A branch-and-cut algorithm for the Edge Interdiction Clique Problem

Fabio Furini[1]

Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" – Consiglio Nazionale delle Ricerche (IASI-CNR), Rome, Italy. fabio.furini@cnr.it

Ivana Ljubić

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

Pablo San Segundo

Univ. Politécnica de Madrid, Centre for Automation and Robotics, Spain, pablo.sansegundo@upm.es

Yanlu Zhao

ESSEC Business School, Cergy-Pontoise, France, yanlu.zhao@essec.edu

Abstract

Given a graph G and an interdiction budget $k \in \mathbb{N}$, the Edge Interdiction Clique Problem (EICP) asks to find a subset of at most k edges to remove from G so that the size of the maximum clique, in the interdicted graph, is minimized. The EICP belongs to the family of interdiction problems with the aim of reducing the clique number of the graph. The EICP optimal solutions, called optimal interdiction policies, determine the subset of most vital edges of a graph which are crucial for preserving its clique number. We propose a new set-covering-based Integer Linear Programming (ILP) formulation for the EICP with an exponential number of constraints, called the *clique-covering inequalities*. We design a new branch-and-cut algorithm which is enhanced by a tailored separation procedure and by an effective heuristic initialization phase. Thanks to the new exact algorithm, we manage to solve the EICP in several sets of instances from the literature. Extensive tests show that the new exact algorithm greatly outperforms the state-of-the-art approaches for the ECIP.

Key words: Combinatorial Optimization, Interdiction Problems, Maximum Clique, Most Vital Edges.

[1]Corresponding author

1. Introduction

Let $G = (V, E)$ be a simple undirected graph with $n = |V|$ vertices and $m = |E|$ edges. Two vertices u and v of V are called *neighbours* if there is an edge $uv \in E$ and a subset of vertices $K \subseteq V$ is called a *clique* if they are all pairwise neighbours. The *Maximum Clique Problem* (MCP) asks for determining the largest clique of the graph whose size is denoted by $\omega(G)$, the *clique number* of the graph. The MCP is one of the most studied problems in combinatorial optimization and graph theory. Many articles addressed it and we refer the interested reader to [2, 15, 16, 20, 21, 22, 25, 23, 26, 28, 27] where efficient exact algorithms are described also for some variants and generalizations of the problem. The MCP is strongly $\mathcal{NP}$-hard and inapproximable in polynomial time to within any polynomial factor unless $\mathcal{P} = \mathcal{ZPP}$ [13].

In this paper, we address the *Edge Interdiction Clique Problem* (EICP) with the goal of developing an efficient exact algorithm to solve it to proven optimality. The EICP belongs to the family of problems aiming at reducing the clique number of the graph. Formally, given a graph G and an *interdiction budget* $k \in \mathbb{N}$, the EICP asks to find a subset of at most k edges to remove (also interdict) from G so that the size of the maximum clique in the remaining graph is minimized. This problem, which has been introduced in Tang et al. [29], is of practical importance for many applications arising in communication, social or biological networks in which cohesive clusters in the underlying network are represented as cliques. Existence of large cohesive clusters allows for much shorter communication paths between pairs of vertices, and hence finding the most critical edges to interdict these clusters is an important question for many real-world applications (see, e.g., [11, 17] and further references therein). In addition, optimal EICP solutions identify the subset of most important (or vital) edges of a graph which are crucial for preserving its clique number.

The subset of edges $S \subset E$ that are interdicted from the graph constitutes an *interdiction policy* and defines the *interdicted graph* $G_I = (V, E \setminus S)$ which corresponds to the original graph after the removal of the interdicted edges. Figure 1 provides a graphical representation of the EICP and of its main features considering a synthetic example graph of 6 vertices and 13 edges. In this graph, the clique number is $\omega(G) = 4$ and there are four maximum cliques, i.e., $\{v_1, v_2, v_5, v_6\}$ (vertices depicted in grey), $\{v_2, v_3, v_4, v_5\}$, $\{v_1, v_2, v_3, v_5\}$ and $\{v_2, v_4, v_5, v_6\}$. In Figures 2(a) and 2(b), we report two optimal interdiction policies with $k = 3$ and $k = 5$, respectively. The edges belonging to the interdiction policies are depicted with red dashed lines. With $k = 3$, the interdiction policy is composed of the edges v_1v_2, v_1v_3 and v_4v_5. After removing these edges, the clique number in the interdicted graph G_I becomes $\omega(G_I) = 3$. There are four remaining maximum cliques of size three, e.g., $\{v_2, v_3, v_4\}$ (vertices depicted in grey) or $\{v_1, v_5, v_6\}$. With $k = 5$, the interdiction policy is composed of the edges v_1v_2, v_1v_5, v_2v_4, v_2v_5 and v_4v_5. After removing these edges, the interdicted graph becomes triangle-free and its clique number is $\omega(G_I) = 2$. There are eight remaining maximum cliques of size two, one of them is depicted with grey vertices, i.e., $\{v_1, v_3\}$. It is important to notice that the optimal interdiction policy is often not unique. For instance, with $k = 5$ another optimal interdiction policy is given by the edges v_2v_3, v_2v_6, v_3v_5, v_2v_5 and v_5v_6.

A trivial lower bound on the size of the clique in the interdicted graph is two, which is always obtained, unless all the edges of the graph are interdicted; without loss of generality, in the remainder of this article we assume $k < |E|$. In the following section, we provide a

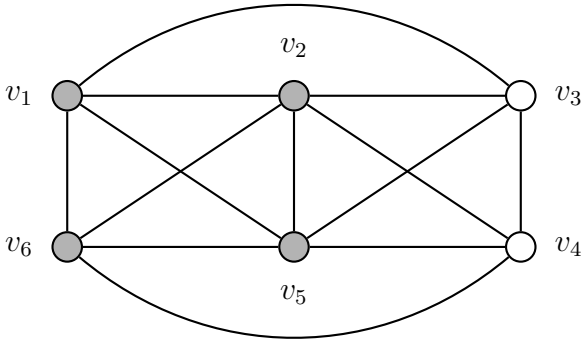
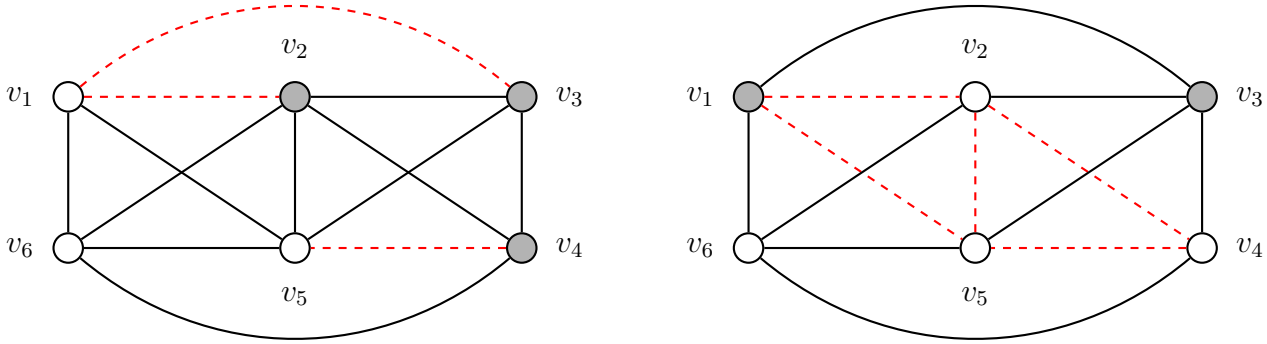


Figure 1: An example graph G with 6 vertices and 13 edges. Before interdiction, the clique number is $\omega(G) = 4$. One of the four maximum cliques is shown in grey, i.e., the clique $\{v_1, v_2, v_5, v_6\}$.



(a) (b)

Figure 2: Two optimal interdiction policies represented by dashed red edges with: (a) $k = 3$ and (b) $k = 5$. With $k = 3$, the interdicted graph G_I has $\omega(G_I) = 3$ and one of the largest cliques is $\{v_2, v_3, v_4\}$ (grey vertices). With $k = 5$, $\omega(G_I) = 2$ and one of the largest cliques is $\{v_1, v_3\}$ (grey vertices).

comprehensive review of the literature addressing the EICP and its closely related problems.

1.1. Literature review

Our research lies at the intersection of several related streams of literature: Research on the edge interdiction clique problem, the node interdiction clique problem and the interdiction games. In this section, we review the key contributions of each of these streams and discuss how we extend them.

The EICP has been introduced by Tang et al. [29]. The authors introduced a generic exact approach for interdiction problems with binary leader variables and used the EICP as a special case study for testing their approach. Due to the generic nature of their method, the largest graphs considered in the computational study of Tang et al. [29] contain at most 15 vertices, and the majority of them could not be solved within an hour of computing time. These instances have been solved to proven optimality by a recent a state-of-the-art exact solver for bilevel mixed integer programs provided in Fischetti et al. [9]. More recently, a MIP-based heuristic for general interdiction problems has been proposed by Fischetti et al.

[8], and larger EICP instances (ranging from 40 to 60 vertices) have been used to test the computational efficiency of their approach. Nevertheless, the question on how to design an efficient algorithm that can solve the EICP instances on large graphs relevant for real-world applications still remains an open issue. To the best of our knowledge, this work is a first study dedicated to the EICP in which a tight problem-specific ILP formulation is presented, along with efficient upper bounding techniques.

Another relevant stream of research is dedicated to the *Node Interdiction Clique Problem* (NICP) and its blocker-variant known as the *Minimum Vertex Blocker Clique Problem* (MVBCP). Furini et al. [11] defined the NICP, in which for a given graph G and an interdiction budget $k \geq 1$, the decision maker has to find a subset of at most k vertices to remove from G so that the clique number in the remaining graph is minimized. The authors designed an exact algorithm `CLIQUE-INTER`, which is able to report optimal solutions for most instances on randomly generated graphs with up to 150 vertices. Optimal solution values are also reported for some larger networks from the SNAP database [2]. Mahdavi Pajouh et al. [17] defined the MVBCP, 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. The authors provided an analytical lower bound for the MVBCP and gave a MIP formulation with an exponential number of constraints, along with the characterization of conditions under which they are facet-defining. It is worth pointing out that the studies by Furini et al. [11] and Mahdavi Pajouh et al. [17] deal with the node interdiction aspect, and as such, cannot be straight-forwardly extended to the EICP. Similarities between the known results for the NICP and their exploitability in the EICP context are addressed in Section 3.

Finally, the NICP also belongs to a larger family of Interdiction Games under Monotonicity, which has been recently addressed by Fischetti et al. [9]. These problems involve two players, a leader and a follower, who play a Stackelberg game. The leader has a limited interdiction budget to remove a subset of items, whereas the follower solves a maximization problem based on the remaining items. The follower's subproblem is assumed to satisfy a monotonicity property, which is then exploited for deriving a single-level integer linear programming reformulation. Whereas this monotonicity property is preserved by the NICP, it no longer holds for the EICP.

1.2. Contribution and outline of the paper

The main contribution of this manuscript is the development of a specialized exact algorithm to solve the EICP to proven optimality. To this end, we have developed an effective and new ILP formulation with an exponential number of constraints. We show that this formulation dominates the Benders-like problem reformulation, typically used for solving interdiction problems, and we develop a dedicated branch-and-cut algorithm. There are two key features of our new branch-and-cut algorithm: (*i*) an efficient combinatorial algorithm which allows to perform the ($\mathcal{NP}$-hard) separation of the inequalities in relatively short computing time; (*ii*) an effective initialization phase in which high-quality feasible solutions are computed thanks

[2] `http://snap.stanford.edu`

to a specialized heuristic algorithm. The specialized separation and the heuristic solutions are crucial to speed up the convergence of our new branch-and-cut algorithm as well as for reducing the exit gaps for the instances which are not solved to proven optimality.

The paper is structured as follows. In Section 2 a bilevel model is presented and in Section 3 we present a Benders-like problem reformulation. The first one can be directly solved by a general purpose bilevel solver, while the second one requires a branch-and-cut algorithm to be solved. In Section 4 we present the new set-covering-based ILP formulation which is the base of the new branch-and-cut algorithm developed in this manuscript. In the same section, we present and analyze the exponential-size family of inequalities called the clique-covering inequalities. Section 4 also contains a theoretical comparison of the strength of the clique-covering inequalities against the ones of the Benders-like reformulation. In Section 5 we present the combinatorial branch-and-bound algorithm used to effectively separate the clique-covering inequalities. This branch-and-bound algorithm is based on a state-of-the-art exact algorithm for the Maximum Clique Problem. In Section 6 we present and discuss the heuristic initialization algorithm which is composed of two main steps: the first step consists of finding a feasible solution solving a compact ILP formulation, and the second step is a specialized heuristic algorithm based on local-branching constraints. In Section 7 we present the computational results testing the performance of the newly developed branch-and-cut algorithm on several sets of benchmark instances. Finally, in Section 8 we present some concluding remarks and elaborate on future research directions.

2. A bilevel model of the EICP

Let w be a vector of binary variables associated with the set of edges E, each variable encoding whether the corresponding edge is interdicted or not. Similarly, let x be a vector of binary variables associated with vertices, indicating whether a vertex $v \in V$ is part of a maximum clique in the interdicted graph. Then, the EICP problem can be stated as the following bilevel optimization problem:

$$\min \quad \vartheta \tag{1a}$$

$$\text{subject to (s.t.)} \quad \sum_{uv \in E} w_{uv} \leq k \tag{1b}$$

$$w_{uv} \in \{0,1\} \qquad uv \in E \tag{1c}$$

$$\text{where } \vartheta = \max \sum_{u \in V} x_u \tag{1d}$$

$$\text{s.t.} \quad x_u + x_v \leq 2 - w_{uv} \qquad uv \in E \tag{1e}$$

$$x_u + x_v \leq 1 \qquad uv \in \overline{E} \tag{1f}$$

$$x_v \in \{0,1\} \qquad v \in V \tag{1g}$$

The auxiliary variable ϑ in the objective function replaces the value of the maximum clique in the interdicted graph. This value is obtained as the optimal solution of the *inner optimization problem* stated by (1d)-(1g). Constraint (1b) guarantees that the interdiction budget is respected, and constraint (1c) expresses the nature of the *edge interdiction variables*. For any

feasible interdiction policy, i.e., for any vector $w \in \mathcal{W}_E$ where

$$\mathcal{W}_E = \left\{ w \in \{0,1\}^m : \sum_{uv \in E} w_{uv} \leq k \right\},$$

the inner optimization problem asks for the size of the maximum clique in the interdicted graph G_I that results from removing the edges e such that $w_e = 1$ from the original graph G. Indeed, the objective function (1d) counts the number of vertices selected, whereas constraints (1e) and (1f) prevent selecting two vertices if an edge between them has been removed, respectively, if they are not connected by an edge in G (i.e., there is an edge in the complement graph $\overline{G} = (V, \overline{E})$ of G).

The major purpose of this article is to develop a single-level ILP formulation for the EICP which can be efficiently solved using branch-and-cut plug-ins available in state-of-the-art ILP solvers. We point out that our proposed methodology (elaborated in the following two sections) remains valid for the more generalized EICP in which the k-cardinality constraint (1b) is replaced by a knapsack-like constraint

$$\sum_{uv \in E} b_{uv} w_{uw} \leq B,$$

where $b_{uv} \geq 0$ represents the cost for interdicting the edge $uv \in E$, and B is the overall interdiction budget. Nevertheless, for the ease of exposition and consistency with the existing literature, we will stick to the k-cardinality constraint in the remainder of the paper.

3. Benders-like problem reformulation

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

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

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. Given an interdiction policy $w^* \in \mathcal{W}_E$, let E_{w^*} be the associated set of interdicted (deleted) edges. We say that a clique K of G is *interdicted* (or, equivalently, *covered*) by w^*, if and only if $E(K) \cap E_{w^*} \neq \emptyset$.

We now provide a way to derive a Benders-like problem reformulation, using the following result:

Proposition 1. *Let $\mathcal{K}$ denote the set of all cliques in G and $\mathcal{W}_E$ the set of all feasible interdiction policies. Then, the EICP can be restated as follows:*

$$\min_{w \in \mathcal{W}_E} \max_{K \in \mathcal{K}} \left\{ |K| - \sum_{e \in E(K)} w_e \right\}. \tag{2}$$

Proof. Let K be a clique of G, let w^* and E_{w^*} be defined as above. We need to show that the inner maximization problem, which maximizes over all cliques in G, provides the clique number of the interdicted graph G_I. We will distinguish two situations: a) $E(K) \cap E_{w^*} \neq \emptyset$ and b) $E(K) \cap E_{w^*} = \emptyset$.

a) If K is interdicted by the current solution w^*, then the objective function value $|K| - \sum_{e \in E(K)} w_e^*$ provides an *underestimation* of the clique number in the subgraph induced by $E(K) \setminus E_{w^*}$. Indeed, let $\tilde{K} \subset K$ be a maximum clique in the subgraph induced by $E(K) \setminus E_{w^*}$. Then if $|E_{w^*}| = 1$ we have $|\tilde{K}| = |K| - 1$, and for $|E_{w^*}| \geq 2$ we have $|\tilde{K}| \geq |K| - |E_{w^*}|$. Hence, there always exists an optimal solution of the inner maximization problem in (2) which is a clique which is not being interdicted by w^*.

b) If K is not interdicted by the current solution w^*, then the objective function value $|K| - \sum_{e \in E(K)} w_e^*$ measures exactly the size of K, and hence, the inner maximization problem focuses on finding the maximum clique among those from $\mathcal{K}$ which are not being interdicted by w^*.

□

Hence, the EICP can be also reformulated as the following ILP:

$$\min_{w \in \mathcal{W}_E} \left\{ \vartheta : \vartheta \geq |K| - \sum_{e \in E(K)} w_e, \quad K \in \mathcal{K} \right\}, \tag{3}$$

where the auxiliary variable ϑ is used to represent the value of the maximum clique in the interdicted graph.

The model (3) resembles the Benders-like reformulation proposed for the NICP by Furini et al. [11] in which the vertex variables (instead of edge variables) are used to model interdiction decisions. The corresponding cuts $\vartheta \geq |K| - \sum_{e \in E(K)} w_e$ are referred to as clique-interdiction cuts. A major difference between the NICP and EICP models is that for the former it is sufficient to impose the clique-interdiction constraints for *maximal cliques* of G only. Furini et al. [11] show that cuts associated to maximal cliques dominate the cuts derived from other cliques, and they also provide conditions under which these maximal-clique-interdiction cuts are facet defining. As we will see in the following sections, formulation (3) is less effective for the EICP, which was the main motivation for us to work on the novel set-covering-based ILP model proposed in Section 4.

4. A set-covering problem reformulation

In this section we propose an alternative single-level problem formulation which uses set-covering arguments and requires an exponential number of constraints.

For $\ell \in \mathbb{N}$, let $\mathcal{K}_\ell$ denote the set of cliques from G, whose size is equal to ℓ, i.e., $|K| = \ell$. Similarly, let $\mathcal{K}_{\ell^+}$ denote the set of all cliques from G, whose size is q, such that $\ell \leq q \leq \ell_{\max}$.

Let $\ell_{\min}$ and $\ell_{\max}$ denote feasible lower and upper bounds to the optimal solution value of the EICP respectively, and let $L = \{\ell_{min} + 1, \ldots, \ell_{max} + 1\}$. To derive our new single-level problem reformulation, we introduce binary variables z_ℓ such that:

$$z_\ell = \begin{cases} 1, & \text{every clique of size } \ell \text{ or larger in } G \text{ is being interdicted,} \\ 0, & \text{otherwise} \end{cases} \qquad \ell \in L.$$

In other words, z_ℓ is equal to 1 if and only if the clique number in the interdicted graph is $\ell - 1$. Correspondingly, the set L contains the range of all possible integer values ℓ such that an optimal solution could take value $\ell - 1$.

We observe that if $z_\ell = 1$, then all cliques of size ℓ in G have to be covered. This leads to the following set-covering-based ILP formulation:

$$\text{(ILP)} \qquad \min_{w \in \mathcal{W}_E} \quad \sum_{\ell \in L} (\ell - 1) z_\ell \tag{4a}$$

$$\text{s.t.} \qquad \sum_{\ell \in L} z_\ell = 1 \tag{4b}$$

$$\sum_{e \in E(K)} w_e \geq z_\ell \qquad \ell \in L \setminus \{\ell_{\max} + 1\}, K \in \mathcal{K}_\ell \tag{4c}$$

$$z_\ell \in \{0, 1\} \qquad \ell \in L. \tag{4d}$$

Constraints (4b) ensure that exactly one value of $\ell \in L$ can be set to one. By minimization of the objective function, the value of z_ℓ will be set to one for the smallest possible value of ℓ for which we are able to cover all cliques of size ℓ (cf. constraints (4c)). Recall that a clique $K \in \mathcal{K}_\ell$ is covered if for at least one of the edges $e \in K$ we have $w_e = 1$. Observe that there is no need to impose constraints (4c) for $\ell_{\max} + 1$. This is because we assume that $\ell_{\max}$ corresponds to a feasible solution value (obtained by e.g., an initialization heuristic). Hence, by the minimization arguments of the objective function, if z_ℓ cannot be set to one for any $\ell \leq \ell_{\max}$ (due to the insufficient budget to cover all cliques of size ℓ) the value of $z_{\ell_{\max}+1}$ will be the only feasible solution, and hence, the interdiction policy found by the heuristic will correspond to the optimal one.

We notice that the above arguments require existence of a heuristic solution, which is used to initialize the value of $\ell_{\max}$. In case no heuristic solution is available at hand, the value of $\ell_{\max}$ can be trivially set to $\omega(G)$. If the optimal solution returned by the ILP formulation remains $\omega(G)$, this indicates that the budget k was not sufficient to interdict every maximum clique of G, and hence any subset of edges can be considered as a feasible (and also optimal) interdiction policy.

4.1. Strengthened set-covering-based formulation

The above formulation uses the fact that at least one edge from each clique of size ℓ has to be covered in order to guarantee that the optimal solution is bounded by $\ell - 1$ from above. This formulation can be strengthened by exploiting the result by Hassan et al. [12], cf. Proposition 14. Their result provides for any two given numbers $q, \ell \in \mathbb{N}$ ($\ell \leq q$) the minimum number of edges that has to be removed from the clique of size q in order to "interdict" all its subcliques of size ℓ.

Proposition 2 (Hassan et al. [12]). *Given a complete graph K_q, i.e., a clique of q vertices, and a number $\ell \in \mathbb{N}$, $3 \leq \ell \leq q$, the minimum number of edges that has to be removed from K_q so that the clique number of the remaining graph is at most $\ell - 1$ is given by:*

$$f(\ell, K_q) = n_{\alpha-1}\binom{\alpha-1}{2} + n_\alpha\binom{\alpha}{2},$$

where $\alpha = \lceil \frac{q}{\ell-1} \rceil$, $n_{\alpha-1} = (\ell-1)\alpha - q$ and $n_\alpha = \frac{q-(n_{\alpha-1})(\alpha-1)}{\alpha}$.

Figures 3 and 4 illustrate the formula of Proposition 2. In Figure 3, we consider the case of a clique of size 6 and in Figure 4, a clique of size 8. In both figures, we depict in grey the vertices of a maximum clique remaining after the removal of the adequate edges. The latter are depicted with red dashed lines. For example, in the case of reducing a clique of size 6 ($q = 6$) to size 4 ($\ell = 5$), which corresponds to Figure 3(a), the values are $\alpha = 2$, $n_1 = 2$ and $n_2 = \frac{4}{2} = 2$, so the function returns $f(5, 6) = 2$ (recall that the binomial coefficient $\binom{\alpha-1}{2}$, for $\alpha = 2$, is 0).

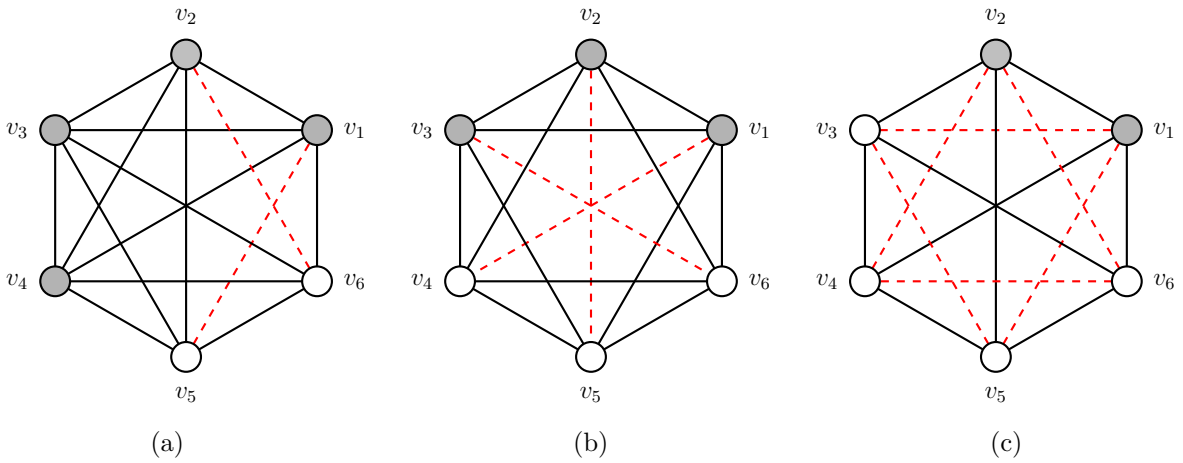


Figure 3: An example clique of size 6. The minimum number of edges to remove from the clique to reduce the clique size to 4 is 2, to 3 is 3 and to 2 is 6.

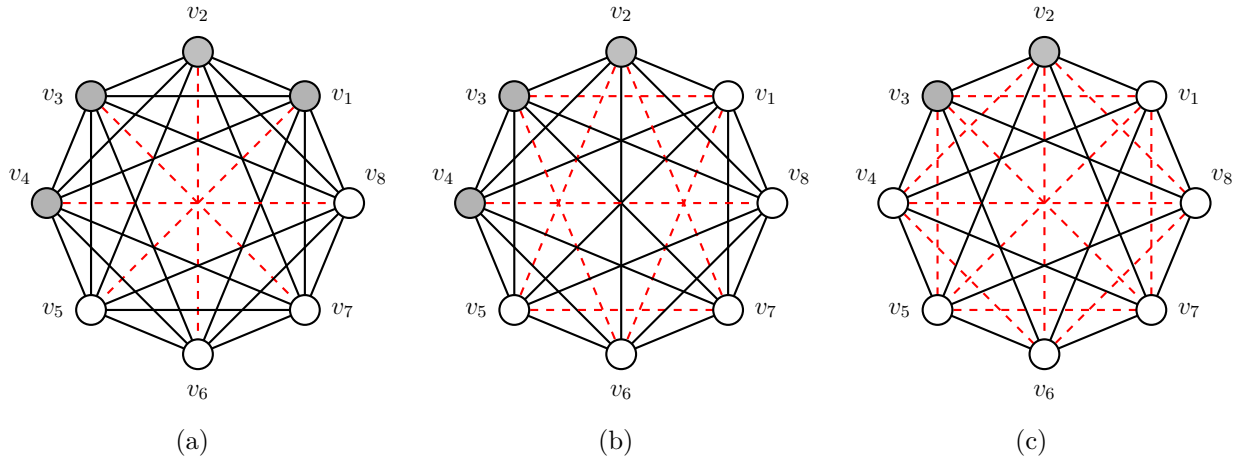


Figure 4: An example clique of size 8. The minimum number of edges to remove from the clique to reduce the clique size to 4 is 4, to 3 is 7 and to 2 is 12.

The result in [12] allows us to replace constraints (4c) with the following stronger ones:

$$\sum_{e \in E(K)} w_e \geq z_\ell \cdot f(\ell, K) \qquad \ell \in L \setminus \{\ell_{\max} + 1\}, K \in \mathcal{K}_{\ell^+},$$

stating that if $z_\ell = 1$, then for any clique K_q of size q, where $\ell \leq q \leq \ell_{\max}$ at least $f(\ell, K_q)$ edges must be covered by the interdiction policy w.

Clique-covering inequalities. Finally, due to the fact that exactly one z variable will be set to one, we can further strengthen the above constraints:

$$\sum_{e \in E(K)} w_e \geq \sum_{\ell' = \ell_{\min}+1}^{\ell} z_{\ell'} \cdot f(\ell', K) \qquad \ell \in L \setminus \{\ell_{\max} + 1\}, K \in \mathcal{K}_\ell. \tag{5}$$

Constraints (5) state that for any clique K_q of size $q \leq \ell_{\max}$, if the value of the optimal solution is ℓ', for $\ell_{\min} \leq \ell' < q$, then at least $f(\ell' + 1, K_q)$ edges must be covered by an optimal interdiction policy. In our default implementation of formulation (4), the weaker constraints (4c) are replaced by the latter inequalities, to which we refer as the *clique-covering inequalities*.

4.2. Comparison against the Benders-like problem reformulation

The mapping between the auxiliary variable ϑ and the objective function term of the ILP model (4) is given as:

$$\vartheta = \sum_{\ell \in L} (\ell - 1) z_\ell.$$

That way, the following family of valid inequalities can be derived for our set-covering-based formulation:

$$\sum_{e \in E(K)} w_e \geq |K| - \sum_{\ell \in L} (\ell - 1) z_\ell, \qquad K \in \mathcal{K}.$$

These cuts are stating that for any clique $K \in \mathcal{K}_q$ of G, if the optimal solution covers all cliques of size ℓ, then at least $q - \ell + 1$ edges from K must be removed. The coefficients of this cut can be tightened for the values of ℓ such that $\ell > |K|$:

$$\sum_{e \in E(K)} w_e \geq |K| - \sum_{\ell' = \ell_{\min}+1}^{\ell} (\ell' - 1) z_{\ell'} - \sum_{\ell' = \ell+1}^{\ell_{\max}+1} |K| z_{\ell'}, \qquad \ell \in L \setminus \{\ell_{\max} + 1\}, K \in \mathcal{K}_\ell. \tag{6}$$

Finally, using the fact that $\sum_{\ell' \in L} z_{\ell'} = 1$, we can rewrite the latter constraint as:

$$\sum_{e \in E(K)} w_e \geq \sum_{\ell' = \ell_{\min}+1}^{\ell} (|K| - \ell' + 1) z_{\ell'}, \qquad \ell \in L \setminus \{\ell_{\max} + 1\}, K \in \mathcal{K}_\ell. \tag{7}$$

We will refer to constraints (7) as *Benders inequalities*, since they are derived from the Benders-like ILP problem reformulation (3).

In what follows, we show that Benders inequalities are dominated by the clique-covering inequalities.

Proposition 3. *For any given clique $K \in \mathcal{K}_q$ of size q ($q \in L \setminus \{\ell_{\max}+1\}$), the associated Benders inequality* (7) *is dominated by the clique-covering inequality* (5).

Proof. Let K be a clique of size q in G. We will show this result by comparing the right-hand sides of inequalities (5) and (7), i.e., by showing that

$$\sum_{e \in E(K)} w_e \geq \sum_{\ell=\ell_{\min}+1}^{q} f(\ell, K) z_\ell \geq \sum_{\ell=\ell_{\min}+1}^{q} (|K| - \ell + 1) z_\ell.$$

To this end, it is enough to show that

$$f(\ell, K) \geq |K| - (\ell - 1), \qquad \text{for } \ell_{\min} + 1 \leq \ell \leq q.$$

The result is trivial for $\ell = q$. Let us therefore focus on $\ell < q$, and let us write $q = k \cdot (\ell - 1) + r$, for some $k \in \mathbb{N}$, and $r \in \{0, \ldots, \ell - 2\}$. We will distinguish two cases: a) $r \neq 0$ (i.e., $\ell - 1$ does not divide q), and b) $r = 0$.

a) In case $r \neq 0$, we have: $\alpha = k + 1$, $n_{\alpha-1} = \ell - 1 - r$, and $n_\alpha = \frac{k \cdot (\ell-1) + r - (\ell - 1 - r)k}{k+1} = r$. Hence, we obtain

$$\begin{aligned} f(\ell, K_q) &= (\ell - 1 - r)\binom{k}{2} + r\binom{k+1}{2} = \\ &= \binom{k}{2}(\ell - 1) + r \cdot k && \geq (\ell - 1)(k - 1) + r = \\ & && = (\ell - 1) \cdot k + r - (\ell - 1) = |K_q| - (\ell - 1). \end{aligned}$$

To see why the latter inequality holds, observe that for $k \geq 2$, we have $\binom{k}{2} \geq k - 1$, and for $k = 1$, the inequality boils down to $r \cdot k \geq r$.

b) For $r = 0$, we have: $\alpha = k$, $n_{\alpha-1} = 0$, and $n_\alpha = \frac{q}{k} = \ell - 1$. Hence, we obtain

$$f(\ell, K_q) = (\ell - 1)\binom{k}{2} = \frac{q(k-1)}{2} \geq |K_q| - (\ell - 1).$$

We notice that the latter inequality holds for $k = 1$, in which case we have $q = \ell - 1$ and therefore the right-hand side turns into zero. For $k \geq 2$, we observe that the inequality can be rewritten as:

$$\frac{q(k-1)}{2} \geq (\ell - 1)(k - 1)$$

which is true as long as $q \geq 2(\ell - 1)$. The latter is always true for $k \geq 2$, $k \in \mathbb{N}$.

□

4.3. Separation of clique-covering inequalities (5)
Separation of both, Benders inequalities and clique-covering inequalities is $\mathcal{NP}$-hard, as it requires solving the maximum clique problem in the interdicted graph. In the following, we detail the separation procedure of clique-covering inequalities for integer points (w^*, z^*) (separation of Benders cuts works similarly). We point out that in modern branch-and-cut solvers, it is sufficient to separate binary LP-relaxation points of our model, while the remaining decomposition of the search space is guaranteed by the branching procedures enhanced by the general-purpose cutting planes.

Exact separation of clique-covering inequalities (5). Given the current binary solution (w^*, z^*) of the relaxed set-covering ILP model, let ℓ^* be the index $\ell \in L$ such that $z_{\ell^*} = 1$. We first calculate a maximum clique in the interdicted graph (i.e., in the graph from which the edges e such that $w_e^* = 1$ are removed). Let K^* be the obtained maximum clique. If $|K^*| \geq \ell^*$, we have detected a violated clique-covering inequality (5), and we add the cut

$$\sum_{e \in E(K^*)} w_e \geq \sum_{\ell' = \ell_{\min}+1}^{\ell^*} f(\ell', K^*) \cdot z_{\ell'}$$

imposed for the detected clique K^* and the current value of ℓ^*.

Heuristic separation of inequalities (5) *using maximal cliques.* During the separation of the inequalities (5), a non-maximal clique of G can be obtained since the current interdiction policy may prevent some vertices to be included in the clique $K \in \mathcal{K}$. In line with what has been proposed in [11], we also explored the possibility of heuristically generating cuts (5) associated to maximal cliques and inserting them into the model, whenever they are violated. Finding the largest clique can be time consuming, for this reason we propose a greedy heuristic procedure which tries to sequentially enlarge the current clique by examining the vertices from the common neighbourhood of a given clique. For the NICP, maximal cliques always lead to stronger cuts (see [11]). On the other hand, for the EICP this is not always the case. To explain this phenomenon, we depict in Figure 5, two possible situations considering the same synthetic graph of Figure 1. We also show the computational efficiency of this heuristic procedure in Section 7.

In Figures 5(a) and 5(b) two different interdiction policies with $k = 5$ ($\{v_1v_2, v_2v_3, v_2v_5, v_2v_6, v_4v_6\}$ and $\{v_1v_3, v_2v_6, v_3v_4, v_4v_5, v_4v_6\}$, respectively) are depicted in red dashed lines. For both of them, a maximum clique in the interdicted graph G_I is $K = \{v_1, v_5, v_6\}$. Considering this clique and assuming $\ell_{\min} = 2$, $\ell_{\max} = 4$ and $\ell^* = 3$, the corresponding cut (5) for both interdiction policies is:

$$w_{v_1v_5} + w_{v_1v_6} + w_{v_5v_6} \geq z_3,$$

given that $f(3, K) = 1$. The cut states that if the optimal solution is equal to two, at least one of the edges from K must be deleted.

In both cases, the vertex v_2 can be added to K resulting in a cut (5) associated to a maximal clique in G. Thus, the cut (5) for maximal clique $\tilde{K} = \{v_1, v_2, v_5, v_6\}$ is:

$$w_{v_1v_2} + w_{v_1v_5} + w_{v_1v_6} + w_{v_2v_5} + w_{v_2v_6} + w_{v_5v_6} \geq 2z_3 + z_4, \tag{8}$$

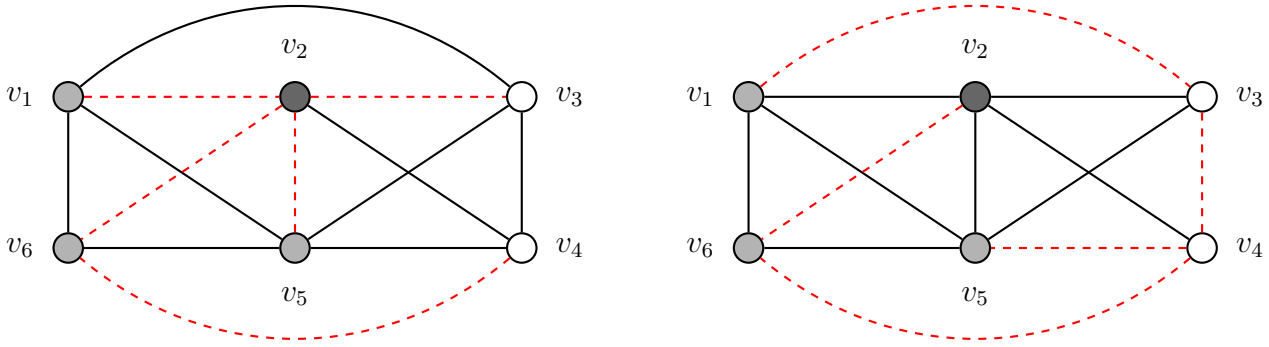


Figure 5: Two different interdiction policies depicted in red dashed lines to show the effects of maximal cliques on the inequalities (5).

given that $f(3, \tilde{K}) = 2$ and $f(4, \tilde{K}) = 1$. Figure 5 shows that this enlarged clique does not always generate a violated inequality: for the interdiction policy given in Figure 5(a), the cut (8) is not violated (we have $3 \geq 2$), whereas for the interdiction policy depicted in Figure 5(b), the cut (8) is violated (we have $1 \geq 2$) and can be added to the model.

5. A combinatorial algorithm to solve the maximum clique separation problem

As explained in the previous section, to separate inequalities (5) and (7) it is necessary to compute a maximum clique in the interdicted graph $G_I = (V_I, E_I)$ derived from each interdiction policy w. To solve the maximum clique problem to proven optimality we have customized the efficient combinatorial branch-and-bound (B&B) algorithm `IMCQ` employed for the node interdiction clique problem in Furini et al. [11], which we will denote `EIMCQ`. One difference between `IMCQ` and `EIMCQ` is that the latter solver is concerned with small and medium graphs only, and has specific customized methods in its initialization phase. Moreover, we have also improved the computation of bounds using bitmask operations wrt to `IMCQ`. The solver's main sources of efficiency derive from techniques employed in [15, 20, 21, 22, 23, 26, 28] amongst others. For the sake of completeness we briefly describe in what follows its main components.

Basic clique enumeration. Each node of the B&B tree is represented by the pair $(\hat{K}_I, \hat{G}_I)$, where $\hat{K}_I$ is a clique being constructed and $\hat{G}_I$ is the subgraph induced by the vertices neighbours to every vertex of $\hat{K}_I$. The n-ary branching scheme explores the vertices of $\hat{G}_I$ in order, i.e., each time a vertex is selected for branching it is added to $\hat{K}_I$ and a new subgraph is determined accordingly. The leaf nodes of the tree correspond to maximal cliques and the algorithm keeps track of the largest one during tree traversal.

Branching. A key idea behind branching in `EIMCQ` is the partitioning of the candidate set $\hat{V}_I$ into two sets denoted the Branching Set B and, its complement, the Pruned Set $P := \hat{V}_I \setminus B$, see, e.g. [15, 28]. The largest Pruned set P (and the corresponding smallest branching set B)

can be obtained by solving the following problem:

$$P := \arg\max_{\bar{P} \subseteq \hat{V}_I} \left\{ |\bar{P}| : LB - |\hat{K}_I| \geq w(\hat{G}_I[\bar{P}]) \right\} \text{ and } \quad B := \hat{V}_I \setminus P,$$

where LB is the size of the incumbent clique, i.e. the largest clique found so far. By construction, P is the largest subset of $\hat{V}_I$ which, by itself, cannot improve the incumbent solution, so branching only occurs on vertices from its complement the Branching Set B. Determining the largest set P is impractical, all the more so in every node of the B&B tree, so `EIMCQ` determines the set heuristically using different bounds for $\omega(\hat{G}_I[\bar{P}])$. The algorithm backtracks when the Branching set B becomes empty.

Bounds. A first upper bound on the clique number used by `EIMCQ` is derived from a κ-colouring C_κ of the subgraph $\hat{G}_I$, by means of the greedy sequential independent set heuristic described in [21, 22]; the Pruned set P is computed as the set of vertices with assigned color number values lower or equal than $LB - |\hat{K}_I|$ in the colouring. In case the Branching set is not empty, it is further reduced by what is known as an *infra-chromatic* bounding procedure [3]. The intuition behind such procedure is to determine subsets $\mathcal{I}$ of color classes (independent sets) in C_κ such that they cannot hold a clique of size $|\mathcal{I}|$. Every time one such set $\mathcal{I}$ is found, the upper bound κ provided by the colouring is reduced by one unit. The infra-chromatic bounding function used by EIMCQ is inspired in the MaxSAT-based bounding procedure of Li et al. [15], and customized with efficient bitstring operations.

Initialization. In the initialization phase, `EIMCQ` reorders the vertices of the graph, computes a feasible solution K_0 and determines a first Branching Set B_0. `EIMCQ` uses the standard minimum width ordering described in the literature for the MCP, see, e.g., [24] for a comparison of different static vertex orderings. Specifically related to the EICP, where `EIMCQ` is called for different edge interdiction policies, we restrict vertex ordering to the original problem graph G for efficiency reasons, and preserve this order in the subsequent calls for every interdicted graph. To compute an initial feasible solution K_0 for the original graph G we use the state-of-the-art Adaptive Multi-start Tabu Search heuristic [30]. In subsequent calls to `EIMCQ` for the different interdiction policies, we switch to a simple greedy clique heuristic which determines a clique by selecting vertices sequentially. We end the preprocessing phase by computing and initial Branching Set B_0 as explained previously, i.e., a first B_0 is derived from a heuristic colouring of the graph and it is further reduced by calling the MaxSAT-based infra-chromatic function.

6. Determining high-quality heuristic solutions

In this section, we describe the heuristic algorithm we designed to quickly obtain high-quality heuristic EICP solutions. Our approach is composed of two phases. The first one consists of an *ILP-based heuristic* which is described in Section 6.1. In this phase, an initial feasible solution is determined by solving a compact ILP model. This initial solution is then improved

[3] the term *infra-chromatic* was first employed in this context in [23].

by a *Local Branching phase* which is described in Section 6.2. This second heuristic phase is based on a *truncated version of our branch-and-cut algorithm*, in which we iteratively impose local-branching constraints. Finally, the computed heuristic solutions are used to initialize and improve the performance of our exact branch-and-cut algorithm designed to solve the EICP to proven optimality (see computational results in Section 7).

6.1. ILP-based heuristic

As described in Section 3, cf. Proposition 1, the EICP can be formulated as follows:

$$\min_{w \in \mathcal{W}_E} \max \sum_{u \in V} x_u - \sum_{uv \in E} w_{uv} y_{uv} \tag{9a}$$

$$\text{s.t.} \qquad x_u + x_v \leq 1 + y_{uv} \qquad uv \in E \tag{9b}$$

$$x_u + x_v \leq 1 \qquad uv \in \overline{E} \tag{9c}$$

$$x_u \in \{0,1\} \qquad u \in V \tag{9d}$$

$$y_{uv} \in \{0,1\} \qquad uv \in E. \tag{9e}$$

This bilevel min-max model is characterized by the leader (or first level) binary variables $w \in \mathcal{W}_E$ which determine the set of interdicted edges. The inner maximization problem has the follower (or second level) binary variables $x_u \in \{0,1\}$ $(u \in V)$ which determine the maximum clique in the interdicted graph. In addition, a second set of binary variables $y_{uv} \in \{0,1\}$ $(uv \in E)$ is used to impose the leader interdiction policy on the follower subproblem using its objective function. The non-linear term in the objective function is the penalty which prevents the follower from using interdicted edges in building the maximum clique in the interdicted graph.

A well-know technique typically used when the follower problem can be modelled as a Linear Program consists in dualizing the inner formulation in order to obtain a compact ILP single level model. We address the interested reader to [14], where these techniques are described in detail and have been successfully used to tackle min-max regret problems with zero duality gap. In our case the inner maximization problem is not an LP but an ILP model. Accordingly, we do not have a zero duality gap. But, after relaxing the follower binary variables and dualizing the resulting LP formulation, we can still obtain a single level ILP model which provides feasible EICP solutions. A similar technique has been successfully used in e.g., [10] for the Min-Max Regret Knapsack Problem or, in [8], for Generalized Interdiction Problems.

As a first consideration, if the x variables are binary, the y variables take binary values as well. So, without loss of generality, we can simply impose $y_{uv} \geq 0$ $(uv \in E)$. Finally, by relaxing the integrality condition (9d) to $x_u \leq 1$ $(u \in V)$ and by introducing dual variables α_{uv} $(uv \in \overline{E})$ for constraints (9c), γ_{uv} $(uv \in E)$ for constraints (9b) and β_u $(u \in V)$ for constraints $x_u \leq 1$ $(u \in V)$, we obtain the following dual problem:

$$\min_{(\alpha,\beta,\gamma)\geq 0} \quad \sum_{uv\in\overline{E}} \alpha_{uv} + \sum_{uv\in E} \gamma_{uv} + \sum_{u\in V} \beta_u \tag{10a}$$

$$\text{s.t.} \quad \sum_{v\in\bar{\delta}(u)} \alpha_{uv} + \sum_{v\in\delta(u)} \gamma_{uv} + \beta_u \geq 1 \qquad u \in V \tag{10b}$$

$$\gamma_{uv} \leq w_{uv} \qquad uv \in E, \tag{10c}$$

where $\bar{\delta}(u)$ is the set of neighbouring vertices of u in the complement graph $\overline{G}$ and $\delta(u)$ is the set of neighbouring vertices of u in the graph G. By embedding this dual in (9), we finally obtain a compact ILP single level model which we call U-EICP :

$$\text{(U-EICP)} \quad \min_{(\alpha,\beta,\gamma)\geq 0} \quad \sum_{uv\in\overline{E}} \alpha_{uv} + \sum_{uv\in E} \gamma_{uv} + \sum_{u\in V} \beta_u \tag{11a}$$

$$\text{s.t.} \quad \sum_{uv\in E} w_{uv} \leq k \tag{11b}$$

$$(10b) - (10c) \tag{11c}$$

$$w_{uv} \in \{0,1\} \qquad uv \in E. \tag{11d}$$

The U-EICP has a polynomial number of constraints and variables. However, the solution value of U-EICP only provides an upper bound for the EICP. Let $\tilde{w}$ be an optimal solution of the U-EICP model. This solution is indeed feasible for the EICP. However, due to the integrality gap introduced by (10), it is not necessarily an optimal EICP solution. Finally, an upper bound on the optimal EICP solution value is computed by solving the maximum clique problem on the interdicted graph using the interdiction policy $\tilde{w}$.

6.2. Local Branching phase

Once the initial feasible interdiction policy $\tilde{w}$ is computed by the ILP-based heuristic, we apply a Local Branching heuristic to further improve its quality. Local branching (LB) is a heuristic approach that uses the power of the general-purpose MIP solvers as black-box tools to strategically explore promising solution subspaces [5]. The main idea is to make the solver explore a neighborhood of a feasible solution with the purpose of (potentially) improving it. The method is based on an ILP formulation where a *local branching constraint* is imposed to truncate the branching tree. This local branching constraint restricts the search space within a certain neighborhood of a given reference solution.

Contrary to the classical Local Branching implementations in which a compact ILP formulation is used to optimally explore a given neighborhood, in our case we are using the set-covering-based ILP formulation, which requires a careful branch-and-cut implementation. In our setting, the benefits of Local Branching are therefore twofold: on the one hand, we are generating stronger upper bounds by iterativelly improving feasible solutions; on the other hand, in each iteration of the neighborhood search, we are generating violated clique-covering inequalities (5), which are valid not only for the local neighborhood of a given reference solution, but

for the global problem formulation as well. Thus, the branch-and-cut which is used in each iteration of the neighborhood search is initialized with the cutting planes generated by the previous iterations (see also [6, 1], where similar combinations of a branch-and-cut with LB have been successfully employed). Moreover, once the Local Branching phase is over, the *CutPool* containing all cutting planes generated during this phase is integrated into the initial LP, before the final branch-and-cut algorithm is launched.

In our case, since the feasible interdiction policy (reference solution) is represented by the binary vector $\tilde{w}$, the local branching constraint explores the δ-neighborhood of the reference solution by *de facto* imposing a maximum *Hamming distance* of δ with the following constraint:

$$\sum_{uv \in E} |w_{uv} - \tilde{w}_{uv}| \leq \delta.$$

In order to linearize the absolute value, we rewrite the local branching constraint as follows:

$$\sum_{uv \in E: \tilde{w}_{uv}=0} w_{uv} + \sum_{uv \in E: \tilde{w}_{uv}=1} (1 - w_{uv}) \leq \delta. \tag{12}$$

Our LB heuristic has two input parameters: (i) the maximum Hamming distance value δ and (ii) the reference solution $\tilde{w}$. As far as the parameter δ is concerned, extensive preliminary experiments showed that a good compromise between the computation time and the quality of the heuristic solutions obtained can be achieved by setting δ equal to $\max\{5, \frac{k}{10}\}$. After the local branching constraint is added to the set-covering-based ILP formulation, we then solve it via the branch-and-cut algorithm. The initial reference solution is obtained by solving the ILP-heuristic proposed in Section 6.1. We also employ the *first improvement* strategy for the neighborhood search, i.e., we restart the branch-and-cut scheme with a new reference solution $\tilde{w}$ as soon a a better solution is found. This update is crucial in order to improve the performance of the method.

7. Computational Results

In this section we assess the computational performance of the newly developed branch-and-cut algorithm which is called `EDGE-INTER` in the remainder of the paper. The purpose of this computational study is threefold: (i) to evaluate the performance of `EDGE-INTER` determining, at the same time, the impact of its main features; (ii) to compare our exact algorithm against the state-of-art method available for general bilevel and interdiction problems from [7]; (iii) to assess the improvements on the computational performance using the additional enhancements presented in Section 4.3.

7.1. Experimental settings and benchmark instances

All the experiments have been performed on a computer with a 3.40 GHz 8-core Intel Core i7-3770 processor and 16 GB RAM, running a 64-bit Linux operating system. The source codes were compiled with gcc 4.8.4 and -03 optimization flag. We used `CPLEX` 12.9.0 (called for brevity `CPLEX` in what follows) and its `CALLABLE LIBRARIES` framework to implement our

branch-and-cut algorithm. `CPLEX` was run in single-thread mode and all `CPLEX` parameters were set to their default values. In all tests we set a time limit of 600 seconds for each run.

We consider four sets of benchmark instances including classical instances from the literature and synthetic graphs:

- *Tang et al. [29] graphs.* The instances by Tang et al. [29] have been proposed to test the performance of an exact algorithm for interdiction problems with binary leader variables. They have been also used in Fischetti et al. [7] to test the performance of a generic exact algorithm for mixed integer bilevel optimization problems. This set of instances is composed of synthetic graphs with $n = |V| \in \{8, 10, 12, 15\}$, edge density $p = \frac{2|E|}{|V|\cdot(|V|-1)}$ chosen as $p \in \{0.7, 0.9\}$ and an interdiction budget $k = \lceil \frac{|E|}{4} \rceil$. Ten instances for each combination of n and p have been created, for a total of 80 EICP instances.

- *Erdös-Rényi (ER) graphs.* We created a set of Erdös-Rényi random graphs $G(n, p)$ with four different number of vertices $n = |V| \in \{25, 50, 75, 100\}$. For the edge densities p, we used 10 different values, i.e., $p \in \{0.1, 0.2, \ldots, 0.9\}$, and $p = 0.95$. Finally, we used 12 different values for the interdiction budged k as a percentage μ of the number of edges $|E|$. Precisely, we used $\mu \in \{1\%, 2\%, \ldots, 10\%\}$ plus $\mu \in \{15\%, 20\%\}$ and we set $k = \lceil \mu \cdot |E| \rceil$. In this way, we created the *ER graphs* data set of 480 EICP instances. Note that also *Tang et al. [29] graphs* are created using the *ER* graph generation procedure. However, in this paper we distinguish between *Tang et al. [29] graphs* which have been used in the previous literature, and the newly generated (larger and more diversified) set of *ER graphs*. Similarly, Fischetti et al. [8] tested their heuristics for generalized interdiction problems (including the EICP) on a family of *ER graphs* with $n \in \{40, 50, 60\}$ and density $p \in \{0, 7, 0.9\}$. These instances can be considered as very similar to the ones from our (much larger) *ER graphs* data set.

- *DIMACS_2-Coloring graphs.* We considered 40 classical instances with $25 \leq |V| < 150$ and $75 \leq |E| \leq 7501$ from the 2nd DIMACS challenge on Graph Coloring (see [4]). These graphs are typically used to test the performance of algorithms for the Vertex Coloring Problem (VCP). Nevertheless, state-of-art exact approaches for the VCP are branch-and-price algorithms [3, 18, 19] which require to solve as pricing problem the Maximum Weighted Stable Set problem. We set the interdiction budget k to the same edge percentage as for the *ER graphs*, obtaining in this manner 480 EICP instances.

- *DIMACS_2-Clique graphs.* We used 16 graphs from the 2nd DIMACS challenge on Maximum Clique with $|V| = 200$ (see [4]). Most of these graphs are still employed as test-bed by state-of-art exact algorithms for the maximum clique problem. For the interdiction budget we used $k \in \{10, 15, 20\}$, creating in this manner 48 EICP instances.

7.2. Testing the impact of the main components of our branch-and-cut algorithm

The newly developed exact branch-and-cut algorithm is based on the set-covering-based ILP formulation of Section 4, in which the clique-covering inequalities (5) are separated on the fly. This algorithm is characterized by the following two main components:

(a) An effective and efficient separation procedure of the constraints (5) using the specialized combinatorial Branch-and-Bound algorithm (`EIMCQ`) described in Section 5. `EIMCQ` is

an enhancement of one of the state-of-the-art MCP algorithms which determines the MCP in the interdicted graphs G_I obtained during tree traversal.

(b) An effective preprocessing phase in which high-quality heuristic EICP solutions are computed. To this end, we designed the two-step heuristic procedure presented in Section 6, i.e., the ILP-based heuristic followed by the Local-Branching phase. Thanks to extensive preliminary experiments we have set the time limit of the heuristic phase to 60 seconds, i.e., 10% of the total time. This choice allows us to obtain the best compromise between the quality of the incumbent solutions found and the overall computational effort.

To measure the impact of these two main components on the computational performance of `EDGE-INTER`, we tested two different configurations in which we gradually enabled these components. Precisely, we tested the following two variants of the main algorithm: `EDGE-INTER-CPLEX` and `EDGE-INTER-NO-LB`. The first configuration corresponds to `EDGE-INTER` in which the Local Branching is turned off and cuts (5) are not separated using `EIMCQ` but using `CPLEX` instead. For the latter, the classical edge-based ILP formulation for solving the maximum clique separation problem is employed. The second configuration, `EDGE-INTER-NO-LB`, corresponds to `EDGE-INTER` in which the Local Branching initialization phase is disabled and the separation is performed using `EIMCQ`. In all three settings, the ILP-based heuristic presented in Section 6.1 is used to initialize $\ell_{\max}$. Solving the underlying ILP model requires only a fraction of a second for most of our instances, so we decided not to disable this feature.

These experiments are performed using the *Erdös-Rényi* and *DIMACS_2-Coloring* sets of instances and, as previously mentioned, setting a time limit of 600 seconds for each run. The results are shown in Table 1. Each row of the table corresponds to 40 instances grouped by the values of the interdiction budget k. The name of the first column is $\mu(\%)$ since the budget k for these instances is computed as a percentage of the number of edges of the graphs ($k = \lceil \mu \cdot |E| \rceil$). The second column reports the number of instances per row of the table. For each method, i.e., `EDGE-INTER` , `EDGE-INTER-CPLEX` and `EDGE-INTER-NO-LB`, we then report the following values: the number of instances solved to proven optimality (columns "*#opt*"), the average computing time in seconds (columns "time"), and finally the average *exit gap* (columns "gap_e") and the average *root-node gap* (columns "gap_r"). The exit gap is calculated as $gap_e = UB - \lceil LB \rceil$, where UB refers to the global upper bound computed by the corresponding method, and LB refers to the global lower bound of the same method in case the time limit is reached. If the instance is solved to proven optimality the exit gap is zero. In order to measure the quality of lower bounds at the root node of the Branch-and-cut tree (denoted by LB_r), we compute the root-node gap with respect to the best known solution value (BKS) as $gap_r = BKS - \lceil LB_r \rceil$. The best known solution value BKS corresponds to the optimal solution value in case one of the methods is able to solve the instance otherwise it is the value of the best incumbent solution found by the three methods. The average values are reported across 40 instances (including those solved to optimality) representing each row.

The results reported in Table 1 demonstrate that both the effective separation procedure and the heuristic initialization are beneficial in improving the computational performance of the branch-and-cut algorithm. As far as the *ER graphs* are concerned, `EDGE-INTER-CPLEX` solves

228 instances to proven optimality and EDGE-INTER-NO-LB solves 234 instances. In contrast, the algorithm EDGE-INTER is capable of solving 240 instances. As far as the *DIMACS_2-Coloring graphs* are concerned, EDGE-INTER-CPLEX solves 357 instances to proven optimality and EDGE-INTER-NO-LB solves 365 instances. The algorithm EDGE-INTER again outperforms the other two algorithms and is capable of solving 374 instances. The table also shows that the value of the interdiction budget k (in terms of μ) has a strong effect on the number of solved instances. Specifically, all the algorithms are capable of solving more instances for small values of k, while as the value of k increases, the number of solved instances consistently decreases with a few exceptions. A similar effect can be observed concerning the computational time, which tends to increase when incrementing the interdiction budget. Large values of the interdiction budget k have an impact on the the number of possible interdiction policies and accordingly increase the difficulty of the instances.

Table 1 also shows that EDGE-INTER guarantees the smallest exit gaps (column gap_e). For the *ER graphs*, the average exit gap is 2.3 while for the *DIMACS_2-Coloring graphs* it is 1.7. The exit gaps of EDGE-INTER-CPLEX and EDGE-INTER-NO-LB are substantially larger. This effect can be explained by the faster separation algorithm of EDGE-INTER, which allows to explore many more nodes of the branching tree; also the heuristic algorithm is capable of finding high-quality incumbent solutions. In addition, these high-quality initial feasible solutions have a beneficial impact on the exploration of the branching tree, *de facto*, pruning dominated branching subtrees. Similarly to the exit gaps, the root node gaps (column gap_r) are also considerably smaller in EDGE-INTER than in the other two algorithms. It is worth noticing that the *ER graphs* are harder to solve compared to the *DIMACS_2-Coloring graphs*, and they are characterized by larger exit and root node gaps.

When comparing EDGE-INTER with EDGE-INTER-CPLEX, we also had a closer look at the percentage of time taken by the separation algorithm with respect to the total time and the number of cuts generated during tree traversal. The separation time taken by EDGE-INTER for the *ER graphs* takes 3.7% of the total time, while for *DIMACS_2-Coloring graphs* it is 22.3%. In addition, the algorithm generates on average 1349 cuts for the *ER graphs* and around 511 for the *DIMACS_2-Coloring graphs*. In the branching tree, EDGE-INTER explores 15336 nodes on average for the *ER graphs* and 47652 nodes for the *DIMACS_2-Coloring graphs*. As far as EDGE-INTER-CPLEX is concerned, the separation time for the *ER graphs* requires 70.9% of the total CPU time and 82.4% for the *DIMACS_2-Coloring graphs*; it generates 726 cuts for the *ER graphs* and 161 cuts for the *DIMACS_2-Coloring graphs*; finally, 15796 and 38617 nodes are explored on average for these two types of graphs, respectively. These results further support the superiority of the separation procedure of EIMCQ. Fast separation allows EDGE-INTER to explore a larger number of branching nodes and, accordingly, substantially reduce the exit gaps.

To summarize, the computational results given in Table 1 show that both components, i.e., the effective separation and the initial heuristic algorithms, are crucial in improving the overall performance of the branch-and-cut algorithm EDGE-INTER.

7.3. Impact of the edge density on the performance of EDGE-INTER

To measure the impact of the edge density on the computational performance of EDGE-INTER, we present Figure 6 considering all the 480 instances of the *ER graphs* set. This figure reports

Table 1: Effect of the main components of `EDGE-INTER` on the *Erdös-Rényi graphs* and *DIMACS_2-Coloring graphs* sets of instances.

			`EDGE-INTER-CPLEX`				`EDGE-INTER-NO-LB`				`EDGE-INTER`			
	$\mu(\%)$	#	#*opt*	time	gap_e	gap_r	#*opt*	time	gap_e	gap_r	#*opt*	time	gap_e	gap_r
Erdös-Rényi graphs	1	40	26	239.8	2.8	5.0	26	210.1	2.2	5.0	26	210.5	1.6	3.2
	2	40	22	287.2	3.2	4.7	25	231.6	2.5	4.7	25	234.8	2.3	3.5
	3	40	21	345.3	3.0	4.5	22	273.8	2.5	4.4	22	282.2	2.5	3.5
	4	40	21	334.2	3.4	4.1	22	286.3	2.9	4.1	22	287.6	2.5	3.4
	5	40	21	330.5	3.8	3.8	21	288.4	2.6	3.8	22	300.2	2.7	3.3
	6	40	19	353.3	4.0	3.6	18	332.5	3.4	3.7	18	348.7	2.7	3.2
	7	40	16	402.7	4.4	3.4	16	363.3	3.6	3.4	17	362.8	2.6	3.2
	8	40	18	377.8	4.4	3.4	17	345.2	3.9	3.2	18	357.0	2.6	3.1
	9	40	17	385.5	4.3	3.2	17	350.0	3.6	3.2	17	364.7	2.5	3.1
	10	40	17	379.1	4.6	3.1	19	335.0	3.6	3.1	19	345.4	2.4	3.0
	15	40	13	455.6	5.2	2.4	14	394.0	3.4	2.4	17	397.1	2.0	2.3
	20	40	17	448.0	4.9	1.7	17	367.8	3.5	1.7	17	383.1	1.7	1.7
			228	361.6	4.0	3.6	234	314.8	3.1	3.5	240	322.8	2.3	3.0
DIMACS_2-Coloring graphs	1	40	33	110.6	2.2	4.6	34	90.4	1.7	4.8	34	100.4	1.0	2.1
	2	40	33	117.1	3.2	4.3	32	121.0	3.2	4.5	33	112.2	1.7	2.6
	3	40	32	135.2	2.8	3.9	33	126.4	2.5	3.9	34	100.9	1.8	2.4
	4	40	29	184.7	3.6	4.2	29	165.0	3.0	4.3	30	164.2	2.0	2.6
	5	40	30	179.5	3.5	3.7	32	124.7	2.4	3.7	32	146.0	1.8	2.5
	6	40	29	214.4	3.3	3.5	27	201.2	2.1	3.5	30	158.3	2.0	2.8
	7	40	30	202.0	3.9	3.3	31	136.6	3.1	3.3	31	145.7	1.8	2.6
	8	40	30	191.2	4.1	3.1	29	165.7	2.5	3.1	30	161.3	1.9	2.6
	9	40	28	242.2	4.2	2.8	30	152.3	3.2	2.8	32	131.4	1.7	2.3
	10	40	27	248.4	5.2	2.8	29	165.8	3.7	2.8	30	169.6	1.8	2.3
	15	40	27	281.3	5.5	2.1	29	169.6	3.6	2.0	28	202.0	1.6	1.9
	20	40	29	280.3	6.0	1.7	30	152.0	3.9	1.6	30	166.4	1.4	1.6
			357	198.9	3.9	3.3	365	147.5	2.9	3.3	374	146.5	1.7	2.4

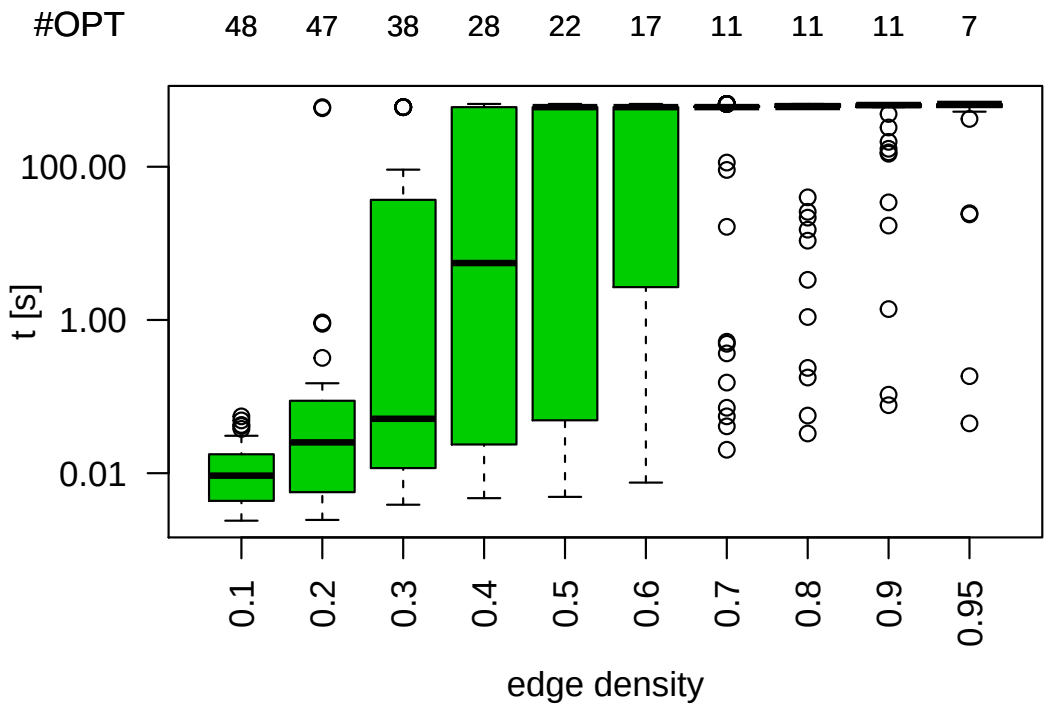
box plots of the computational times grouping the instances by edge density. In the figure, we graphically show the computing times (in logarithmic scale) through their quartiles; the lines extending vertically from the boxes indicate the variability outside the upper and lower quartiles. Finally the outliers are plotted as individual points. In Figure 6, there are 10 groups of 48 instances each. Above the box plots, and for each of the 10 groups, we show the number of instances solved to optimality within the given time limit (`#OPT`).

The results indicate that the computational time increases with the edge density. This

phenomenon can be explained by the fact that dense graphs typically contain many more large cliques and, accordingly, the optimal distribution of the interdiction budget is harder to determine. Every instance with edge density 0.1 can be solved to proven optimality (48/48), but the percentage of instances solved decreases quickly as density rises, i.e., one instance remains unsolved for a density of 0.2 (47/48), whereas for a density of 0.95 (the worst scenario) `EDGE-INTER` is only able to solve 7 (7/48).

Density has also a big impact on CPU times. Solved instances with densities of 0.1 and 0.2 require, on average, less than 0.1 seconds of CPU time, whereas, for the remaining groups, the average CPU time of the solved instances is much greater: typically less than 100 seconds for densities ranging from 0.3 to 0.8, and several hundred seconds for densities ≥ 0.9.

Figure 6: Impact of the edge density on the computational performance of `EDGE-INTER` for the set of *Erdös-Rényi graphs* used for benchmarking.



7.4. Computational performance of `EDGE-INTER` on the DIMACS_2-Clique graphs

In this section we discuss the computational performance of `EDGE-INTER` on the *DIMACS_2-Clique graphs* set of 48 instances with $|V| = 200$. We test three levels of interdiction budget: $k = 10, k = 15$ and $k = 20$. The results are reported in Table 2. The time limit for these tests was set to 600 seconds as for all previous runs. In case an instance could not be solved within the time limit, we report in the table the symbol "t.l.". Besides the clique number $\omega(G)$ of the 16 original graphs, the table shows for each value k the final lower and upper bounds (columns LB and UB) as well as the total CPU time spent in solving the instance

(in seconds). Furthermore, it shows the initial upper bound l_1 computed by the ILP-based heuristic and the improved upper bound l_2 computed by the Local Branching phase. In case an instance is solved to proven optimality we have $LB = UB$.

The algorithm `EDGE-INTER` is able to solve 11, 9 and 7 instances (out of a possible 16) for $k = 10, k = 15$ and $k = 20$ respectively. From the table, we can see that the Local Branching heuristic significantly tightens the upper bounds compared to the ILP heuristic, and that the gaps between l_2 and the optimal solution values (or the lower bound) are very small for several instances, e.g., `brock200_1, brock200_2, brock200_3, brock200_4, san200_0.9_1, sanr200_0.7, sanr200_0.9`. These tests validate the Local Branching heuristic, showing its ability to compute high-quality upper bounds. As already observed in the previous tests, increasing the value of the interdiction budget k also increases the computational difficulty of the *DIMACS_2-Clique graphs*.

Table 2: Computational results of `EDGE-INTER` on the *DIMACS_2-Clique graphs*.

		`EDGE-INTER` ($k = 10$)					`EDGE-INTER` ($k = 15$)					`EDGE-INTER` ($k = 20$)				
	$\omega(G)$	UB	LB	time	l_1	l_2	UB	LB	time	l_1	l_2	UB	LB	time	l_1	l_2
`brock200_1`	21	19	19	33.1	21	20	19	19	21.7	21	20	19	18	t.l.	21	20
`brock200_2`	12	10	10	0.6	12	10	10	10	0.8	12	10	10	10	2.7	12	10
`brock200_3`	15	13	13	1.4	15	13	13	13	6.2	15	13	13	13	9.8	15	13
`brock200_4`	17	15	15	4.0	17	15	15	15	9.1	17	15	14	14	19.0	17	15
`c-fat200-1`	12	11	11	0.1	12	12	10	10	0.1	12	12	10	10	0.1	12	12
`c-fat200-2`	24	22	22	0.2	23	22	21	21	0.2	23	22	20	20	0.2	23	22
`c-fat200-5`	58	55	54	t.l.	58	55	54	52	t.l.	57	54	52	51	t.l.	58	54
`san200_0.7_1`	30	20	20	5.5	30	20	17	17	119.4	28	18	38	33	t.l.	42	39
`san200_0.7_2`	18	15	15	0.7	18	15	15	15	2.7	18	15	41	37	t.l.	54	41
`san200_0.9_1`	70	60	60	66.2	68	68	55	52	t.l.	67	55	17	16	t.l.	30	18
`san200_0.9_2`	60	50	50	66.5	58	58	45	44	t.l.	60	45	15	15	55.6	17	15
`san200_0.9_3`	44	37	36	t.l.	44	37	36	34	t.l.	44	37	50	48	t.l.	68	68
`sanr200_0.7`	18	17	17	15.4	18	17	16	16	20.0	18	17	42	39	t.l.	58	42
`sanr200_0.9`	42	41	37	t.l.	42	42	41	36	t.l.	41	41	36	32	t.l.	44	37
`gen200_p0.9_44`	44	38	37	t.l.	44	38	38	34	t.l.	43	39	16	16	21.9	18	17
`gen200_p0.9_55`	55	45	42	t.l.	53	45	41	39	t.l.	53	41	40	29	t.l.	41	41

7.5. Comparison with a state-of-the-art bilevel solver

In this section we compare the computational performance of `EDGE-INTER` against `BILEVEL`, the exact framework proposed by Fischetti et al. [7] to solve generic bilevel and interdiction problems and applied to solve the EICP. For this comparison, we considered both *Tang et al. [29] graphs* and *ER graphs* sets of instances; as in the previous experiments, a time limit of 600 seconds was set for each run. The results of these tests are reported in Table 3. For the *Tang et al. [29] graphs*, each row of the table corresponds to 20 instances grouped by the number of vertices, i.e., $|V| \in \{8, 10, 12, 15\}$. For the *ER graphs*, each row of the table

represents 120 instances with $|V| \in \{25, 50, 75, 100\}$. For each algorithm, the table reports the number of solved instances per group, the average computing time in seconds, the average exit gap and the average root-node gap (over all instances). The gaps are defined in the same way as in Section 7.2. Finally, the table also reports the nodes explored during tree traversal by both algorithms (column "nodes").

Table 3 demonstrates that `EDGE-INTER` outperforms `BILEVEL` by several orders of magnitude. As far as the *Tang et al. [29] graphs* are concerned, both approaches are able to solve all the 80 instances of this testbed. However, for the largest instances ($|V| = 15$), `EDGE-INTER` spends ≈ 0.2 seconds on average, while `BILEVEL` averages ≈ 20 seconds, two orders of magnitude more. Similar differences in performance can also be observed for the remaining instances of this set. Concering the *ER graphs*, out of the 480 instances `EDGE-INTER` manages to solve a total of 240 (50%), while `BILEVEL` is only able to solve 126 (slightly more than 25%). In addition, `EDGE-INTER` is also characterized by much better exit gaps, which are very small for the instances with $|V| = 25$. For larger instances the exit gaps slightly increase, but the average exit gap is never bigger than ≈ 4 units for the largest instances ($|V| = 100$). In the case of `BILEVEL`, the exit gaps are considerably bigger, the average exit gap reaching up to 11.5 units for the largest group. One of the main drivers of this difference in performance is the quality of the root node lower bounds computed by `EDGE-INTER` with respect to those of `BILEVEL`. By looking at the root node gaps reported in Table 3, we can observe that for both sets of instances these average gaps are much smaller for `EDGE-INTER`, ranging from 0 (instances solved at the root node) to a maximum of 4.9. In the case of `BILEVEL` these gaps are considerably larger with a maximum average value of 6.2. The small root node gaps computationally indicate the stength of our set-covering-based ILP formulation, and, in particular, the effectiveness of clique-covering inequalities (5), which are specifically designed for the EICP, contrary to the generic *intersection cuts* employed by `BILEVEL`.

Finally, by looking at the number of nodes explored during the branching trees by the two algorithms for the *Tang et al. [29] graphs*, we can conclude that for small instances ($|V| < 15$) `EDGE-INTER` explores on average less nodes than `BILEVEL`. On the other hand, for the group with the largest instances ($|V| = 15$) `EDGE-INTER` explores more branching nodes. Notwithstanding, `EDGE-INTER` explores the nodes much faster resulting in a better global computational performance. In the case of the *ER graphs*, `EDGE-INTER` explores many more branching nodes than `BILEVEL`, and it clearly outperform the latter.

7.6. Impact of the heuristic separation of inequalities (5)

In this final section of the computational results, we discuss the impact on the performance of `EDGE-INTER` of the greedy enlargement of the cliques generated during the exact separation problem . The goal of this greedy enlargement is to make the cliques maximal by potentially including extra vertices which are endpoints of the edges interdicted by a given interdiction policy. As mentioned in Section 4.3, maximal cliques do not always produce stronger cuts and, for this reason, we evaluate computationally the impact of this feature. This additional variant of `EDGE-INTER`, is denoted by `EDGE-INTER-MAX`.

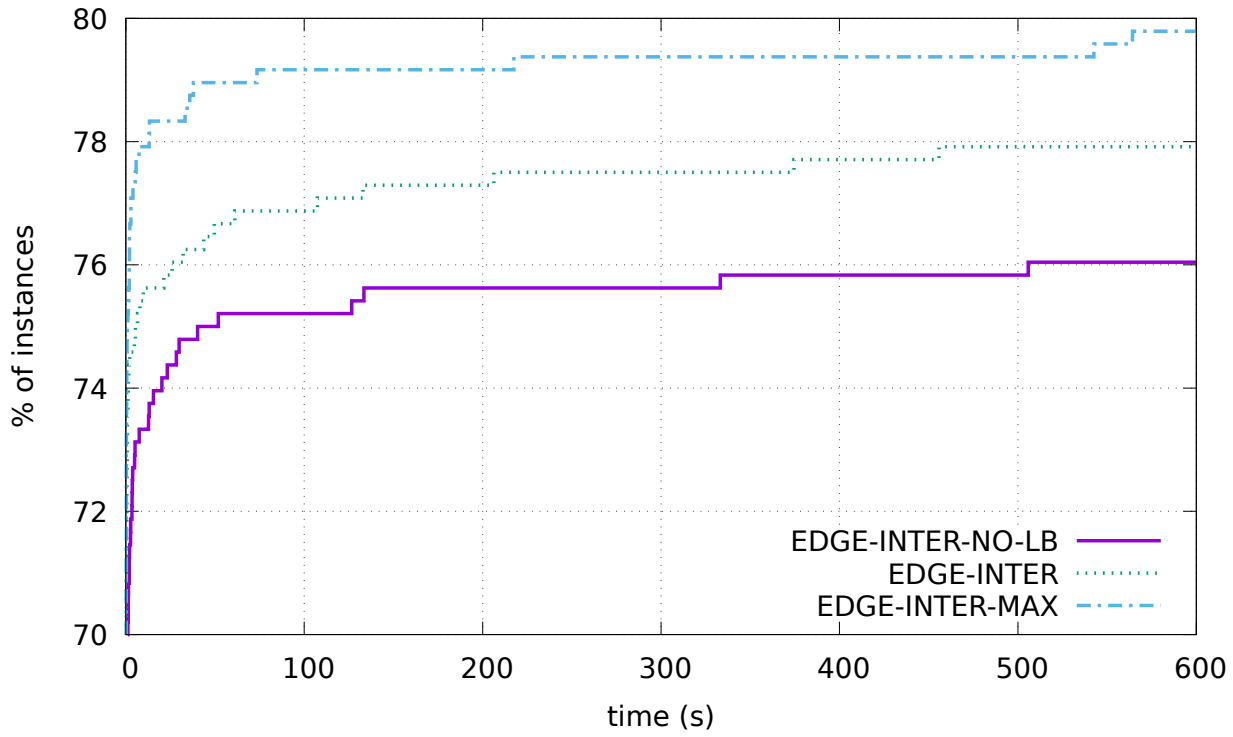
We ran `EDGE-INTER-MAX` on all four sets of instances, but for the *Tang et al. [29] graphs*, *ER graphs* and *DIMACS_2-Clique graphs* the improvements were negligible. However, on the

Table 3: Comparison between EDGE-INTER and BILEVEL, the state-of-the-art bilevel solver from [7].

		EDGE-INTER					BILEVEL [7]				
$\|V\|$	#	*#opt*	time	gap_e	gap_r	nodes	*#opt*	time	gap_e	gap_r	nodes
Tang et al. [29] graphs											
8	20	20	0.0	0	0.0	0	20	0.1	0	0.6	7
10	20	20	0.0	0	0.2	1	20	0.6	0	1.0	28
12	20	20	0.0	0	0.3	15	20	2.7	0	1.0	56
15	20	20	0.2	0	0.3	414	20	20.6	0	1.0	98
Erdös-Rényi graphs											
25	120	112	69.2	0.1	0.7	14060	67	271.9	1.5	1.6	1981
50	120	56	330.4	1.8	2.6	21224	34	444.8	5.8	3.8	417
75	120	40	424.0	3.2	3.9	15148	15	534.4	8.7	5.2	158
100	120	32	467.6	4.3	4.9	10911	10	568.0	11.5	6.2	55

DIMACS_2-Coloring graphs the impact is significant, hence we present results for just this set.

Figure 7: Performance profile of the different configurations of EDGE-INTER for the *DIMACS_2-Coloring graphs*.



To assess the performance of EDGE-INTER-MAX, we report the performance profile of Figure 7. For completeness, we include in this analysis also the variant EDGE-INTER-NO-LB. This performance profile gives a graphical representation of the relative performance of the different configurations of EDGE-INTER, where the horizontal axis represents the CPU time in seconds and the vertical axis represents the cumulative percentage of instances being solved within a given CPU time (reported on the horizontal axis). Figure 7 clearly shows that each of the components is necessary to achieve the best computational performance for the *DIMACS_2-Coloring graphs*.

EDGE-INTER -MAX is the fastest algorithm in most of cases and it manages to solve to proven optimality almost 80% of the instances of this set. More precisely, EDGE-INTER-NO-LB solves 365 instances to optimality, EDGE-INTER solves 374 and EDGE-INTER-MAX solves 383. From these results, we can conclude that the impact of separating the additional valid inequalities (5) is positive in this set.

8. Conclusions

In this manuscript we have addressed the *Edge Interdiction Clique Problem* (EICP), a very challenging problem from a computational perspective. The problem was introduced in [29], and belongs to the family of interdiction problems. The aim of the EICP is to reduce as much as possible the clique number of a graph by removing a given number of edges.

The main goal of this paper was to design an effective exact algorithm to solve the EICP to proven optimality. To this end, we have proposed a new Integer Linear Programming (ILP) formulation with an exponential number of constraints, called the *clique-covering inequalities*. We have shown that the latter inequalities dominate Benders cuts derived from a Benders-like problem reformulation. Moreover, we also designed a branch-and-cut algorithm based on the new ILP formulation which managed to solve the EICP for various instances from literature and for Erdös-Rényi graphs of varying sizes and densities. Our tests showed that the new exact algorithm clearly outperforms the state-of-the-art exact algorithms for the ECIP. This computational performance has been achieved by enhancing the new branch-and-cut algorithm with several tailored components, including a specialized algorithm to separate the clique-covering inequalities and an effective heuristic initialization phase in which high-quality feasible solutions are computed.

As potential future lines of research, we highlight the generalization of the EICP in which, instead of reducing the clique number of a graph, the goal is to minimize the maximum weighted clique in the interdicted graph. Different kinds of weights can be considered, i.e., weights on the vertices of the graph, on its edges or on both (see e.g., [27] for the recent results on the *maximum edge-weighted clique problem* and [28] for the *maximum weighted clique problem*). This generalization of the EICP would be able, for example, to model priorities in the vertices/edges of the graph to be interdicted and, thus, we envisage new interesting practical applications.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Science, Innovation and Universities through the project COGDRIVE (DPI2017-86915-C3-3-R).

References

[1] E. Álvarez-Miranda, M. Goycoolea, I. Ljubić, and M. Sinnl. The generalized reserve set covering problem with connectivity and buffer requirements. *European Journal of Operational Research*, 2019. doi: doi.org/10.1016/j.ejor.2019.07.017. to appear.

[2] S. Coniglio, F. Furini, and P. San Segundo. A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *European Journal of Operational Research*, 2020. doi: doi.org/10.1016/j.ejor.2020.07.023. to appear.

[3] D. Cornaz, F. Furini, and E. Malaguti. Solving vertex coloring problems as maximum weight stable set problems. *Discrete Applied Mathematics*, 217:151–162, 2017.

[4] DIMACS2. 2nd DIMACS Implementation Challenge - NP Hard Problems: Maximum Clique, Graph Coloring, and Satisfiability. URL `http://dimacs.rutgers.edu/Challenges/`.

[5] M. Fischetti and A. Lodi. Local branching. *Mathematical programming*, 98(1-3):23–47, 2003.

[6] M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017.

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

[8] M. Fischetti, M. Monaci, and M. Sinnl. A dynamic reformulation heuristic for generalized interdiction problems. *European Journal of Operational Research*, 267(1):40 – 51, 2018.

[9] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. Interdiction games under monotonicity. *INFORMS Journal on Computing*, 31(2):390–410, 2019.

[10] F. Furini, M. Iori, S. Martello, and M. Yagiura. Heuristic and exact algorithms for the interval min–max regret knapsack problem. *INFORMS Journal on Computing*, 27(2): 392–405, 2015.

[11] F. Furini, I. Ljubić, S. Martin, and P. San Segundo. The maximum clique interdiction problem. *European Journal of Operational Research*, 277(1):112 – 127, 2019.

[12] M.-A. Hassan, I. Kacem, S. Martin, and I. M. Osman. On the m-clique free interval subgraphs polytope: polyhedral analysis and applications. *Journal of Combinatorial Optimization*, 36(3):1074–1101, Oct 2018.

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

[14] A. Kasperski. *Discrete Optimization with Interval Data - Minmax Regret and Fuzzy Approach*, volume 228 of *Studies in Fuzziness and Soft Computing*. Springer, 2008.

[15] C. Li, Z. Fang, H. Jiang, and K. Xu. Incremental upper bound for the maximum clique problem. *INFORMS Journal on Computing*, 30(1):137–153, 2018.

[16] C. Li, Y. Liu, H. Jiang, F. Manyà, and Y. Li. A new upper bound for the maximum weight clique problem. *European Journal of Operational Research*, 270(1):66–77, 2018.

[17] F. Mahdavi Pajouh, V. Boginski, and E. L. Pasiliao. Minimum vertex blocker clique problem. *Networks*, 64(1):48–64, 2014.

[18] E. Malaguti and P. Toth. A survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1):1–34, 2010.

[19] E. Malaguti, M. Monaci, and P. Toth. An exact approach for the vertex coloring problem. *Discrete Optimization*, 8(2):174–190, 2011.

[20] P. San Segundo and C. Tapia. Relaxed approximate coloring in exact maximum clique search. *Computers & Operations Research*, 44:185–192, 2014.

[21] P. San Segundo, D. Rodríguez-Losada, and A. Jiménez. An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research*, 38(2):571–581, 2011.

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

[23] P. San Segundo, A. Nikolaev, and M. Batsyn. Infra-chromatic bound for exact maximum clique search. *Computers & Operations Research*, 64:293–303, 2015.

[24] P. San Segundo, A. Lopez, M. Batsyn, A. Nikolaev, and P. M. Pardalos. Improved initial vertex ordering for exact maximum clique search. *Applied Intelligence*, 45(3):868–880, 2016.

[25] P. San Segundo, A. Lopez, and P. M. Pardalos. A new exact maximum clique algorithm for large and massive sparse graphs. *Computers & Operations Research*, 66:81–94, 2016.

[26] P. San Segundo, A. Nikolaev, M. Batsyn, and P. M. Pardalos. Improved infra-chromatic bound for exact maximum clique search. *Informatica*, 27(2):463–487, 2016.

[27] P. San Segundo, S. Coniglio, F. Furini, and I. Ljubić. A new branch-and-bound algorithm for the maximum edge-weighted clique problem. *European Journal of Operational Research*, 278(1):76 – 90, 2019.

[28] P. San Segundo, F. Furini, and J. Artieda. A new branch-and-bound algorithm for the maximum weighted clique problem. *Computers & Operations Research*, 110:18 – 33, 2019.

[29] Y. Tang, J.-P. P. Richard, and J. C. Smith. A class of algorithms for mixed-integer bilevel min–max optimization. *Journal of Global Optimization*, 66(2):225–262, 2016.

[30] Q. Wu and J. Hao. An adaptive multistart tabu search approach to solve the maximum clique problem. *Journal of Combinatorial Optimization*, 26(1):86–108, 2013.