2 (Nov 20, 2017) 2nd DIMACS Implementation Challenge - NP Hard Problems: Maximum Clique, Graph Coloring, and Satisfiability. URL `http://dimacs.rutgers.edu/Challenges/`.

Fischetti M, Ljubić I, Monaci M, Sinnl M (2016) Interdiction games under monotonicity Under Revision.

Fischetti M, Ljubić I, Monaci M, Sinnl M (2017) A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research* 65(60):1615–1637.

García-Martnez C, Blum C, Rodriguez F, Lozano M (2015) The firefighter problem: Empirical results on random graphs. *Computers & Operations Research* 60:55 – 66.

Kim H, Feamster N (2013) Improving network management with software defined networking. *IEEE Communications Magazine* 51(2):114–119.

Li C, Jiang H, Manyà F (2017) On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem. *Computers & OR* 84:1–15.

Mahdavi Pajouh F, Boginski V, Pasiliao EL (2014) Minimum vertex blocker clique problem. *Networks* 64(1):48–64.

Nemhauser GL, Wolsey LA (1988) *Integer and Combinatorial Optimization*, volume 18 (Wiley New York).

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

Rothenberg RB, Potterat JJ, Woodhouse DE (1996) Personal risk taking and the spread of disease: Beyond core groups. *The Journal of Infectious Diseases* 174:S144–S149.

Rutenburg V (1991) Complexity classification of truth maintenance systems. Choffrut C, Jantzen M, eds., *STACS 91: 8th Annual Symposium on Theoretical Aspects of Computer Science Hamburg, Germany, February 14–16, 1991 Proceedings*, 372–383 (Springer Berlin Heidelberg).

Rutenburg V (1994) Propositional truth maintenance systems: Classification and complexity analysis. *Annals of Mathematics and Artificial Intelligence* 10(3):207–231.

Sageman M (2004) *Understanding Terrorist Networks* (Philadelphia, PA, USA: University of Pennsylvania Press).

Sampson RJ, Groves WB (1989) Community structure and crime: Testing social-disorganization theory. *American Journal of Sociology* 94(4):774–802.

San Segundo P, Lopez A, Pardalos PM (2016a) A new exact maximum clique algorithm for large and massive sparse graphs. *Computers & OR* 66:81–94.

San Segundo P, Matia F, Rodriguez-Losada D, Hernando M (2013) An improved bit parallel exact maximum clique algorithm. *Optimization Letters* 7(3):467–479.

San Segundo P, Nikolaev A, Batsyn M (2015) Infra-chromatic bound for exact maximum clique search. *Computers & OR* 64:293–303.

San Segundo P, Nikolaev A, Batsyn M, Pardalos PM (2016b) Improved infra-chromatic bound for exact maximum clique search. *Informatica, Lith. Acad. Sci.* 27(2):463–487.

San Segundo P, Rodriguez-Losada D, Jimenez A (2011) An exact bit-parallel algorithm for the maximum clique problem. *Computers & OR* 38(2):571–581.

San Segundo P, Tapia C (2014) Relaxed approximate coloring in exact maximum clique search. *Computers & OR* 44:185–192.

Snyder LV, Atan Z, Peng P, Rong Y, Schmitt AJ, Sinsoysal B (2016) OR/MS models for supply chain disruptions: a review. *IIE Transactions* 48(2):89–109.

Song Y, Shen S (2016) Risk averse shortest path interdiction. *INFORMS Journal on Computing* 28(3):527–539.

Tang Y, Richard JPP, Smith JC (2016) A class of algorithms for mixed-integer bilevel min–max optimization. *Journal of Global Optimization* 66(2):225–262.

Washburn A, Wood K (1995) Two-person zero-sum games for network interdiction. *Operations Research* 43(2):243–251.

Wasserman S, Faust K (1994) *Social Network Analysis* (Cambridge University Press).

APPENDIX

In this Appendix we provide the proof of Proposition 10, which is deliberately left out from the manuscript, for the sake of conciseness and clarity.

Proof of Proposition 10: ($\Leftarrow$) See Lemma 1 and Corollary 2.

($\Rightarrow$) Let us denote by $\alpha' w + \beta'\theta \geq \gamma'$ inequality (13) associated with K. Let $\alpha w + \beta\theta \geq \gamma$ be a facet defining an equality (13) such that $\{(w,\theta) \in \mathcal{P}(G,k) : \alpha' w + \beta'\theta = \gamma'\} \subseteq \{(w,\theta) \in \mathcal{P}(G,k) : \alpha w + \beta\theta = \gamma\}$. We will show that $\alpha = \rho\alpha'$ and $\beta = \rho\beta'$ for some $\rho \in \mathbb{R}$.

Let us first consider a set V' of vertices which is an optimal interdiction policy such that $V' \cap K \neq \emptyset$. Observe that such set must exist, given that the optimal solution value is ℓ_{opt} and $|K| \geq \ell_{\text{opt}} + 1$, so at least one vertex from K has to be interdicted. Let $V'' = V' \setminus K$.

- We will first show that $\alpha_v = 0$ for all $v \in V''$. Let $v \in V''$. We consider the solution $(V_1 = V'', |K|)$. This solution is feasible since $\omega(G[V \setminus V']) = \ell_{\text{opt}} \leq |K|$. Consider a second solution $(V_2 = V'' \setminus \{v\}, |K|)$. This solution is also feasible because by interdicting one vertex less from V'', the size of the maximum clique in $G[V \setminus (V'' \setminus \{v\})]$ is at most $\ell_{\text{opt}} + 1 \leq |K|$. These two solutions satisfy (13) with the equality. We deduce $\alpha^{V_1} + \beta \cdot |K| = \alpha^{V_2} + \beta \cdot |K|$. This implies $\alpha_v = 0$ for all $v \in V''$.
- We now show that $\alpha_v = 0$ for all $v \in (V \setminus K) \setminus V''$. Let $v \in (V \setminus K) \setminus V''$. Let us consider the solution $(V_3 = V_2 \cup \{v\}, |K|)$. This solution is valid since we interdict an additional vertex compared to V_2, so $\omega(G[V \setminus (V_2 \cup \{v\}]) \leq \omega(G[V \setminus V_2]) \leq |K|$. Clearly this solution satisfies (13) with equality. We deduce $\alpha^{V_1} + \beta \cdot |K| = \alpha^{V_3} + \beta \cdot |K|$. This implies $\alpha_v = 0$ for all $v \in (V \setminus K) \setminus V''$.

Let $v_k \in K$. By the hypothesis there exists a set V_k of vertices which is a feasible interdiction policy ($|V_k| \leq k$) such that $v_k \in V_k$ and $\omega(G[V \setminus V_k]) + |V_k \cap K| \leq |K|$. By the inequality (13) ($\theta + \sum_{u \in K} w_u \geq |K|$) we deduce that $\omega(G[V \setminus V_k]) + |V_k \cap K| = |K|$. Let $V'_k = V_k \setminus K$.

- We will show that $\beta = \alpha_{v_k}$, for the given $v_k \in K$. Consider two feasible solutions $(V_k, \omega_0 = \omega(G[V \setminus V_k]))$ and $(V_1 = V_k \setminus \{v_k\}, \omega_1 = \omega(G[V \setminus V_k]) + 1)$ – they both satisfy (13) with equality. This implies $\alpha^{V_k} + \beta \cdot \omega_0 = \alpha^{V_1} + \beta \cdot \omega_1$ and thus $\alpha_{v_k} = \beta$.

- Finally, we will show that $\alpha_{\tilde{v}} = \alpha_{v_k}$, for all $\tilde{v} \in K$, $\tilde{v} \neq v_k$. Let $\tilde{v} \in K$ and let $\tilde{V}$ be a feasible interdiction policy such that $\tilde{v} \in \tilde{V}$, and $\omega(G[V \setminus \tilde{V}]) + |\tilde{V} \cap K| \leq |K|$. Let $\tilde{V}' = \tilde{V} \setminus K$ and $V'_k = V_k \setminus K$. We consider $(V_4 = V'_k \cup \{v_k\}, |K|-1)$ and $(V_5 = \tilde{V}' \cup \{\tilde{v}\}, |K|-1)$ two valid solutions. Indeed, in the graph $G[V \setminus V'_k]$ there remains the clique K and if we interdict one vertex of K plus the vertices of V'_k then the biggest clique has a size at most $|K|-1$. The same argument holds for the solution $(V_5, |K|-1)$. Hence we can deduce that these two solutions satisfy (13) with equality. This implies $\alpha^{V_4} + \beta \cdot (|K|-1) = \alpha^{V_5} + \beta \cdot (|K|-1)$. Since $\alpha_v = 0$ for all $v \in V \setminus K$, it follows that $\alpha_{\tilde{v}} = \alpha_{v_k}$, for all $\tilde{v}, v_k \in K$.

To finish the proof, we set $\beta = \rho$ and thus $\alpha = \rho\alpha'$ and $\beta = \rho\beta'$. □