On integer and bilevel formulations for the k-vertex cut problem

Published Version:
On integer and bilevel formulations for the k-vertex cut problem / Furini F.; Ljubic I.; Malaguti E.; Paronuzzi P.. - In: MATHEMATICAL PROGRAMMING COMPUTATION. - ISSN 1867-2949. - STAMPA. - 12:2(2020), pp. 133-164. [10.1007/s12532-019-00167-1]

Availability:
This version is available at: https://hdl.handle.net/11585/760455 since: 2020-09-02

Published:
DOI: http://doi.org/10.1007/s12532-019-00167-1

Noname manuscript No.
(will be inserted by the editor)

On Integer and Bilevel Formulations for the k-Vertex Cut Problem

Fabio Furini · Ivana Ljubić · Enrico Malaguti · Paolo Paronuzzi

Received: date / Accepted: date

Abstract The family of Critical Node Detection Problems asks for finding a subset of vertices, deletion of which minimizes or maximizes a predefined connectivity measure on the remaining network. We study a problem of this family called the k-vertex cut problem. The problems asks for determining the minimum weight subset of nodes whose removal disconnects a graph into at least k components. We provide two new integer linear programming formulations, along with families of strengthening valid inequalities. Both models involve an exponential number of constraints for which we provide poly-time separation procedures and design the respective branch-and-cut algorithms. In the first formulation one representative vertex is chosen for each of the k mutually disconnected vertex subsets of the remaining graph. In the second formulation, the model is derived from the perspective of a two-phase Stackelberg game in which a leader deletes the vertices in the first phase, and in the second phase a follower builds connected components in the remaining graph. Our computational study demonstrates that a hybrid model in which valid inequalities of both formulations are combined significantly outperforms the state-of-the-art exact methods from the literature.

Keywords Vertex Cut · Mixed-Integer Linear Programming · Bilevel Programming · Branch-and-Cut algorithm.

Fabio Furini
LAMSADE, Université Paris-Dauphine,
E-mail: fabio.furini@dauphine.fr

Ivana Ljubić
ESSEC Business School of Paris,
E-mail: ivana.ljubic@essec.edu

Enrico Malaguti
DEI, University of Bologna,
E-mail: enrico.malaguti@unibo.it

Paolo Paronuzzi
DEI, University of Bologna,
E-mail: paolo.paronuzzi@unibo.it

1. Introduction

In the analysis of networks, their correct functioning frequently depends on a small number of important vertices whose malfunctioning can significantly degrade the performance of the whole network. Depending on the crucial properties that need to be maintained (or achieved) in the network, different vertices may be considered as important. So, for example, if the major concern of a decision maker is the way how information is diffused in the network, we might be interested in finding the key-player vertices or the most influential vertices in the network (see [25]). Similarly, if the decision maker wants to protect the network against malicious attacks that may affect or destroy connectivity, we are talking about the detection of critical vertices of a network. Although there may be some vertices that remain critical no matter which connectivity measure is considered, very often the importance of a vertex changes with the definition of the connectivity measure (see, e.g. [18, 26]).

The family of Critical Node Detection Problems asks for finding a subset of vertices, deletion of which minimizes or maximizes a predefined connectivity measure on the remaining network (see, e.g., [26] for a recent survey). Related to CNDPs is the family of problems in which we are searching for a subset of vertices of minimum weight, deletion of which changes the predefined connectivity measure of the remaining network by a certain value, specified by the decision maker in advance. In this article we study the k-Vertex Cut Problem, which belongs to the latter family of problems, and which is defined as follows.

Definition 1 (k-Vertex-Cut) A *vertex cut* is a set of vertices whose removal disconnects the graph into several connected components. If the number of connected components is at least k, this set is called a *k-vertex cut.* Given a graph $G = (V, E)$, a positive weight w_u for each vertex $u \in V$, and an integer $k \geq 2$, the *k-vertex cut problem* is to find a k-vertex cut of minimum weight.

Besides applications in the analysis of networks, the k-vertex cut problem also models relevant applications in matrix decomposition for solving systems of equations by parallel computing [30]. Given a system of equations with the coefficient matrix A, the *intersection graph* associated to A has one vertex for each column and an edge between a pair of vertices if and only if there exists a row in A where both variables have a nonzero coefficient. When the system is solved by decomposition, it is divided into smaller subsystems that are solved separately. The solutions of the subsystems have to be merged in a consistent way to obtain a solution of the whole system (i.e., if the same variable appears in multiple subsystems, it must take the same value in all of them). The effort for performing this task increases with the number of variables that appear in more than one subsystem. If one wants to partition the equations into k subsystems, the problem of minimizing the number of common variables can be formulated as a vertex k-cut problem.

Figure 1 illustrates an example of a graph with 10 vertices, all with the same weight, along with an optimal solution for the 3-vertex-cut problem: a vertex-cut is of size 3 (given in black), and removal of these vertices results in 3 connected components in the remaining graph.

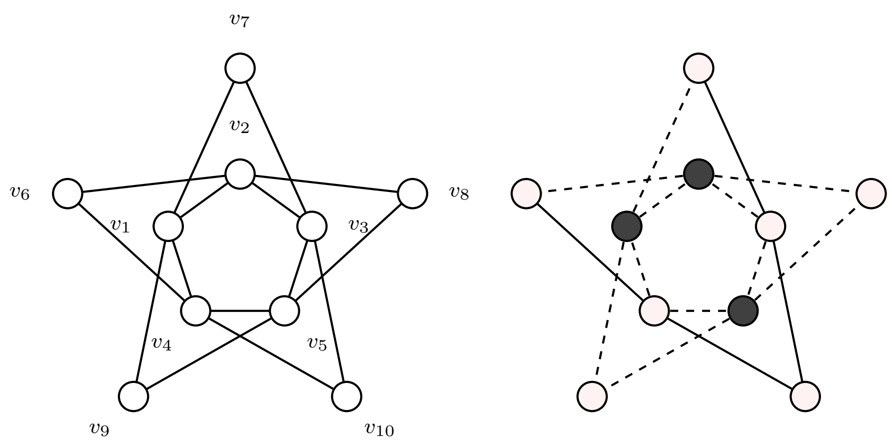


Fig. 1 A graph with 10 vertices of equal weight and an optimal 3-vertex cuts (on the right) represented by the black vertices $\{v_1, v_2, v_5\}$.

By the equivalence with the vertex k-multiclique problem on the complement graph, it has been shown that for any fixed $k \geq 3$, even with unitary weights, the problem is NP-hard [12]. On the other hand, for $k = 2$, the problem can be solved in polynomial time: For uniform vertex weights, the problem is equivalent to calculating the vertex-connectivity of the graph; For the more general case of non-uniform weights, the problem boils down to calculating $O(n^2)$ maximum flows, see [6].

Our Contribution. In this article, we study exact solution approaches to the k-Vertex-Cut problem. We first provide two new Integer Linear Programming (ILP) formulations, along with some families of strengthening valid inequalities. Both models involve an exponential number of constraints for which we provide separation procedures and implement branch-and-cut algorithms. The first formulation, to which we refer to as *Representative Formulation*, asks to choose one representative for each of the k mutually disconnected subsets of the remaining graph. In the second, so-called *Natural Formulation*, we derive the model from the perspective of a two-stage Stackelberg game in which a leader deletes the vertices in the first stage, and in the second stage a follower builds connected components in the remaining graph. In our computational study, we implement these models, compare them with the state-of-the-art approach from [12] and report results of a Hybrid approach in which the Representative and Natural formulations are combined, to provide the new best performing method for the k-vertex cut problem.

The paper is organized as follows: in the remainder of this section, we introduce the notation, we provide a detailed literature overview, and we recall a compact formulation for the problem that was introduced in [9,12]. In Section 2, we derive theoretical properties that allow us to fix some vertices in the optimal solution. The Representative Formulation, along with valid inequalities is given in Section 3, and the bilevel modeling approach is shown in Section 4. Separation procedures for both models are provided in Section 5. Finally, a detailed computational study is provided in Section 6 and conclusions are drawn in Section 7.

Notation. Let K denote the set of integers $\{1, ..., k\}$. Given a simple undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, for an edge $uv \in E$, we say that u and v are *neighbours*. The complement of graph $G = (V, E)$ is a graph $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{uv : uv \notin E\}$. Let $N(u) = \{v \in V | uv \in E\}$ denote the *neighborhood* of u and $\overline{N}(u) = V \setminus (N(u) \cup \{u\})$ denote the *anti-neighborhood* of u. A subset of vertices $W \subset V$ is a *clique of* G, if any two vertices of W are neighbours. A subset of vertices $W \subset V$ is a *stable set* if it is a clique in $\overline{G}$; the cardinality of the largest stable set of G, called the *stability* number of G, is denoted as $\alpha(G)$. We indicate by $deg_G(v)$ the number of edges incident on v in graph G. Given a subset of edges $E' \subseteq E$ of G, we say that E' is *spanning* if for every vertex v of G there is at least an edge in E' incident with v.

We denote by *component* of a graph G a connected subgraph, while a generic *subset* of vertices of G can induce several components. This distinction is relevant because the removal of a k-vertex cut from a graph G can disconnect G in *more* than k components, and we may need to refer instead to exactly k subgraphs, induced by k subsets of vertices.

We will use the observation that a k-vertex cut V_0 is a set of vertices such that $V \setminus V_0$ can be partitioned into k non-empty subsets $V_1, ..., V_k$ that are pairwise disconnected, i.e., there is no edge between two subsets V_i and V_j for all $i \neq j \in \{1, \ldots, k\}$. A necessary and sufficient condition for G to have a k-vertex cut is given in the following

Observation 1 *A graph $G = (V, E)$ admits a k-vertex cut if and only if $\alpha(G) \geq k$.*

Without loss of generality we will assume the condition of Proposition 1 to be satisfied (otherwise, the input instance can be discarded as infeasible). If q is the number of (connected) components of G, we will also assume that $q < k$, otherwise the problem can be trivially solved (empty vertex cut).

1.1 Literature Review

The k-vertex cut problem is polynomially solvable for $k = 2$ [6], and it is NP-hard for $k \geq 3$, when k is part of the input [8]. Only very recently, in [12] the authors show that even for a fixed value of k, the problem remains NP-hard for $k \geq 3$. In addition, the first study on exact methods for the vertex k-cut problem is given in [12]. The authors provide a compact integer programming formulation and a formulation with an exponential number of variables, for which a branch-and-price algorithm is implemented and tested on benchmark instances with up to 200 vertices.

A well studied problem in combinatorial optimization is a closely related problem of finding the minimum-weight *edge k-cut*. The problem consists of finding a subset of edges (instead of vertices) of minimum weight, whose removal separates the graph in at least k connected components. Mainly complexity results are known about this problem: in [4], the author exploits submodularity property to obtain a poly-time lower bound for the problem. For a fixed value of k, the problem reduces to $O(n^{k^2})$ minimum cut problems [21]. Better running times for a fixed value of k are given in [24]. Very recently in [22] an FPT algorithm is given in which the value of k is used as a parameter and which improves the 2-approximation results from e.g., [29].

Another well-studied problem variant is the *multiway cut problem* (sometimes also called the *multiterminal cut problem*), in which a set of terminal vertices T is given and one has to find a minimum-weight subset of edges that separates each terminal from all others. For this problem, complexity is studied in [14] where the authors show that for $|T| \geq 3$ the problem is already NP-hard, and that for a fixed size of T, the problem is solvable in polynomial time on planar graphs. A polyhedral study is given in [11].

There also exists the vertex-counterpart of the multiway cut problem, called the *multi-terminal vertex k-cut problem*, in which one searches for the minimum-weight subset of vertices to remove from a graph, so that every pair of terminals is disconnected (here $k = |T|$). Clearly, a vertex multiway cut exists only if the terminals form an independent set. For this problem, the authors of [19, 20] give an approximation preserving reduction from the vertex cover problem, and provide a 2-approximation algorithm. In [28] the W[1]-hardness of this problem is shown. A path-based integer programming formulation along with some valid inequalities is given in [13]. In addition, a polyhedral analysis is also performed and an efficient branch-and-cut algorithm is developed.

Finally, there also exist problem variants in which cardinality bounds on each component/vertex set are imposed. In the *k-separator problem* the goal is to find a vertex cut whose removal results in a disconnected graph such that the maximum size of each connected component is bounded by k. A bound on the number of components may also be imposed. This problem is introduced in [9] and motivated by matrix decomposition. The authors propose a model (with binary variables indicating the assignment of vertices to the partitions), which is solved by a tailored branch-and-cut algorithm. The complexity of this problem is studied in [7], where also an approximation algorithm is given, along with a integer programming formulations and a polyhedral study. Recently, the authors of [5] present an exponential size integer programming formulation which they solve by branch-and-price, and perform an extensive computational study, in particular on graphs coming from matrix decomposition. The proposed approach consistently solves instances with a large bound on the number of components, and thus complements previous exact approaches that work better/only for smaller number of components. A closely related problem is the one where the cardinality constraints are imposed not on the size of the connected components but on vertex sets. More precisely, the problem consists in finding a subset of vertices to remove from G so that the remaining graph can be partitioned into two sets of cardinality at most k with no edge being incident to both sets. Observe that each set may contain several connected components. This problem is NP-hard even for planar graphs [17] or maximum degree 3 graphs [10]. A first polyhedral study on this problem is done in [3] from which a branch-and-cut algorithm is derived [30].

1.2 Compact Formulation

In this section, we recall the compact formulation, which has been introduced in [12] (for the case where $w_u = 1$ for all $v \in V$). The formulation exploits the fact that a k-vertex cut V_0 is a set of vertices such that $V \setminus V_0$ can be partitioned into k non-empty subsets $V_1, ..., V_k$ that are pairwise disconnected. This formulation is similar to the one introduced in [9] for the the k-separator problem, in particular, it uses the same variables: for each vertex $v \in V$ and each integer $i \in K$, a binary variable y_v^i is defined, such that

$$y_v^i = \begin{cases} 1 & \text{if vertex } v \text{ belongs to subset } i \\ 0 & \text{otherwise} \end{cases} \qquad i \in K, v \in V.$$

The vertices that remain unassigned to any of the subsets V_k (i.e., for which $y_v^i = 0$, for all $i \in K$), are the ones defining the k-vertex cut. This is why instead of minimizing the weight of the k-vertex cut, one can equivalently maximize the sum of the weights of vertices out of the vertex cut (i.e., the weight of vertices in the union $\cup_{i \in K} V_i$).
This compact ILP formulation (denoted as $COMP$) reads as follows:

$$(COMP) \quad \min \sum_{v \in V} w_v - \sum_{i \in K} \sum_{v \in V} w_v y_v^i \tag{1}$$

$$\sum_{i \in K} y_v^i \leq 1 \qquad v \in V \tag{2}$$

$$y_u^i + \sum_{j \in K \setminus \{i\}} y_v^j \leq 1 \qquad i \neq j \in K, uv \in E \tag{3}$$

$$\sum_{v \in V} y_v^i \geq 1 \qquad i \in K \tag{4}$$

$$y_v^i \in \{0, 1\} \qquad i \in K, v \in V. \tag{5}$$

Constraints (2) impose that each vertex belongs to at most one of the subsets V_i, $i \in K$. Constraints (3) ensure that the subsets are pairwise disconnected, i.e., whenever there is an edge between a pair of vertices u and v, these two vertices are not permitted to belong to two different subsets V_i and V_j, $i, j \in$

$K, i \neq j$. Finally, constraints (4) avoid having empty subsets in a feasible solution.

The model $COMP$ has some serious drawbacks. First the number of variables increases linearly with the value of k, and the LP relaxation bound of this model is always equal to zero (we can obtain an optimal LP-solution by setting $y_v^i = 1/k$, for all $v \in V$, $i \in K$, see [12]). Second, the model suffers from symmetries, as the variables can be permuted by obtaining an equivalent solution. This is why an alternative modeling approach has been considered in [12]. A model with an exponential number of variables has been proposed, in which each column represents one of the subsets V_i, $i \in K$, and the corresponding branch-and-price algorithm has been implemented. In what follows, we derive two alternative ways to model the problem after having presented some preprocessing techniques.

2. Preprocessing

In this section we discuss necessary conditions under which a vertex must belong to any optimal k-vertex cut and, *de facto*, the size of the input graph can be reduced.

Assume that a vertex $u \in V$ is not in a k-vertex cut, so that all the vertices in its neighbourhood $N(u)$ either belong to the same subset as u, or are in the k-vertex cut. Therefore, the size of the anti-neighbourhood of u gives an upper bound on the number of disconnected non-empty components that can be obtained.

Proposition 1 *In any feasible solution to the k-vertex cut problem, if for a vertex $u \in G$ we have $k \geq |\overline{N}(u)|+2$, then vertex u must belong to any optimal k-vertex cut.*

Proof Observe that $|\overline{N}(u)|$ is a straight-forward upper bound on the number of components in the anti-neighborhood of u, assuming that the anti-neighborhood defines a stable set, i.e., $\alpha(\overline{N}(u)) = |\overline{N}(u)|$. Vertex u, if not in the k-vertex cut, makes at most a single component along with the vertices in its neighborhood, which leads to at most $k-1$ components, and hence, such a solution would be infeasible. □

We can strengthen this upper bound by analyzing the connected components in the graph induced by the anti-neighborhood of $u \in V$. Let n_C be the number of connected components $(C_1, \dots, C_{n_C})$ in the subgraph $G[\overline{N}(u)]$ induced by $\overline{N}(u)$. Let

$$m(C_i) = \max_{S \subset V(C_i)} \{ \text{ number of connected components of } G[V(C_i) \setminus S] \}$$

(where $V(C_i)$ is the vertex set of the component C_i). Therefore we have:

Proposition 2 *Consider $u \in V$ and let $(C_1, \ldots, C_{n_C})$ be connected components in $G[\overline{N}(u)]$. If we have*

$$k \geq \sum_{i=1}^{n_C} m(C_i) + 2,$$

then vertex u must belong to the k-vertex cut.

Proof Same reasoning as for Proposition 1. □

The following proposition allows us to compute the exact values of $m(C)$ for each of the n_C components:

Proposition 3 *The maximum number of components that can be obtained by deleting some vertices from a connected component C of G is equal to the stability number of C, that is, $m(C) = \alpha(C)$.*

Proof If C contains a stable set of cardinality $\alpha(C)$, we have $\alpha(C)$ non-empty components composed by the vertices of the stable set, so $m(C) \geq \alpha(C)$. Viceversa, if C can be decomposed in $m(C)$ non-empty components, each of these components contains vertices that are not adjacent to any vertex of the other components. By picking a vertex per component, we define a stable set of cardinality $m(C)$, so $m(C) \leq \alpha(C)$, i.e., $m(C) = \alpha(C)$. □

See the computational Section 6, for further implementation details concerning the preprocessing and its effectiveness in reducing the size of input graphs.

3. Representative Formulation

We now propose a novel, alternative formulation for the k-Vertex Cut Problem which is based on the idea of identifying a vertex that is the *representative* of each subset V_i, $i \in K$. This way, it is enough to impose non-connectivity among the representatives to obtain pairwise disconnected subsets. Connected components that are disconnected from any representative can be feasibly assigned to any subset.

The non-connectivity of the representatives can be obtained via an exponential number of path inequalities, similarly to what was done by [13, 27] for the multi-terminal vertex k-cut problem, where each representative is denoted as terminal and it is fixed as an input. We consider two sets of binary variables associated with the vertices, denoting whether a vertex is a representative, and whether a vertex is in the k-vertex cut, respectively. We have

$$z_v = \begin{cases} 1 & \text{if vertex } v \text{ is the representative of a subset} \\ 0 & \text{otherwise} \end{cases} \qquad v \in V,$$

$$x_v = \begin{cases} 1 & \text{if vertex } v \text{ is in the } k\text{-vertex cut} \\ 0 & \text{otherwise} \end{cases} \qquad v \in V,$$

and the corresponding *Representative Formulation* reads as follows:

$$(REP)\ \min \sum_{v \in V} w_v x_v \tag{6}$$

$$\sum_{v \in V} z_v = k \qquad v \in V \tag{7}$$

$$z_u + z_v \leq 1 \qquad uv \in E \tag{8}$$

$$\sum_{t \in V(P) \setminus \{u,v\}} x_t \geq z_u + z_v - 1 \qquad u, v \in V, P \in \Pi_{uv}, uv \notin E \tag{9}$$

$$x_v, z_v \in \{0,1\} \qquad v \in V. \tag{10}$$

In this model, P denotes a simple path in G, $V(P)$ are the vertices connected by P, and Π_{uv} is the set of all simple paths between vertices u and v. The objective function (6) minimizes the weight of the vertices in the k-vertex cut. Constraint (7) ensures that exactly k representative vertices are selected, and constraints (8) impose the set of representative vertices to be a stable set. *Path constraints* (9), in exponential number, impose that at least one vertex of each path $P \in \Pi_{uv}$ between a pair of representative u and v is in the vertex cut (thus disconnecting the two representatives). Note that condition $uv \notin E$ in (9) serves to remove redundant inequalities for which the right-hand-side is equal to zero due to (8).

Proposition 4 *For $k \leq n/2$, the LP relaxation bound of the formulation* (6)-(10) *is equal to zero.*

Proof It can be checked that for $k \leq n/2$, setting $z_v = k/n$ results in a feasible solution in which $x_v = 0$, for all $v \in V$. □

Strengthening Inequalities. Constraints (9) can be lifted by observing that, each time a path in Π_{uv} includes a representative vertex, an additional vertex of the path must be in the vertex-cut:

$$\sum_{t \in V(P) \setminus \{u,v\}} x_t \geq z_u + z_v + \sum_{t \in V(P) \setminus \{u,v\}} z_t - 1, \\ u, v \in V, P \in \Pi_{uv}, uv \notin E. \tag{11}$$

Other families of constraints in polynomial number can be considered in order to strengthen the linear relaxation of the representative model.

Given a vertex u and its neighbourhood $N(u)$, if u is not in a k-vertex cut, then together with (some of) its neighbors it belongs to the same connected component, and hence, at most one of the vertices from $N(u) \cup \{u\}$ can be chosen as representative. Alternatively, if u is in the k-vertex cut, at most $\deg_G(u) = |N(u)|$ vertices can be representatives, which can be expressed by the following *neighborhood constraints*:

$$z_u + \sum_{v \in N(u)} z_v \leq 1 + (\deg_G(u) - 1)x_u \qquad u \in V, \tag{12}$$

paired with the additional condition that a vertex u cannot be a representative and be in the vertex cut at the same time:

$$x_u + z_u \leq 1 \qquad u \in V. \tag{13}$$

Note that an integer solution violating (13) cannot be optimal, so these constraints are not necessary for the correctness of formulation REP. Indeed, consider a solution where for a vertex u we have $x_u = z_u = 1$: by (9) any path from u to another representative vertex w must be disconnected, so u cannot be the (only) vertex disconnecting a path from w to a third representative vertex v. As a consequence, we can set $x_u = 0$ and reduce the cost of the solution while keeping feasibility.

4. Bilevel Approach

We now provide a bilevel point-of-view to the problem, which will allow us to derive a valid ILP formulation in the natural space of the $x_v,\ v \in V$, variables only.

We can see the k-vertex cut problem as a sequential two-player Stackelberg game in which there are two players: a leader and a follower. In the first step, the leader "interdicts" the follower by deleting some vertices from the graph, and in the following step, the follower looks for the largest cycle-free subgraph problem in the remaining graph. The solution of the leader is feasible, if and only if the number of connected components in the subgraph corresponding to the the follower's optimal response is at least k. The leader wants to find a feasible solution where the set of deleted vertices (i.e., the k-vertex cut) has minimum weight.

In the following, we first provide a bilevel integer programming formulation (BILP), which follows the description of the two sequential steps described above. We start by describing a graph property that allows us to model the follower's subproblem as an ILP. It is well known that a graph G is connected if and only if it contains a spanning tree, i.e., the number of edges in its spanning cycle-free subgraph is $|V| - 1$. If G contains multiple connected components, this property can be generalized as follows:

Observation 2 *A graph G has at least k connected components if and only if any cycle-free subgraph of G contains at most $|V| - k$ edges.*

Clearly, a graph G contains at least k connected components if and only if any *maximum* cycle-free subgraph (with respect to the number of edges) contains at most $|V| - k$ edges. By exploiting this property, the k-vertex cut problem can be seen as a Stackelberg game in which the leader searches the smallest subset of vertices V_0 to delete from G, and the follower maximizes the size of the cycle-free subgraph on the remaining graph.

Observation 3 *The solution $V_0 \subset V$ of the leader is feasible if and only if the value of the optimal follower's response (i.e., the maximum number of edges of a cycle-free subgraph in the remaining graph) is at most $|V| - |V_0| - k$.*

4.1 A Bilevel Integer Programming Formulation

The leader decisions are encoded by the same x variables used for the Representative Formulation, where x_v is one if vertex v is "interdicted" (e.g., vertex v is in the k-vertex cut), and zero otherwise. To model the decisions of the follower, we use additional binary variables associated with the edges of G:

$$e_{uv} = \begin{cases} 1 & \text{if edge } uv \text{ is selected to be in the cycle-free subgraph} \\ 0 & \text{otherwise} \end{cases} \qquad uv \in E,$$

The BILP formulation of the k-vertex cut problem reads as follows:

$$(BILP) \quad \min \sum_{v \in V} w_v x_v \tag{14}$$

$$\Phi(x) \leq n - k - \sum_{v \in V} x_v \tag{15}$$

$$x_v \in \{0, 1\} \qquad v \in V. \tag{16}$$

Constraint (15) ensures Observation 3, i.e., it guarantees the feasibility of the solution x of the leader. Thereby, $\Phi(x)$ is the solution value of the follower subproblem, in which the follower searches for cycle-free subgraph on the remaining graph having the largest number of edges. For a solution x^* of the leader, which represents an incidence vector of a set V_0 of interdicted vertices, the follower's subproblem is:

$$\Phi(x^*) = \max \sum_{uv \in E} e_{uv} \tag{17}$$

$$e(S) \leq |S| - 1 \qquad S \subseteq V, |S| \geq 3 \tag{18}$$

$$e_{uv} \leq \begin{cases} 1 - x_v^* \\ 1 - x_u^* \end{cases} \qquad uv \in E \tag{19}$$

$$e_{uv} \in \{0, 1\} \qquad uv \in E, \tag{20}$$

where $e(S) = \sum_{uv \in E; u,v \in S} e_{uv}$. In this model, the subtour elimination constraints (18) ensure that solution of the follower contains no cycles, where constraints (19) guarantee that the follower cannot use the edges that are adjacent to interdicted (deleted) vertices.

It is straightforward to see that any optimal solution of the follower spans the subgraph $G^* = G[V \setminus V_0]$ (except for the vertices with a completely interdicted neighborhood). Indeed, assume that there is a vertex which is not isolated in G^* but has a degree of zero in an optimal follower solution; then adding a random edge adjacent to this vertex improves the value of the follower solution without creating any cycle, fact that leads to a contraction. Hence, the only vertices not spanned by an optimal follower solution are the isolated vertices in the interdicted graph G^*.

The BILP formulation (14)-(16) is non-continuous and non-linear, hence it cannot be plugged into a general purpose solver. Instead, we propose a linearization of the BILP model that results in a new formulation to which we refer as *Natural Formulation*, since it lays in the space of the natural $x_v,\ v \in V$, variables.

4.2 Single-Level Reformulation

In the following, we propose a linearization of the BILP model (14)-(16), by reformulating the follower's subproblem in such a way that the set of its feasible solutions does not depend on the leader. We then derive a single-level reformulation with an exponential number of constraints, associated to extreme points of the follower's polytope. This idea, which resembles the Benders decomposition approach for mixed ILPs, is often applied to (network) interdiction problems [16,23]. The major challenge of this approach is in finding the tightest possible way to reformulate the follower's subproblem, since this reformulation directly affects the quality of the LP relaxation bounds of the associated single-level model. It is known that a tight reformulation is possible in some special cases. For example, if the leader interdicts vertices (edges), and the follower's subproblem admits a hereditary property[1] for its vertex (resp., edge) induced subgraphs, a tight single-level reformulation is possible (see [16, 18]). However, there is no clear rule on how to derive a tight reformulation in general.

In our setting, the leader interdicts *vertices*, but the follower's subproblem is hereditary with respect to *edge-induced* subgraphs, so that the results from [16] cannot be directly applied. Instead, we have the following result:

Proposition 5 *The follower subproblem can be equivalently restated as*

[1] A hereditary property is a property of a graph which also holds for its induced subgraphs.

$$\Phi(x^*) = \max \sum_{uv \in E} e_{uv} \cdot \big(1 - x_u^* - x_v^*\big) \tag{21}$$

$$e(S) \leq |S| - 1 \qquad S \subseteq V, |S| \geq 3 \tag{22}$$

$$e_{uv} \in \{0,1\} \qquad uv \in E. \tag{23}$$

Proof Any optimal solution e^* of (17) -(20) corresponds to a maximum cycle-free subgraph in the *interdicted* graph G^*. Instead, notice that in (21) -(23) the follower solves the *maximum weighted cycle free subgraph* problem on the *original* graph G, with edge weights $w_{uv} := 1 - x_u^* - x_v^*$. However, the weights of an edge uv in E are positive if and only if this edge is not adjacent to any interdicted vertex in V^*. Otherwise, the weight of an edge is zero or -1 (if both its end points are interdicted). Hence, any optimal solution in G^* can be mapped to an optimal solution on G (with the same weight). On the contrary, there always exists an optimal solution on G of the problem (21)-(23) with positive edge weights only, which corresponds to an optimal solution on G^*.□

Observe that the space of feasible solutions of the redefined follower subproblem does not depend on the leader anymore; the only dependence to the solution of the leader is through the objective function. Hence, we can enumerate all feasible solutions of the follower and restate the whole problem as a single-level formulation. This formulation has an exponential number of constraints, one for each extreme point of the follower polytope.

Let $\mathcal{T}$ denote the set of all cycle-free subgraphs of G corresponding to extreme points of the polytope defined as the convex hull of all points satisfying constraints (22) and (23). The non-linear constraint (15) from the BILP formulation can now be replaced by the following exponential family of inequalities:

$$\sum_{uv \in E(T)} \big(1 - x_u - x_v\big) \leq n - \sum_{v \in V} x_v - k \qquad T \in \mathcal{T}. \tag{24}$$

Since every vertex v is counted $\deg_T(v)$ many times in the above constraints (24), they can also be restated as:

$$\sum_{v \in V} \big(\deg_T(v) - 1\big) x_v \geq k - n + |E(T)| \qquad T \in \mathcal{T}. \tag{25}$$

The following result shows that constraints (25) do not have to be imposed for any extreme point from $\mathcal{T}$, it is namely sufficient to concentrate on spanning subgraphs from $\mathcal{T}$ only. Let $\mathcal{T}_G$ denote the *subset* of extreme points from $\mathcal{T}$ being *spanning subgraphs* in G. The following result holds:

Proposition 6 *The following single-level formulation, denoted as* Natural Formulation*, is a valid model for the k-vertex cut problem:*

$$(NAT) \qquad \min \sum_{v \in V} w_v x_v \tag{26}$$

$$\sum_{v \in V} \big(\deg_T(v) - 1\big) x_v \geq k - n + |E(T)| \qquad T \in \mathcal{T}_G \tag{27}$$

$$x_v \in \{0,1\} \qquad v \in V. \tag{28}$$

Proof It is sufficient to show that any inequality associated to a subgraph $T \in \mathcal{T} \setminus \mathcal{T}_G$ can be replaced by an inequality associated to some $T' \in \mathcal{T}_G$. Let us assume for a moment that $|T| = n - 2$ and let $v \notin V(T)$. To create T', given an integer solution x^* that violates the constraint (25), we choose to connect v with some $u \in V(T)$ such that $x_v^* + x_u^* \leq 1$ (this is always possible, unless v and all its neighbours are interdicted). By setting $T' = T \cup \{uv\}$ we obtain a spanning subgraph inequality of type (27) with the same violation as for the inequality (25). For $|T| < n - 2$, this "growing" of the subgraph T can be subsequently repeated until all vertices of G are spanned by T', without changing the violation of the inequality. Finally, in case an interdicted vertex v has an interdicted neighbourhood (however, this cannot happen in an optimal solution, because removing the interdicted vertex from the vertex-cut would improve the leader solution) we need to add the extra constraints:

$$x_u + \sum_{v \in N(u)} x_v \leq \deg_G(u) \qquad u \in V. \tag{29}$$

□

Coefficient lifting. For any $T \in \mathcal{T}_G$, the coefficients next to x_v variables are all non-negative, and hence inequalities (27) can be lifted to:

$$\sum_{v \in V} \big(\min\{\gamma, \deg_T(v) - 1\}\big) x_v \geq \gamma \qquad T \in \mathcal{T}_G, \tag{30}$$

where $\gamma = k - n + |E(T)|$.

Figures 2-3 illustrate a cycle-free subgraph $T \in \mathcal{T} \setminus \mathcal{T}_G$ and a spanning cycle-free subgraph $T' \in \mathcal{T}_G$, along with the associated inequalities. Both inequalities are able to cut off the infeasible solution of Figure 2.

Finally, given the fact that imposing the inequalities (25) associated to spanning subgraphs from $\mathcal{T}_G$ guarantees a valid formulation, a natural question arises: would it be sufficient to impose these inequalities only for spanning trees of G? The following result provides a negative answer to this question:

Proposition 7 *Inequalities* (25) *derived from spanning trees only are not sufficient to ensure a valid formulation for the k-vertex cut problem.*

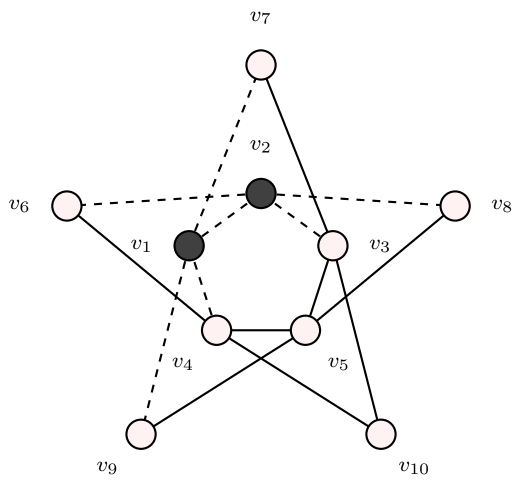


Fig. 2 Infeasible solution for $k = 3$, with the black vertices $\{v_1, v_2\}$ in the vertex cut (the remaining vertices form one connected component).

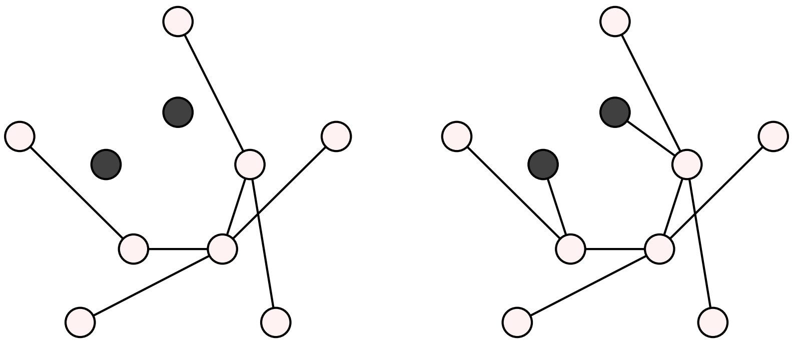
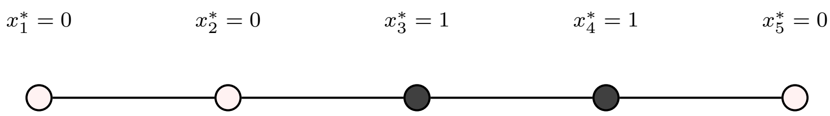


Fig. 3 A cycle-free subgraph $T \in \mathcal{T} \setminus \mathcal{T}_G$ and the associated inequality (25): $-x_1 - x_2 + 2x_3 + x_4 + 3x_5 \geq 0$ (left part). A spanning cycle-free subgraph $T \in \mathcal{T}_G$ and the associated inequality (27): $3x_3 + 2x_4 + 3x_5 \geq 2$; downlifted according to (30) to $x_3 + x_4 + x_5 \geq 1$ (right part).

Proof To prove this result, we provide an instance in which an infeasible solution x^* is not cut off by spanning tree inequalities. Consider a graph composed by a path of 5 vertices, $k = 3$, and the solution x^* depicted in the figure below, where the black vertices represent interdicted ones ($x_3^* = x_4^* = 1$, the remaining values are zero). The solution x^* separates G into only 2 components, hence it is infeasible.



There is a single spanning tree in G, and the associated cut, which is $x_2 + x_3 + x_4 \geq 2$, does not cut off the infeasible point x^*. □

The following propositions characterize the strength of the LP relaxation of the Representative and Natural formulations.

Proposition 8 *If $k \leq n/2$, the bound for the k-vertex cut problem provided by the optimal solution value of the LP relaxation of formulation* (26)-(28) *strictly dominates the corresponding bound provided by the formulation* (6)-(10).

Proof We first show that any feasible solution x^* of the LP relaxation of (26) -(28) can be mapped into a feasible solution of the LP relaxation of (6)-(10) with the same objective function value. The two objective functions are the same, thus we only have to determine z^* satisfying all the constraints of formulation (6)-(10). By exploiting Proposition 4, z^*_u can be fixed to n/k, for each $u \in V$. It is straightforward to check that all the constraints of formulation (6)-(10) are satisfied by (x^*, z^*).

To prove the strictness of the relation, we show that the value of the optimal solution of the LP relaxation of (26) -(28) is strictly larger then 0 for any graph G which is not yet disconnected in at least k components, while by Proposition 4 those of (6)-(10) is always 0. Indeed, any solution of value 0 for (26)-(28) must have $x_u = 0 \; \forall u \in V$. Consider a graph G with q connected components. Any acyclic subgraph of G has at most $n - q$ edges, let t be a subgraph with exactly $n - q$ edges (so it is spanning). By plugging $x_u = 0 \; \forall u \in V$ in (24) (which are equivalent to (27)) for t, we get $n - q \leq n - k$ which is violated if $k > q$, so the solution of cost 0 is infeasible. □

5. Separation Algorithms

In this section, we address separation procedures for the valid inequalities introduced in Sections 3 and 4.2.

Separation of constraints (9). Given a (fractional) solution $x^*, z^* \in [0,1]^V$ to the LP relaxation of model REP, separation of constraints (9) asks for finding a pair of vertices u, v such that there is a path $P^* \in \Pi_{uv}$ with

$$z^*_u + z^*_v > \sum_{t \in V(P^*) \setminus \{u,v\}} x^*_t + 1. \tag{31}$$

For each pair of vertices, we can search for such a path in polynomial time by solving a shortest path problem from u to v on $G(V, E)$, where we define the length of each edge $(i, j) \in E$ as

$$l_{ij} = \frac{x^*_i + x^*_j}{2} \tag{32}$$

(note that the constant term $\frac{x^*_u + x^*_v}{2}$ has to be removed from the length of each path).

Concerning the computation of shortest paths, for fractional solutions, one can solve the All Pairs Shortest Path problem through the Floyd Warshall algorithm. In the case of integer solutions, finding a shortest path between a vertex u and all other vertices can be done by performing a simple breath-first search (BFS) procedure in the support graph G^* in which vertices v such that $x^*_v = 1$ are removed. The BFS tree guarantees that each vertex v at layer ℓ in that tree has the shortest distance ℓ from the source u. If the vertices are not connected, the distance is ∞. Hence, separation of integer solutions can be done in $O(|V||E|)$ time.

Observation 4 *Separation of constraints* (9) *can be performed in polynomial time.*

Separation of constraints (11). Constraints (11), that are the lifted version of (9), can be still carried on by solving a shortest path problem from u to v on $G(V,E)$, where we define the length of each edge $(i,j) \in E$ as

$$l_{ij} = \frac{x^*_i + x^*_j}{2} - \frac{z^*_i + z^*_j}{2}, \tag{33}$$

(still the constant term $\frac{x^*_u + x^*_v}{2}$ has to be removed from the length of each path). Since edges can have negative weight, we solve heuristically this problem in two steps (where the first step can be skipped):

- first we heuristically solve a *longest path* problem with lengths as defined in (33) with opposite sign. We implemented a greedy procedure that obtains such *long path* starting from the edge with the largest weight and then it builds a path by adding the edge with the largest and positive weight that is adjacent to the current path, without closing a cycle;
- second, if in the previous step no violated cut is found, we compute the shortest paths P^* with the nonnegative lengths as defined in (32), and then check the value of the z^*_w variables for $w \in V(P^*) \setminus \{u,v\}$. This way, we have a separation procedure that is exact for (9) and heuristic for (11).

Separation of constraints (27). Let x^* be the current solution to the LP relaxation of model NAT. We define edge-weights as

$$w^*_{uv} = 1 - x^*_u - x^*_v, \quad uv \in E$$

and search for the maximum-weighted cycle-free subgraph in G. Let W^* denote the weight of the obtained subgraph; if $W^* > n - k - \sum_{v \in V} x^*_v$, we have detected a violated inequality.

For fractional points x^* the maximum-weighted cycle-free subgraph can be detected in $O(|E| \log |V|)$ by running an adaptation of Kruskal's algorithm for minimum-spanning trees. Edges are sorted in a non-increasing order according to their weight, and then Kruskal's algorithm is applied, i.e., each edge in this ordering is selected to be included in the subgraph being constructed, provided it does not close a cycle. The algorithm stops as soon as an edge with negative weight is encountered in the ordering.

Separation of integer points x^* can be performed in $O(|E|)$ time. In this case, all edge weights are equal to 1, 0 or -1. Following the result of Proposition 5, it is sufficient to consider the graph defined by edges with weight equal to one, which corresponds to $G^* = G[V \setminus V_0]$ where V_0 are interdicted vertices encoded by x^*. Hence, it is sufficient to run any graph traversal algorithm on G^* (like, e.g., BFS) to find connected components in G^*.

Observation 5 *Separation of constraints* (27) *can be performed in polynomial time.*

Since there are several alternative subgraphs describing connected components, to avoid shallow cuts, when separating integer points we shuffle the set of edges of G^* before each separation call. This procedure guarantees to find a cut of type (25), where the associated subgraph T is not necessarily spanning all vertices from V.

However, it is (always) possible to construct a Spanning Subgraph cut starting from an infeasible integer solution x^* and a cut associated with a (nonspanning) acyclic subgraph $T \in \mathcal{T}$ violated by this solution. To do so, we scan first isolated vertices in the interdicted graph G^* and we assign them an interdicted neighbor, then for all still non-spanned interdicted vertices we assign them to one of their neighbors in $V \setminus V_0$. In this way we do not change the weight of the obtained $T \in \mathcal{T}$ since these edges have 0 weight. This repairing step requires $O(|E|)$ steps, so that the total separation time remains $O(|E|)$.

6. Computational results

The goal of our computational experiments is to test the performance of the proposed formulations, i.e., the Representative Formulation REP (Section 3) and the Natural Formulation NAT (Section 4.2). Both formulations, having an exponential number of constraints, are solved within a branch-and-cut framework. We have proposed several variants and valid inequalities for each formulation, and thus a second goal of this section is to identify their best configuration. In addition, we propose and test a *Hybrid Formulation* obtained by combining elements of the two formulations.

Finally, we assess the computational performance of our best branch-and-cut algorithm by comparison with the Compact Formulation $COMP$ (Section 1.2), and with a state-of-the-art branch-and-price algorithm proposed in [12], and based on a formulation with exponentially many variables.

6.1 Experimental Setting

Benchmark instances. In our experiments we have two sets of instances, which are the ones considered in the computational experiments of [12]. All instances have weights $w_v = 1$ for all $v \in V$. The first set includes all the classical Vertex Coloring instances [1] having up to 200 vertices, and all the 10th DIMACS instances [2] having up to 300 vertices (instances with $\alpha(G) \geq 5$). The features of the 59 selected instances are reported in the first part of Table 1 where, after the instance name, we show the number of vertices (n) and edges (m), the stability number ($\alpha(G)$), and the optimal solution value of the k-vertex cut problem (size of the optimal vertex cut) for values of $k \in \{5, 10, 15, 20\}$, when it can be found by one of the methods discussed in this section or in [12]. Missing entries correspond to infeasible problems ($\alpha(G) < k$), while unknown optimal values are indicated by a "-" (these are the instances which are not solved within timelimit). Trivially solved instances are indicated by a "·" (these are the instances which, before or after preprocessing, have q connected components, with $q \geq k$). The second set of 59 instances, whose features are given in Table 2, were proposed in [30]. This set is a collection of intersection graphs of the coefficient matrices of linear equations systems, arising from various applications. When solving the k-vertex cut problem for a given value of k, we remove from our analysis all infeasible and trivial instances.

All the instances are preprocessed off-line by checking the condition of Proposition 3. In particular, for each vertex the stability number of its anti-neighborhood is computed and, when the condition of the proposition is met, the vertex is removed. Although this asks for solving a NP-hard Maximum Stable Set problem, the associated computing time in negligible for the size of graphs we consider. As long as at least a vertex is removed from the graph, the procedure is iteratively repeated. In our testbed, graph reductions are achieved only for a limited subset of instances, namely, 20, 11, 17 and 16 instances for $k = 5, 10, 15$ and 20, respectively. While for many instances only one or two vertices are removed, in some cases many vertices are removed, with up to 113 vertices out of 125. In 6 cases the resulting instance is solved (i.e., it is disconnected in q components, with $q \geq k$). These instances are marked as trivial in Tables 1 and 2. Preprocessing is applied before instances are tackled by any of the solution algorithms here described, in other worlds, all methods receive the same input (preprocessed) graph.

Detailed results for the preprocessing are reported in the Appendix.

Computational environment. All the experiments, including the runs of the branch-and-price algorithm from [12], are performed on a computer with an i7-6900K processor clocked at 3.20 GHz and 64 GB RAM under GNU/Linux Ubuntu 16.04. We use `CPLEX 12.7.1` and the `Concert Technology` framework to implement our branch-and-cut algorithms. The Compact Formulation is solved with the `CPLEX` MIP solver. `CPLEX` is run in single-threaded mode and all `CPLEX` parameters are set to their default values. A time limit of one hour is set for each tested instance.

				Optimal Values								Optimal Values			
	n	m	$\alpha(G)$	$k=5$	$k=10$	$k=15$	$k=20$		n	m	$\alpha(G)$	$k=5$	$k=10$	$k=15$	$k=20$
1-FullIns_3	30	100	14	7	11			miles750	128	2113	12	20	75		
1-FullIns_4	93	593	45	9	13	18	22	mug100_1	100	166	33	5	10	15	20
1-Insertions_4	67	232	32	7	12	16	22	mug100_25	100	166	33	5	10	15	20
2-FullIns_3	52	201	25	8	13	17	23	mug88_1	88	146	29	4	9	15	20
2-Insertions_3	37	72	18	6	10	16		mug88_25	88	146	29	4	9	14	19
2-Insertions_4	149	541	74	7	11	17	22	mulsol.i.2	188	3885	90	·	·	·	18
3-FullIns_3	80	346	37	9	14	17	25	mulsol.i.3	184	3916	86	·	·	18	19
3-Insertions_3	56	110	27	6	11	16	21	mulsol.i.4	185	3946	86	·	·	18	19
4-FullIns_3	114	541	55	9	15	18	23	mulsol.i.5	186	3973	88	·	·	18	19
4-Insertions_3	79	156	39	6	11	16	21	myciel3	11	20	5	·			
5-FullIns_3	154	792	72	9	15	19	22	myciel4	23	71	11	7	12		
adjnoun	112	425	53	2	6	11	16	myciel5	47	236	23	8	13	18	23
anna	138	493	80	1	1	2	2	myciel6	95	755	47	9	14	19	24
celegansneural	297	2148	110	1	1	2	6	myciel7	191	2360	95	10	15	20	25
chesapeake	39	170	17	7	12	17		polbooks	105	441	43	8	15	19	25
david	87	406	36	·	·	4	9	queen10_10	100	1470	10	-	90		
dolphins	62	159	28	2	7	13	19	queen11_11	121	1980	11	-	-		
DSJC125.1	125	736	34	-	-	-	-	queen12_12	144	2596	12	-	-		
DSJC125.5	125	3891	10	-	115			queen13_13	169	3328	13	-	-		
football	115	613	21	-	-	-	71	queen14_14	196	4186	14	-	-		
games120	120	638	22	-	-	-	67	queen5_5	25	160	5	20			
huck	74	301	27	1	3	6	9	queen6_6	36	290	6	28			
jazz	198	2742	40	4	12	25	-	queen7_7	49	476	7	38			
jean	80	254	38	1	1	2	4	queen8_12	96	1368	8	-			
karate	34	78	20	2	4	6	11	queen8_8	64	728	8	48			
lesmis	77	254	35	1	2	3	5	queen9_9	81	1056	9	59			
miles1000	128	3216	8	53				r125.1	125	209	49	·	·	1	5
miles1500	128	5198	5	115				r125.1c	125	7501	7	116			
miles250	128	387	44	·	·	4	11	r125.5	125	3838	5	91			
miles500	128	1170	18	7	-	-									

Table 1 Instance features (Coloring and DIMACS)

				Optimal Values								Optimal Values			
	n	m	$\alpha(G)$	$k=5$	$k=10$	$k=15$	$k=20$		n	m	$\alpha(G)$	$k=5$	$k=10$	$k=15$	$k=20$
arc130	130	7763	6	83				L120.fidap022	120	4307	5	87			
ash219	85	219	29	7	16	26	34	L120.fidap025	120	2787	5	·			
ash331	104	331	30	8	21	-	-	L120.fidapm02	120	4626	5	91			
ash85	85	616	14	22	-			L120.rbs480a	120	3273	6	76			
bcspwr01	39	118	13	7	16			L120.wm2	120	3387	23	3	8	13	41
bcspwr02	49	177	16	7	16	24		L125.ash608	125	390	37	8	-	-	-
bcspwr03	118	576	32	10	23	35	46	L125.bcsstk05	125	2701	9	41			
bfw62a	62	639	8	22				L125.can__161	125	1257	15	-	-	-	
can__144	144	1656	12	-	-			L125.can__187	125	1022	20	-	-	-	102
can61	61	866	6	39				L125.dwt__162	125	943	16	-	-	-	
can62	62	210	18	7	17	27		L125.dwt__193	125	2982	8	56			
can73	73	652	13	28	-			L125.fs_183_1	125	3392	9	16			
can96	96	912	10	-	-			L125.gre__185	125	1177	19	27	-	-	
curtis54	54	337	9	16				L125.lop163	125	1218	17	-	-	-	
dwt___59	59	256	15	10	25	41		L125.west0167	125	444	39	5	11	17	24
dwt66	66	255	13	15	-			L125.will199	125	386	45	5	13	20	27
dwt72	72	170	24	7	16	26	36	L80.cavity01	80	1201	31	10	10	20	31
dwt87	87	726	16	11	29	54		L80.fidap025	80	1201	5	·			
gre__115	115	576	33	12	24	-	-	L80.steam2	80	1272	6	48			
ibm32	32	179	8	16				L80.wm1	80	1786	15	15	36	49	
impcol_b	59	329	20	5	13	23	38	L80.wm2	80	1848	11	4	48		
L100.cavity01	100	1844	36	10	19	21	32	L80.wm3	80	1739	13	4	12		
L100.fidap025	100	2031	5	·				lund_a	147	2837	10	-	-		
L100.fidapm02	100	3090	5	80				pores_1	30	179	6	20			
L100.rbs480a	100	2550	5	66				rw136	136	641	39	7	-	-	-
L100.steam2	100	1766	6	56				steam3	80	712	7	32			
L100.wm1	100	2956	17	15	28	48		west0067	67	411	12	20	-		
L100.wm2	100	3039	12	4	41			west0132	132	560	39	5	12	21	29
L100.wm3	100	2934	15	4	12	53		will57	57	304	10	7	22		
L120.cavity01	120	2972	36	10	21	23	32								

Table 2 Instance features (Intersection graphs)

6.2 Results for Representative, Natural and Hybrid Formulations

We tested several different configurations of the Representative Formulation (e.g., changing the separation strategy, removing strengthening constraints, etc.), and we report detailed computational results for the following two configurations:

- we denote by REP the formulation (6) - (8), (10) - (13). Constraints (11) are separated by only applying the second step of the procedure described in Section 5, that is, by computing shortest paths on a graph with positive edge weights;
- we denote by REP_{lp} the same formulation, where (11) are separated by applying both steps of the procedure described in Section 5, that is, by heuristically computing a long path in a graph with positive and negative edge weights.

Different frequencies and tolerances for the separation procedure were tested for all configurations. According to our extensive preliminary computational experiments, the best choice is to stop the cut separation when the absolute violation is smaller than 0.5 (*violation tolerance*). We call the separation procedure for all integer points and for fractional points every 100 nodes of the branching tree.

Inequalities (8), that are expressed for each edge in $E(G)$, can be strengthened to clique inequalities. However (as confirmed by our preliminary computational experiments) modern MIP solvers are very effective in the automatic separation of clique inequalities, and hence we keep edge constraints in our formulation.

Similarly, we tested several different configurations of the Natural Formulation, and we report detailed computational results for the following two:

- we denote by NAT the formulation (25), (26) and (28), where (25) are lifted to (30) when spanning;
- we denote by NAT_s the previous formulation where the family of constraints (25) are made spanning for all integer solutions, and then lifted to (30).

We tested different frequencies and tolerances of the separation procedure and the best choice for the violation tolerance is also in this case 0.5. We call the separation procedure for all integer points and for fractional points at all the nodes of the branching tree.

The Representative and the Natural Formulations use the same natural variables $x_v,\ v \in V$, to describe which vertices are in the k-vertex cut, and implement alternative sets of contraints to impose the required number of nonempty disconnected components. Although the Natural Formulation showed more effective than the Representative Formulation (see results in the following), there are some instances on which the latter has a better performance. In addition, in our preliminary computational experiments we observed that, thanks to the presence of a stable set constraints (8), the Representative Formulation is much faster in detecting infeasible instances (i.e., those with $\alpha(G) < k$). Infeasible instances were removed from our testbed, however, we expect the Representative Formulation to be fast in detecting infeasibilities also at the nodes on the branch-and-cut tree. Therefore, it makes sense trying to obtain a more effective formulation by integrating the two into a *Hybrid* model.

In order to explore the direction of embedding into the Natural Formulation the advantages of the Representative one (i.e., solving some specific instance and fast detection of infeasibilities after branching), we designed the following Hybrid configuration:

- we denote by HYB Formulation NAT_s with additional constraints (7), (8), (10), (12) and (13).

Aggregated results for the first set of instances (Vertex Coloring and DIMACS) are reported in Table 3, where the first column gives the considered value of k. Then the table reports, for each configuration of the Representative, Natural and Hybrid Formulations described above, the number of instances solved to optimality; the average computing time in seconds (for the subset of instances solved to optimality by all configurations), the average number of explored nodes in the branching tree (for the subset of instances solved to optimality by all configurations); the average percentage gap of the LP relaxation computed as $100 \cdot ((UB - LP)/UB)$, where UB is the optimal or best known solution value and LP is the optimal value of the LP relaxation; the average time to solve the LP relaxation. Violation tolerance is set to 0.1 when solving LPs. The last three rows of the table report the averages over all values of k.

The configurations reported in Table 3 have improving performance. When moving from REP to REP_{lp}, the number of instances solved to optimality is increased for all value of k, except $k = 5$. The improved results are explained by comparing the values of the LP gap of REP and REP_{lp}: the table clearly shows that separating inequalities (11) by applying both steps of the procedure described in Section 5 allows to close much more LP gap. Using Natural Formulations (NAT and NAT_s) for all values of k the number of instances solved to optimality is increased, and the number of nodes explored by the branch-and-cut algorithm is reduced by 3 orders of magnitude. This can be attributed to the significantly smaller LP relaxation gaps of Natural Formulations, when compared to those obtained using Representative Formulations. Comparing formulations NAT and NAT_s, the latter has a slightly better performance, and can solve 2 more instances on the whole set. Finally, the table shows that the best computational performances is provided by HYB which is able to solve 132 instances (out of 169). The number of explored nodes by the branch-and-cut algorithm is one third of that of NAT_s. As anticipated, this is as a result of the introduction of the constraints from the Representative Formulation, which allow to fast detect infeasible nodes in the branching tree. Summarizing from Table 3 we can conclude that HYB is the best formulation

Table 3 Performance comparison for different configurations of the Representative, Natural and Hybrid Formulations on the first set of instances (Vertex Coloring and DIMACS).

k		REP	REP_{lp}	NAT	NAT_s	HYB
	Opt. (out of 51)	29	27	33	34	35
	Avg Time	148.70	105.57	7.40	3.79	1.07
5	Avg Nodes	61524	24174	70	73	29
	LP Avg Gap	89.55	67.15	22.96	22.76	22.85
	LP Avg Time	0.01	0.17	0.24	0.21	0.32
	Opt. (out of 41)	20	23	29	30	32
	Avg Time	201.66	319.21	2.11	1.52	2.43
10	Avg Nodes	41683	32568	6	7	5
	LP Avg Gap	72.27	46.34	13.88	13.94	14.00
	LP Avg Time	0.05	1.32	0.37	0.33	0.54
	Opt. (out of 38)	22	24	33	32	33
	Avg Time	96.75	52.17	316.91	226.47	3.57
15	Avg Nodes	48078	10923	39	35	12
	LP Avg Gap	65.99	48.75	16.91	16.96	16.94
	LP Avg Time	0.06	138.57	0.18	0.17	0.33
	Opt. (out of 36)	18	22	31	32	32
	Avg Time	141.32	351.13	190.94	41.70	3.66
20	Avg Nodes	47735	25595	58	48	11
	LP Avg Gap	58.65	38.37	17.12	17.11	17.12
	LP Avg Time	0.07	1.93	0.24	0.24	0.49
	Total Opt. (out of 166)	89	96	126	128	132
	Total Avg Time	146.75	194.04	121.10	66.20	2.55
	Total Avg Nodes	50656	23169	45	43	15
	Total Avg LP Gap	73.19	51.69	18.11	18.07	18.11
	Total Avg LP Time	0.04	34.98	0.25	0.24	0.41

proposed in this paper. We now compare its performances with the state-of-the-art algorithm present in the literature for the k-vertex cut problem.

6.3 Comparison with state-of-the-art solution methods

In this section we compare the results of our best formulation (HYB) with the solution of the *Compact Formulation* (denoted as $COMP$) solved by means of the general purpose `CPLEX` MIP solver, and with a state-of-the-art branch-and-price algorithm proposed in [12] (denoted as BP).

When solving the *Compact Formulation*, as suggested in [12], the formulation is enhanced by a preprocessing phase in which a subset of variables is removed so as to reduce the symmetry of the formulation and to improve the quality of the associated LP relaxation. In this preprocessing, we search for $k-1$ vertex-disjoint cliques $C_1, \ldots, C_i, \ldots, C_{k-1}$ of the graph G, and remove the following variables

$$y_v^h, \quad i = 1, \ldots, k-1, \quad v \in C_i, \quad h = i+1, \ldots, k. \tag{34}$$

Indeed, two vertices u, v of a clique cannot be in two different subsets V_i and V_j. Then for all solutions we can reorder the sets $V_1, ..., V_k$ to ensure that each vertex of a clique C_i must be in one set V_j $j \leq i$ or in the vertex cut. Thus we can remove the variables (34) to reduce the symmetry.

The comparison, whose results are reported in Table 4, is performed on the whole set of instances including Vertex Coloring, DIMACS and Intersection graphs described in Section 6.1. The table has the same structure of the previous one, and reports the number of instances solved to optimality, the average computing time in seconds and the average number of explored nodes (for solved instances). The table clearly shows that HYB is the best performing method on average, being able to solve 202 out of the 304 tested instances. $COMP$ and BP can both solve 168 instances. On the subset of instances that are solved by all the three methods, the computing time of BP is approximately 2/3 the computing time of $COMP$, while the computing time of HYB is approximately halved with respect to the computing time of $COMP$. An important information is given by the average number of nodes explored in the branch-and-cut tree, in particular $COMP$ explores ≈33,000, BP ≈22 and HYB ≈64 nodes, respectively. By analyzing these figures, it clearly emerges that $COMP$ explores many more nodes than the other two methods. This fact is due to the poor quality of the LP relaxation bound provided by the Compact Formulation. BP and HYB explore fewer nodes, and the reason is the quality of the LP bounds provided by these formulations. BP is the algorithm which explores the smallest number of nodes on average. By analyzing the results for each value of k separately, the table shows that $COMP$ provides the best computational performances for $k = 5$ but then, as far as $k \geq 10$, HYB always guarantees the best computational performances on this set of instances, being able to solve 49 out of 80 instances, 46 out of 65 and 38 out of 52, for $k = 10$, $k = 15$ and $k = 20$, respectively. Also the BP algorithm shows a better performance than $COMP$ as soon as $k \geq 10$.

A graphical representation of the relative performance of the three compared approaches is given by the performance profiles of Figures 4 and 5, for unweighed and weighted (see Section 6.3.1) instances respectively. Following the guidelines suggested by [15], the performance profiles are defined as follows. Let m be any solution method and i denote an instance of the problem. In addition let $t_{i,m}$ be the time required by method m to solve instance i. We define the *performance ratio* for pair (i, m) as

$$r_{i,m} = \frac{t_{i,m}}{\min_{m \in M}\{t_{i,m}\}}$$

where M is the set of the considered methods. Then, for each method $m \in M$, we define:

$$\rho_m(\tau) = \frac{|\{i \in I : r_{i,m} \leq \tau\}|}{|I|}$$

Table 4 Performance comparison between the Hybrid Formulation and the state-of-the-art methods on the complete instance set (Vertex Coloring, DIMACS and Intersection graphs).

k		*COMP*	*BP*	*HYB*
	Opt. (out of 107)	92	60	71
5	Avg Time	31.84	59.93	84.78
	Avg Nodes	10768	30	106
	Opt. (out of 80)	37	43	51
10	Avg Time	105.64	52.19	1.39
	Avg Nodes	67123	7	26
	Opt. (out of 65)	29	36	46
15	Avg Time	219.33	23.38	2.81
	Avg Nodes	41750	19	25
	Opt. (out of 52)	19	29	38
20	Avg Time	196.06	169.52	0.39
	Avg Nodes	58673	16	6
	Total Opt. (out of 304)	177	168	206
	Total Avg Time	98.66	61.78	43.66
	Total Avg Nodes	33040	22	64

Table 5 Performance comparison between the Hybrid Formulation and the state-of-the-art methods on the complete instance set with weights (Vertex Coloring, DIMACS and Intersection graphs).

k		*COMP*	*BP*	*HYB*
	Opt. (out of 107)	92	60	71
5	Avg Time	35.99	67.55	210.67
	Avg Nodes	11350	77	217
	Opt. (out of 80)	37	43	51
10	Avg Time	69.61	174.96	2.30
	Avg Nodes	22872	21	26
	Opt. (out of 65)	29	37	47
15	Avg Time	343.26	36.61	21.76
	Avg Nodes	109726	180	86
	Opt. (out of 52)	19	30	39
20	Avg Time	559.17	300.40	1.15
	Avg Nodes	180529	31	15
	Total Opt. (out of 304)	177	170	208
	Total Avg Time	151.21	112.23	106.13
	Total Avg Nodes	48594	77	127

where I is the set of the instances. Intuitively, $r_{i,m}$ denotes the worsening (with respect to computing time) incurred when solving instance i using method m instead of the best possible one, whereas $\rho_m(\tau)$ gives the percentage of instances for which the computing time of method m was not larger than τ times the time of the best performing method. For each value of τ in the horizontal axis, the vertical axis reports the percentage of the instances for which the corresponding algorithm spends no more than τ times the computing time of the fastest algorithm. The curves originates from a point denoting the percentage of instances for which the corresponding algorithm is the fastest, and at the right end of the chart, they show the percentage of instances solved within time limit. The best performance algorithm is graphically represented by the curve in the upper part of the Figures. The horizontal axis is represented in logarithmic scale. The figures clearly show that the relative performance of the 3 algorithms depends on the value k considered.

For $k = 5$, Figure 4 shows that HYB and $COMP$ are the fastest method in ≈40% of the instances. HYB can solve ≈65% of the instances, while $COMP$ can solve ≈85%, and the corresponding curve dominates those of HYB in most of the chart. BP is the fastest method in ≈5% and can solve ≈55% of the instances. For $k = 5$, the best option appears to solve the problem by means of the $COMP$ formulation. As soon as the value of k increases, the performance of the three solution methods changes. For $k = 10$, the figure shows that HYB is the fastest method in ≈50% and it can solve ≈60% of the instances. It dominates the other two methods on the whole chart; BP is the fastest method in ≈20% and can solve ≈50% of the instances, while $COMP$ is the fastest method in ≈10% and can solve ≈40% of the instances. The primacy of HYB increases with increasing k: for $k = 15$, the figure shows that HYB is the fastest method in ≈60% and it is able to solve ≈70% of the instances. It dominates the other two methods on the whole chart; BP is the fastest method in ≈15% and can solve ≈60% of the instances, while $COMP$ is the fastest method in ≈5% and can solve ≈40% of the instances. For $k = 20$, the figure shows that shows that HYB is the fastest method in ≈70% and it is able to solve ≈75% of the instances. It dominates the other two methods on the whole chart; BP is the fastest method in ≈15% and can solve ≈55% of the instances, while $COMP$ is the fastest method in less than 5% and can solve ≈30% of the instances.

Summarizing for $k = 5$ the best method on average is $COMP$ which is able to solve the largest percentage of the instances, even if HYB remains the fastest in almost half of them. For all the other values of k, i.e., $k \in \{10, 15, 20\}$, the best computational performance is provided by HYB which is always able to solve the largest percentage of the instances and it is always the fastest methods in more that 50% of them. As far as the comparison between $COMP$ and BP is concerned, the results we obtain are in line with the results presented in [12], i.e., BP is dominated by $COMP$ when $k = 5$, while an opposite behavior is experienced for larger values of k.

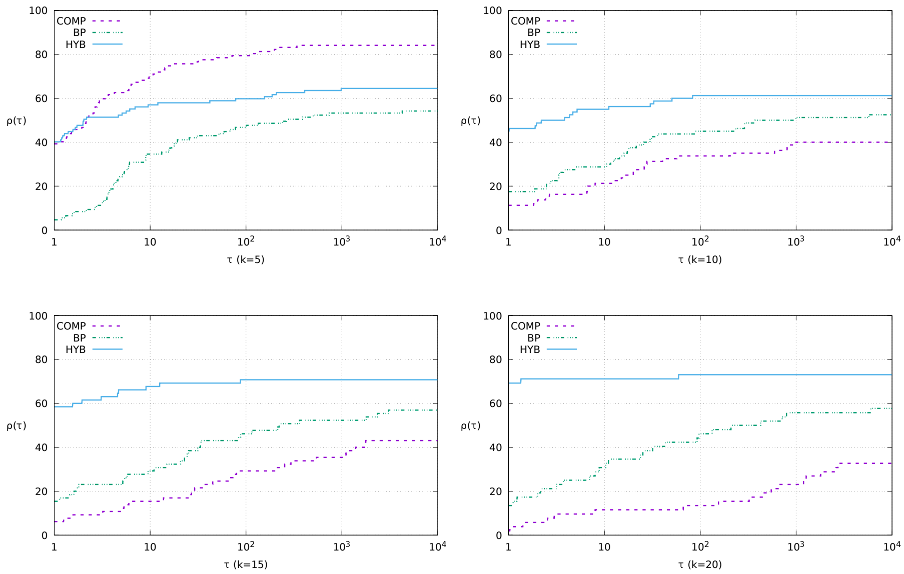


Fig. 4 Performance profile of exact methods for the k-vertex cut problem.

6.3.1 Weighted case

In the previous sections we focused the computational analysis on the case where vertices have the same weight (without loss of generality, equal to 1), but all the described formulations, as well as the BP algorithm can also tackle the *weighted case*, that is, the case in which each vertex $v \in V$ has an integer weight w_v. According to our computational experiments the best among the formulations proposed in this paper for the weighted case is still HYB. Hence, in this section we report on the performance of HYB, $COMP$ and BP on the complete set of instances including Vertex Coloring, DIMACS and Intersection graphs, where a random integer weight with uniform distribution in $\{1, \ldots, 10\}$ is generated for each vertex $v \in V$. As reported in Table 5, the results in terms of number of solved instances are very similar to those obtained in the unweighted case, confirming the superior performance of HYB, with 208 out of 304 instances solved to optimality, followed by $COMP$ and BP with 177 and 170 solved instances, respectively. The distribution of optimal solution among the separate values of k shows that $COMP$ provides the best computational performances for $k = 5$ but then, as far as $k \geq 10$, HYB is always the best method, and BP always performs better than $COMP$. Although the (almost identical) number of solved instances by each algorithm, the weighted instances appear more challenging for what concerns computing times and number of Branch-and-Bound nodes: $COMP$ requires approximately 50% more nodes and seconds while both BP and HYB approximately double the number of Branch-and-Bound nodes and the computing time.

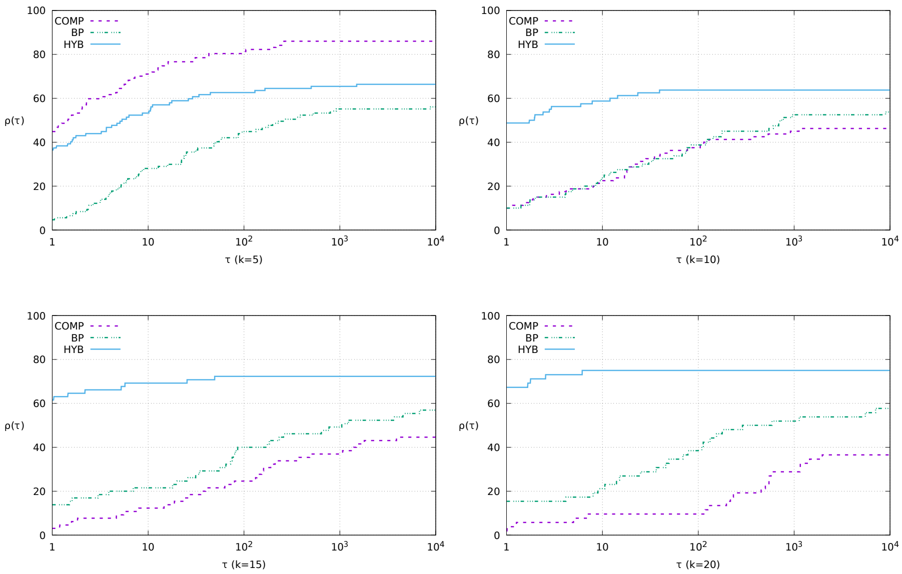


Fig. 5 Performance profile of exact methods for the k-vertex cut problem with weights.

Performance profiles for the weighted case are reported in Figure 5, and are very close to the profiles obtained in the unweighted case. For $k = 5$, the curve corresponding to $COMP$ dominates that of HYB, and the best option appears to solve the problem by means of the $COMP$ formulation. The performance of BP is the worst. As soon as $k = 10$, the performance of HYB becomes the best. The primacy of HYB increases with increasing k and it largely dominates the other solution methods. Further details on the experiments for the weighted case are reported in the Appendix.

7. Conclusions

We have considered a prototype problem in the family of Critical Node Detection Problems, that is, the problem of removing a (minimum weight) set of vertices from a graph so as to disconnect the resulting graph in several components. The so-called k-vertex cut problem has relevant applications not only in network analysis, but also in matrix decomposition for solving systems of equations by parallel computing.

We have described two new integer linear programming formulations, both involving an exponential number of constraints for which we provided separation procedures and implemented branch-and-cut algorithms.

Both formulations use a *natural* set of variables to identify the removed vertices (the k-vertex cut). The first considers additional variables to denote which vertex is *representative* of each component of the disconnected graph, while in the second formulation, the model is derived from the perspective of a two-phase Stackelberg game in which a leader deletes the vertices in the first phase, and in the second phase a follower builds connected components in the remaining graph.

Extensive computational experiments on a set of benchmark instances allowed us to identify the strengths and weaknesses of the two formulations, that in the end we combined in a hybrid one. The experiments also showed that the hybrid formulation significantly outperforms a state-of-the-art branch-and-price method recently proposed for the problem.

The presented idea of looking into the k-vertex cut problem from the perspective of a two-players Stackelberg game can be used in a more general setting for solving Critical Node/Edge Detection Problems. Derivation of new formulations in the natural space of decision variables for this large family of problems will be subject of future research.

Acknowledgements The authors are indebted to two anonymous referees and one technical editor for their constructive and useful comments. Enrico Malaguti is supported by the Air Force Office of Scientific Research under Award Number FA9550-17-1-0025.
This is a pre-print of an article published in Mathematical Programming Computation.
The final authenticated version is available online at: https://doi.org/10.1007/s12532-019-00167-1.

References

1. Color02/03/04: Graph coloring and its generalizations. `http://mat.gsia.cmu.edu/COLOR03/`. Accessed: 2018-07-20
2. Dimacs implementation challenges. `http://dimacs.rutgers.edu/archive/Challenges/`. Accessed: 2018-07-20
3. Balas, E., de Souza, C.C.: The vertex separator problem: a polyhedral investigation. Mathematical Programming **103(3)**, 583–608 (2005)
4. Barahona, F.: On the k-cut problem. Operations Research Letters **26**(3), 99 – 105 (2000)
5. Bastubbe, M., Lübbecke, M.: A branch-and-price algorithm for capacitated hypergraph vertex separation. Technical Report, Optimization Online (2017)
6. Ben-Ameur, W., Biha, M.D.: On the minimum cut separator problem. Networks **59(1)**, 30–36 (2012)
7. Ben-Ameur, W., Mohamed-Sidi, M.A., Neto, J.: The k-separator problem. In Computing and Combinatorics pp. 337–348 (2013)
8. Berger, A., Grigoriev, A., v. d. Zwaan, R.: Complexity and approximability of the k-way vertex cut. Networks **63(2)**, 170–178 (2014)
9. Borndörfer, R., Ferreira, C., Martin, A.: Decomposing matrices into blocks. SIAM Journal on Optimization **9**(1), 236–269 (1998)
10. Bui, T.N., Jones, C.: Finding good approximate vertex and edge partitions is NP-hard. Information Processing Letters **42**(3), 153 – 159 (1992)
11. Chopra, S., Rao, M.R.: On the multiway cut polyhedron. Networks **21(1)**, 51–89 (1991)
12. Cornaz, D., Furini, F., Lacroix, M., Malaguti, E., Mahjoub, A.R., Martin, S.: The vertex k-cut problem. Discrete Optimization, in press (2018)
13. Cornaz, D., Magnouche, Y., Mahjoub, A.R., Martin, S.: The multi-terminal vertex separator problem: Polyhedral analysis and branch-and-cut. Conference on Computers & Industrial Engineering (CIE45) pp. 857–864 (2015)
14. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakiss, M.: The complexity of multiterminal cuts. SIAM Journal on Computing **23(4)**, 864–894 (1994)

15. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**(2), 201–213 (2002)
16. Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: Interdiction games under monotonicity. INFORMS Journal on Computing (2018). To appear
17. Fukuyama, J.: NP-completeness of the planar separator problems. Journal of Graph Algorithms and Applications **10**(2), 317–328 (2006)
18. Furini, F., Ljubić, I., San Segundo, P., Martin, S.: The maximum clique interdiction problem. European Journal of Operational Research **to appear** (2019). DOI https://doi.org/10.1016/j.ejor.2019.02.028
19. Garg, N., Vazirani, V.V., Yannakakis, M.: Multiway cuts in directed and node weighted graphs. In Automata, Languages and Programming pp. 487–498 (1994)
20. Garg, N., Vazirani, V.V., Yannakakis, M.: Multiway cuts in node weighted graphs. Journal of Algorithms **50(1)**, 49–61 (2004)
21. Goldschmidt, O., Hochbaum, D.S.: A polynomial algorithm for the k-cut problem for fixed k. Mathematics of Operations Research **19**(1), 24–37 (1994)
22. Gupta, A., Lee, E., Li, J.: An FPT algorithm beating 2-approximation for k-cut. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '18, pp. 2821–2837. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2018)
23. Israeli, E., Wood, R.K.: Shortest-path network interdiction. Networks **40**(2), 97–111 (2002)
24. Karger, D.R., Motwani, R.: An nc algorithm for minimum cuts. SIAM Journal on Computing **26**(1), 255–272 (1997)
25. Kempe, D., Kleinberg, J., Tardos, É.: Influential nodes in a diffusion model for social networks. In: L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (eds.) Automata, Languages and Programming, pp. 1127–1138. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
26. Lalou, M., Tahraoui, M.A., Kheddouci, H.: The critical node detection problem in networks: A survey. Computer Science Review **28**, 92 – 117 (2018)
27. Magnouche, J.: The multi-terminal vertex separator problem : Complexity, polyhedra and algorithms (2017)
28. Marx, D.: Parameterized graph separation problems. Theoretical Computer Science **351(3)**, 394–406 (2006)
29. Saran, H., Vazirani, V.: Finding k cuts within twice the optimal. SIAM Journal on Computing **24**(1), 101–108 (1995)
30. de Souza, C., Balas, E.: The vertex separator problem: algorithms and computations. Mathematical Programming **103(3)**, 609–631 (2005)

Table 6 Number of vertices removed by preprocessing

	n	m	$k = 5$	$k = 10$	$k = 15$	$k = 20$
2-FullIns_3	52	201				6
2-Insertions_3	37	72			1	
chesapeake	39	170	1	2	5	
david	87	406	1	1	1	1
DSJC125.5	125	3891		113		
football	115	613				3
huck	74	301			1	1
karate	34	78		1	2	7
miles1500	128	5198	108			
mulsol.i.2	188	3885				3
mulsol.i.3	184	3916			3	3
mulsol.i.4	185	3946			3	3
mulsol.i.5	186	3973			3	3
myciel3	11	20	6			
myciel4	23	71		7		
myciel5	47	236			3	8
r125.1c	125	7501	97			
r125.5	125	3838	8			
bcspwr02	49	177			11	
can61	61	866	11			
dwt___59	59	256			31	
dwt87	87	726			13	
impcol_b	59	329				34
L100.cavity01	100	1844				2
L100.fidap025	100	2031	66			
L100.fidapm02	100	3090	57			
L100.rbs480a	100	2550	64			
L100.wm1	100	2956		10	37	
L100.wm3	100	2934			50	
L120.cavity01	120	2972				2
L120.fidap022	120	4307	80			
L120.fidap025	120	2787	80			
L120.fidapm02	120	4626	50			
L120.rbs480a	120	3273	34			
L120.wm2	120	3387				23
L125.can__161	125	1257			32	
L125.can__187	125	1022				73
L125.dwt__162	125	943			5	
L125.dwt__193	125	2982	4			
L125.fs_183_1	125	3392	1			
L80.cavity01	80	1201				8
L80.fidap025	80	1201	52			
L80.steam2	80	1272	4			
L80.wm1	80	1786	2	15	47	
L80.wm2	80	1848		29		
lund_a	147	2837		21		
pores_1	30	179	4			
west0067	67	411		3		
will57	57	304		16		

8. Appendix

	COMP				*BP*				*HYB*			
	$k=5$	$k=10$	$k=15$	$k=20$	$k=5$	$k=10$	$k=15$	$k=20$	$k=5$	$k=10$	$k=15$	$k=20$
1-FullIns_3	0.38	2.10			0.66	2.25			0.15	0.14		
1-FullIns_4	48.53	tl	tl	tl	tl	tl	tl	tl	13.51	16.40	21.42	5.78
1-Insertions_4	16.74	tl	2717.20	tl	174.89	48.54	56.18	55.01	2.96	2.80	2.46	4.99
2-FullIns_3	5.70	1329.02	217.61	560.95	51.79	34.25	23.20	20.38	8.95	1.34	0.74	0.47
2-Insertions_3	0.93	51.14	334.47		1.90	21.97	20.11		0.45	0.25	0.24	
2-Insertions_4	247.02	tl	tl	tl	tl	tl	tl	tl	198.26	163.67	674.54	1159.77
3-FullIns_3	25.32	tl	tl	tl	tl	tl	70.38	579.20	43.78	39.64	2.89	23.00
3-Insertions_3	8.91	tl	tl	tl	13.35	24.88	24.64	24.35	0.83	1.73	2.59	2.76
4-FullIns_3	60.07	tl	tl	tl	tl	tl	tl	tl	2520.50	829.77	159.96	113.36
4-Insertions_3	36.61	tl	tl	tl	29.07	34.23	34.50	34.33	2.20	13.72	19.13	15.83
5-FullIns_3	311.26	tl	tl	tl	tl	tl	tl	tl	2968.00	2161.91	247.28	72.32
adjnoun	1.35	82.85	854.49	tl	2200.58	866.14	tl	1777.17	0.01	2.98	10.71	18.32
anna	0.31	0.26	0.73	1.55	42.60	57.32	23.51	60.09	0.00	0.00	0.01	0.01
celegansneural	3.96	16.08	88.06	tl	tl	tl	tl	tl	0.02	0.02	0.05	11.17
chesapeake	0.39	6.74	6.59		0.62	5.70	11.53		0.17	0.54	0.10	
david	·	·	12.28	753.15	·	·	78.22	27.56	·	·	0.89	1.16
dolphins	0.15	8.22	165.39	tl	5.03	15.57	26.81	26.44	0.00	0.52	0.79	2.56
DSJC125.1	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
DSJC125.5	tl	0.27			tl	0.01			tl	0.00		
football	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
games120	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
huck	0.06	5.96	7.60	5.31	3.89	9.95	53.86	10.64	0.00	0.01	0.03	0.08
jazz	32.79	1817.51	tl	tl	66.46	tl	82.90	tl	0.66	269.73	tl	tl
jean	0.06	0.20	0.36	4.56	2.64	2.89	3.62	7.15	0.00	0.00	0.00	0.01
karate	0.03	0.08	0.06	0.03	0.09	0.18	0.11	0.08	0.00	0.00	0.00	0.01
lesmis	0.14	0.57	0.45	6.47	2.41	3.72	2.14	8.53	0.00	0.00	0.00	0.02
miles1000	1307.71				tl				tl			
miles1500	0.12				0.06				0.01			
miles250	·	·	tl	tl	·	·	35.09	36.08	·	·	1.69	9.65
miles500	116.00	tl	tl		31.14	tl	tl		11.88	tl	tl	
miles750	359.62	tl			tl	tl			tl	tl		
mug100_1	269.56	tl	tl	tl	22.89	27.89	28.30	29.29	1.22	8.34	17.58	23.91
mug100_25	397.28	tl	tl	tl	22.56	23.90	29.64	35.81	1.17	8.65	5.71	17.77
mug88_1	62.20	tl	tl	tl	22.23	27.61	33.55	36.35	0.87	2.37	5.83	11.45
mug88_25	16.36	tl	tl	tl	18.83	27.29	28.67	30.66	0.14	2.15	5.28	0.96
mulsol.i.2	·	·	·	382.63	·	·	·	tl	·	·	·	0.14
mulsol.i.3	·	·	230.54	594.89	·	·	tl	tl	·	·	0.13	0.33
mulsol.i.4	·	·	197.86	186.83	·	·	tl	tl	·	·	37.05	0.34
mulsol.i.5	·	·	176.49	559.76	·	·	tl	tl	·	·	0.16	0.44
myciel4	0.29	0.21			1.58	0.05			0.10	0.01		
myciel5	3.80	732.31	222.53	625.69	18.71	26.63	90.83	22.24	0.60	0.86	0.41	0.24
myciel6	43.87	tl	tl	tl	tl	tl	tl	tl	6.26	18.63	16.34	19.99
myciel7	940.63	tl	tl	tl	tl	tl	tl	tl	580.86	1733.41	1416.07	865.15
polbooks	78.21	tl	tl	tl	67.08	388.26	50.71	38.09	411.21	394.90	237.70	2259.55
queen10_10	tl	1.06			tl	5.76			tl	0.16		
queen11_11	tl	tl			tl	tl			tl	tl		
queen12_12	tl	tl			tl	tl			tl	tl		
queen13_13	tl	tl			tl	tl			tl	tl		
queen14_14	tl	tl			tl	tl			tl	tl		
queen5_5	0.01				0.09				0.02			
queen6_6	1.38				4.40				2.29			
queen7_7	25.63				1338.27				tl			
queen8_12	tl				tl				tl			
queen8_8	411.59				tl				tl			
queen9_9	3078.70				tl				tl			
r125.1	·	·	203.76	tl	·	·	30.70	35.53	·	·	0.00	0.17
r125.1c	0.34				0.79				0.13			
r125.5	364.18				tl				tl			

Table 7 Computational times (Coloring and DIMACS)

	COMP				*BP*				*HYB*			
	$k=5$	$k=10$	$k=15$	$k=20$	$k=5$	$k=10$	$k=15$	$k=20$	$k=5$	$k=10$	$k=15$	$k=20$
arc130	0.10				tl				40.89			
ash219	57.19	tl	tl	tl	26.01	43.36	435.94	263.85	132.71	tl	tl	tl
ash331	190.01	tl	tl	tl	24.67	534.41	tl	tl	1923.06	tl	tl	tl
ash85	219.87	tl			tl	tl			tl	tl		
bcspwr01	1.04	9.50			1.89	11.62			0.37	3.57		
bcspwr02	4.55	88.12	0.58		9.39	25.13	2.77		1.54	13.42	0.11	
bcspwr03	175.02	tl	tl	tl	230.01	44.96	474.05	230.46	tl	tl	tl	tl
bfw62a	2.33				180.84				tl			
can__144	tl	tl			tl	tl			tl	tl		
can61	0.97				4.68				2.11			
can62	11.61	tl	tl		17.87	25.25	24.89		13.49	2099.11	2179.20	
can73	160.08	tl			tl	tl			tl	tl		
can96	tl	tl			tl	tl			tl	tl		
curtis54	9.67				39.62				1812.30			
dwt___59	17.02	966.14	0.54		20.11	23.88	0.67		20.88	tl	0.02	
dwt66	566.29	tl			tl	tl			tl	tl		
dwt72	59.40	tl	tl	tl	23.85	28.15	163.40	436.94	6.41	tl	tl	tl
dwt87	44.31	tl	485.65		32.90	1994.16	28.33		20.34	tl	16.66	
gre__115	998.31	tl	tl	tl	2158.27	3139.10	tl	tl	tl	tl	tl	tl
ibm32	0.67				1.85				8.07			
impcol_b	0.35	4.17	9.60	0.08	0.91	5.17	13.91	0.07	0.17	2.07	29.60	0.01
L100.cavity01	0.86	5.22	5.48	5.78	tl	tl	tl	tl	0.40	170.85	3.49	7.77
L100.fidapm02	0.60				15.62				0.23			
L100.rbs480a	0.03				22.14				0.14			
L100.steam2	290.48				tl				tl			
L100.wm1	3.74	6.80	0.51		tl	tl	0.58		21.04	3.71	0.99	
L100.wm2	0.10	7.49			0.41	tl			0.14	225.32		
L100.wm3	0.11	2.88	0.44		0.39	tl	0.35		0.14	11.09	3.17	
L120.cavity01	1.03	9.89	6.56	7.43	tl	tl	tl	tl	0.68	501.24	10.24	7.18
L120.fidap022	0.05				0.19				7.80			
L120.fidapm02	0.31				tl				64.38			
L120.rbs480a	19.99				335.74				tl			
L120.wm2	0.16	0.50	0.69	12.04	1.07	1.88	tl	tl	0.06	0.95	3.16	4.71
L125.ash608	843.17	tl	tl	tl	26.46	tl	tl	tl	tl	tl	tl	tl
L125.bcsstk05	159.35				tl				tl			
L125.can__161	tl	tl	tl		tl	tl	tl		tl	tl	tl	
L125.can__187	tl	tl	tl	tl	tl	tl	tl	38.72	tl	tl	tl	34.61
L125.dwt__162	tl	tl	tl		tl	tl	tl		tl	tl	tl	
L125.dwt__193	1271.68				tl				tl			
L125.fs_183_1	2.23				tl				2195.21			
L125.gre__185	3244.87	tl	tl		tl	tl	tl		tl	tl	tl	
L125.lop163	tl	tl	tl		tl	tl	tl		tl	tl	tl	
L125.west0167	38.97	tl	tl	tl	23.80	30.37	34.33	39.01	2.74	157.35	431.59	tl
L125.will199	16.82	tl	tl	tl	21.25	35.26	47.00	41.12	4.60	tl	tl	tl
L80.cavity01	0.53	0.22	4.81	2.75	tl	tl	tl	1538.51	0.39	0.09	1.50	1.94
L80.steam2	29.73				tl				tl			
L80.wm1	0.82	3.27	0.26		tl	tl	0.15		5.71	6.38	0.01	
L80.wm2	0.05	0.66			0.27	14.19			0.06	2.86		
L80.wm3	0.04	0.16			0.23	37.53			0.08	0.75		
lund_a	tl	tl			tl	tl			tl	tl		
pores_1	0.24				0.65				0.18			
rw136	157.57	tl	tl	tl	37.21	tl	tl	tl	75.83	tl	tl	tl
steam3	186.29				tl				tl			
west0067	144.60	tl			tl	688.55			tl	tl		
west0132	24.78	tl	tl	tl	40.00	34.71	39.58	169.35	12.83	385.95	tl	tl
will57	1.12	0.05			2.78	0.17			0.82	0.11		

Table 8 Computational times (Intersection graphs)

	COMP				BP				HYB			
	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 5$	$k = 10$	$k = 15$	$k = 20$
1-FullIns_3	0.41	1.28			1.33	2.95			0.18	0.16		
1-FullIns_4	11.21	438.59	tl	tl	tl	tl	tl	tl	5.62	15.44	10.81	8.58
1-Insertions_4	4.41	190.63	tl	tl	tl	913.62	2283.63	796.23	23.58	10.79	12.99	7.31
2-FullIns_3	1.64	147.15	52.52	72.91	343.85	72.01	26.37	21.16	1.50	1.29	1.32	0.58
2-Insertions_3	0.41	22.00	16.84		5.24	24.21	20.21		0.06	0.17	0.11	
2-Insertions_4	331.64	tl	tl	tl	tl	tl	tl	tl	tl	2679.20	2090.26	566.87
3-FullIns_3	19.10	263.12	432.30	2564.03	tl	105.84	tl	281.41	31.64	7.53	2.70	5.67
3-Insertions_3	1.86	241.05	tl	tl	16.18	33.77	31.82	23.50	0.32	0.45	0.44	0.34
4-FullIns_3	69.45	tl	tl	tl	tl	tl	tl	tl	50.74	1064.56	41.78	99.47
4-Insertions_3	4.20	842.10	tl	tl	24.89	70.35	67.74	30.89	0.26	0.92	2.16	2.05
5-FullIns_3	29.44	tl	tl	tl	tl	tl	tl	tl	108.07	420.77	166.03	126.83
adjnoun	5.22	77.60	340.18	tl	tl	570.49	145.39	389.75	0.05	0.74	1.79	15.56
anna	0.43	0.77	4.84	14.50	88.39	89.78	67.74	70.59	0.01	0.01	0.01	0.01
celegansneural	9.87	22.50	1559.04	tl	tl	tl	tl	tl	0.05	0.02	0.40	599.76
chesapeake	0.31	2.88	3.52		0.42	2.77	8.24		0.05	0.31	0.24	
david	·	·	5.46	2931.76	·	·	25.65	26.84	·	·	0.04	2.53
dolphins	0.31	6.18	247.26	1042.79	5.28	36.77	48.72	26.80	0.01	0.30	1.87	1.91
DSJC125.1	2477.17	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
DSJC125.5	tl	0.25			tl	0.02			tl	0.00		
football	1062.01	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
games120	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
huck	0.41	7.17	6.34	25.56	9.17	19.09	23.11	14.61	0.01	0.02	0.03	0.11
jazz	67.51	2980.98	tl	tl	32.00	tl	73.38	tl	30.33	334.29	tl	tl
jean	0.11	0.18	14.85	5.50	7.96	7.11	12.29	5.88	0.01	0.00	0.01	0.01
karate	0.12	0.17	0.08	0.05	0.16	0.26	0.18	0.09	0.01	0.01	0.01	0.01
lesmis	0.15	0.40	3.75	3.93	1.55	6.90	10.46	5.78	0.00	0.01	0.01	0.02
miles1000	1197.96				tl				tl			
miles1500	0.14				0.05				0.01			
miles250	·	·	tl	tl	·	·	227.79	240.53	·	·	0.05	1.55
miles500	204.84	tl	tl		28.15	tl	tl		313.19	tl	tl	
miles750	458.78	tl			tl	tl			tl	tl		
mug100_1	15.42	tl	tl	tl	10.84	109.05	30.32	29.65	0.06	0.18	0.13	0.38
mug100_25	18.26	tl	tl	tl	24.71	31.93	51.74	34.30	0.07	0.18	0.20	0.75
mug88_1	94.87	tl	tl	tl	24.21	55.01	72.20	34.92	0.44	0.81	0.96	61.92
mug88_25	17.73	tl	tl	tl	23.95	43.42	69.66	31.22	0.18	0.36	0.81	7.50
mulsol.i.2	·	·	·	393.14	·	·	·	tl	·	·	·	0.20
mulsol.i.3	·	·	305.87	758.72	·	·	tl	tl	·	·	0.17	0.66
mulsol.i.4	·	·	153.20	325.31	·	·	tl	tl	·	·	0.11	0.65
mulsol.i.5	·	·	299.17	338.89	·	·	tl	tl	·	·	0.28	0.59
myciel4	0.27	0.13			6.79	0.12			0.14	0.00		
myciel5	1.76	40.91	66.09	45.47	153.19	66.42	53.73	23.64	1.80	2.18	0.83	0.21
myciel6	49.44	tl	tl	tl	tl	tl	tl	tl	31.89	63.06	47.48	51.60
myciel7	477.80	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	1556.57
polbooks	23.36	tl	tl	tl	35.81	445.88	130.27	40.37	20.19	256.03	80.52	103.30
queen10_10	tl	0.89			tl	84.83			tl	0.49		
queen11_11	tl	tl			tl	tl			tl	tl		
queen12_12	tl	tl			tl	tl			tl	tl		
queen13_13	tl	tl			tl	tl			tl	tl		
queen14_14	tl	tl			tl	tl			tl	tl		
queen5_5	0.09				0.09				0.01			
queen6_6	1.69				6.51				1.87			
queen7_7	30.27				761.89				tl			
queen8_12	tl				tl				tl			
queen8_8	181.76				tl				tl			
queen9_9	1051.71				tl				tl			
r125.1	·	·	tl	tl	·	·	37.11	55.21	·	·	0.01	0.01
r125.1c	0.46				0.95				0.12			
r125.5	574.91				tl				tl			

Table 9 Computational times for instances with weights (Coloring and DIMACS)

	COMP				*BP*				*HYB*			
	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 5$	$k = 10$	$k = 15$	$k = 20$
arc130	0.13				tl				65.34			
ash219	112.44	tl	tl	tl	185.48	185.63	604.08	1205.06	251.06	2673.94	tl	tl
ash331	320.54	tl	tl	tl	25.31	tl	tl	tl	664.89	tl	tl	tl
ash85	762.38	tl			tl	tl			tl	tl		
bcspwr01	0.68	5.98			7.36	16.43			0.33	1.68		
bcspwr02	2.07	87.13	2.47		33.68	36.41	3.96		1.42	20.78	0.07	
bcspwr03	407.38	tl	tl	tl	85.71	666.73	671.30	1796.12	tl	tl	tl	tl
bfw62a	2.20				437.56				3293.22			
can__144	tl	tl			tl	tl			tl	tl		
can61	0.98				7.86				8.54			
can62	9.83	tl	3279.77		70.55	56.08	51.94		2.14	156.57	272.13	
can73	216.41	tl			tl	tl			tl	tl		
can96	tl	tl			tl	tl			tl	tl		
curtis54	7.69				18.52				128.46			
dwt___59	18.52	2904.33	0.80		67.36	58.75	2.32		195.93	tl	0.03	
dwt66	30.47	tl			159.22	tl			1032.61	tl		
dwt72	40.69	tl	tl	tl	62.20	39.54	40.08	79.69	1.32	77.44	1982.92	tl
dwt87	49.22	tl	2778.26		1165.64	tl	223.66		2193.55	tl	147.73	
gre__115	115.53	tl	tl	tl	23.57	215.78	tl	tl	240.00	tl	tl	tl
ibm32	0.68				1.81				3.46			
impcol_b	0.25	3.08	40.15	0.07	3.45	4.82	7.22	0.08	0.11	1.18	7.47	0.01
L100.cavity01	0.95	6.56	2.05	5.70	tl	tl	tl	tl	0.55	51.38	1.71	5.61
L100.fidapm02	0.04				15.56				0.13			
L100.rbs480a	0.03				8.15				0.53			
L100.steam2	939.05				tl				tl			
L100.wm1	3.76	7.95	1.10		tl	tl	0.60		23.76	6.87	1.32	
L100.wm2	0.13	8.98			0.51	tl			0.21	353.92		
L100.wm3	0.13	4.97	0.42		0.41	2710.89	0.35		0.23	9.81	0.09	
L120.cavity01	0.65	8.16	6.66	6.02	tl	tl	tl	tl	0.41	191.10	9.72	10.02
L120.fidap022	0.05				0.28				8.27			
L120.fidapm02	0.36				tl				10.47			
L120.rbs480a	32.84				811.22				tl			
L120.wm2	0.13	0.57	1.31	580.75	0.89	1.97	tl	tl	0.04	0.30	7.51	4.92
L125.ash608	577.32	tl	tl	tl	102.70	tl	tl	tl	609.68	tl	tl	tl
L125.bcsstk05	392.61				tl				tl			
L125.can__161	tl	tl	tl		tl	tl	tl		tl	tl	tl	
L125.can__187	tl	tl	tl	tl	tl	tl	tl	23.10	tl	tl	tl	0.13
L125.dwt__162	tl	tl	tl		tl	tl	tl		tl	tl	tl	
L125.dwt__193	516.22				tl				tl			
L125.fs_183_1	2.40				139.12				312.08			
L125.gre__185	tl	tl	tl		tl	tl	tl		tl	tl	tl	
L125.lop163	tl	tl	tl		tl	tl	tl		tl	tl	tl	
L125.west0167	21.70	tl	tl	tl	23.82	42.86	45.71	39.75	3.89	10.32	15.21	244.72
L125.will199	6.46	tl	tl	tl	15.36	76.29	86.93	38.87	9.34	224.00	2214.50	tl
L80.cavity01	0.40	0.33	2.75	3.50	tl	7.43	tl	3127.89	0.36	0.22	1.73	2.77
L80.steam2	32.78				tl				tl			
L80.wm1	2.07	3.64	0.48		tl	tl	0.14		9.79	21.64	0.02	
L80.wm2	0.06	1.56			0.28	7.60			0.25	3.75		
L80.wm3	0.06	0.55			0.25	5.74			0.22	0.95		
lund_a	tl	tl			tl	tl			tl	tl		
pores_1	0.30				0.55				0.24			
rw136	230.13	tl	tl	tl	324.92	tl	tl	tl	2459.85	tl	tl	tl
steam3	78.03				249.55				tl			
west0067	107.00	tl			tl	49.39			tl	tl		
west0132	7.29	tl	tl	tl	18.66	62.73	56.18	39.44	3.19	44.36	tl	tl
will57	0.79	0.02			10.09	0.17			1.23	0.24		

Table 10 Computational times for instances with weights (Intersection graphs)

	Optimal Values					Optimal Values			
	$k = 5$	$k = 10$	$k = 15$	$k = 20$		$k = 5$	$k = 10$	$k = 15$	$k = 20$
1-FullIns_3	35	53			miles500	42	-	-	
1-FullIns_4	35	66	90	122	miles750	120	-		
1-Insertions_4	40	68	100	125	mug100_1	10	27	46	69
2-FullIns_3	42	71	92	125	mug100_25	11	30	52	77
2-Insertions_3	18	50	73		mug88_1	20	43	68	99
2-Insertions_4	42	69	99	124	mug88_25	14	38	63	93
3-FullIns_3	33	53	76	106	mulsol.i.2	·	·	·	96
3-Insertions_3	22	47	72	95	mulsol.i.3	·	·	96	98
4-FullIns_3	40	81	98	127	mulsol.i.4	·	·	96	98
4-Insertions_3	17	43	68	94	mulsol.i.5	·	·	96	98
5-FullIns_3	35	72	95	113	myciel4	38	68		
adjnoun	11	29	51	81	myciel5	47	77	105	129
anna	7	7	9	15	myciel6	57	87	115	138
celegansneural	5	5	15	37	myciel7	67	-	-	148
chesapeake	28	60	92		polbooks	34	79	103	136
david	·	·	17	50	queen10_10	-	486		
dolphins	10	30	66	89	queen11_11	-	-		
DSJC125.1	106	-	-	-	queen12_12	-	-		
DSJC125.5	-	645			queen13_13	-	-		
football	101	-	-	-	queen14_14	-	-		
games120	-	-	-	-	queen5_5	103			
huck	7	17	33	54	queen6_6	149			
jazz	23	70	133	-	queen7_7	199			
jean	2	4	14	19	queen8_12	339			
karate	11	23	34	61	queen8_8	239			
lesmis	4	6	13	21	queen9_9	296			
miles1000	297				r125.1	·	·	2	9
miles1500	626				r125.1c	648			
miles250	·	·	7	30	r125.5	505			

Table 11 Optimal values of the instances with weights, instances that are infeasible and/or trivial for all values of k are not reported (Coloring and DIMACS).

	Optimal Values					Optimal Values			
	$k=5$	$k=10$	$k=15$	$k=20$		$k=5$	$k=10$	$k=15$	$k=20$
arc130	442				L120.cavity01	49	100	115	168
ash219	36	78	120	164	L120.fidap022	486			
ash331	39	-	-	-	L120.fidapm02	509			
ash85	117	-			L120.rbs480a	433			
bcspwr01	28	70			L120.wm2	7	28	67	239
bcspwr02	38	87	133		L125.ash608	37	-	-	-
bcspwr03	53	113	168	235	L125.bcsstk05	218			
bfw62a	114				L125.can__161	-	-	-	
can__144	-	-			L125.can__187	-	-	-	541
can61	207				L125.dwt__162	-	-	-	
can62	31	78	130		L125.dwt__193	291			
can73	144	-			L125.fs_183_1	71			
can96	-	-			L125.gre__185	-	-	-	
curtis54	74				L125.lop163	-	-	-	
dwt___59	59	141	226		L125.west0167	19	46	73	109
dwt66	54	-			L125.will199	21	60	92	127
dwt72	26	64	105	169	L80.cavity01	43	49	92	154
dwt87	66	-	313		L80.steam2	257			
gre__115	44	108	-	-	L80.wm1	88	218	281	
ibm32	80				L80.wm2	24	264		
impcol_b	22	58	109	202	L80.wm3	23	74		
L100.cavity01	49	91	100	162	lund_a	-	-		
L100.fidapm02	443				pores_1	99			
L100.rbs480a	370				rw136	40	-	-	-
L100.steam2	303				steam3	145			
L100.wm1	78	169	274		west0067	97	188		
L100.wm2	20	237			west0132	21	57	97	132
L100.wm3	20	76	303		will57	33	113		

Table 12 Optimal values of the instances with weights, instances that are infeasible and/or trivial for all values of k are not reported (Intersection graphs).