Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Casting Light on the Hidden Bilevel Combinatorial Structure of the Capacitated Vertex Separator Problem

Fabio Furini, Ivana Ljubić, Enrico Malaguti, Paolo Paronuzzi

informs
http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 70, No. 4, July–August 2022, pp. 2399–2420
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

Casting Light on the Hidden Bilevel Combinatorial Structure of the Capacitated Vertex Separator Problem

Fabio Furini,[a] Ivana Ljubić,[b] Enrico Malaguti,[c] Paolo Paronuzzi[c]

[a] Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", Consiglio Nazionale delle Ricerche (IASI-CNR), 00185 Roma, Italy; [b] ESSEC Business School of Paris, 95021 Cergy-Pontoise, France; [c] Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi," Università di Bologna, 40136 Bologna, Italy

Contact: fabio.furini@iasi.cnr.it, https://orcid.org/0000-0002-1839-5827 (FF); ivana.ljubic@essec.edu, https://orcid.org/0000-0002-4834-6284 (IL); enrico.malaguti@unibo.it, https://orcid.org/0000-0002-5884-9360 (EM); paolo.paronuzzi@unibo.it, https://orcid.org/0000-0002-0001-8628 (PP)

Received: September 26, 2019
Revised: April 20, 2020; September 16, 2020
Accepted: November 2, 2020
Published Online in Articles in Advance: September 16, 2021

Area of Review: Optimization

https://doi.org/10.1287/opre.2021.2110

Copyright: © 2021 INFORMS

Abstract. Given an undirected graph, we study the capacitated vertex separator problem that asks to find a subset of vertices of minimum cardinality, the removal of which induces a graph having a bounded number of pairwise disconnected shores (subsets of vertices) of limited cardinality. The problem is of great importance in the analysis and protection of communication or social networks against possible viral attacks and for matrix decomposition algorithms. In this article, we provide a new bilevel interpretation of the problem and model it as a two-player Stackelberg game in which the leader interdicts the vertices (i.e., decides on the subset of vertices to remove), and the follower solves a combinatorial optimization problem on the resulting graph. This approach allows us to develop a computational framework based on an integer programming formulation in the natural space of the variables. Thanks to this bilevel interpretation, we derive three different families of strengthening inequalities and show that they can be separated in polynomial time. We also show how to extend these results to a min-max version of the problem. Our extensive computational study conducted on available benchmark instances from the literature reveals that our new exact method is competitive against the state-of-the-art algorithms for the capacitated vertex separator problem and is able to improve the best-known results for several difficult classes of instances. The ideas exploited in our framework can also be extended to other vertex/edge deletion/insertion problems or graph partitioning problems by modeling them as two-player Stackelberg games and solving them through bilevel optimization.

Funding: This work was supported by the Air Force Office of Scientific Research [Grant FA8655-20-1-7019 to E. Malaguti and P. Paronuzzi].
Supplemental Material: The e-companion is available at https://doi.org/10.1287/opre.2021.2110.

Keywords: bilevel optimization • Stackelberg games • graph decomposition • branch-and-cut • Benders decomposition

1. Introduction

Given a graph, we are interested in finding the smallest subset of its vertices to remove such that the remaining graph has a bounded number of pairwise disconnected shores (subset of vertices) of limited cardinality. Formally, we study the following problem:

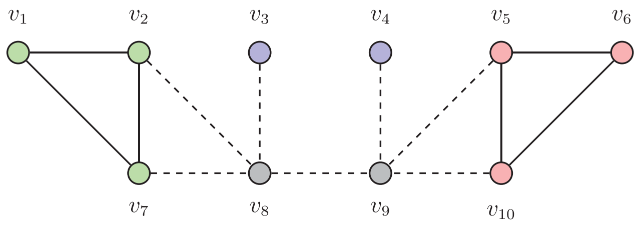
Definition 1 (Capacitated Vertex Separator Problem). Given a simple undirected graph $G=(V,E)$ and two integer values $k,b\in\mathbb{N}, k\geq 2$, the *capacitated vertex separator problem* (CVSP) asks for a partition of V into $k+1$ disjoint subsets $V=\{V_1,V_2,\dots,V_k\}\cup S$, where V_i $(i=1,\dots,k)$ are denoted as *shores* and S is denoted as *separator* such that $v\in V_i$ and $w\in V_j$ with $i,j\in\{1,\dots,k\}, j>i$ implying edge $vw\notin E$; the size of each shore is bounded by b; empty shores are allowed, and the cardinality of S is minimized.

In Figure 1, we give an example graph and provide an optimal CVSP solution for $k=b=3$. The separator is composed by the gray vertices, that is, the set $S=\{v_8,v_9\}$. Dashed lines represent the edges that are incident to the removed vertices. After the removal of S, the graph is partitioned into three pairwise disconnected shores, namely $V_1=\{v_1,v_2,v_7\}$ (first shore), $V_2=\{v_3,v_4\}$ (second shore), and $V_3=\{v_5,v_6,v_{10}\}$ (third shore).

Notice that in the definition of the CVSP, the parameter k is an upper bound on the number of shores, since empty shores are allowed. Hence, in some extreme cases (e.g., complete graphs), it may happen that the removal of S results in a graph having a single shore that consists of a connected component of size b (or smaller).[1]

The CVSP is equivalent to the *matrix decomposition problem* studied by Borndörfer et al. (1998). Indeed,

Figure 1. (Color online) An Example Graph G for the *Capacitated Vertex Separator Problem* (CVSP), with 10 Vertices and 13 Edges



it can be viewed as the problem of assigning the *rows* of a matrix A to k disjoint *blocks*. The objective is to remove a minimum number of rows from A and to assign the remaining rows to the blocks so that (*i*) each row is assigned to at most one block, (*ii*) each block contains at most b rows, and (*iii*) no two rows in different blocks have nonzero entries in the same column. The problems are equivalent by defining a binary matrix A as having a row for each vertex of G and a column for each edge of G. Each column has two nonzero entries at the rows corresponding to the end points of the associated edge (whereas all other entries are 0). Conversely, given A, we define a vertex in G for each row and an edge for each pair of vertices if there is a column in A with nonzero entries in the corresponding rows. The problem is $\mathcal{NP}$-hard, as discussed by Borndörfer et al. (1998). The matrix decomposition problem has relevant applications in parallel computing, namely, in the parallel solution of linear systems of equations. The number of shores k corresponds to the number of parallel machines on which the subsystems of equations are solved. At the same time, the bound b guarantees that the workload assigned to each of the machines is balanced.

Another relevant CVSP application (in which $k = |V|$) concerns the protection of communication or social networks against viral attacks. We assume that the decision maker has resources to vaccinate/protect some vertices in the network, without knowing at which vertex the attack will take place. Such situations are commonly analyzed in epidemic control (Tao et al. 2006) or in preventing the spread of fake news in social networks (see the work of Baggio et al. 2021 and further references therein). An attack can suddenly occur at any vertex of the network, and the virus can spread from the attacked vertex to its neighbors, as long as the neighbors are not protected/vaccinated. To contain the virus, one has to isolate the infected community from the rest of the network. At the same time, the maximum number of infected vertices need to be kept under control. In this context, the vertex separator found by the CVSP determines the smallest possible set of "critical" vertices that need to be protected/vaccinated in order to reduce the network vulnerability against the viral attack. The obtained CVSP solution guarantees that the number of potentially infected vertices (corresponding to the largest connected subgraph of the network once protected vertices are removed) is bounded by the parameter b.

The CVSP is closely related to another *interdiction* problem in which the available resources for network protection are limited, and the goal is to find a subset of critical vertices to protect so that the size of the largest connected component in the remaining graph is minimized (Albert et al. 2000, Shen and Smith 2012). A (maximal) connected component of an undirected graph is given by its connected subgraph such that no path exists between a vertex outside the subgraph and a vertex belonging to it. More formally, this problem is defined as follows:

Definition 2 (Minimize the Maximal Connected Component Problem). Given a simple undirected graph $G = (V, E)$ and an integer budget $B \in \mathbb{N}$, the maximal connected component problem (MinMaxC) asks for finding a subset of vertices $S \subset V$ to remove from G such that $|S| \leq B$ and such that the number of vertices of the largest connected component in the remaining graph is minimized.

We notice that in the MinMaxC, the size of connected components that are strictly smaller than the largest one does not play any role. In addition, there is no need to pack the connected components into k shores. Finally, the MinMaxC is strongly $\mathcal{NP}$-hard (Shen and Smith 2012). Besides the absence of this "packing" aspect, another major difference between the CVSP and the MinMaxC is in the type of objective function; instead of dealing with a min-max objective function, in the CVSP we are minimizing the number of vertices to be removed (i.e., according to the MinMaxC terminology of Shen et al. (2012), we are minimizing the budget) while making sure the largest component in the remaining graph will contain no more than b vertices. Although the two problems are different, we demonstrate that, thanks to the bilevel interpretation of the problem, many ideas developed for the CVSP can be directly applied to the MinMaxC as well.

1.1. Notation

Let K denote the set of integers $\{1, \ldots, k\}$. Given a simple undirected graph, $G = (V, E)$ for each edge $wv \in E$, we say that w and v are *neighbors*. Let $N(w) = \{v \in V | wv \in E, w \neq v\}$ denote the *neighborhood* of w. For each edge $vw \in E$, we define two arcs, $(v, w), (w, v)$, and A denotes the set of all of these arcs. Given a vertex $v \in V$, we indicate by $\delta^-(v)$ and $\delta^+(v)$ its subset of incoming and outgoing arcs, respectively, from A. A subset of vertices $W \subset V$ is a *clique* of G if any two vertices of W are neighbors. Given a tree T, we indicate by $deg_T(v)$ the number of edges incident to v in T. Given a subset of edges, $E' \subseteq E$ of G, we say that E' is *spanning* G if for every vertex v of G there is at least an edge in E' incident with v. Given $W \subset V$, the subgraph $G[W] = (W, E[W])$ is the graph induced by W, which contains all vertices of W and all edges $E[W] \subset E$, both of whose ends belong to W. Given a connected subgraph C of G, the vertex set of C is denoted by $V(C)$. Vectors are denoted in boldface; $\mathbf{0}$ and $\mathbf{1}$ denote the null vector and a vector of 1 entries, respectively.

1.2. Paper Contributions

This article studies a canonical IP formulation for solving the CVSP in which several new families of valid inequalities are derived by exploiting a "bilevel" point of view. The problem is seen as a two-player Stackelberg game in which a leader interdicts the network by removing some of its vertices, and a follower determines the *maximum connected component* in the remaining graph (we refer the interested reader to, for example, Baïou and Barahona (2016), Brotcorne et al. (2008), Casorrán et al. (2019), and Cormican et al. (1998) for other relevant problems related to Stackelberg games). In addition, the leader has to make sure that the connected components can be packed in at most k shores, each of the size at most b. We first provide a basic canonical formulation and show how to use the value function reformulation of the follower's optimization problem to derive new sets of valid inequalities. The value function reformulation is convexified in three different manners; the first one adds penalties for the violation of some constraints in the objective function, the second one is a Benders reformulation derived from an *extended formulation*, and the third one exploits necessary conditions on the number of vertices to remove in order to disconnect a graph. Theoretical analysis reveals that Benders inequalities are dominated by the first family of inequalities, whereas there is no domination between the second and third. We show that the new inequalities can be separated at integer points in polynomial time and explain the details of an efficient branch-and-cut implementation. We also show how to extend these results to a min-max version of the problem. A computational study that is performed on a large set of publicly available benchmark instances shows that our new exact method is competitive against the state-of-the-art branch-and-price procedure for the CVSP proposed by Bastubbe and Lübbecke (2020). Moreover, we are able to improve the best-known results for several difficult classes of instances and to provide optimal solution values for 60 previously unsolved instances from the literature.

The paper is structured as follows. In the remainder of this section, we provide an overview of the related literature and illustrate several optimal solutions for a real-world social network. In Section 2, we present a compact integer programming formulation for the CVSP. In Section 3, we develop our new formulation in the natural space of the variables obtained through a bilevel interpretation of the problem. In this section, we present several families of valid inequalities whose separation procedures are presented in Section 4. In Section 5, we discuss how to extend the bilevel interpretation and the developed inequalities to a min-max version of the problem. In Section 6, we discuss extensive computational results comparing a newly developed branch-and-cut algorithm with the state-of-the-art algorithms for the CVSP, and we also present results for the considered min-max version. Finally, in Section 7, we present the conclusions of our work and some future lines of research.

1.3. Literature Review

In this section, we provide a review of the exact algorithms proposed in the CVSP literature, and we present closely related problems.

To the best of our knowledge, the first exact algorithm for the CVSP, addressed as the *matrix decomposition problem*, has been proposed by Borndörfer et al. (1998). An *integer programming* (IP) formulation is proposed, and a branch-and-cut algorithm, based on polyhedral investigations, has been designed. The main motivation of the study was to verify whether the constraint matrix of a linear or integer program can be decomposed into the so-called *bordered block diagonal form* (see also Bergner et al. 2015 for further details). Recently, an alternative exact algorithm for the CVSP has been proposed by Bastubbe and Lübbecke (2020). In this paper, the CVSP has been called the *capacitated hypergraph vertex separator problem*, and a branch-and-price algorithm has been designed based on specialized algorithms to solve the pricing problems. In addition, a branching scheme tailored for the problem is proposed and enhanced by a number of speed-up techniques. It is worth mentioning that, even though Bastubbe and Lübbecke (2020) defined the problem on hypergraphs, an equivalent problem defined on simple graphs is obtained by replacing each hyperedge with a clique. We compare the computational performance of this branch-and-price algorithm with our newly developed branch-and-cut algorithm in Section 6.

Concerning the MinMaxC, the problem has been introduced by Albert et al. (2000), who proposed a greedy heuristic in which the vertices are sequentially removed from the network, starting with those with the highest degrees. Shen et al. (2012) introduced an exact approach based on an extended MIP formulation and a family of valid inequalities. A dynamic programming procedure that runs in polynomial time on trees and series-parallel graphs can be found in the work of Shen and Smith (2012). These authors also showed that for the problem variant in which each vertex is associated with a nonnegative weight, and the goal is to minimize the maximum-weighted connected component in the remaining graph, the problem becomes $\mathcal{NP}$-hard even on trees.

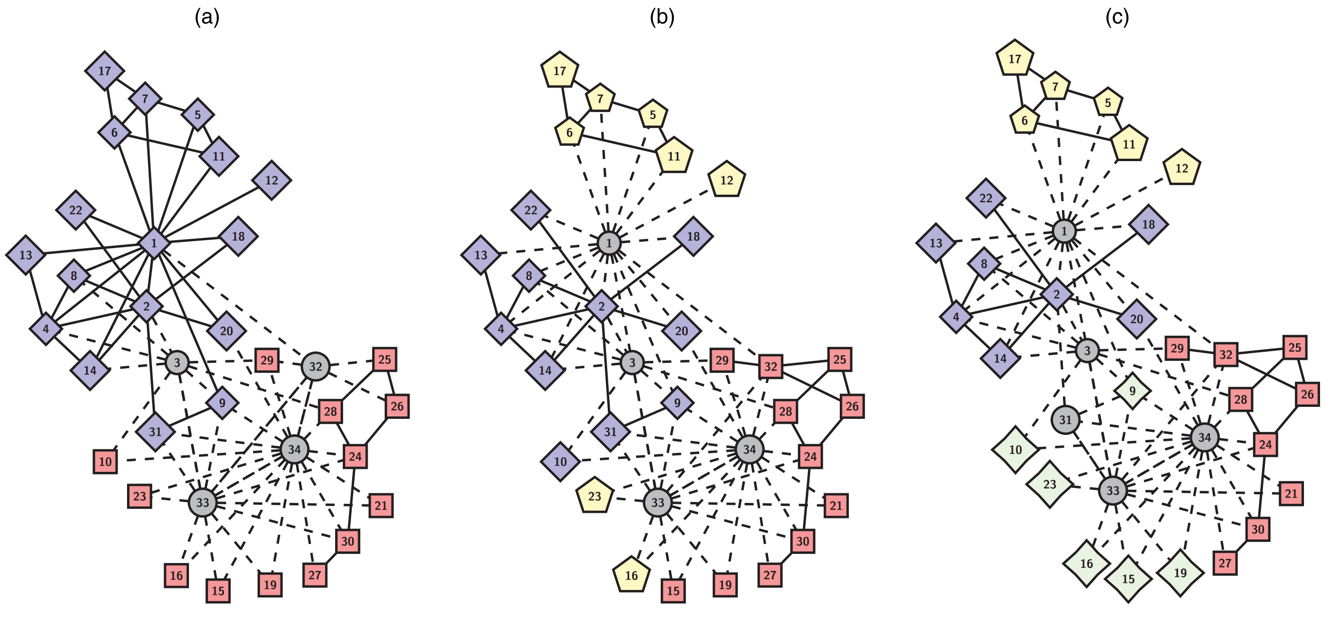
Another problem related to the CVSP is the *k-vertex cut problem*. Formally, a vertex cut is a set of vertices whose removal disconnects the graph into several connected components. If the number of connected components is at least k, this set is called a k-vertex cut. Given a graph G, a positive weight c_v for each vertex $v \in V$, and an integer $k \geq 2$, the k-vertex cut problem (k-VCP) is to find a k-vertex cut of minimum weight. The k-VCP has been object of research in recent years, and we address the interested reader to, for example, the work of Cornaz et al. (2019), where an exact branch-and-price algorithm has been proposed. Recently, Furini et al. (2019) proposed a branch-and-cut algorithm for the k-VCP, exploiting a bilevel point of view of the problem, which allowed us to derive a valid IP formulation in the natural space of the variables and to beat state-of-the-art results achieved by Cornaz et al. (2019). One of the main differences between the k-VCP and the CVSP is that no capacity restriction on the size of the components is considered in the former one. In addition, the CVSP imposes an *upper bound* k on the number of shores, whereas for the k-VCP the value k represents the *lower bound* on the number of connected components obtained after the vertices belonging to the k-vertex cut are removed. For these reasons, the two problems are structurally very different, and even though they both can be seen through bilevel lenses, there is no result for the k-VCP that straightforwardly translates into a related result for the CVSP. Moreover, the optimal solutions of the k-VCP are comprised of k connected components that can be very imbalanced, and hence, they are of little use for the practical applications that motivate our research.

Finally, the CVSP is also related to the *vertex separator problem* (VSP), considered by de Souza and Balas (2005a, b). In the VSP, we are given an integer $b \in \mathbb{N}$ and a cost $c_v \in \mathbb{N}$ associated with each vertex $v \in V$. The VSP asks for a partition of V into three disjoint *nonempty* subsets, V_1, V_2, and S, where V_1 and V_2 are the *shores* of the *separator* S such that $v \in V_1$ and $w \in V_2$ imply edge $vw \notin E$, the size of each shore is bounded by b, and the function $\Sigma_{v \in S} c_v$ is minimized. The VSP is $\mathcal{NP}$-hard even for planar graphs (Fukuyama 2006) or maximum degree 3 graphs (Bui and Jones 1992), and it has several applications for different connectivity problems (we refer the interested reader to Djidjev (2000), Garg et al. (1999), and Lipton and Tarjan (1979) and to de Souza and Balas (2005b) for a survey of such applications); one of the most important ones is related to the efficient solution of linear systems (Heath et al. 1991, Lipton and Tarjan 1977).

1.4. Examples of Optimal Solutions

In this section, we depict some examples of optimal solutions of the CVSP and of the MinMaxC for a classical social network from the literature. We consider the instance introduced by Zachary (1977) that consists of a network modeling the interactions of a university-based karate club. This network has 34 vertices and 78 edges, where the vertices represent active members of the club and the edges represent strong interactions between the members.

Figure 2. (Color online) Optimal CVSP Solutions: In Part (a) with $k = 2$ and $b = 17$, in Part (b) with $k = 3$ and $b = 12$, and in Part (c) with $k = 4$ and $b = 9$



In Figure 2, we report three optimal solutions of the CVSP with different values for the maximum number of shores k, that is, $k \in \{2,3,4\}$, and we set $b = \lceil |V|/k \rceil$. The vertices belonging to the separators are depicted with gray circle and their incident edges are depicted with dashed lines. The vertices belonging to the different shores are depicted with different colors and different shapes. In Figure 2(a), for $k = 2$ and $b = 17$, the size of the largest connected component is 17, and there is one more disconnected shore of 13 vertices. In Figure 2(b), for $k = 3$ and $b = 12$, the size of the largest connected component is 10, and there are three disconnected shores with 11, 11, and 8 vertices, respectively. Finally, Figure 2(c), for $k = 4$ and $b = 9$, shows that the size of the largest connected component is 8, and there are four disconnected shores of size 9, 9, 6, and 6, respectively. We notice that the size of the optimal separator is the same for $b \in \{12, 17\}$, whereas it increases by just one unit for $b = 9$. This can be attributed to the small-world effect and scale-free property of real-world networks (Watts and Strogatz 1998, Tao et al. 2006), as opposed to regular or randomly generated networks. These results are in line with the observations of Albert et al. (2000); in homogeneous networks, all vertices have approximately the same number of links, and hence, they all contribute equally to the connectivity of the network. However, the power-low distribution of vertex degrees in scale-free networks implies that targeting of a relatively small number of the "most-critical" vertices may significantly reduce the connectivity of the network.

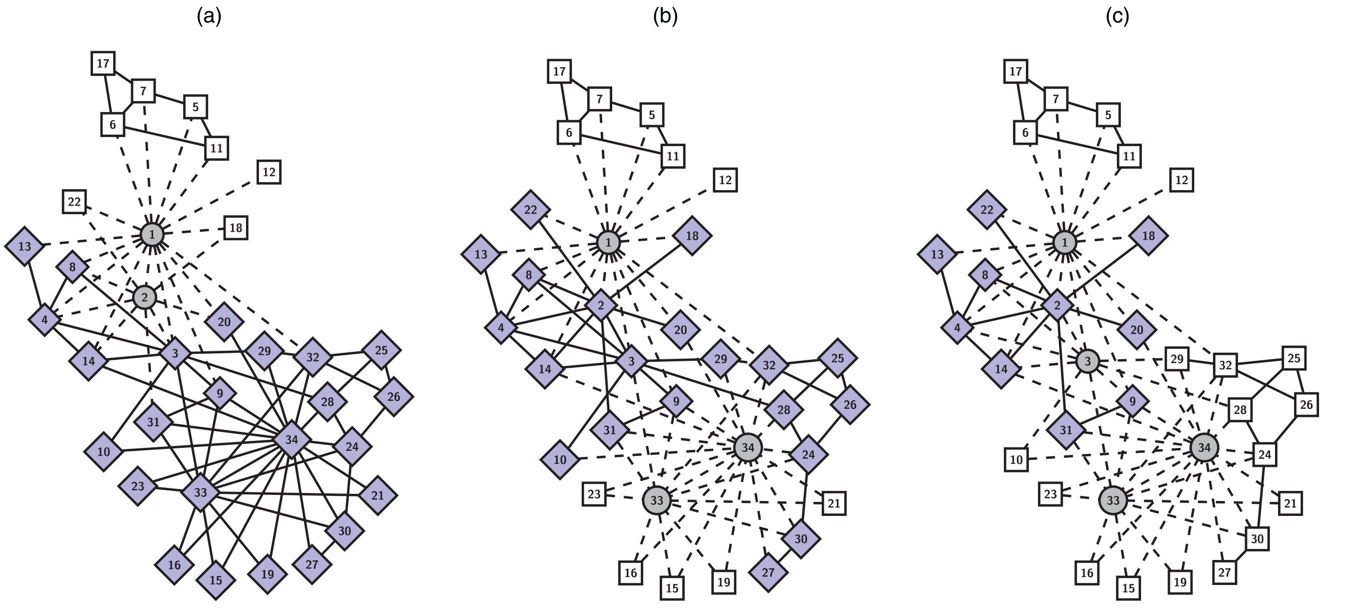
We observe similar effects when solving the MinMaxC; three optimal solutions of the MinMaxC with different budget levels are reported in Figure 3.

We depict with blue diamonds the vertices belonging to the largest connected component and in white the remaining vertices. For $B \in \{2,3,4\}$, the size of the largest connected component reduces from 34 to 24, 20, and 10, respectively. Hence, a minimal increase in the budget B allows for a significant reduction of the largest connected component. Because of the scale-free property of such networks, one might be tempted to apply an intuitive approach in which the most-connected vertices (i.e., those with the highest degree) should be removed first, as suggested, for example, by Albert et al. (2000). However, Figure 3(a) illustrates that it is not always true that optimal separators contain the highest-degree vertices (i.e., the degree of vertex 2 is only six). Moreover, it has been shown that a greedy heuristic in which the vertices are removed based on their degrees can lead to solutions in which the size of the largest component can be arbitrarily bad when compared with the value of the optimal solution (Shen et al. 2012).

2. A Compact Integer Programming Formulation

A first IP formulation for the CVSP has been introduced by Borndörfer et al. (1998), who defined a binary variable ξ_v^i for each vertex $v \in V$ and each integer $i \in K = \{1, 2, \ldots, k\}$, such that $\xi_v^i = 1$ if vertex v belongs to the shore V_i and 0 otherwise. In this formulation, the vertices that remain unassigned to any of the shores (i.e., for which $\xi_v^i = 0$, for all $i \in K$), are the ones defining the separator S. This is why instead of minimizing the cardinality $|S|$ of the separator, one can equivalently maximize the number of vertices in the

Figure 3. (Color online) Optimal MinMaxC Solutions: In Part (a) with $B = 2$, in Part (b) with $B = 3$, and in Part (c) with $B = 4$



shores (i.e., the vertices in $\cup_{i\in K} V_i$), thus obtaining the following IP formulation:

$$\max \sum_{i\in K}\sum_{v\in V} \xi_v^i \tag{1a}$$

$$\sum_{i\in K} \xi_v^i \le 1 \qquad v \in V \tag{1b}$$

$$\xi_w^i + \sum_{j\in K\setminus\{i\}} \xi_v^j \le 1 \qquad i \in K,\ wv \in E \tag{1c}$$

$$\sum_{v\in V} \xi_v^i \le b \qquad i \in K \tag{1d}$$

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

The objective function (1a) maximizes the number of vertices assigned to the shores of the separator. Constraints (1b) impose that each vertex is assigned to at most one shore, and (1c) imposes that the shores induce pairwise disconnected subgraphs. Constraints (1d) impose that the capacity of each shore is not exceeded, that is, the number of vertices assigned to each shore is not larger than the capacity b. This formulation is known to suffer from symmetries, given that any permutation of indices $\{1,\dots,k\}$ results in the same feasible (LP-)solution.

Bastubbe and Lübbecke (2020) reformulated this model by defining an *edge clique cover* $\mathcal{Q}$ of G. An edge clique cover is a collection $\mathcal{Q}$ of cliques of G such that for each edge wv of E there exists a clique $Q \in \mathcal{Q}$ with $w, v \in Q$. The model is then obtained by introducing a binary variable ψ_Q^i for each integer $i \in K$ and each clique $Q \in \mathcal{Q}$ such that $\psi_Q^i = 1$ if some vertex $v \in Q$ belongs to the shore i and 0 otherwise. Constraints (1b) and (1c) are then replaced by

$$\sum_{i\in K} \psi_Q^i \le 1 \qquad Q \in \mathcal{Q}, \tag{2a}$$

$$\xi_v^i - \psi_Q^i \le 0 \qquad i \in K,\ Q \in \mathcal{Q},\ v \in Q. \tag{2b}$$

Constraints (2a) impose that each clique $Q \in \mathcal{Q}$ is assigned to at most one shore, whereas constraints (2b) impose that the vertices can be assigned to a shore if and only if they belong to a clique $Q \in \mathcal{Q}$ selected for the shore.

Borndörfer et al. (1998) strengthened formulation (1a)–(1e) through several valid inequalities, and they solved it by means of a tailored branch-and-cut algorithm. Among the inequalities introduced in their work, the so-called *block-invariant inequalities* are invariant under a permutation of the indices of the shores $i \in K$ (called blocks by Borndörfer et al. 1998). These inequalities can be expressed for aggregated variables defined as

$$z_v = \sum_{i\in K} \xi_v^i, \quad v \in V,$$

which define whether a vertex v is assigned to a shore ($z_v = 1$) or is removed from G ($z_v = 0$); that is, it is in the separator S. In the next section, we present an IP formulation based on the complement of these variables and recall some of the block-invariant inequalities that we exploit to strengthen our formulation.

3. A Canonical IP Formulation

In this section, we study an IP model for the CVSP that exploits the complement of the aggregated variables z_v, $v \in V$ introduced in the previous section. Our goal is to provide a "thin" formulation that lives in the natural space of decision variables, namely those constituting the objective function. This will allow us to tackle more challenging (and potentially denser) instances, using only a linear number of decision variables.

To this end, for each vertex $v \in V$, a binary variable x_v is defined such that $x_v = 1$ if vertex v belongs to the separator S and 0 otherwise. Given an arbitrary subset of vertices $W \subseteq V$, the connected components C in $G[W]$ can be viewed as items of weight $|V(C)|$, that is, the number of vertices in the component. Let $\sigma(W)$ be equal to the number of bins of size b needed to pack the connected components of $G[W]$. If the size of a connected component is larger than b, the packing is not feasible, and we set $\sigma(W) = \infty$. The CVSP can be then modeled as follows:

$$\min \sum_{v\in V} x_v \tag{3a}$$

$$\sum_{v\in W} x_v \ge 1 \qquad W \subseteq V : \sigma(W) > k \tag{3b}$$

$$x_v \in \{0,1\} \qquad v \in V, \tag{3c}$$

where the objective function (3a) minimizes the number of deleted vertices, that is, the vertices in the separator S. Constraints (3b), denoted as *bin-packing constraints* in the following, guarantee that any vertex separator S (encoded by $\boldsymbol{x}$ variables) that does not allow for "packing" of the connected components of $G[V\setminus S]$ into k shores of size b has to be discarded. The bin-packing constraints model the fact that if more than k bins must be used for a vertex subset $W \subseteq V$, the connected components induced by W cannot be "packed" into the k shores of capacity b, so at least one vertex in W must belong to the separator. This exponential-size family of constraints has been proposed by Borndörfer et al. (1998), who showed that their separation problem corresponds to solving the bin-packing problem (BPP). The BPP is strongly $\mathcal{NP}$-hard, and we refer the reader to the work of Delorme et al. (2016) for a recent survey on the problem.

As a special case, if for a vertex subset $W \subseteq V$ there exists a component C in $G[W]$ such that $|V(C)| > b$ (which can be determined in polynomial time), the solution is infeasible, and hence, at least one vertex from C must belong to the separator. It is not difficult to see that in the latter case, the associated bin-packing constraints (3b) are dominated by

$$\sum_{v\in C} x_v \geq 1, \quad C\subseteq W : C \text{ connected and } |V(C)| = b+1. \tag{4}$$

Nevertheless, constraints (4) are not sufficient to build a valid formulation unless the value of k is such that the condition on the maximum number of shores is not binding (i.e., any solution satisfying constraints (4) also satisfies constraints (3b)). It is not hard to see that this is the case for any $k \geq |V|$, whatever the value of capacity b. The following proposition provides a tighter result:

Proposition 1. *Any solution satisfying constraints (4) defines a separator S and a graph $G[V\backslash S]$ whose connected components can be packed into at most $k = 2(|V|-1)/(b+1)$ bins for odd values of b and $k = 2|V|/(b+2)$ bins for even values of b, respectively.*

Proof. Let S be a separator, satisfying constraints (4).

Let us define a BPP instance with bins of capacity b for packing the connected components of $G[V\backslash S]$ and such that the number of necessary bins in the largest. This happens when all (but possibly one) item weights just exceed $b/2$, that is:

- the item weights are all equal to $b+1/2$ when b is odd; and
- all of the item weights but one are $b/2+1$, and one is possibly $b/2$, when b is even.

We can now compute the overall weight that is packed in the two cases and set it equal to the residual number of vertices $|V|-|S|$:

$$\begin{aligned} \frac{b+1}{2}k &= |V|-|S| \quad \text{if } b \text{ is odd;} \\ \left(\frac{b}{2}+1\right)(k-1)+\frac{b}{2} &= |V|-|S| \quad \text{if } b \text{ is even.} \end{aligned}$$

Because at least one vertex must be removed in a nontrivial instance of the CVSP, we can impose $|S| = 1$, and the result follows. □

Besides its sparsity, another major advantage of model (3) compared with the formulation (1) from Section 2 is that we get rid of the symmetries (i.e., the degeneracy caused by index permutations). This comes at a cost of having an $\mathcal{NP}$-hard procedure to check feasibility of any integer point of the branch-and-cut tree.

To (partially) overcome this difficulty, in the remainder of this section we propose new valid inequalities in the space of x variables that can be used to enhance this basic model and whose separation can be performed in polynomial time. To derive these inequalities, we approach the problem from a bilevel perspective.

3.1. A Bilevel Interpretation of the Problem

Bilevel optimization has recently attracted a lot of attention from the research community, not only because of its relevance for the real-world applications but also because of the recent advancements in the development of off-the-shelf MILP solvers. The latter ones are the major driving force for the methods of computational optimization to be pushed to the next frontiers (Dempe and Zemkoho 2013, Fischetti et al. 2017, Lozano and Smith 2017, Tahernejad et al. 2020, Kleinert et al. 2020). We propose a novel way of interpreting the CVSP as a defender-attacker game. Such problems are typically solved using the tools and methods of bilevel optimization (Borrero et al. 2019, Fischetti et al. 2019, Baggio et al. 2021). Our ideas based on bilevel optimization allow us to improve the modeling power and understanding of the CVSP.

The CVSP can be viewed as a two-player Stackelberg game, that is, a game where players take decisions sequentially and are denoted as a *leader* (i.e., defender) and a *follower* (i.e., attacker). In the first step, the leader “interdicts” the follower by deleting (i.e., protecting, vaccinating) some vertices from the graph. In the following step, the follower determines a maximum connected component in the remaining graph. Hence, from the perspective of the leader, the problem is to find the smallest subset of vertices to delete from G so that the size of the optimal follower solution (i.e., the number of vertices in the maximum connected component in the remaining graph) is at most b. For binding values of k (cf. Proposition 1), we are interested in finding at most k shores; hence, the leader solution must additionally satisfy the bin-packing constraints (3b).

Independently on the value of k, using the value function reformulation for the follower, we can impose the following condition:

$$\Phi(\boldsymbol{x}) \leq b, \tag{5}$$

where $\Phi(\boldsymbol{x})$ denotes the optimal solution value of the follower subproblem for a given vector $\boldsymbol{x}$. In general, the value function $\Phi(\boldsymbol{x})$ does not need to be convex. Hence, one possible way to deal with the problem and to derive a single-level problem reformulation is to try to *convexify* the value function. In the following, we discuss two possible ways to convexify this function and derive valid inequalities.

3.1.1. Convexification by Penalization: Component Inequalities.

Given a *binary* realization of the leader variables $\boldsymbol{x}^*$, we denote by $V(\boldsymbol{x}^*) \subset V$ the subset of *interdicted vertices* and by G^* the *interdicted graph*, which is the subgraph of G induced by $V\backslash V(\boldsymbol{x}^*)$. The value $\Phi(\boldsymbol{x}^*)$ can be calculated in $O(|E|)$ time by simply removing the vertices $v \in V(\boldsymbol{x}^*)$ and searching for the largest connected component in the resulting graph G^*. Nevertheless, because our next goal is to use the value function reformulation to derive valid linear constraints in the x space, in the following we are providing a sparse IP formulation for finding $\Phi(\boldsymbol{x}^*)$.

In this follower's subproblem, for each vertex $v \in V$, a binary variable y_v is defined such that $y_v = 1$ if vertex v belongs to a maximum connected component and 0 otherwise, recalling that a maximum connected component has to be determined in the interdicted graph. The follower IP formulation reads

$$\Phi(\boldsymbol{x}^*) = \max \sum_{v \in V} y_v \tag{6a}$$

$$y_v \le 1 - x_v^* \qquad v \in V \tag{6b}$$

$$\sum_{u \in F} y_u \ge y_w + y_v - 1 \qquad wv \notin E, F \in \mathcal{F}_{wv} \tag{6c}$$

$$y_v \in \{0, 1\} \qquad v \in V. \tag{6d}$$

The objective function (6a) maximizes the number of the selected vertices. Constraints (6b) ensure that the interdicted vertices cannot be selected. Constraints (6c) impose that the optimal follower solutions correspond to connected components (in the interdicted graph G^*). Given a nonadjacent pair of distinct vertices $w, v \in V$, a set $F \subset V$ is called *v-w-separator* if and only if removing F from G disconnects w from v. These constraints are then defined with respect to the collection $\mathcal{F}_{wv}$ of all of the (minimal) *w-v-separators* for each pair of vertices $wv \notin E$. More precisely, constraints (6c) impose that if a pair of vertices w and v ($wu \notin E$) is selected, at least one vertex in each $F \in \mathcal{F}_{wv}$ must be selected as well (see, e.g., the work of Fischetti et al. 2017 for further details).

We aim at finding a reformulation of the follower's subproblem whose feasible space does not depend on $\boldsymbol{x}^*$, with an adapted objective function, so that for any choice of $\boldsymbol{x}^*$, the two problems provide the same optimal solution. In our setting, we apply *convexification by penalization*, as it is done, for example, by Brown et al. (2006), Cormican et al. (1998), and Fischetti et al. (2019). The major goal is to remove interdiction constraints (6b) from the follower's subproblem and to introduce penalty terms $M_v x_v^* y_v$ in the objective function instead so that the existence of the optimal follower solution satisfying $y_v x_v^* = 0$ is guaranteed. The reformulation can be obtained as stated in the following observation.

Observation 1. The follower subproblem can be restated as

$$\Phi(\boldsymbol{x}^*) = \max \left\{ \sum_{v \in V} y_v - \sum_{v \in V} M_v \, x_v^* \, y_v : \ (6c), \ (6d) \right\} \tag{7}$$

where M_v are sufficiently large values that guarantee that $y_v = 0$ whenever $x_v^* = 1$.

With the above observation and a proper choice of multipliers M_v, $v \in V$, the value function $\Phi(\boldsymbol{x})$ becomes a piece-wise convex function defined as

$$\Phi(\boldsymbol{x}) = \max_{y^* \in \mathcal{Y}} \sum_{v \in V} y_v^* - \sum_{v \in V} M_v \, y_v^* \, x_v,$$

where $\mathcal{Y}$ denotes all feasible points of the follower subproblem defined by constraints (6c) and (6d). Therefore, $\boldsymbol{y}^*$ is the indicator vector of the vertex sets $V(C)$, $C \in \mathcal{C}$, where $\mathcal{C}$ is the collection of the connected subgraphs of G.

Hence, constraint (5) can now be replaced by the following family of inequalities:

$$\sum_{v \in V(C)} (1 - M_v x_v) \le b, \quad C \in \mathcal{C}. \tag{8}$$

The new constraints (8) have been obtained by replacing in (5) the expression of $\Phi(\boldsymbol{x})$ by the objective function of (7). They can be equivalently restated as

$$\sum_{v \in V(C)} M_v x_v \ge |V(C)| - b \qquad C \in \mathcal{C}, \tag{9}$$

imposing that, for each connected subgraph C of G, the sum of the M_v coefficients associated with the interdicted vertices is greater than or equal to the cardinality of the vertex set of C, denoted as $V(C)$, minus the capacity b.

A straightforward tightening of the coefficients gives

$$\sum_{v \in V} \min\{ |V(C)| - b \, , \, M_v \} x_v \ge |V(C)| - b \quad C \in \mathcal{C}. \tag{10}$$

In order to obtain a tight formulation, the values of M_v should be as small as possible. Finding the tightest possible coefficients M_v is a nontrivial task, and the existing literature on bilevel optimization provides recipes for their calculation under some very specific assumptions related to the follower's subproblem. For example, Wood (2010) and Brown et al. (2006) assume that the follower's subproblem is a linear program, whereas Fischetti et al. (2019) provide a more general result for the discrete and continuous follower's subproblems satisfying the downward monotonicity property. Other results can be found for the case in which the follower solves a graph optimization problem that satisfies vertex- or edge-hereditary property (see Furini et al. 2019). Unfortunately, none of these assumptions are satisfied by the follower's subproblem defined in (6). In the following, we discuss how to derive tight inequalities for this nontrivial situation.

Given a tree T with $|V(T)| > b$, let $\text{comp}_T(v)$ denote the cardinality of the vertex set of a largest connected component obtained after removing v.

Proposition 2. *Let $C \in \mathcal{C}$ be a tree T, and then Formulation (7) is correct if we choose*

$$M_v = |V(C)| - \text{comp}_T(v). \tag{11}$$

Proof. The value of M_v exactly models the reduction of the objective value of the follower subproblem (7) when a single vertex v is interdicted. When more than one vertex is interdicted, the overall reduction of the objective value is overestimated, still guaranteeing that the optimal follower solution satisfies $y_v x_v^* = 0$. □

In the general case, that is, when C is a generic connected subgraph from $\mathcal{C}$ with $|V(C)| > b$, the above-mentioned constraints remain valid when imposed for a tree of C, and hence, they can be imposed for *any spanning tree* T of C. We have the following result:

Proposition 3. *Let C be a connected subgraph of G with $|V(C)| > b$, and let $\mathcal{T}(C)$ be the set of all spanning trees of C; then the following component inequalities,*

$$\sum_{v \in V(C)} \Big(|V(C)| - \text{comp}_T(v)\Big) x_v \geq |V(C)| - b \qquad T \in \mathcal{T}(C), \tag{12}$$

are valid for model (3).

Proof. By contradiction, assume there exists a graph $\tilde{G}$, an instance of the CVSP, and an associated feasible solution $\tilde{\boldsymbol{x}}$ that violates one of the inequalities (12). Hence, there exists in $\tilde{G}$ a tree $\tilde{T}$ spanning noninterdicted vertices according to $\tilde{\boldsymbol{x}}$ and for which (12) is violated. This means that $\tilde{T}$ is spanning a connected subgraph of $\tilde{G}$ of noninterdicted vertices having cardinality larger than b, contradicting the assumption that $\tilde{\boldsymbol{x}}$ is feasible. □

Notice that the component inequalities can be tightened by taking the minimum of the coefficient next to each variable and the right-hand side, as in (10).

3.1.2. Convexification By Dualization: Benders Inequalities.

In this section, we show how to model the followers' subproblem as a linear program. For each couple of vertices $v, l \in V$, we define a nonnegative continuous variable $\sigma_{vl} \leq 1$ that takes value one if vertices v and l belong to the same component. Furthermore, we introduce an additional continuous variable λ that represents the size of the largest component in the interdicted graph G^*. Using these variables, $\Phi(\boldsymbol{x}^*)$ can be determined using the following compact LP formulation (see also the work of Shen et al. 2012):

$$\Phi(\boldsymbol{x}^*) = \min_{\lambda \geq 0,\ \boldsymbol{0} \leq \boldsymbol{\sigma} \leq \boldsymbol{1}} \lambda \tag{13a}$$

$$\lambda \geq \sum_{v \in V} \sigma_{vl} \qquad l \in V \tag{13b}$$

$$x_w^* + x_v^* \geq \sigma_{vl} - \sigma_{wl} \qquad (v,w) \in A, l \in V \tag{13c}$$

$$\sigma_{ll} \geq 1 \qquad l \in V. \tag{13d}$$

Constraints (13d) impose that each vertex belongs to its own component. Constraints (13c) guarantee that if two neighboring vertices, v and w, are not interdicted, and v is connected to l in G^*, then w must be connected to l as well. For a pair of vertices $vw \in E$, if $x_w^* = 1$ or $x_v^* = 1$, the corresponding constraints (13c) are deactivated. In constraints (13b), the right-hand side represents an upper bound on the size of the connected component in G^* containing vertex l. Accordingly, constraints (13b) impose that λ is greater than or equal to the maximum of $\Sigma_{v \in V} \sigma_{vl}$ over all $l \in V$.

The following proposition guarantees the validity of the model to compute $\Phi(\boldsymbol{x}^*)$.

Proposition 4. *Given a binary realization of the leader variables $\boldsymbol{x}^*$, there exists an optimal solution to* (13) *in which $\tilde{\sigma}_{vl} = 1$ if and only if node v and node l belong to the same component in the interdicted graph G^*, and $\tilde{\sigma}_{vl} = 0$ otherwise.*

Proof. We first observe that the solution $\tilde{\boldsymbol{\sigma}}$ is feasible for (13) and accordingly the optimal solution value $\lambda^* \leq \max_{l \in V}\left\{\Sigma_{v \in V} \tilde{\sigma}_{vl}\right\}$. Consider any feasible solution $\hat{\boldsymbol{\sigma}}$ satisfying constraints (13c) and (13d). Let l be a noninterdicted vertex in G^*; $\hat{\sigma}_{ll} = 1$ and, due to (13c) for all the noninterdicted neighbors $v \in N(l) \backslash V(\boldsymbol{x}^*)$, we have $\hat{\sigma}_{vl} = 1$. By repeating this argument for each $v \in N(l) \backslash V(\boldsymbol{x}^*)$, we conclude that $\hat{\sigma}_{vl} = 1$ for all v belonging to the connected component of l in G^*. From constraint (13b), we then have $\lambda^* \geq \max_{l \in V}\left\{\Sigma_{v \in V} \hat{\sigma}_{vl}\right\}$. This proves that the solution $\tilde{\boldsymbol{\sigma}}$ is optimal. Similar arguments have been used by Shen et al. (2012), proposition 2. □

The following Corollary allows us to strengthen formulation (3) by using the additional $\boldsymbol{\sigma}$ variables.

Corollary 1. *The following set of constraints is valid for model* (3):

$$\sum_{v \in V} \sigma_{vl} \leq b \qquad l \in V \tag{14a}$$

$$x_w + x_v \geq \sigma_{vl} - \sigma_{wl} \qquad (v,w) \in A, l \in V \tag{14b}$$

$$\sigma_{ll} \geq 1 \qquad l \in V \tag{14c}$$

$$\sigma_{vw} \geq 0 \qquad v, w \in V. \tag{14d}$$

Proof. The meaning of the $\boldsymbol{\sigma}$ variables is the same as in model (13), and constraints (14a) ensure that the size of any connected component in the interdicted graph G^* does not exceed the capacity b. □

To the best of our knowledge, the extended formulation, obtained by adding constraints (14) to (3), is new and has not been considered in the previous literature. Because our major motivation is to study the IP models in the natural space of $\boldsymbol{x}$, our next goal is to project out $\boldsymbol{\sigma}$ variables from this model. This can be done in a Benders fashion by dualizing the function $\Phi(\boldsymbol{x})$ defined in (13).

By associating nonnegative dual variables $\boldsymbol{\alpha}, \boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ to the constraints (13b), (13c), and (13d), respectively, and by dropping the redundant constraints $\boldsymbol{\sigma} \le \mathbf{1}$, we get the following dual LP:

$$\Phi(\boldsymbol{x}^*) = \max_{(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})\ge \mathbf{0}} \sum_{l\in V}\left(\gamma_l - \sum_{vw\in A}\beta^l_{vw}(x^*_v + x^*_w)\right) \tag{15a}$$

$$\sum_{wv\in\delta^-(v)}\beta^l_{wv} - \sum_{vw\in\delta^+(v)}\beta^l_{vw} \le \alpha_l \qquad v,l\in V, v\ne l \tag{15b}$$

$$\sum_{wv\in\delta^-(v)}\beta^l_{wv} - \sum_{vw\in\delta^+(v)}\beta^l_{vw} \le \alpha_l - \gamma_l \qquad v,l\in V, v = l \tag{15c}$$

$$\sum_{l\in V}\alpha_l = 1. \tag{15d}$$

The following Proposition provides the Benders reformulation of model (3) extended by (14).

Proposition 5. *Constraints* (14) *can be equivalently replaced by the following family of Benders feasibility inequalities:*

$$\sum_{l\in V}\left(\tilde{\gamma}_l - \sum_{vw\in A}\tilde{\beta}^l_{vw}(x_v + x_w)\right) \le b \tag{16}$$

where $(\tilde{\boldsymbol{\alpha}},\tilde{\boldsymbol{\beta}},\tilde{\boldsymbol{\gamma}})$ *represent the extreme points of the dual polyhedron defined by* (15b)–(15d).

Proof. The proof follows from LP duality theory. We point out that Benders inequalities (16) correspond to *normalized and aggregated Benders feasibility cuts* derived by the standard projection of $\boldsymbol{\sigma}$ variables from the model (3)+(14). To see why this is the case, observe that the relaxed Benders master problem consists of model (3), whereas the Benders subproblem consists of checking the feasibility of constraints (14) for any given solution $\boldsymbol{x}^*$ of the master. By Farkas Lemma, the system (14) is infeasible if the following LP is unbounded (observe that the Benders subproblem is separable by l):

$$D_l(\boldsymbol{x}^*) = \max_{(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})\ge \mathbf{0}} \gamma_l - b\alpha_l - \sum_{vw\in A}\beta^l_{vw}(x^*_v + x^*_w) \tag{17a}$$

$$\sum_{wv\in\delta^-(v)}\beta^l_{wv} - \sum_{vw\in\delta^+(v)}\beta^l_{vw} \le \alpha_l \qquad v,l\in V, v\ne l \tag{17b}$$

$$\sum_{wv\in\delta^-(v)}\beta^l_{wv} - \sum_{vw\in\delta^+(v)}\beta^l_{vw} \le \alpha_l - \gamma_l \qquad v,l\in V, v = l \tag{17c}$$

For a binary solution $\boldsymbol{x}^*$, let C_l be the connected component in the interdicted graph containing l. The optimal solution of this LP corresponds to a single-commodity flow in which α_l units are sent from l to all other vertices from $V(C_l)$. The flow is sent along a spanning tree T rooted at l of C_l, and each value β^l_{vw} counts the total amount of flow carried along the arc (v,w) of that tree. Hence, $\gamma_l = |V(C_l)|\cdot\alpha_l$, and the value of the optimal solution is $(|V(C_l)| - b)\alpha_l$. The problem is unbounded if $|V(C_l)| > b$, and the standard Benders feasibility cut (associated to the l-th subproblem) reads

$$\tilde{\gamma}_l - b\,\tilde{\alpha}_l - \sum_{vw\in A}\tilde{\beta}^l_{vw}(x_v + x_w) \le 0$$

where $(\tilde{\boldsymbol{\alpha}},\tilde{\boldsymbol{\beta}},\tilde{\boldsymbol{\gamma}})$ corresponds to an extreme ray from the dual cone defined by (17b)–(17c), along with the nonnegativity constraints. It is now not difficult to see that our Benders inequalities (16) correspond to latter cuts after aggregating all $|V|$ subproblems and adding a normalization hyperplane (15d). □

Similarly, given an interdicted graph G^*, the dual model (15) represents a single-commodity flow formulation imposed for each "root" $l\in V$. For a connected component C in G^*, a vertex l is chosen as a root, and α_l units of flow are sent from l to every other vertex $v\in V(C)$. Thereby, the value γ_l contains the total amount of flow sent from l plus α_l (which is exactly the size of C, assuming $\alpha_l = 1$). Because we are looking for a distribution of the values of α_l among the vertices of G, and we penalize each arc (v,w) whose end vertices are interdicted (cf. the 2nd term in the objective function), an optimal solution of problem (15) is obtained by choosing the largest component in the interdicted graph, arbitrarily picking one of its vertices l as a root and setting $\alpha_l = 1$. Hence, instead of detecting Benders inequalities using a black-box LP formulation, based on the above arguments we can use a combinatorial procedure to detect following subfamily of inequalities (16).

Proposition 6. *Let C be a connected subgraph of G with $|V(C)| > b$, and let $\mathcal{T}(C)$ be the set of all spanning trees of C, and assume that one unit of flow is sent from a chosen root $l\in V(C)$ to all other $v\in V(C), v\ne l$ along the edges of T. Let a^l_v be the sum of flows sent into the vertex v and out of v. Then, the following Benders inequalities*

$$\sum_{v\in V(C)} a^l_v x_v \ge |V(C)| - b \qquad T\in\mathcal{T}(C),\quad l\in V(C) \tag{18}$$

are valid for model (3).

Proof. We first observe that inequalities (16) can be rewritten as

$$\sum_{l\in V}\sum_{vw\in A}\tilde{\beta}^l_{vw}(x_v + x_w) \ge \sum_{l\in V}\tilde{\gamma}_l - b. \tag{19}$$

Following the discussion from above, we then choose a root $l\in V(C)$, calculate the coefficients $\tilde{\beta}^l$, and set

$$a^l_v = \sum_{vw\in\delta^+(v)\cup\delta^-(v)}\tilde{\beta}^l_{vw}.$$

Recall that the value of $\tilde{\gamma}_l$ is $|V(C)|$ for the chosen l and that all $\tilde{\beta}^{l'}$ and $\tilde{\gamma}_{l'}$ are zero for $l'\ne l$. □

Notice that the Benders inequalities (18) can be tightened as in (10).

3.2. Another Bilevel Point of View: Degree Inequalities

Let $C\in\mathcal{C}$ be a connected subgraph of G such that $|V(C)| > b$. The minimum number $q(C)$ of components into which C has to fall apart, so that each resulting component contains no more than b vertices, is given as

$$q(C) = \left\lceil \frac{|V(C)| - \sum_{v \in V(C)} x_v}{b} \right\rceil. \tag{20}$$

Hence, from an alternative bilevel perspective, we could see this as a Stackleberg game; the leader interdicts some vertices, and, for each connected subgraph C such that $|V(C)| > b$, the follower calculates the number of connected components in the interdicted graph. If the number of components is smaller than $q(C)$, then the solution of the leader is infeasible. Let $\Psi_C(\boldsymbol{x})$ be the number of connected components of subgraph C in the interdicted graph. The latter condition can be imposed as the following constraint:

$$\Psi_C(\boldsymbol{x}) \geq q(C) \quad C \in \mathcal{C}, |V(C)| > b.$$

In Furini et al. (2019; equation (24)), we showed that the condition for a generic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to be partitioned into at least κ nonempty components by interdicting vertices can be expressed by the following exponential family of inequalities:

$$\sum_{uv \in \mathcal{E}(S)} (1 - x_u - x_v) \leq |\mathcal{V}| - \sum_{v \in \mathcal{V}} x_v - \kappa \quad S \in \mathcal{S}, \tag{21}$$

where $\mathcal{S}$ denotes the set of all cycle-free spanning subgraphs of $\mathcal{G}$, S is one such subgraph, $\mathcal{E}(S)$ is its edge set, and $\kappa \geq 2$. Hence, we can derive a new family of valid inequalities for the CVSP by applying this result to any connected subgraph $C \in \mathcal{C}$ with $|V(C)| > b$ and by replacing in (21) the constant term κ with (20). In addition, we restrict ourselves to spanning trees $T \in \mathcal{T}(C)$:

$$\sum_{uv \in E(T)} (1 - x_u - x_v) \leq |V(T)| - \sum_{v \in V(T)} x_v - \left\lceil \frac{|V(T)| - \sum_{v \in V(T)} x_v}{b} \right\rceil \quad T \in \mathcal{T}(C). \tag{22}$$

After removing the rounding and using $|E(T)| = |V(T)| - 1$, we obtain the following result:

Proposition 7. *Let C be a connected subgraph of G with* $|V(C)| > b$, *and let* $\mathcal{T}(C)$ *be the set of all spanning trees of* C, and *then the following degree inequalities,*

$$\sum_{v \in V(T)} [b(\deg_T(v) - 1) + 1] x_v \geq |V(T)| - b \quad T \in \mathcal{T}(C), \tag{23}$$

are valid for model (3).

Notice that, according to Furini et al. (2019, proposition 6), constraints (21) should be imposed for each acyclic spanning subgraph of C. In the context of the current paper, however, it is correct to consider only spanning trees of C, because if there is a disconnected acyclic spanning subgraph (i.e., a forest) violating the constraint, we would add the corresponding constraint for each tree of the forest as well. Furthermore, for (23), the right-hand side is always positive, and all coefficients next to the vertices are nonnegative so that the constraints can be tightened as in (10).

3.3. Comparison Between Component, Degree, and Benders Inequalities

By comparing component inequalities (12) and degree inequalities (23), we observe that the latter are obtained by selecting a tree T spanning C and by setting

$$M_v = b\left(\deg_T(v) - 1\right) + 1, \qquad v \in V(T).$$

Despite the fact that both families of inequalities are associated with trees, where each vertex v in the selected tree appears with a coefficient M_v, the values of these coefficients differ in the two cases. An example given in Figure 4(a) for $b = 2$ shows that, when these inequalities are imposed for the same $C \in \mathcal{C}$, the two inequalities do not dominate each other (notice that the inequalities are always tightened as in (10) when possible). For the given example, the component inequality and, respectively, the degree inequality are given as:

$$x_1 + 3x_2 + 4x_3 + 3x_4 + x_5 + x_6 + x_7 \geq 5$$
$$x_1 + 5x_2 + 3x_3 + 5x_4 + x_5 + x_6 + x_7 \geq 5.$$

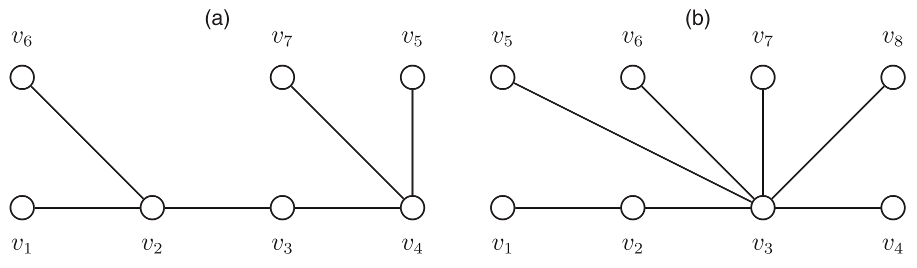
For this example, the Benders inequalities (with the root being any nonleaf vertex, e.g., $l = v_2$) are dominated by component and degree inequalities and read as

$$x_1 + 5x_2 + 5x_3 + 5x_4 + x_5 + x_6 + x_7 \geq 5.$$

However, another example depicted in Figure 4(b) for $b = 3$ shows a case where, when imposed for the same $C \in \mathcal{C}$, Benders inequalities dominate degree inequalities. The Benders inequality (with the root $l = v_3$) and, respectively, the degree inequality are

$$x_1 + 3x_2 + 5x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \geq 5$$
$$x_1 + 4x_2 + 5x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \geq 5.$$

Figure 4. Two Examples Demonstrating Relationships Between Studied Inequalities: a) $b = 2$, b) $b = 3$



From the two reported examples, we can conclude that in general there is no dominance between Benders inequalities and degree inequalities when imposed for the same $C \in \mathcal{C}$.

The following proposition determines the relative strength of the Benders inequalities (18) with respect to the component inequalities (12):

Proposition 8. *For a given connected subgraph C and its spanning tree $T \in \mathcal{T}(C)$, Benders inequalities (18) are dominated by the component inequalities (12).*

Proof. We prove this result by showing that, for each vertex $v \in V(T)$, the coefficient a_v in the Benders inequality is not smaller than the coefficient M_v in the component inequality. Let $|V(T)| = \rho$. When v is a leaf, it is trivial. If v is the root l, $a_v = \rho - 1$, whereas $M_v \leq \rho - 1$. If v is neither the root l nor a leaf, by removing v we partition T in $\deg(v)$ components: a component containing l and $\deg(v) - 1$ components not containing l. Let these components include κ vertices in total. We have $a_v = 2\kappa + 1$. If the largest component of T after removing v is the one including l, $M_v = \rho - (\rho - (\kappa + 1)) = \kappa + 1$, and hence, $a_v > M_v$. If the largest component of T after removing v does not include l, let $p \leq \kappa$ be its cardinality; we have $M_v = \rho - p$. Having $M_v > a_v$ would imply $\rho > 2\kappa + 1 + p$. But this would imply that the component including l has cardinality $\rho - \kappa - 1 > 2\kappa + 1 + p - \kappa - 1 = \kappa + p$, which is a contradiction since we assumed that the largest component has cardinality p. □

3.4. Cover Inequalities

These inequalities exploit the concept of connectivity. Let $W \subseteq V$ be a subset such that the induced subgraph $G[W]$ is r-vertex-connected (i.e., at least r vertices have to be removed to disconnect W) and $|W| > b$. Then, the following inequalities, which can be derived from the corresponding block-invariant inequalities proposed by Borndörfer et al. (1998), are valid

$$\sum_{v \in W} x_v \geq \min\Big\{|W| - b\,,\, r\Big\}. \tag{24}$$

This exponential-size family of constraints is referred to as *cover inequalities* in the remainder of this paper.

3.5. Star Inequalities

Additional sets of valid inequalities, derived from the corresponding block-invariant inequalities proposed by Borndörfer et al. (1998), can be exploited in order to strengthen the formulations of the previous sections. The following polynomial-size family of constraints is referred to as *star inequalities* in the remainder of this paper:

$$\sum_{w \in N(v)} x_w \geq (|N(v)| + 1 - b)(1 - x_v) \quad v \in V,\ |N(v)| \geq b. \tag{25}$$

Constraint (25) imposes that for each vertex $v \in V$ having a degree larger than or equal to the capacity, if vertex v is not interdicted, then at least $|N(v)| + 1 - b$ of its adjacent vertices have to be interdicted. After some rewriting, it is not difficult to see that the star inequalities are a special case of the tightened version of the component inequalities (12) imposed for the stars centered at $v \in V$, with $|N(v)| \geq b$. Furthermore, for the sufficiently large value of r ($r > |N(v)| - b$), we notice that cover inequalities (24) dominate star inequalities. Indeed, by setting $W = N(v) \cup \{v\}$, the cover inequality is reduced to $\Sigma_{v \in N(v)} x_v \geq |N(v)| + 1 - b$, which dominates (25).

3.6. Precedence Constraints

Finally, we can impose some precedence conditions between the interdiction of the vertices, as in Borndörfer et al. (1998). When the neighborhood of a vertex w is strictly included in the neighborhood of a vertex v, we can impose that the vertex w can be interdicted only if vertex v is interdicted. Indeed, any feasible solution where $x_w = 1$ and $x_v = 0$ can be transformed to a feasible solution of the same cost where $x_w = 0$ and $x_v = 1$. These precedence conditions can be stated as

$$x_w \leq x_v \qquad v, w \in V,\ N(w)\backslash\{v\} \subset N(v)\backslash\{w\}. \tag{26}$$

In addition, when two vertices share the same neighborhood, we can impose an order in the interdiction of the two vertices. In this case, we remove the symmetries with constraints (27) by imposing that the vertex with the lowest index can be interdicted only after the vertex with the largest index is interdicted. These additional precedence conditions are stated as

$$x_w \leq x_v \qquad v, w \in V,\ w < v,\ N(w)\backslash\{v\} = N(v)\backslash\{w\}. \tag{27}$$

These two polynomial-size families of constraints are referred to as *precedence constraints* in the remainder of this paper. Observe that at least one optimal solution exists that satisfies these constraints, whereas many (equivalent) feasible solutions can be cut off by imposing (26) and (27).

4. Separation Routines

This section describes separation strategies for the presented inequalities. All inequalities we propose (with the exception of cover inequalities) are given for a specific tree associated with a connected component. Trees are constructed during the detection of connected components, which can be performed by depth-first search (DFS) or breadth-first search (BFS). By construction, trees obtained through BFS define inequalities where one vertex (having a large degree) receives a large coefficient, and the other vertices receive a small coefficient. After tightening, these constraints tend to be computationally more effective than the corresponding constraints where the initial trees are detected by DFS.

4.1. Separation of Degree Inequalities

Given an integer solution $\boldsymbol{x}^*$, the separation problem is reduced to finding a connected subgraph C of the interdicted graph G^* such that $|V(C)| > b$, yielding a violation $|V(C)| - b$ of constraints (23). The most violated inequality is obtained by choosing the (element-wise) maximal subgraph C. However, our inequalities (23) are not associated with connected subgraphs but with subtrees contained in the subgraphs, whose number can be exponential in the size of each subgraph.

So, during the separation procedure, a tree for each connected subgraph of the interdicted graph that exceeds the capacity is built by means of a BFS, where the edges are processed in an arbitrary order. As soon as the size of the tree under construction is larger than b, and for each edge later on added to the tree, an inequality is defined and included in the formulation. So at the end of the procedure, for each connected subgraph C (of the interdicted graph) that exceeds the capacity b, $|V(C)| - b$ inequalities of type (23) are added to the model.

Observation 2. For integer solutions, the exact separation of degree inequalities (23) can be performed in polynomial time.

Given a fractional solution $\boldsymbol{x}^*$, after rewriting (23), we can see that checking whether a violated constraint exists is equivalent to finding a subtree T that maximizes the following function

$$\sum_{vw\in E(T)} (1 - x_v^* - x_w^*) - \sum_{v\in V(T)} (1 - x_v^*)\left(1 - \frac{1}{b}\right). \tag{28}$$

If the obtained value is positive, a violated constraint (23) is added to the model. The above separation problem can be formulated as the following IP:

$$\max\left\{\sum_{v\in V} w_v\, y_v - \sum_{vw\in E} w_{vw}\, z_{vw} : \ (z, y) \text{ is a subtree of } G\right\} \tag{29}$$

with $w_{vw} = x_v^* + x_w^*$, $vw \in E$ and $w_v = \frac{1}{b} + (1 - \frac{1}{b})x_v^*$, $v \in V$. This problem can be seen as an instance of the prize-collecting Steiner tree problem (PCSTP). To solve it, one can use an IP, a specialized algorithm for the PCSTP (see, e.g., the one proposed by Leitner et al. 2018), or a heuristic procedure.

We propose a heuristic procedure that builds a tree starting from the edge with the largest weight, where the weight of an edge is defined according to the contribution of the edge itself and of its end points to (28). Iteratively, the procedure adds edge-vertex pairs to the current tree according to an arbitrary order, as long as edge-vertex pairs with a positive contribution can be found. The contribution for adding a vertex v and an edge vw is again defined as in (28). When the procedure stops, if the tree has a positive weight, then a violated inequality has been detected.

4.2. Separation of Component Inequalities

For integer solutions, any tree T that is contained in the interdicted graph and whose vertex set $V(T)$ exceeds the capacity produces a violation $|V(T)| - b$ of constraints (12). In this case, however, computing the coefficients of each variable x_v, $v \in T$ requires us to run a BFS procedure with vertex v as a root. So, adding an inequality for each intermediate tree (i.e., trees that do not span a maximal connected subgraph) can be time consuming. For this reason, when separating inequalities (12), we only consider a spanning tree for each maximal subgraph C in the interdicted graph that exceeds the capacity b. In this case as well, the tree is built according to an arbitrary order of the vertices in $V(C)$.

Observation 3. For integer solutions, the exact separation of component inequalities (12) can be performed in polynomial time.

No separation is performed for fractional solutions, for which a well-defined associated optimization problem is lacking. Although it is always possible to define the separation problem by means of a MIP, the use of a general-purpose solver for solving the latter would be inefficient when embedded within a branch-and-cut scheme.

4.3. Separation of Benders Inequalities

Benders inequalities are separated for integer solutions, as discussed in Section 3.1.2; for each connected component C of the interdicted graph whose vertex set $V(C)$ exceeds the capacity, a vertex of maximum degree l is chosen as a root (this choice produces lower values of the coefficients in the Benders inequality). A spanning tree rooted at l is constructed by BFS (as this choice produces lower values of the coefficients in the Benders inequality), where vertices are sorted according to an arbitrary order, and the corresponding Benders inequality is defined.

Observation 4. For integer solutions, the exact separation of Benders inequalities (18) can be performed in $O(|E|)$ time. Fractional solutions of the master problem can be separated in polynomial time by solving the associated LP (15).

4.4. Separation of Cover Inequalities

Given a connected subgraph C of the interdicted graph with $|V(C)| > b$, inequalities (24) impose either to disconnect the set (by removing at least r vertices, when the component is r-connected) or to remove a number of vertices to reduce the cardinality of the vertex set to b. Because for each connected subgraph $r \geq 1$, if there exists a violated inequality (24), then there is one with $r = 1$. For integer solutions it is hence enough to find a connected subgraph C of the interdicted graph with $|V(C)| > b$.

Observation 5. For integer solutions, the exact separation of cover inequalities (24) can be performed in $O(|E|)$ time.

However, in order to separate strong inequalities, we have to increase the value of r and to search for r-connected components of cardinality $b + r$. There are different options for computing the vertex-connectivity r of a connected subgraph:

1. The vertex-connectivity r of a connected subgraph C can be obtained in polynomial time by maximum-flow computations. Even though this procedure runs in polynomial time, it can be time consuming in dense graphs.
2. Borndörfer et al. (1998) proposed a greedy procedure to possibly detect biconnected components and so to derive inequalities (24) with $r = 2$.
3. Computing biconnected components in a connected undirected graph can be performed in linear time with the sequential algorithm proposed by Hopcroft and Tarjan (1973) during the execution of a DFS.

This procedure can be used to detect whether a component is (at least) biconnected and to derive inequalities (24) with $r = 2$.

4.5. Separation of Bin-Packing Inequalities

Inequalities (3b) are separated for integer solutions only. Given an integer solution $\boldsymbol{x}^*$ and the associated interdicted graph G^*, defined by the set of interdicted vertices $V(\boldsymbol{x}^*)$, the bin-packing problem instance associated with the connected components of the latter graph is defined and solved. If the optimal solution to the bin-packing problem uses more than k bins, an inequality (3b) is defined for $W = V \setminus V(\boldsymbol{x}^*)$.

Observation 6. The exact separation of *bin-packing inequalities* (3b) *is* $\mathcal{NP}$-hard.

5. Extension to the MinMaxC

Although not the main focus of this paper, in this section we discuss how to extend some of the developed ideas to the solution of the MinMaxC (Shen et al. 2012), that is, the problem of disconnecting a graph $G = (V, E)$ by deleting a subset of no more than $B < |V|$ vertices, while minimizing the maximum cardinality of a connected component in the residual graph. Notice that Shen et al. (2012) considered a second objective; among all equivalent solutions, the second objective is to minimize the number of deleted vertices. In order to properly consider this hierarchy of objectives, a lexicographic optimization approach is advisable. However, this would move the analysis far from the scope of the present paper, which is to investigate how to disconnect a graph by vertex removal, and hence, we do not consider the second objective.

A first compact formulation of the MinMaxC is obtained by adapting the compact model (1a)–(1e) presented in Section 2, where in addition to the binary variables ξ_v^i taking value 1 if vertex v belongs to the shore V_i, a continuous variable λ denotes the cardinality of the largest component. The model reads

$$\min \quad \lambda \tag{30a}$$

$$\lambda \geq \sum_{v \in V} \xi_v^i \qquad i \in K \tag{30b}$$

$$|V| - \sum_{i \in K} \sum_{v \in V} \xi_v^i \leq B \tag{30c}$$

$$(1b) - (1c)$$

$$\xi_v^i \in \{0, 1\} \qquad i \in K, \ v \in V, \tag{30d}$$

where $K = \{1, \ldots, |V| - B\}$ is the index set of possible shores, constraint (30b) imposes that λ is at least the cardinality of each (and hence, of the largest) shore, and constraint (30c) imposes the budget constraint. The model can be strengthened by introducing extra variables and by replacing (1b)–(1c) with clique inequalities (2a)–(2b), as explained in Section 2.

An alternative compact model is the one considered by Shen et al. (2012), which we adapt for completeness to the notation and variables of the present paper. The model considers interdiction variables x_v, $v \in V$ and the $\boldsymbol{\sigma}$ variables (introduced in Section 3.1.2) and reads

$$\min_{\boldsymbol{x} \in \{0,1\}^{|V|}} \ \lambda \tag{31a}$$

$$\sum_{v \in V} x_v \leq B \tag{31b}$$

$$(13b) - (13d).$$

5.1. Extension of the Bilevel Approach

By considering the usual binary interdiction variables x_v for $v \in V$, the *leader problem* in the space of $\boldsymbol{x}$ variables reads

$$\min \quad \lambda \tag{32a}$$

$$\sum_{v \in V} x_v \leq B \tag{32b}$$

$$\Phi(\boldsymbol{x}) \leq \lambda \tag{32c}$$

$$x_v \in \{0, 1\} \qquad v \in V, \tag{32d}$$

where $\Phi(\boldsymbol{x})$ denotes the optimal solution value of the follower subproblem for a given vector $\boldsymbol{x}$. By a derivation equivalent to the one in Section 3.1, we have that constraint (32c), for each connected subgraph C of G with $|V(C)| > b$, can be replaced by the following inequalities:

$$\sum_{v \in V(C)} \Big(|V(C)| - \text{comp}_T(v)\Big) x_v \geq |V(C)| - \lambda \qquad T \in \mathcal{T}(C), \tag{33}$$

where $\mathcal{T}(C)$ is the set of all spanning trees of C. These inequalities for the MinMaxC are the counterparts of inequalities (12).

Similarly, a derivation equivalent to the one in Section 3.1.2 allows us to write Benders inequalities that are the counterparts of (18).

Observation 7. We remark that, among the inequalities discussed in this paper, only component inequalities, Benders inequalities, and the precedence conditions can be extended to the MinMaxC while keeping linear formulations. Indeed:

- degree inequalities remain valid at the cost of introducing nonlinearities;
- cover inequalities are not valid, since they are defined for each subset of vertices whose cardinality violates the capacity b, which is substituted by a variable (λ) in the MinMaxC;
- star inequalities are not valid, since they are defined for each vertex with a degree greater or equal to the capacity b, which is substituted by a variable (λ) in the MinMaxC.

6. Computational Results

In this section, we present the results of the computational experiments with the aim of assessing the performance of the mathematical models described in the previous sections. We implemented a branch-and-cut framework based on formulation (3), strengthened by constraints (4) for appropriate k, which has a polynomial number of variables and an exponential number of constraints, namely, the bin-packing inequalities (3b) and their feasibility counterpart (4). Although correct, this formulation asks to solve a $\mathcal{NP}$-hard problem to check feasibility of any integer point of the branch-and-cut tree. Hence, this basic model is enhanced by four different families of constraints for which we have developed polynomial separation algorithms for integer points: (i) component inequalities (12), (ii) degree inequalities (23), (iii) Benders inequalities (18), and (iv) cover inequalities (24).

The first goal of this computational section is to assess the relative computational performance of each family of inequalities and their computational interaction when embedded in a branch-and-cut algorithm. Based on the results of these experiments, which are presented in Section 6.1, the best (and hence, default) configuration of our newly developed branch-and-cut algorithm is determined. The latter is then used in a second set of experiments (cf. Section 6.2) in which the performance of the branch-and-cut algorithm is compared with the state-of-the-art exact methods for the CVSP present in the literature.

6.1. Benchmark Instances

We tested the same four sets of benchmark instances considered by Bastubbe and Lübbecke (2020). The first two sets of graphs are obtained from matrix decomposition problems, as discussed in Section 1. The considered matrices are the constraint matrices of several Netlib (Gay 1985) and MIPLIB (Koch et al. 2011) instances. There are 55 graphs constituting the `Netlib` data set, with the number of vertices ranging from 51 to 500. The `MIPLIB` data set contains 37 graphs, whose number of vertices ranges from 19 to 490.[2] The other two sets are 40 instances from the second DIMACS challenge (Johnson and Trick 1996) and 50 Random graphs representing hypergraphs generated by Bastubbe and Lübbecke (2020). For the `DIMACS` set, the number of vertices ranges from 23 to 496, whereas for the `Random` set, the number of vertices ranges from 68 to 164. Because Bastubbe and Lübbecke (2020) considered hypergraphs, we adapted the latter instances to our case by defining a clique for each hyperedge. In summary, our computational study is conducted on a set of 182 graphs with different structures and densities; the exact number of vertices and edges of these graphs are reported in the tables provided in the Appendix.

As far as the values of k (maximum number of shores) and b (maximum capacity of each shore) are concerned, we borrow the same setting used by Bastubbe and Lübbecke (2020), in which $k \in \{4, 8, 12, 16, 24, 32, 64, 218, 256\}$ and $b = \lceil |V|/k \rceil$. In our analysis, we do not consider instances for which the value of b equals to 1, because, in these cases, the problem reduces to the maximum stable set problem. The values of k are clustered into three major categories: (i) `small` $\rightarrow k \in \{4, 8, \ \ 12\}$, (ii) `medium` $\rightarrow k \in \{16, 24, 32\}$, and (iii) `large` $\rightarrow k \in \{64, 128, 256\}$. In summary, using nine different values of k and 182 graphs, we obtained a testbed of 1,397 different instances. In the remainder of this article, aggregated results for small, medium, and large values of k are reported (whereas the detailed results can be found in the appendix).

6.2. Computational Environment

All of the reported experiments are performed on a computer equipped with an i7 processor clocked at 3.20 GHz and 64 GB of RAM under the Linux operating system. We use the CPLEX 12.7.1 MIP framework to implement our branch-and-cut algorithms and to solve the compact formulations for which we report the results. CPLEX is run in single-threaded mode, and all CPLEX parameters are set to their default values. A time limit of 30 minutes is set for each tested instance.

6.3. Determining the Best Configuration of the Branch-and-Cut Algorithm

As previously discussed, a basic valid formulation for the CVSP is given by (3a)–(4). Each one of the four families of inequalities, the component inequalities (12), the degree inequalities (23), the Benders inequalities (18), and the cover inequalities (24), can be used to enhance this basic model. These families of inequalities are composed by an exponential number of constraints, and thus they are separated within the branching tree for integer solutions. In addition, they can be separated for fractional solutions in order to strengthen the dual bounds and (potentially) improve the computational convergence. In the following, we report results for the following branch-and-cut configurations:

- C: the branch-and-cut separating the component inequalities (12), tightened as in (10), for integer solutions;
- D: the branch-and-cut separating the degree inequalities (23), tightened as in (10), for integer solutions;
- B: the branch-and-cut separating the Benders inequalities (18), tightened as in (10), for integer solutions;
- CV: the branch-and-cut separating the cover inequalities (24) for integer solutions, via the detection of biconnected components. Among the three separation procedures given in Section 4.4, after extensive preliminary computational tests, we determined that the best-performing way is via the application of the algorithm proposed by Hopcroft and Tarjan (1973).

Concerning the separation of fractional points for (12), this is not performed because we could not identify a well-defined associated optimization problem (see Section 4.2). Regarding the constraints (23), we tested the separation of fractional points either in an exact or heuristic fashion. Although improvements were obtained for some specific instances, on average the computational performance was worsened; additional computational effort was needed to solve the LP relaxation at the branching nodes because of a large number of violated cuts detected. Therefore, we do not report the results for this particular setting. Similar considerations apply to the separation at fractional points of (18), which can be performed by solving a LP, and (24), which was performed by means of the greedy procedure proposed by Borndörfer et al. (1998). Also in these cases, the average computational performance was worsened.

In all of the configurations of the branch-and-cut algorithm, bin-packing inequalities (3b) are separated at integer points only when no other violated inequalities have been detected. This guarantees that the resulting connected components in the interdicted graph can be packed into k shores of capacity b. The associated bin-packing instances were not challenging, and hence, a standard MIP formulation for the bin-packing problem was used, with CPLEX as the off-the-shelf solver. Indeed, the performance of our configurations was not affected by the efficiency of the latter separation procedure, which is why we refrained from developing a tailored algorithm for the bin-packing problem.

In Sections 3.5 and 3.6, we presented two additional families of inequalities that are polynomial in number: (i) the *stars inequalities* (25) and (ii) the *precedence constraints* (26) and (27). Thanks to extensive preliminary experiments, we observed that these inequalities are useful to strengthen the formulation and to speed up the computational convergence. For this reason, they are always included into our models.

In Table 1, we present the results of the computational experiments performed with the previously discussed configurations of the branch-and-cut algorithm. Specifically, we report the performance of five different configurations: C, D, B, and CV, that is, the four basic variants separating the component inequalities, degree inequalities, Benders inequalities, and cover inequalities for integer solutions, respectively. In addition, we report the performance of C + CV, which corresponds to the separation of the component inequalities and of the cover inequalities for each integer

Table 1. Performance Comparison for Different Configurations of Our Branch-and-Cut Algorithm

k		C	D	B	CV	C+CV
small	Opt. (out of 546)	294	226	258	219	305
	Avg time	66.52	89.38	74.72	78.92	85.12
	Avg nodes	87,355	12,375	70,370	29,922	75,221
medium	Opt. (out of 540)	382	354	363	356	386
	Avg time	54.37	47.18	47.33	32.97	45.03
	Avg nodes	83,435	21,368	65,391	32,864	54,991
large	Opt. (out of 311)	249	248	249	248	249
	Avg time	35.72	33.08	40.71	29.41	36.70
	Avg nodes	77,583	74,830	79,778	71,280	75,227
	Total opt. (out of 1397)	925	828	870	823	940
	Avg time	53.21	54.47	53.55	44.13	55.83
	Avg nodes	83,106	34,926	70,985	43,658	66,915

solution. Indeed, whereas the first three families of inequalities have the same structure (i.e., are all associated with trees of the graph G), the latter family is structurally different. Hence, we tried to combine cover inequalities with the other inequalities, and we report the results for the best configuration, that is, C + CV. Table 1 is horizontally divided in four sections, the first three reporting aggregated results for the three classes of instances by the choice of parameter k, that is, small $\rightarrow k \in \{4, 8, 12\}$, medium $\rightarrow k \in \{16, 24, 32\}$, and large $\rightarrow k \in \{64, 128, 256\}$. These three sections report, for each configuration of the branch-and-cut algorithm, the total number of instances solved to proven optimality (rows "Opt"), the average computing time in seconds (rows "Avg time"), and the average number of nodes explored by the branching tree (rows "Avg nodes"). The values "Avg time" and "Avg nodes" are computed over the instances solved to optimality by the respective method. Finally, the fourth section of the table reports the same information for the entire set of the 1,397 instances. All of the averages are computed separately for each configuration by considering only the instances solved to proven optimality by that configuration.

As far as the comparison of the four basic variants (C, D, B, and CV) is concerned, from the table it emerges that with 925 instances solved to proven optimality, C is the best configuration, followed by B, which is able to solve 870 instances, D, which is able to solve 828 instances, and CV, which only solves 823 instances. A similar pattern can also be seen for the three different categories of values for k. The number of instances solved to proven optimality and the computational times suggest that the class small is the hardest to solve for all of our branch-and-cut algorithms.

Separating both the component and the cover inequalities pays off in terms of the number of instances solved to proven optimality; precisely, C + CV is able to solve 940 instances (15 more than C alone). The average number of branch-and-bound nodes suggests that CV explores on average fewer nodes than C, especially for small values of k. By combining the two families of inequalities, on average the number of explored nodes is reduced compared with C alone.

6.4. Comparison with State-of-the-Art Solution Methods

In this section, we compare the performances of our best branch-and-cut configuration identified in the previous section (i.e., the configuration C + CV), with the state-of-the-art exact methods available in the literature for the CVSP:

- BP: the branch-and-price algorithm proposed by Bastubbe and Lübbecke (2020), and
- Cplex: the direct solution of the compact model (1a), (1d)–(2b) via CPLEX, a state-of-the-art commercial MIP solver.

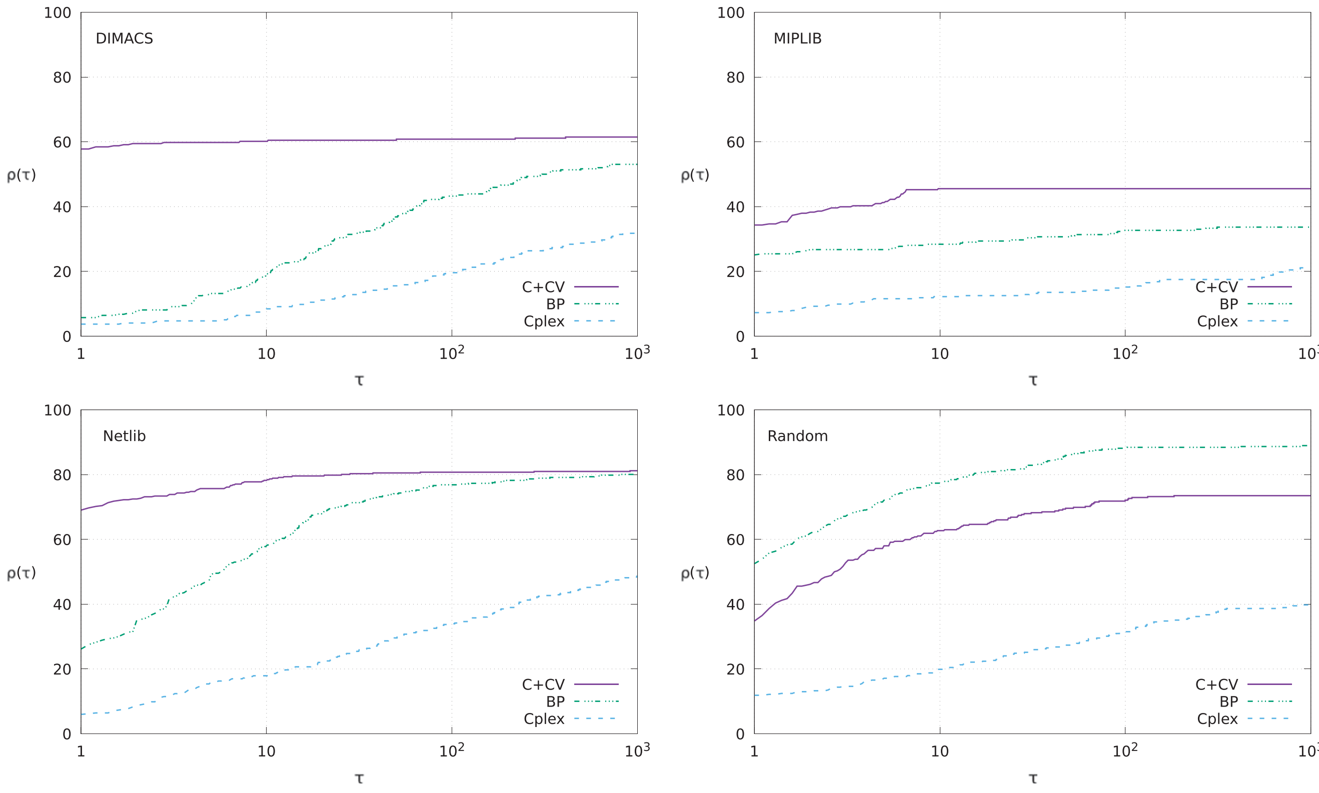
For these tests, we used the same testbed of 1,397 instances proposed by Bastubbe and Lübbecke (2020) and described in the previous section. We recall that, in our analysis, we do not consider instances for which the value of b results equal to 1 and that a time limit of 30 minutes is set for each run as for the experiments reported by Bastubbe and Lübbecke (2020). The results of BP are directly borrowed from the tables reported by these authors in their work (the performance of our machine is comparable with the machine used for their experiments, which is equipped with a i7 processor clocked at 3.40 GHz).

The information reported in Table 2 summarizes the results of this second set of tests. The table follows the same structure given in Table 1, but in addition to the disaggregation concerning the category of k, we also report disaggregated information for each class of instances. All of the averages are computed separately for each method by considering only the instances solved to proven optimality by that method. We discuss now the results for each class of instances separately.

Table 2. Performance Comparison Between Cplex, BP, and Our Best Branch-and-Cut Algorithm (C + CV)

Class	k		Cplex	BP	C+CV
DIMACS	small	Opt. (out of 120)	73	59	66
		Avg time	213.87	164.37	45.57
	medium	Opt. (out of 117)	47	70	75
		Avg time	382.73	114.47	29.82
	large	Opt. (out of 59)	4	35	41
		Avg time	601.75	19.19	72.58
	Total Opt. (out of 296)		124	164	182
	Avg time		290.39	112.09	45.17
MIPLIB	small	Opt. (out of 111)	47	23	45
		Avg time	179.34	147.41	22.99
	medium	Opt. (out of 108)	22	43	45
		Avg time	278.15	186.35	2.83
	large	Opt. (out of 84)	4	43	48
		Avg time	180.73	182.70	95.31
	Total opt. (out of 303)		73	109	138
	Avg time		209.19	176.70	41.57
Netlib	small	Opt. (out of 165)	133	114	122
		Avg time	103.29	182.35	36.34
	medium	Opt. (out of 165)	91	141	134
		Avg time	301.08	66.15	30.52
	large	Opt. (out of 106)	13	96	98
		Avg time	589.42	40.33	15.95
	Total opt. (out of 436)		237	351	354
	Avg time		205.90	96.83	28.49
Random	small	Opt. (out of 150)	106	110	72
		Avg time	247.72	276.42	242.86
	medium	Opt. (out of 150)	60	150	132
		Avg time	554.37	43.92	82.78
	large	Opt. (out of 62)	2	62	62
		Avg time	1281.65	0.97	0.39
	Total opt. (out of 362)		168	322	266
	Avg time		369.55	115.07	106.91

Figure 5. (Color online) Performance Profiles by Class of Instances: DIMACS, MIPLIB, Netlib, and Random



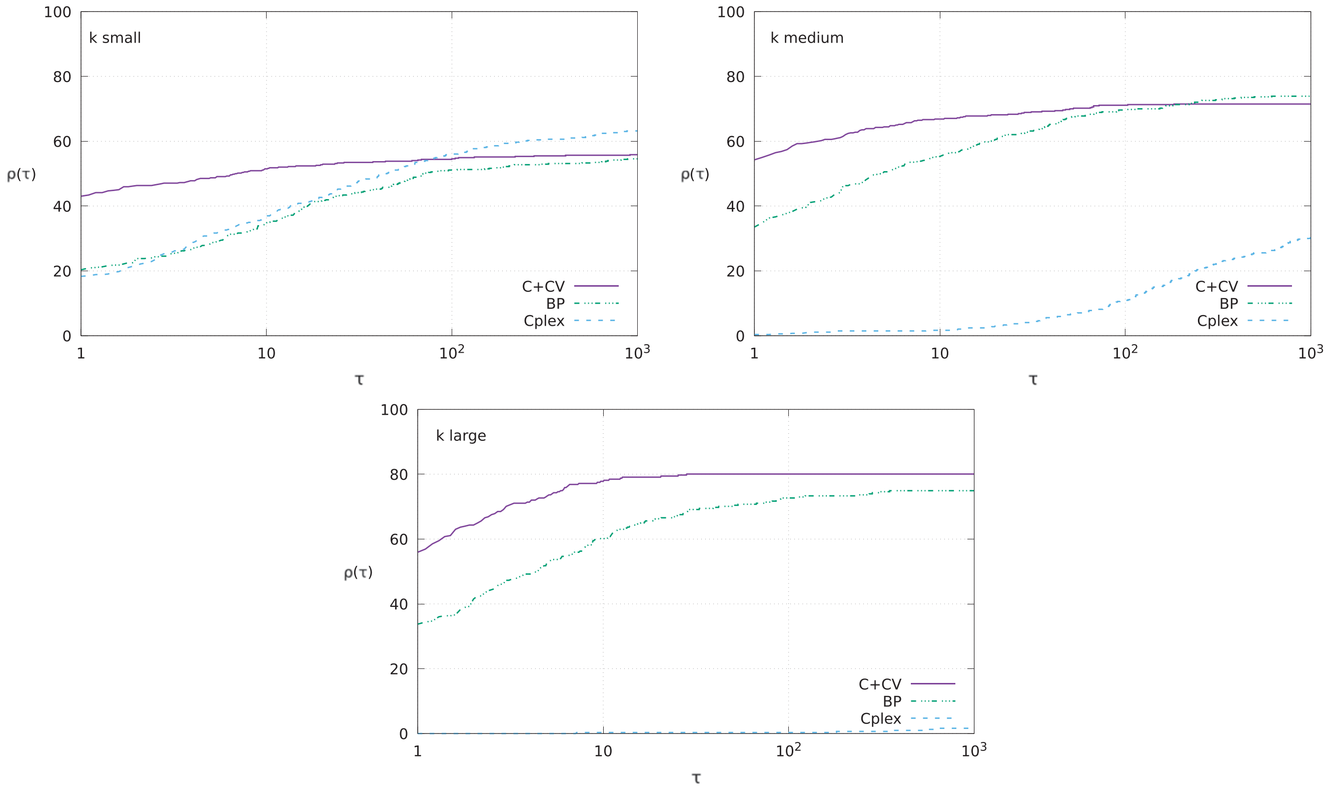
For 296 DIMACS instances, Cplex is able to solve 124 instances, BP 164 instances, and 182 instances. For 303 MIPLIB instances, Cplex is able to solve 73 instances, BP 109 instances, and C+CV 138 instances. For 436 Netlib instances, Cplex is able to solve 237 instances, BP 351 instances, and C+CV 354 instances. For the 362 Random instances, Cplex is able to solve 168 instances, BP 322 instances, and C+CV 266 instances.

Summarizing, in terms of the number of instances solved, our branch-and-cut algorithm C+CV outperforms both Cplex and BP for the DIMACS and MIPLIB classes of instances, and it has a performance comparable with that of BP for the Netlib instances, whereas it is outperformed by BP for the Random instances. For the category small of the k values, Cplex instead remains the best option. This is because of the small number of variables of the compact formulation, which linearly depends on k. For medium and large values of k, Cplex is largely dominated by BP and C+CV, which improve their performance for increasing values of k, thus showing a complementary performance with respect to Cplex. In particular, C+CV is the best option for all classes of instances when k is large.

Finally, performance profiles depicted in Figures 5 and 6 give a graphical representation of the relative performance of the three compared methods, that is, C+CV, BP, and Cplex. In Figure 5, the instances are gathered by class of instances, that is, Netlib, MIPLIB, DIMACS, and Random. In Figure 6, the instances are gathered by values of k, that is, small, medium, and large. As proposed by Dolan and Moré (2002), let s be any solution method; for each value of τ in the horizontal axis, the vertical axis $\rho_s(\tau)$ gives the percentage of instances for which the computing time of method s was not larger than τ times the time of the best-performing method. Notice that these values may sum up to a value larger than 100% if more than one algorithm is classified as the fastest for a specific instance (because of ties in the computing time), and they may sum up to a value smaller than 100% if there are instances that have not been solved by any method. All computing times smaller than 0.1 seconds were scaled to 0.1, which is the granularity of the profile. This way we avoid comparisons between tiny values, which would produce inaccurate conclusions. The curves originate 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-performing algorithm is graphically represented by the curve in the upper part of the respective figure. The horizontal axis is represented in logarithmic scale.

From Figure 5, it emerges that C+CV is the fastest exact method for around 60% of the DIMACS instances, whereas this is the case for the BP and Cplex

Figure 6. (Color online) Performance Profiles by Values of k: `small` $\rightarrow k \in \{4, 8, 12\}$, `medium` $\rightarrow k \in \{16, 24, 32\}$, and `large` $\rightarrow k \in \{64, 128, 256\}$



for only ≈10% of instances. Even by allowing larger computing times, C+CV outperforms BP and Cplex on this class of instances. For the MIPLIB instances, C+CV is the fastest exact method for approximately 35% of the instances, whereas this is true for BP (resp., Cplex) for approximately 25% (resp., approximately 10%) of the instances. By allowing larger computing times, C+CV outperforms BP and Cplex also on this class of instances. For the Netlib instances, C+CV is the fastest exact method for 70% of the instances, whereas this is true for BP (resp., Cplex) for less than 30% (resp., less than 10%) of the instances. Even by allowing larger computing times, the fraction of instances solved by C+CV is slightly larger than that of instances solved by BP. For the Random instances, BP is the fastest method for more than 50% of the instances, followed by C+CV (less than 40%) and Cplex (around 10%). BP is the best method for this class of instances. From Figure 5, it also emerges that the hardest set of instances are the MIPLIB ones, since only less than 50% of these instances can be solved to proven optimality by the best-performing method. Instead, more than 60% of the DIMACS, around 80% of the Netlib, and more than 90% of the Random instances, respectively, can be solved by the best exact method.

Figure 6, where instances are gathered by values of k, confirms that the hardest instances are the ones of category small, for which more than 60% of the instances can be solved to proven optimality by the best-considered exact method. C+CV is the fastest method for around 40% of the instances, whereas for BP and Cplex this is true for around 20% of the instances. However, by allowing larger computing times, Cplex is the best-performing method for this class. The situation is slightly improved for the category medium, where more than 70% of the instances can be solved to proven optimality by C+CV and BP. C+CV is the fastest method for around 55% of the instances, BP is the fastest for around 35% of the instances, and Cplex is completely outperformed. For large computing times, C+CV and BP have a similar performance. As far as the category large is concerned, approximately 80% of these instances can be solved by C+CV, which is also the fastest method for almost 60% of these instances, whereas this is true for BP for around 35% of the instances, and Cplex is completely outperformed. C+CV is the best-performing method for this class.

6.5. Computational Experiments on the MinMaxC

We conclude the section with a concise analysis of the computational results obtained for the MinMaxC. As discussed in Section 5, this is a closely related problem to the CVSP.

6.5.1. Benchmark Instances. For this problem, we considered the same classes of instances (`DIMACS`, `Netlib`, `MIPLIB`, and `Random`) described in the previous section. We fix the value of the budget B (i.e., the maximum number of vertices that can be removed) as a percentage of the number of vertices. More precisely, for each instance, we considered $B = \lceil p|V|\rceil$ with $p \in \{0.05, 0.1, 0.2\}$. This way, we obtained a testbed of 564 different instances. We also generated another set of 15 random instances (as described by Shen et al. 2012): five 20-, five 30-, and five 40-vertex instances. In this case, we tested all of the different values of B considered by Shen et al. (2012).

The computational environment is the same as described in the previous section. Also, in this case, a time limit of 30 minutes is set.

6.5.2. Performance Comparison. We compare the computational results obtained with the following exact methods:

1. `CPX`: the direct solution of compact model (30) where (1b) and (1c) are replaced by (2a) and (2b);
2. `SIG`: the direct solution of compact model (31).
3. `CC`: the branch-and-cut algorithm based on formulation (32a), (32b), (32d), and component inequalities (33);

We do not report results of the tests performed on the instances generated as described by Shen et al. (2012), because these instances turned out to be very easy for state-of-the-art MIP solvers (all of these instances are solved in short computing time by `CPX`). In addition, we do not report results for a branch-and-cut algorithm based on a formulation exploiting Benders inequalities, because the previous analysis showed that component inequalities lead to computationally more effective formulations.

Table 3 summarizes the results on the set of instances of classes `DIMACS`, `Netlib`, `MIPLIB`, and `Random`. We aggregated the results according to three different ranges of the number of vertices of the instances (up to 100 vertices, between 100 and 200 vertices, and more than 200 vertices). The first column of the table indicates the range, and the second column reports the percentage p of the vertices that can be removed. Then, for each solution method, we report the number of instances solved to optimality within the time limit (rows "Opt.") and the average time in seconds (rows "Avg time", computed over instances that are solved to optimality by the respective method) for each range and for each p. In the last two rows, we also report for each method the total number of solved instances and the average time.

Table 3. Performance Comparison Between `CC`, `CPX`, and `SIG` on Instances of `DIMACS`, `Netlib`, `MIPLIB`, and `Random` Classes

$\|V\|$	p		`CPX`	`SIG`	`CC`
$[-, 100]$	0.05	Opt. (out of 51)	45	39	45
		Avg time	260.04	385.65	19.10
	0.1	Opt. (out of 51)	38	31	32
		Avg time	326.14	315.92	138.77
	0.2	Opt. (out of 51)	32	30	29
		Avg time	485.52	303.77	85.27
$(100, 200]$	0.05	Opt. (out of 69)	15	11	44
		Avg time	798.00	409.03	135.92
	0.1	Opt. (out of 69)	11	10	36
		Avg time	905.83	445.88	144.22
	0.2	Opt. (out of 69)	9	12	31
		Avg time	899.18	203.88	58.88
$(200, -]$	0.05	Opt. (out of 62)	1	8	25
		Avg time	329.95	386.94	47.26
	0.1	Opt. (out of 62)	2	8	24
		Avg time	154.17	81.69	117.67
	0.2	Opt. (out of 62)	8	15	24
		Avg time	222.06	105.86	114.73
		Total opt. (out of 564)	161	164	290
		Avg time	447.66	309.08	94.93

These results show that `CPX` outperforms `CC` and `SIG` on small instances having up to 100 vertices. Indeed, `CPX` is able to solve 115 out of the 153 instances in range $[-, 100]$, whereas `CC` solves 106 instances and `SIG` solves 100 instances. On the other hand, as soon as the number of vertices increases, our branch-and-cut algorithm improves its performance, whereas `CPX` and `SIG` drastically get the worst results due to the increasing number of variables in the associated formulations. More precisely, out of the 207 instances in the range $(100, 200]$, `CC` is able to solve 111 instances, `CPX` solves 35 instances, and `SIG` solves 33 instances. Out of the 186 instances in the range $(200, -]$, CC is able to solve 73 instances, `CPX` solves 11 instances, and `SIG` solves 31 instances. For the last two ranges, we also observed that in many cases `CPX` and `SIG` failed because CPLEX does not even have the capability to load the MIP model due to the huge size of the latter. Overall, our branch-and-cut algorithm `CC` turns out to be the best method both in terms of solved instances (290 out of 564) and in terms of computational speed.

7. Conclusions

In this article, we studied the capacitated vertex separator problem in which a subset of vertices of minimum cardinality has to be removed from a given graph so that the size of each connected component in the remaining graph is bounded by b, and all components can be packed into k shores, each one containing no more than b vertices. For this hard problem with applications in network protection against the spread of a virus, detection of critical nodes in social and communication networks, networks analysis, and matrix decomposition, extended formulations have been well studied in the previous literature, but very little is known about solving the problem using a canonical IP formulation. The major drawback of the canonical IP formulation is that it requires solving an ($\mathcal{NP}$-hard) bin-packing problem in order to verify the feasibility of a solution. To improve the computational efficiency of the underlying IP formulation, we proposed three new families of valid inequalities that have been derived from the perspective of a two-player sequential game in which a leader removes the vertices, and a follower solves another combinatorial optimization problem that (partially) guarantees the feasibility of the solution. The effects of the introduced inequalities on the basic IP formulation have been studied from both the theoretical and computational perspectives. In addition, we also showed how to extend some of the developed ideas to tackle a related min-max problem, where, given a maximum number of vertices to remove, the objective is to minimize the cardinality of the largest connected component in the residual graph.

On a large benchmark set of the instances available in the current literature, we demonstrated that our new branch-and-cut approach is competitive with the state-of-the-art branch-and-price algorithm and a compact formulation proposed by Bastubbe and Lübbecke (2020). In particular, our approach computationally outperformed the branch-and-price algorithm from Bastubbe and Lübbecke (2020) for large values of the number of shores k and for structured graphs from various applications, whereas the latter had a better performance for random graphs and average values of the k parameter. Our computational analysis revealed that exact approaches for the capacitated vertex separator problem can tackle graphs with up to 500 vertices. Solving the problem for graphs with thousands of vertices is a relevant open problem for which heuristic and approximate methods should be considered as an interesting stream of research. The computational analysis has been further extended to the related min-max problem, where a variant of our new branch-and-cut algorithm outperformed two compact formulations.

Finally, we hope that this article raises the awareness on the importance and merit of bilevel optimization for solving difficult combinatorial optimization problems by modeling them as two-player Stackelberg games. Many vertex/edge deletion/insertion problems or graph partitioning problems could benefit from this new modeling paradigm. The same is true for problems that ask for finding the most central or most critical vertices/edges with respect to various centrality measures.

Acknowledgments

The authors are grateful to Michael Bastubbe and Marco Lübbecke for stimulating discussions on the capacitated vertex separator problem and for sharing detailed information about the computational experiments reported in their paper Bastubbe and Lübbecke (2020), and to the three anonymous referees and the associate editor for their comments that helped to significantly improve the quality of the paper.

Endnotes

[1] In case a single connected component results from the removal of S, the latter is not strictly a separator of G, which is defined as a subset of vertices, the removal of which disconnects the graph. However, with a slight abuse of notation, we still call it a separator in the remainder of the paper.

[2] The constraint matrices determining these graphs have been presolved and reduced by SCIP 3.2.0, with default settings by Bastubbe and Lübbecke (2020).

References

Albert R, Jeong H, Barabási AL (2000) Error and attack tolerance of complex networks. *Nature* 406(6794):378–382.

Baggio A, Carvalho M, Lodi A, Tramontani A (2021) Multilevel approaches for the critical node problem. *Oper. Res.* 69(2):1–23.

Baïou M, Barahona F (2016) Stackelberg bipartite vertex cover and the preflow algorithm. *Algorithmica* 74(3):1174–1183.

Bastubbe M, Lübbecke M (2020) A branch-and-price algorithm for capacitated hypergraph vertex separation. *Math. Programming Comput.* 12(1):39–68.

Bergner M, Caprara A, Ceselli A, Furini F, Lübbecke M, Malaguti E, Traversi E (2015) Automatic Dantzig–Wolfe reformulation of mixed integer programs. *Math. Programming* 149(1):391–424.

Borndörfer R, Ferreira C, Martin A (1998) Decomposing matrices into blocks. *SIAM J. Optim.* 9(1):236–269.

Borrero JS, Prokopyev OA, Sauré D (2019) Sequential interdiction with incomplete information and learning. *Oper. Res.* 67(1):72–89.

Brotcorne L, Labbé M, Marcotte P, Savard G (2008) Joint design and pricing on a network. *Oper. Res.* 56(5):1104–1115.

Brown G, Carlyle M, Salmerón J, Wood K (2006) Defending critical infrastructure. *INFORMS J. Appl. Anal.* 36(6):530–544.

Bui TN, Jones C (1992) Finding good approximate vertex and edge partitions is np-hard. *Inform. Process. Lett.* 42(3):153–159.

Casorrán C, Fortz B, Labbé M, Ordóñez F (2019) A study of general and security Stackelberg game formulations. *Eur. J. Oper. Res.* 278(3):855–868.

Cormican KJ, Morton DP, Wood RK (1998) Stochastic network interdiction. *Oper. Res.* 46(2):184–197.

Cornaz D, Furini F, Lacroix M, Malaguti E, Mahjoub AR, Martin S (2019) The vertex k-cut problem. *Discrete Optim.* 31:8–28.

de Souza C, Balas E (2005a) The vertex separator problem: a polyhedral investigation. *Math. Programming* 103(3):583–608.

de Souza C, Balas E (2005b) The vertex separator problem: algorithms and computations. *Math. Programming* 103(3):609–631.

Delorme M, Iori M, Martello S (2016) Bin packing and cutting stock problems: Mathematical models and exact algorithms. *Eur. J. Oper. Res.* 255(1):1–20.

Dempe S, Zemkoho AB (2013) The bilevel programming problem: reformulations, constraint qualifications and optimality conditions. *Math. Programming* 138(1-2):447–473.

Djidjev HN (2000) Partitioning planar graphs with vertex costs: Algorithms and applications. *Algorithmica* 28(1):51–75.

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. *Math. Programming* 91(2):201–213.

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

Fischetti M, Ljubić I, Monaci M, Sinnl M (2019) Interdiction games under monotonicity. *INFORMS J. Comput.* 31(2):390–410.

Fischetti M, Leitner M, Ljubić I, Luipersbeck M, Monaci M, Resch M, Salvagnin D, Sinnl M (2017) Thinning out Steiner trees: a node-based model for uniform edge costs. *Math. Programming Comput.* 9:203–229.

Fukuyama J (2006) NP-completeness of the planar separator problems. *J. Graph Algorithms Appl.* 10(2):317–328.

Furini F, Ljubić I, Malaguti E, Paronuzzi P (2019) On integer and bilevel formulations for the k-vertex cut problem. *Math. Programming Comput.* 12:133–164.

Garg N, Saran H, Vazirani V (1999) Finding separator cuts in planar graphs within twice the optimal. *SIAM J. Comput.* 29(1):159–179.

Gay DM (1985) Electronic mail distribution of linear programming test problems. *Math. Programming Soc. COAL Newsletter* 13:10–12.

Heath M, Ng E, Peyton B (1991) Parallel algorithms for sparse linear systems. *SIAM Rev.* 33(3):420–460.

Hopcroft J, Tarjan R (1973) Algorithm 447: efficient algorithms for graph manipulation. *Comm. ACM* 16(6):372–378.

Johnson DS, Trick MA (1996) Cliques, coloring, and satisfiability. *Second DIMACS Implementation Challenge, October 11-13, 1993*, vol. 26, American Mathematical Socety (Rutgers University, Piscataway, NJ).

Kleinert T, Labbé M, Plein F, Schmidt M (2020) Technical note–There's no free lunch: On the hardness of choosing a correct Big-M in bilevel optimization. *Oper. Res.* 68(6):1716–1721.

Koch T, Achterberg T, Andersen E, Bastert O, Berthold T, Bixby RE, Danna E, et al. (2011) Miplib 2010. *Math. Programming Comput.* 3(2):103–163.

Leitner M, Ljubić I, Luipersbeck M, Sinnl M (2018) A dual ascent-based branch-and-bound framework for the prize-collecting steiner tree and related problems. *INFORMS J. Comput.* 30(2): 402–420.

Lipton R, Tarjan R (1979) A separator theorem for planar graphs. *SIAM J. Appl. Math.* 36(2):177–189.

Lipton RJ, Tarjan RE (1977) Applications of a planar separator theorem. *18th Annual Sympos. Foundations Comput. Sci. (SFCS 1977 Providence, RI)*, 162–170.

Lozano L, Smith J (2017) A value-function-based exact approach for the bilevel mixed integer programming problem. *Oper. Res.* 65 (3):768–786.

Shen S, Smith JC (2012) Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks* 60(2):103–119.

Shen S, Smith JC, Goli R (2012) Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optim.* 9(3):172–188.

Tahernejad S, Ralphs T, DeNegre S (2020) A Branch-and-Cut Algorithm for Mixed Integer Bilevel Linear Optimization Problems and Its Implementation. *Math. Program. Comput.* 12:529–568.

Tao Z, Zhongqian F, Binghong W (2006) Epidemic dynamics on complex networks. *Progress Natl. Sci.* 16(5):452–457.

Watts DJ, Strogatz SH (1998) Collective dynamics of "small-world" networks. *Nature* 393(6684):440–442.

Wood RK (2011) Bilevel network interdiction models: Formulations and solutions. Cochran JJ, Cox LA, Keskinocak P, Kharoufeh JP, Smith JC, eds. *Wiley Encyclopedia of Operations Research and Management Science*.

Zachary WW (1977) An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* 33(4):452–473.

Fabio Furini is a research fellow at the Italian National Research Council (CNR). From 2013 to 2019, he was a maître de conférences (equivalent to associate professor) at LAMSADE laboratory of Université Paris-Dauphine. Fabio Furini is a researcher in the field of mathematical programming, and his core research project aims at studying and developing effective exact algorithms, based on reformulation and decomposition techniques, for mixed integer linear and nonlinear programs.

Ivana Ljubić is full professor of operations research at the ESSEC Business School of Paris. Research interests of Ivana Ljubić include combinatorial optimization, optimization under uncertainty, and bilevel optimization. She uses tools and methods of mixed integer (non-) linear programming, metaheuristics, and their successful combinations for solving challenging optimization problems with applications in network design, telecommunications, transportation, and logistics.

Enrico Malaguti is associate professor of operations research at the Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" of the University of Bologna, Italy. His main research interests include decomposition and reformulation techniques for mixed-integer linear and nonlinear programs and for stochastic optimization. He is also interested in discrete optimization and in particular in the design and computational evaluation of algorithms for problems on graphs.

Paolo Paronuzzi is a research fellow at the Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" of the University of Bologna, Italy. His research interests focus on integer programming and combinatorial optimization, with an emphasis in decomposition approaches and model tightening techniques.