Discrete Applied Mathematics 290 (2021) 86–111

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

The multi-terminal vertex separator problem: Branch-and-Cut-and-Price

Y. Magnouche [a,*], A.R. Mahjoub [a], S. Martin [b]

[a] Université Paris-Dauphine PSL Research University, CNRS, UMR [7243], LAMSADE, 75016 Paris, France
[b] Laboratoire de conception, optimisation et modélisation des systèmes, Université de Lorraine, Metz, France

ARTICLE INFO

Article history:
Received 12 January 2019
Received in revised form 16 June 2020
Accepted 23 June 2020
Available online 14 September 2020

Keywords:
Vertex separator problem
Integer linear programming
Combinatorial optimization
Column generation
Branch-and-Price
Branch-and-Cut-and-Price

ABSTRACT

We are given a graph $G = (V \cup T, E)$, with $V \cup T$ the set of vertices where T is a set of terminals and E the set of edges. The multi-terminal vertex separator problem consists in finding a subset of vertices $S \subseteq V$ of minimum size intersecting all paths between every pair of terminals. In this paper we present three extended linear integer programming formulations for the multi-terminal vertex separator problem and we develop Branch-and-Price and Branch-and-Cut-and-Price algorithms. For each formulation we present the pricing problem, the branching scheme and the computation of the dual bound used during the column generation phase. Computational results are reported comparing the performance of the formulations on a set of instances.

1. Introduction

Given a simple graph $G = (V \cup T, E)$ with $V \cup T$ the set of vertices, where T is a set of distinguished vertices called *terminals*, a *multi-terminal vertex separator* S in G is a subset of vertices such that the graph induced by $(V \cup T) \setminus S$ consists of $|T|$ disjoint components, each with exactly one terminal. The *multi-terminal vertex separator problem* (MTVSP for short), consists in finding a multi-terminal vertex separator in G of minimum size. When a weight $w(v) \in \mathbb{Z}^+$ is associated with a vertex $v \in V$, finding the multi-terminal vertex separator of a minimum weight can, polynomially, be reduced to solving the size version of the MTVSP in a transformed graph. This latter consists in replacing v by a clique $K_{w(v)}$ of size $w(v)$ such that each vertex of $K_{w(v)}$ is adjacent to the neighbors of v. The MTVSP is a NP-hard problem [14]. It has many applications [8]. A multi-terminal vertex separator can be used to evaluate the robustness of the network connecting the terminals. The MTVSP has also applications in social networks. Indeed, by solving a multi-terminal vertex separator problem, one gets different components each containing exactly one terminal. These components may represent communities with one leader. The multi-terminal vertex separator may then represent persons connecting communities. The MTVSP has also other applications in different areas like VLSI conception, linear algebra, connectivity problems and parallel algorithms.

In this paper, we propose three extended formulations for the MTVSP. We develop a Branch-and-Price algorithm for each formulation and a Branch-and-Cut-and-Price algorithm for two of them.

The MTVSP was already considered in [11]. The authors propose two integer programming formulations for the problem and present a polyhedral study. They describe several classes of valid inequalities and develop a Branch-and-Cut algorithm. In [17], the authors give a linear system for the MTVSP, and characterize the class of graphs for which this

* Corresponding author.
E-mail addresses: youcef.magnouche@dauphine.fr (Y. Magnouche), ridha.mahjoub@lamsade.dauphine.fr (A.R. Mahjoub), sebastien.martin@univ-lorraine.fr (S. Martin).

https://doi.org/10.1016/j.dam.2020.06.021
0166-218X/

system is totally dual integral (i.e., the dual problem has an integer optimal solution for any integer weight vector for which the primal has a bounded optimal solution). The MTVSP is a variant of the *k-separator problem* that consists, given a graph $G = (V, E)$, in partitioning V into $k + 1$ subsets $\{S, V_1, \ldots, V_k\}$ in such a way that S is minimum and there is no edge between any two different subsets V_i and V_j. Many variants of this problem have been studied in the literature [6–10,18]. In [6,18], the authors consider the following problem. Given a simple graph $G = (V, E)$ and an integer $\beta(n)$ with $n = |V|$, partition V into three subsets A, B and C such that $|C|$ is minimum, no vertex in A is adjacent to a vertex in B and $\max\{|A|, |B|\} \leq \beta(n)$. In [7], a different variant of the k-separator problem is considered. Given a simple graph $G = (V, E)$ and $a, b \in V$ two terminals, the problem here is to partition V into three subsets A, B and C minimizing the size of the cut induced by C such that $a \in A$, $b \in B$ and no vertex in A is adjacent to a vertex in B. It is shown that the problem can be solved in polynomial time and is equivalent to a minimum cut problem in an auxiliary graph. In [10], the balanced vertex k-separator problem is considered. This consists, given a graph $G = (V, E)$, in finding a subset $S \subset V$ of minimum cardinality such that after removing S from G, the remaining vertices can be partitioned into k disjoint subsets with balanced sizes. The authors present a compact formulation and a polyhedral analysis for the problem. They also propose an extended formulation and develop a Branch-and-Price algorithm.

Column generation is an efficient approach for solving linear programs with large number of variables [13]. In each step of the method, we consider a restricted linear program given by a few number of variables. This is called the restricted master problem (RMP). The idea of the method is to generate in each iteration new columns (variables) to add to the restricted master. This is called the Pricing. If such algorithm is combined with a Branch-and-Bound algorithm, then the resulting approach is called a Branch-and-Price algorithm. A Branch-and-Cut-and-Price algorithm is a combination of a Branch-and-Price and a cutting plane algorithm [15]. Such algorithms are used for solving integer linear programs with large number of variables and constraints.

In this paper we will present three extended formulations for the MTVSP. For each formulation we develop a Branch-and-Price algorithm. We first define the pricing problem that we solve at each iteration of the column generation algorithm. We also identify the basic columns that we use in the first restricted master problem. This helps the column generation algorithm to start with a good primal bound. Moreover, we detail the computation of the Lagrangian bound which represents a dual bound for the problem. Therefore, if the two bounds are equal, we stop the column generation algorithm even if there are other variables to add to the RMP. Then we present a strategy for adding the new variables to the RMP in order to speed up the resolution of the RMP. Finally, we explain how the master problem structure is exploited to add new valid inequalities without changing the pricing problem. This permits to develop a Branch-and-Cut-and-Price algorithm.

In the rest of this section we give some notations. We denote by $G = (V \cup T, E)$ a simple graph with $V \cup T$ a set of vertices where T is the set of terminals and E the set of edges. Let $n = |V|$, $k = |T|$ and $m = |E|$. In the rest of this paper, and for sake of convenience, we will refer to the multi-terminal vertex separator as *separator*. Given a vertex $v \in V \cup T$, we denote by $N(v) \subseteq (V \cup T) \setminus \{v\}$ the set of vertices adjacent to v and by $d(v)$ the size of $N(v)$. Given a subset $W \subseteq (V \cup T)$, we denote by $N(W) \subseteq (V \cup T) \setminus W$ the set of vertices adjacent to at least one vertex in W. A path between two terminals is called a *terminal path*. For two terminals $t, t' \in T$, let $P_{tt'}$ be a terminal path. For a set of vertices $W \subset V \cup T$, the subgraph of G induced by W, denoted by $G[W]$, is the graph whose set of nodes is W and set of edges consists of all the edges of E having their end vertices in W. For $v \in V$, let $\delta(v)$ denote the set of edges incident to v and $\delta(W)$ the set of edges having exactly one vertex in W. If $x \in \mathbb{R}^{V \cup T}$, let $x(W) = \sum_{v \in W} x(v)$. In all figures of this paper, the terminals are represented by triangles.

In this paper we consider the following hypotheses

1- $|T| \geq 3$, otherwise the MTVSP is polynomial [7].
2- There is no edge between two terminals, otherwise the problem has no solution.
3- For every two different terminals $t, t' \in T$, we have $N(t) \cap N(t') = \emptyset$. Otherwise all vertices in $N(t) \cap N(t')$ must belong to the separator. In this case we can remove these vertices from the graph.
4- For each vertex $v \in V$, there is at least one terminal path, containing v. Otherwise v cannot belong to a minimal separator. In this case we can delete it from the graph. Checking if a node v belongs to a terminal path, can be done in polynomial time. This consists in computing a flow of value 2 of minimum weight between v and an artificial sink connected to all the terminals of T, where the edges have a capacity and a weight equal to 1.
5- Graph G is connected. Otherwise, we consider the MTVSP on each component.

The paper is organized as follows. In Sections 2–4, we present the three extended linear integer programming formulations for the MTVSP. For each formulation we give a column generation scheme to solve the linear relaxation, the way to compute the Lagrangian bound. We then present the branching scheme used in the Branch-and-Price algorithm. Section 5 is devoted to the Branch-and-Cut-and-Price algorithms, and Section 6 is devoted to the numerical results.

2. The terminal-set formulation

A *terminal-set* $W \subseteq V \cup T$ is a set of vertices containing exactly one terminal and such that the neighbors of W do not contain terminals, *i.e.*, $|W \cap T| = 1$ and $N(W) \cap T = \emptyset$. Fig. 1 shows three graphs with different configurations. Also, the subset of vertices in Fig. 1(a) is not a terminal-set since there is one terminal adjacent to this subset. The subset of vertices

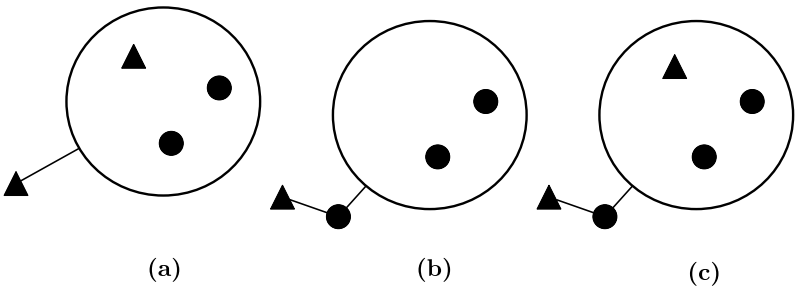

Fig. 1. Three different subsets.

in Fig. 1(b) does not define a terminal-set since it contains no terminals. The subset of vertices in Fig. 1(c) represents a terminal-set.

In this section we will introduce a formulation for the MTVSP based on the terminal-sets. Indeed, any solution of the MTVSP can be seen as a partition of the vertex set into $k+1$ subsets such that k of them are disjoint terminal-sets. Thus the variables of this formulation are associated to the terminal-sets. Then the multi-terminal vertex separator problem reduces to finding k disjoint terminal-sets such that the cardinality of the multi-terminal vertex separator is minimum. Let $\mathcal{W}$ be the set of all terminal-sets of G. For $W \in \mathcal{W}$, let x^W be a $0-1$ variable which takes 1 if W is selected and 0 if not. We notice that these variables are exponential in number. For $v \in V$, let $y(v)$ be a $0-1$ variable which takes 1 if v is selected in the multi-terminal vertex separator and 0 if not.

Given a terminal-set $W \in \mathcal{W}$, let $a^W \in \{0,1\}^E$ and $b^W \in \{0,1\}^{V \cup T}$ be the vectors defined as follows,

$$a^W_{uv} = \begin{cases} 1 & \text{if } \{u,v\} \cap W \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } uv \in E,$$

$$b^W_v = \begin{cases} 1 & \text{if vertex } v \in W, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } v \in V \cup T.$$

The MTVSP is equivalent to the following integer linear program,

$$\min \sum_{v \in V} y(v) \tag{1}$$

$$\sum_{W \in \mathcal{W}} -a^W_e x^W \geq -1 \qquad \forall e \in E, \tag{2}$$

$$\sum_{W \in \mathcal{W}} b^W_v x^W + y(v) = 1 \qquad \forall v \in V, \tag{3}$$

$$\sum_{W \in \mathcal{W}} b^W_t x^W = 1 \qquad \forall t \in T, \tag{4}$$

$$x^W \geq 0 \qquad \forall\, W \in \mathcal{W}, \tag{5}$$

$$y(v) \geq 0 \qquad \forall\, v \in V, \tag{6}$$

$$x \text{ integer}, \tag{7}$$

$$y \text{ integer}. \tag{8}$$

Inequalities (2) ensure the disjunction of the terminal-sets, inequalities (3) ensure that every vertex in V is either in the multi-terminal vertex separator or in exactly one terminal-set and not in both. Finally, inequalities (4) guarantee that each terminal belongs to a terminal-set.

Theorem 2.1. *For each feasible solution (x, y) of (2)–(7), y is the incidence vector of a multi-terminal vertex separator.*

Proof. From Eqs. (3), if x is integer, y so is. ■

Theorem 2.2. *For each feasible solution (x, y) of (2)–(6) and (8), y is the incidence vector of a multi-terminal vertex separator.*

Proof. Suppose that y is not an incidence vector of a multi-terminal vertex separator. It follows that there exists a terminal path $P_{tt'}$ between two terminals t and t' such that $y(P_{tt'}) = 0$. Note that, since x may be fractional, each vertex $v \in V \cup T$

may belong to several fractions of terminal-sets. We claim that, for an edge uv from $P_{tt'}$, u and v must belong to the same terminal-set of the solution. In fact suppose that there are W_1 and W_2 such that $u \in W_1$, $v \in W_2$ such that $x^{W_1} > 0$ and $x^{W_2} > 0$. As $y(u) = 0$, by inequalities (3), it follows that $\sum_{W \in \mathcal{W}} b_u^W x^W = 1$. As $x^{W_2} > 0$, it follows that $\sum_{W \in \mathcal{W}} a_e^W x^W > 1$, contradicting inequalities (2). Therefore, all vertices of $P_{tt'}$ are in the terminal-set. On the other hand, t and t' cannot belong to the same terminal-set, a contradiction. ■

Model (1)–(5) is called the *master problem*. It has an exponential size, thus we need a column generation procedure to solve its continuous relaxation.

Let $u \in \mathbb{R}_+^E$, $\eta \in \mathbb{R}^V$ and $\lambda \in \mathbb{R}^T$ be the dual variables associated with inequalities (2), (3), (4), respectively. The dual of the linear relaxation of (1)–(5), denoted by DLMP, is given by

$$\max \sum_{e \in E} -u^e + \sum_{v \in V} \eta_v + \sum_{t \in T} \lambda_t$$

$$\sum_{e \in E} -a_e^W u^e + \sum_{v \in V} \eta_v b_v^W + \sum_{t \in T} \lambda_t b_t^W \leq 0 \qquad \forall\, W \in \mathcal{W}, \tag{9}$$

$$\sum_{v \in V} \eta_v \leq 1 \qquad \forall\, v \in V, \tag{10}$$

$$u_e \geq 0 \qquad \forall e \in E,$$

$$\eta_v \in \mathbb{R} \qquad \forall v \in V,$$

$$\lambda_t \in \mathbb{R} \qquad \forall t \in T.$$

The master problem is initialized with a subset of variable and then the additional variables necessary to solve its linear relaxation, are generated by separating the associated dual constraints (9). This constitutes the *pricing problem*. DLMP will be used for both defining the pricing problem and computing the Lagrangian bound.

2.1. Pricing problem

Given a dual solution $\pi = (u, \eta, \lambda)$, the pricing problem is nothing but the separation problem for the dual constraints (9), that is to say, it consists in finding a subset $W \in \mathcal{W}$ such that

$$\sum_{e \in E} -u^e a_e^W + \sum_{v \in V} \eta_v b_v^W + \sum_{t \in T} \lambda_t b_t^W > 0.$$

If such W does not exist, then the current solution is optimal. This can be reduced to a binary linear program. In fact consider the following binary variables

$$r(v) = \begin{cases} 1 & \text{if vertex } v \in W, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } v \in V \cup T,$$

$$z(uv) = \begin{cases} 1 & \text{if } \{u, v\} \cap W \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } uv \in E.$$

Note that r represents a terminal-set W. Thus, inequality (9), associated with W, can be written as

$$\sum_{e \in E} -u^e z(e) + \sum_{v \in V} \eta_v r(v) + \sum_{t \in T} \lambda_t r(t) \leq 0,$$

Therefore, the pricing problem is equivalent to the following mixed integer linear program P',

$$\max \sum_{e \in E} -u^e z(e) + \sum_{v \in V} \eta_v r(v) + \sum_{t \in T} \lambda_t r(t)$$

$$z(uv) \geq r(u) \qquad \forall uv \in E, \tag{11}$$

$$z(uv) \geq r(v) \qquad \forall uv \in E, \tag{12}$$

$$z(uv) \leq r(u) + r(v) \qquad \forall uv \in E, \tag{13}$$

$$z(ut') \leq 1 - \sum_{t \in T \setminus \{t'\}} r(t) \qquad \forall ut' \in E, \tag{14}$$

$$\sum_{t \in T} r(t) = 1, \tag{15}$$

$$z(e) \geq 0 \qquad \forall e \in E, \tag{16}$$

$$z(e) \leq 1 \qquad \forall e \in E, \tag{17}$$
$$r(v) \geq 0 \qquad \forall v \in V, \tag{18}$$
$$r(v) \leq 1 \qquad \forall v \in V, \tag{19}$$
$$r(v) \text{ integer} \qquad \forall v \in V \cup T, \tag{20}$$
$$z(e) \text{ integer} \qquad \forall e \in E. \tag{21}$$

Inequalities (11)–(13) link variables z and r. If a terminal $t \in T$ is in the solution, inequalities (14) ensure that all nodes adjacent to the terminals of $T \setminus \{t\}$ are not in the solution. Finally, inequalities (15) guarantee that exactly one terminal is selected in the solution.

If the optimal value of P' is greater than 0, then we add the column corresponding to W, the optimal solution of P', to the restricted master problem (RMP). Otherwise, the current solution of the RMP is optimal.

Program P' can be simplified. The following lemmas show that some inequalities in P' can be deleted.

Lemma 2.1. *Inequalities (13) can be deleted.*

Proof. Let P'' be the linear program obtained from P' by deleting inequalities (13). Consider an optimal solution (r, z) of P''. Let (r', z') be the solution obtained from (r, z) by setting $z'(uv) = 0$ for all $uv \in E$ with $r(u) = r(v) = 0$. Clearly, (r', z') satisfies the inequalities of P'. Moreover, since the weight of $z(e)$ in the objective function is less than or equal to 0, (r', z') is an optimal solution of P'. ■

Lemma 2.2. *Inequalities (16) can be deleted.*

Proof. By inequalities (18), $r(v) \geq 0$ for all $v \in V \cup T$. Moreover, by inequalities (12), $z(uv) \geq r(v)$ for all $uv \in E$. Therefore $z(uv) \geq 0$, and hence inequalities (16) are redundant and can then be deleted. ■

Lemma 2.3. *For an edge $e = uv$ such that $u^e \neq 0$, inequalities (17) can be deleted.*

Proof. Note that $\alpha \leq 1$ for all inequalities of P' of the form $z(e) \geq \alpha$. Moreover, for all $e \in E$, the weight of $z(e)$ in the objective function is less than or equal to 0. Since $u^e \neq 0$, it follows that $z(e) \leq 1$ in the optimal solution of P' for all $e \in E$ such that $u^e \neq 0$. ■

Lemma 2.4. *Constraints (21) can be deleted.*

Proof. Let P'' be the program obtained from P' by deleting constraints (21). Suppose there is an optimal fractional solution (r, z) of P'', with $z(e)$ fractional for some $e \in E$. Since $r(v) \in \{0, 1\}$ for all $v \in V$, and the coefficient of the z variables in the objective function is less than or equal to 0, we have that $0 < z(e) < 1$. As a consequence, any inequality among (11)–(14) cannot be tight for (r, z), if it is induced by e. In addition, if $e = uv$, it follows that $r(u) = r(v) = 0$. Let (r', z') be the solution obtained from (r, z) by setting $z'(e) = 0$. We have in consequence that (r', z') is a solution of P'. Moreover, the value of (r', z') is greater than that of (r, z), a contradiction. ■

In what follows we denote by P'' the linear program obtained from P' by deleting inequalities (13), (16) and constraints (21). Hence solving the pricing problem reduces to solving P''.

A $m \times n$ integral matrix A is said to be *totally unimodular* if and only if each set $R \subseteq \{1, \ldots, m\}$ can be divided into two disjoints sets R_1 and R_2 such that $\sum_{i \in R_1} A_{ij} - \sum_{i \in R_2} A_{ij} \in \{-1, 0, +1\}, j = 1, 2, \ldots, n$. Remark that one of the characterization of totally unimodular matrices, is that A is totally unimodular if and only if for any integer vector b, the polyhedron $\{Ax \geq b, x \geq 0\}$ is integer.

Theorem 2.3. *The pricing problem can be solved in polynomial time.*

Proof. As a terminal-set contains exactly one terminal, looking for a terminal-set can be decomposed into a sequence of subproblems, each consisting in finding a terminal-set with a specific terminal t^*. This is equivalent to solving P'' by fixing one variable $r(t^*)$ to 1 and all other variables $r(t),\ t \in T \setminus \{t^*\}$ to 0. As a consequence, equality (15) becomes redundant and can then be deleted. The linear relaxation of the resulting program associated with t^* is as follows,

$$\max \sum_{e \in E} -u^e z(e) + \sum_{v \in V} \eta_v r(v) + \lambda_{t^*} r(t^*)$$
$$z(uv) \geq r(u) \qquad \forall uv \in E,$$
$$z(uv) \geq r(v) \qquad \forall uv \in E,$$
$$z(ut') \leq 0 \qquad \forall ut' \in E, t' \neq t^*,$$
$$z(e) \leq 1 \qquad \forall e \in E,$$

$$r(v) \geq 0 \qquad \forall v \in V,$$
$$r(v) \leq 1 \qquad \forall v \in V.$$

Let A be the matrix of the above linear program. Now we need to prove that an optimal integer solution of the above linear program can be found in a polynomial time. To this end, we show that A is totally unimodular which is enough to prove the theorem.

Claim 2.1. *Matrix A is totally unimodular.*

Proof. Note that A is totally unimodular if its transpose $A^\top$ so is. For any subset of rows R in $A^\top$ we can divide it into two subsets R_1 and R_2, one containing the rows associated with variables z and the other with variables r, respectively. Since, all elements of A are in $\{-1, 0, +1\}$, each constraint contains at most two variables (at most one variable of z and at most one variable of r), it follows that for each column $j \in \{1, 2, \ldots, n\}$, $\sum_{i \in R_1} A_{ij} - \sum_{i \in R_2} A_{ij} \in \{-1, 0, +1\}$. Therefore, A is a totally unimodular. ■

By the remark above and Claim 2.1, it follows that an optimal solution of P'' can be obtained by solving its linear relaxation. Consequently, the pricing problem reduces to solving k linear programs with polynomial size, which can be done in polynomial time. ■

2.2. Heuristic algorithm for the pricer

In what follows, we propose a heuristic for the pricing problem. The aim is to use the heuristic as a first phase in the pricing in order to speed up its resolution. The first step of the algorithm consists in selecting a terminal which may be an appropriate terminal for a terminal-set violating (9). We have remarked from numerical tests that terminal t^* maximizing $(\lambda_t + \sum_{tv \in E} -u^{tv})$ would be a good choice. The second step consists in adding one by one each vertex of V, not adjacent to terminals of $T \setminus \{t^*\}$, to W and to check whether the value of W increases or not. If it decreases, we do not add the node to W. This heuristic is given in Algorithm 1

Algorithm 1: Heuristic for the pricing problem

Data: Graph $G = (V \cup T, E)$ and $\pi = (u, \eta, \lambda)$
Result: Inequality (9) violated by π

begin
  $t^* \leftarrow \underset{t \in T}{\text{argmax}}(\lambda_t + \sum_{tv \in E} -u^{tv})$;
  $W \leftarrow \{t^*\}$;
  $Z^W \leftarrow \lambda_{t^*} + \sum_{t^*v \in E} -u^{t^*v}$;
  for $(v \in V \setminus N(T \setminus \{t^*\}))$ **do**
    $tmp \leftarrow Z^W + \eta_v$;
    for $(vu \in E)$ **do**
      if $(u \notin W)$ **then**
        $tmp \leftarrow tmp - u^{vu}$;
      end
    end
    if $(tmp > Z^W)$ **then**
      $Z^W \leftarrow tmp$;
      $W \leftarrow W \cup \{v\}$;
    end
  end
  Check if inequality (9) associated with W is violated by π ;
end

The above algorithm runs in $O(n + m)$-time.
Now we will discuss the first restricted master problem.

2.3. Basic columns

The first restricted master problem is induced by a collection $\mathcal{W}^1 \subset \mathcal{W}$ of terminal-sets. $\mathcal{W}^1$ contains a first set of k terminal-sets of cardinality 1, each consists of only one terminal, see Fig. 2(a). Note that as the terminals are pairwise non adjacent, each singleton, consisting of one terminal, is a terminal-set. Set $\mathcal{W}^1$ also contains k maximal terminal-sets $W_{t_1}, \ldots, W_{t_k}$, with terminals $t_1, \ldots, t_k$ of T, respectively *i.e.,* $W_{t_i} = \{t_i\} \cup (V \setminus N(T \setminus \{t_i\}))$, see Fig. 2(b).

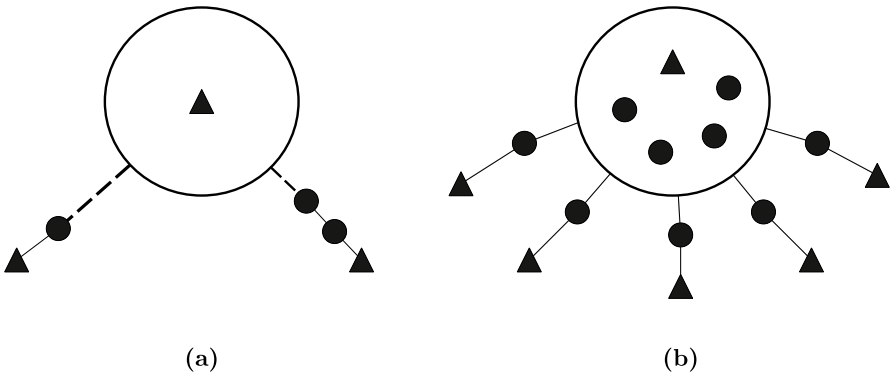

Fig. 2. Basic Columns.

2.4. Column generation strategy

The strategy for finding new columns is to solve the pricing problem. This is first handled using the heuristic presented in Algorithm 1. If the heuristic fails to find an appropriate terminal-set, then we use the exact method by solving P'' to generate columns with positive values, if there is any.

2.5. Branching scheme

By Theorems 2.2 and 2.1, two branching schemes can be considered :

1- $y \in \{0, 1\}$ and $x \in [0, 1]$: The branching scheme is reduced to the classical branching on y, it generates two nodes by imposing either $y(v) = 1$ or $y(v) = 0$ for some fractional variable $y(v)$.

2- $y \in [0, 1]$ and $x \in \{0, 1\}$: Consider a restricted master problem associated with a subset of terminal-sets $\mathcal{W}' \subseteq \mathcal{W}$. For two vertices $u, v \in V \cup T$, let $\mathcal{W}'_{u,v} \subseteq \mathcal{W}'$ be the set of all terminal-sets in $\mathcal{W}'$ containing both u and v. The branching scheme that we will use is as follows. If for $u, v \in V \cup T$, $0 < \sum_{W \in \mathcal{W}'_{u,v}} x^W < 1$, then the branching generates two nodes by imposing either

$$\sum_{W \in \mathcal{W}'_{u,v}} x^W = 1,$$

or

$$\sum_{W \in \mathcal{W}'_{u,v}} x^W = 0.$$

Lemma 2.5. *For any fractional solution, there exists a pair of terminals $u, v \in V \cup T$ such that $0 < \sum_{W \in \mathcal{W}'_{u,v}} x^W < 1$.*

Proof. Consider a restricted master problem (RMP) associated with a set of terminal-sets $\mathcal{W}' \subseteq \mathcal{W}$. Suppose that the solution x of this RMP is fractional. Consider a terminal-set $W_0 \in \mathcal{W}$ such that $0 < x^{W_0} < 1$, and suppose that $|W_0|$ is maximum. Since $W_0 \neq \emptyset$, let $v \in W_0$. We distinguish two cases.

- Case 1: $|W_0| = 1$. It then follows that v is a terminal. By (4), there must exist another terminal-set $W'_0 \in \mathcal{W}$ such that $v \in W'_0$ and $0 < x^{W'_0} < 1$. As $W_0 \neq W'_0$ and $v \in W_0 \cap W'_0$, it follows that $|W'_0| > |W_0|$. This is a contradiction with the maximality of $|W_0|$.
- Case 2: $|W_0| \geq 2$. By (3), $\sum_{W \in \mathcal{W}'} b_v^W x^W \leq 1$.
  - If $\sum_{W \in \mathcal{W}'} b_v^W x^W < 1$, then it is clear that for all $u \in W_0 \setminus \{v\}$, $0 < \sum_{W \in \mathcal{W}'_{u,v}} x^W < 1$ and the lemma follows.
  - If $\sum_{W \in \mathcal{W}'} b_v^W x^W = 1$, then again the result follows. For otherwise, for all vertex $u \in W_0 \setminus \{v\}$, $\sum_{W \in \mathcal{W}'_{u,v}} x^W = 1$. Let
    - $\overline{\mathcal{W}}'_v = \{W \in \mathcal{W}' \mid v \in W \text{ and } x^W > 0\}$,
    - $\overline{\mathcal{W}}'_{u,v} = \{W \in \mathcal{W}'_{u,v} \mid x^W > 0\}$ for all $u \in W_0 \setminus \{v\}$.

    Clearly, for all $u \in W_0 \setminus \{v\}$, $\sum_{W \in \mathcal{W}'} b_v^W x^W = \sum_{W \in \overline{\mathcal{W}}'_v} x^W = \sum_{W \in \overline{\mathcal{W}}'_{u,v}} x^W = 1$.

We claim that $W_0 \subset W_i$ for all $W_i \in \overline{W}'_{u,v} \setminus \{W_0\}$ and $u \in W_0 \setminus \{v\}$. In fact, suppose, on the contrary that $W_0 \not\subseteq W_i$ for some set $W_i \in \overline{\mathcal{W}}'_{u,v} \setminus \{W_0\}$ and $u \in W_0 \setminus \{v\}$. Hence, there is $u' \in W_0 \setminus W_i$. As $\sum_{W \in \overline{\mathcal{W}}'_{u',v}} x^W = 1$, $W_i \notin \overline{\mathcal{W}}'_{u',v}$, and $\sum_{W \in \overline{\mathcal{W}}'_v} x^W = 1$, it follows that $x^{W_i} = 0$, a contradiction and the claim follows.
However this contradicts again the fact that $|W_0|$ is maximum. ■

A branching scheme is said to be *complete*, if it can generate any feasible solution. As corollary, we obtain the following.

Corollary 2.1. *The two branching schemes, presented above, are complete.*

2.6. Lagrangian bound

The Lagrangian bound is a value that represents the dual bound of the linear relaxation of the master problem (LMP). It is the value of the objective function of DLMP (dual of the LMP).

To compute the Lagrangian bound, we need a feasible solution of DLMP. In what follows we will show how to construct a feasible solution of DLMP during the column generation phase. Let $\pi = (u, \eta, \lambda)$ be a dual vector obtained from a restricted master problem induced by $\mathcal{W}' \subseteq \mathcal{W}$. And let Z be the optimal value of the pricer with respect to π. For all $W \in \mathcal{W}$, we then have that

$$\sum_{e \in E} -u^e a_e^W + \sum_{v \in V} \eta_v b_v^W + \sum_{t \in T} \lambda_t b_t^W \leq Z.$$

Each terminal-set $W \in \mathcal{W}'$ contains exactly one terminal. It follows that $\sum_{t \in T} b_t^W = 1$. Moreover, for all terminal $t \in T$, b_t^W is integer. Let $\overline{\lambda} \in \mathbb{R}^T$ be the vector defined as follows

- $\overline{\lambda}_t = \lambda_t - Z$ for all terminal $t \in T$.

It follows that for all $W \in \mathcal{W}$,

$$\sum_{e \in E} -u^e a_e^W + \sum_{v \in V} \eta_v b_v^W + \sum_{t \in T} \overline{\lambda}_t b_t^W \leq 0.$$

Thus, $(u, \eta, \overline{\lambda})$ is a feasible solution for DLMP and $LB = \sum_{e \in E} -u^e + \sum_{v \in V} \eta_v + \sum_{t \in T} \overline{\lambda}_t$ is a lower bound for LMP.

Fig. 3 illustrates the state of the Lagrangian bound and the upper bound (optimal solution of the restricted master problem) during the column generation phase at the root node.

3. The isolating-separator formulation

In this section we will give a second extended formulation for the MTVSP based on isolating separators.

An *isolating-separator* $S \subseteq V$, associated with a terminal $t \in T$, is a set of vertices that intersects all paths between t and the terminals of $T \setminus \{t\}$. For a terminal $t \in T$, let $\mathcal{S}^t$ be the set of all the isolating-separators in G associated with t. Let $\mathcal{S} = \bigcup_{t \in T} \mathcal{S}^t$ be the set of all isolating-separators in G. Note that $\mathcal{S}$ may contain similar isolating-separators but each is associated with a particular terminal.

In Fig. 4, set S^t represents an isolating-separator since it intersects all paths (represented by dashed lines) between t and all other terminals. We remark the following.

Remark 3.1. Any MTV separator is an isolating-separator that can be associated with any terminal.

In what follows we will introduce a formulation for the MTVSP based on the isolating-separators. Indeed, any solution of the MTVSP can be seen as the union of several isolating-separators, each intersecting all terminal paths between one specific terminal and all the other terminals. Then, the multi-terminal vertex separator problem reduces to finding a set of isolating-separators such that the cardinality of their union is minimum. Given an isolating-separator $S \in \mathcal{S}$, let x^S be a variable which takes 1 if S is chosen and 0 if not. And for $v \in V$, let y_v be a variable such that $y(v)$ takes 1 if v belongs to at least one selected isolating-separator and 0 if not. We notice that the variables x are exponential in number.

Given an isolating-separator $S \in \mathcal{S}$, let $a^S \in \{0,1\}^{V \times T}$, $\overline{a}^S \in \{0,1\}^V$ and $b^S \in \{0,1\}^T$ be the vectors defined as follows

$$a^S_{v,t} = \begin{cases} 1 & \text{if } v \text{ belongs to } S \text{ and } S \in \mathcal{S}^t, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } v \in V,\ t \in T,$$

$$b^S_t = \begin{cases} 1 & \text{if } S \in \mathcal{S}^t, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } t \in T.$$

The MTVSP is equivalent to the following integer linear program

$$\min \quad y(V) \tag{22}$$

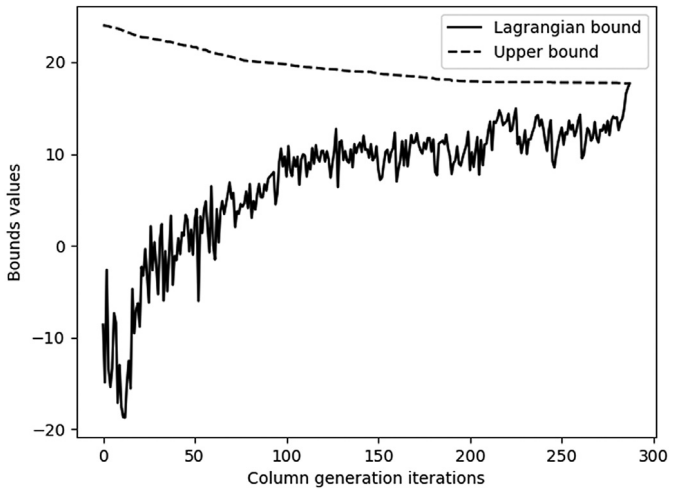

Fig. 3. Example of the Lagrangian bound.

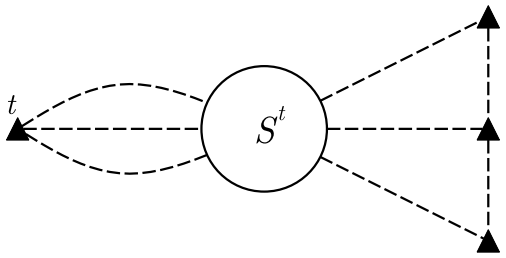

Fig. 4. Example of an isolating-separator.

$$y(v) - \sum_{S \in \mathcal{S}} a^S_{v,t} x^S \geq 0 \qquad \forall t \in T, \quad \forall v \in V, \tag{23}$$

$$\sum_{S \in \mathcal{S}} b^S_t x^S = 1 \qquad \forall t \in T, \tag{24}$$

$$x^S \geq 0 \qquad \forall\, S \in \mathcal{S}, \tag{25}$$

$$y(v) \in \{0, 1\} \qquad \forall\, v \in V. \tag{26}$$

Inequalities (23) ensure that a vertex belonging to a selected isolating-separator, also belongs to the multi-terminal vertex separator. Equalities (24) guarantee that exactly one isolating-separator, associated with each terminal, is selected. Fig. 5 illustrates a solution of the MTVSP. The three subsets in the Figure represent selected isolating-separators. The union of the three isolating-separators is a multi-terminal vertex separator for G. We note that one isolating-separator can be associated to several terminals.

Model (22)–(26) is the *master problem*. It has an exponential size, thus we need a column generation procedure to solve its continuous relaxation.

Let $u \in \mathbb{R}_+^{T \times V}$ and $\eta \in \mathbb{R}^T$ be the dual variables associated with inequalities (23) and Eq. (24). The dual of the linear program (22)–(25), denoted by DLMP, is given by the following program

$$\begin{aligned}
\max \sum_{t \in T} \eta_t & \\
\sum_{t \in T} u^v_t &\leq 1 \qquad \forall v \in V, \\
\sum_{v \in V} \sum_{t \in T} -a^S_{v,t} u^v_t + \sum_{t \in T} b^S_t \eta_t &\leq 0 \qquad \forall S \in \mathcal{S}, \\
u^v_t &\geq 0 \qquad \forall t \in T, v \in V, \\
\eta_t &\in \mathbb{R} \qquad \forall t \in T.
\end{aligned} \tag{27}$$

The master problem is initialized with a subset of variables, and the additional variables, necessary to solve its linear relaxation, are generated by separating the associated dual constraints (27). This constitutes the *pricing problem*.

3.1. Pricing problem

Given a dual solution $\pi = (u, \eta)$, the pricing problem is nothing but to the separation problem for the dual constraints (27), *i.e.,* it consists in finding a subset $S \in \mathcal{S}$ such that

$$\sum_{v \in V} \sum_{t \in T} -a^S_{v,t} u^v_t + \sum_{t \in T} b^S_t \eta_t > 0.$$

Then the pricing problem can be solved by finding the terminal $t \in T$ and an isolating separator S^t for it such that $-\sum_{v \in V} u^v_t a^S_{vt} + \eta_t$ is maximum. If this value is greater than 0, then we add the corresponding column to the RMP. Otherwise, the current solution of the RMP is optimal.

Theorem 3.1. *The pricing problem can be solved in polynomial time.*

Proof. For a terminal $t \in T$, let $G^t = (V \cup \{t, t_s\}, E)$ be the graph obtained from G by merging all terminals of $T \setminus \{t\}$ into one terminal t_s. Let $c \in \mathbb{R}^V$ such that $c(v) = u^v_t$ for all $v \in V$. To solve the pricing problem, we need to iterate over each terminal $t \in T$ and solve the minimum weight two-terminal vertex separator problem in graph G^t. The latter is polynomially solvable [7,16]. Let Z^*_t be the value of the optimal two-terminal vertex separator in graph G^t. The optimal solution for the pricer is the two-terminal separator with minimum value $Z^*_t - \eta_t$. It follows that the pricer problem can be solved in polynomial time. ■

3.2. Basic columns

Let us remark that set V represents an isolating-separator for any terminal of T. Thus, $V \in \mathcal{S}^t$ for all $t \in T$. So, for the basic columns, we can add a variable associated with set V for each terminal of T. Moreover, any multi-terminal vertex separator is also an isolating-separator for any terminal of T. Thus, for each terminal $t \in T$, we add a variable associated with an isolating-separator S^t that is a multi-terminal vertex separator.

The following heuristic permits to compute a MTV separator of $G = (V \cup T, E)$ by merging the MTV separators of $G = (V \cup \{t, t'\}, E)$ for all $t, t' \in T$.

Algorithm 2: Isolating terminal heuristic.

```
Data: Graph G = (V ∪ T, E)
Result: Vertex separator S ⊆ V
begin
    S = ∅;
    for t, t' ∈ T do
        S^t_{t'} ← a multi-terminal vertex separator of (V ∪ {t, t'}, E);
        S ← S ∪ S^t_{t'};
    end
    Sort S = {v_0, v_1, ..., v_{|S|}} from the vertex of the smallest degree to the vertex of the largest degree in G;
    for (i ∈ {0, ..., |S|}) do
        if S \ {v_i} is a multi-terminal vertex separator then
            S ← S \ {v_i};
        end
    end
    return S;
end
```

The multi-terminal vertex separator between two terminals can be reduced to the max flow problem. We have used the lemon implementation of the Goldberg's preflow algorithm that runs in $O(n^2\sqrt{m})$. Therefore, the above algorithm runs in $O(k^2 n^2 \sqrt{m})$-time.

3.3. Column generation strategy

The column generation strategy aims at generating several columns at each iteration of the column generation phase. This may speed up the resolution time. It is based on the fact that an isolating-separator S may be related to several terminals of T. Therefore, each time we generate a new column, we check if the associated isolating-separator can also

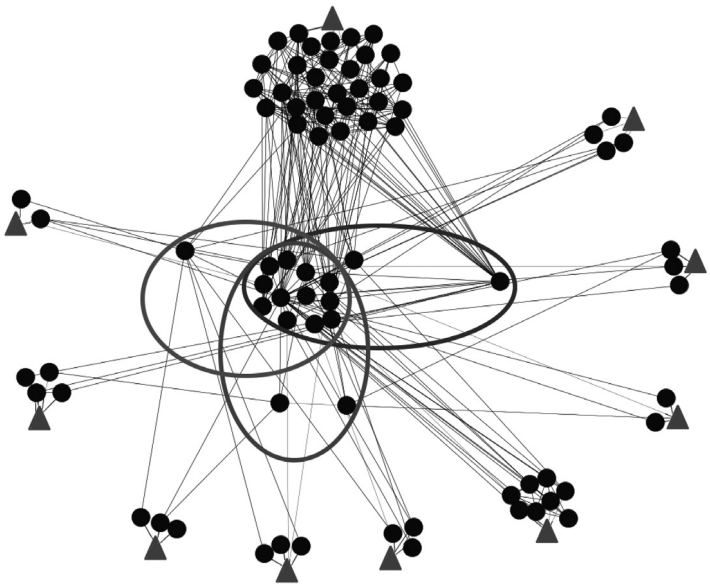

Fig. 5. Example of a solution of the isolating-separator formulation.

correspond to other terminals. Then, we add several columns by iteration. This can be done in a polynomial time. It consists, for a terminal $t \in T$, in adding a super terminal t^* adjacent to all terminal of $T \setminus \{t\}$ and solving a min-cut problem between t and t^*. This can be done in $O(kn^2\sqrt{m})$ time.

3.4. Branching scheme

The branching scheme is reduced to the classical branching on y, it generates two nodes by imposing either $y(v) = 1$ or $y(v) = 0$ for some fractional variable $y(v)$.

3.5. Lagrangian Bound

Let $\pi = (u, \eta)$ be a dual vector obtained from the linear relaxation of a restricted master problem. Let Z be the optimal value of the pricer. Remark that

$$\sum_{v\in V}\sum_{t\in T} -a^S_{v,t}u^v_t + \sum_{t\in T} b^S_t\eta_t \leq Z \quad \forall S \in \mathcal{S}.$$

Let $\overline{\eta} \in \mathbb{R}^T_+$ be a vector such that $\overline{\eta}_t = \eta_t - Z$ for all terminal $t \in T$.

Since $\sum_{t\in T} b^S_t = 1$, it follows that

$$\sum_{v\in V}\sum_{t\in T} -a^S_{v,t}u^v_t + \sum_{t\in T} b^S_t\overline{\eta}_t \leq 0 \quad \forall S \in \mathcal{S}$$

Consequently, $\pi = (u, \overline{\eta})$ is a feasible solution for the DLMP and $LB = \sum_{t\in T} \overline{\eta}_t$ is a lower bound for LMP.

Fig. 6 illustrates the state of the Lagrangian bound and the upper bound (optimal solution of the restricted master problem) during the column generation phase at the root node.

4. The two-terminal vertex separator formulation

A *two-terminal vertex separator* S is a set of vertices whose removal disconnects a pair of terminals. We say that S *separates* this pair of terminals. For $t, t' \in T$, let $\mathcal{S}^t_{t'} \subseteq \mathcal{S}$ be the set of all two-terminal vertex separators of G that separate t and t'. Let $\mathcal{S} = \bigcup_{t,t'\in T} \mathcal{S}^t_{t'}$ be the set of all two-terminal vertex separators of G. Note that $\mathcal{S}$ may contain similar two-terminal vertex separators but each is associated with a particular pair of terminals.

Fig. 7 gives an example of a two-terminal vertex separator. We can see that S is a two-terminal vertex separator for terminals t_1 and t_3 as well as t_2 and t_3 but not a two-terminal vertex separator for terminals t_1 and t_2.

To clarify the definition, let us remark the following which can be easily seen.

Remark 4.1. Any MTV separator is a two-terminal vertex separator that can be associated to any pair of terminals.

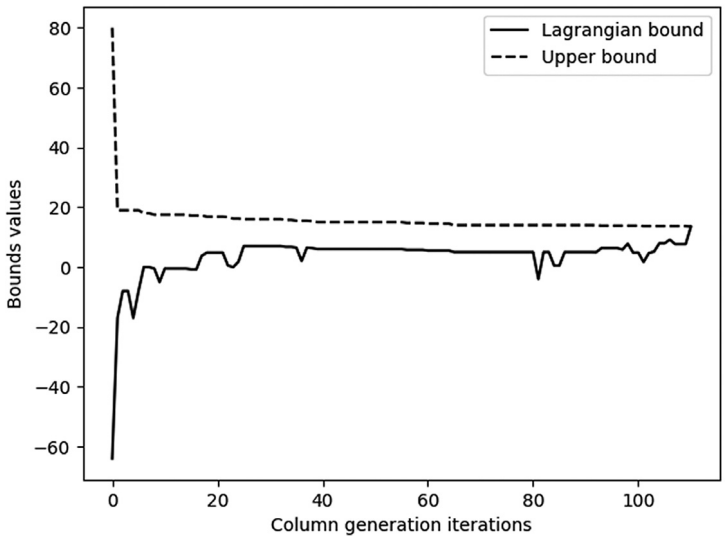

Fig. 6. Example of the Lagrangian bound.

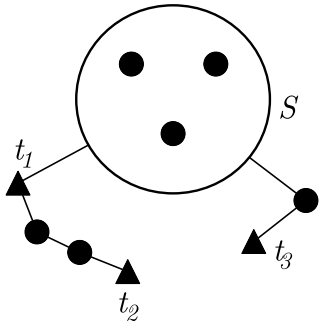

Fig. 7. Example of a two-terminal vertex separator.

In what follows we will introduce a formulation for the MTVSP based on the two-terminal vertex separators. Any MTV separator of G can be seen as the union of a set of two-terminal vertex separators such that each pair of terminal can be represented by at least one element of the set. Then, the multi-terminal vertex separator problem reduces to finding such a set of two-terminal vertex separators such that the cardinality of their union is minimum. For $S \in \mathcal{S}$, let x^S be a variable which takes 1 if S is chosen and 0 if not. And for $v \in V$, let y_v be a variable which takes 1 if v belongs to the multi-terminal vertex separator and 0, if not. Note that the variables x are exponential in number.

Given a two-terminal vertex separator $S \in \mathcal{S}$, let $a^S \in \{0,1\}^{V \times T \times T}$ and $b^S \in \{0,1\}^{T \times T}$ be the vectors defined as follows

$$a^S_{tt'v} = \begin{cases} 1 & \text{if } v \in S \text{ and } S \in \mathcal{S}^t_{t'}, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } t, t' \in T \text{ and } v \in V,$$

$$b^S_{tt'} = \begin{cases} 1 & \text{if } S \in \mathcal{S}^t_{t'}, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } t, t' \in T,$$

The MTVSP is equivalent to the following integer linear program

$$\min \quad y(V) \tag{28}$$

$$y(v) - \sum_{S \in \mathcal{S}} a^S_{tt'v} x^S \geq 0 \qquad \forall v \in V, \forall t, t' \in T, \tag{29}$$

$$\sum_{S \in \mathcal{S}} b^S_{tt'} x^S = 1 \qquad \forall t, t' \in T, \tag{30}$$

$$x^S \geq 0 \qquad \forall S \in \mathcal{S}, \tag{31}$$

$$y(v) \in \{0,1\} \qquad \forall v \in V. \tag{32}$$

Inequalities (29) ensure that each vertex belonging to at least one selected two-terminal vertex separator, also belongs to the multi-terminal vertex separator. Inequalities (30) allow to select exactly one two-terminal vertex separator per pair of terminals.

Lemma 4.1. *Constraints (32) can be replaced by $y(v) \in \mathbb{N}$ for all $v \in V$.*

Proof. By (30), for all $t, t' \in T$, $1 = \sum_{S \in \mathcal{S}} b^S_{tt'} x^S \geq \sum_{S \in \mathcal{S}} a^S_{tt'v} x^S$. Therefore, by (29), $y(v) \leq 1$ in any optimal solution. ■

Theorem 4.1. *For each feasible solution (x, y) of (29)–(32), y is the incidence vector of a multi-terminal vertex separator.*

Proof. By (30), for each pair of terminals $t, t' \in T$, there is one $S \in \mathcal{S}^t_{t'}$ such that $x^S > 0$. Denote this two-terminal vertex separator by $S^t_{t'}$. It is clear that $\widehat{S} = \bigcup_{t,t'} S^t_{t'}$ is a multi-terminal vertex separator of G. As y is integer, by (29), it follows that for all $t, t' \in T$ and $v \in S^t_{t'}$, $y_v = 1$. ■

Model (28)–(31) is the *master problem*. As it has an exponential size, we need a column generation procedure to solve its continuous relaxation.

Let $u \in \mathbb{R}_+^{V \times T \times T}$ and $\eta \in \mathbb{R}^{T \times T}$ be the dual variables associated with inequalities (29) and (30). The dual of the linear relaxation of (28)–(31), denoted by DLMP, is given by

$$\max \sum_{t,t' \in T} \eta_{tt'}$$

$$\sum_{t,t' \in T} \Big(\sum_{v \in V} -a^S_{tt'v} u^v_{tt'} + b^S_{tt'} \eta_{tt'} \Big) \leq 0, \qquad \forall S \in \mathcal{S}, \tag{33}$$

$$\sum_{t,t' \in T} u^v_{tt'} \leq 1, \qquad \forall v \in V, \tag{34}$$

$$u^v_{tt'} \geq 0, \qquad \forall v \in V, \forall t, t' \in T, \tag{35}$$

$$\eta_{tt'} \in \mathbb{R}, \qquad \forall t, t' \in T, \tag{36}$$

The master problem is initialized with a subset of variables, and then the additional variables necessary to solve its linear relaxation are generated by separating the associated dual constraints (33). This constitutes the pricing problem.

4.1. Pricing problem

Given a dual solution $\pi = (u, \eta)$, the pricing problem consists in finding a subset $S \in \mathcal{S}$ such that

$$\sum_{t,t' \in T} \sum_{v \in V} -a^S_{tt'v} u^v_{tt'} + b^S_{tt'} \eta_{tt'} > 0.$$

Thus, the pricing problem reduces to generating a two-terminal vertex separator S associated with two terminals $t, t' \in T$ such that $\sum_{v \in V} -u^v_{tt'} a^S_{tt'v} + b^S_{tt'} \eta_{tt'}$ is maximum.

Theorem 4.2. *The pricing problem can be solved in polynomial time.*

Proof. For all $t, t' \in T$, let $c^t_{t'} : V \cup (T \setminus \{t, t'\}) \to \mathbb{R}$ such that for all $v \in V$, $c^t_{t'}(v) = u^v_{tt'}$ and for all $t'' \in T \setminus \{t, t'\}$, $c^t_{t'}(t'') = +\infty$. Solving the pricing problem consists in iterating over each pair of terminals $t, t' \in T$, and finding a two-terminal vertex separator $S \in \mathcal{S}^t_{t'}$ of minimum weight in G w.r.t $c^t_{t'}$. This is equivalent to finding a minimum cut separating t and t', which can then be solved in polynomial time [7].

4.2. Basic columns

It is clear that any multi-terminal vertex separator is also a two-terminal vertex separator for any pair of terminals. Thus, for each pair of terminals $t, t' \in T$, we add a variable associated with a multi-terminal vertex separator obtained using the heuristic in Algorithm 2. We also use this heuristic to obtain a good primal bound.

4.3. Column generation strategy

The idea of this column generation strategy is to iterate over all pairs of terminals $t, t' \in T$ and to look for the minimum two-terminal vertex separator intersecting all paths between t and t' in G w.r.t $u_{tt'}$. We stop when a two-terminal vertex separator S is found with a positive reduced cost, *i.e.*, $\sum_{t,t' \in T} \sum_{v \in V} -u^v_{tt'} a^S_{tt'v} + b^S_{tt'} \eta_{tt'} > 0$. We then add the associated column.

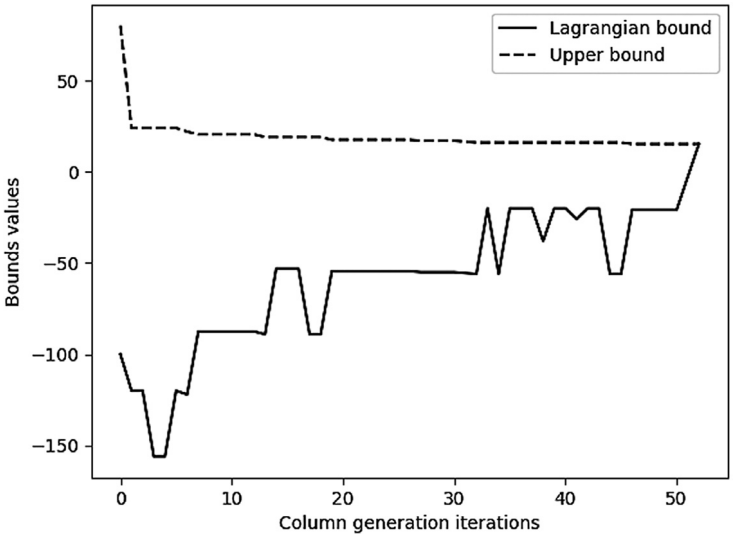

Fig. 8. Example of the Lagrangian bound.

We remarked from the numerical tests that when we iterate over all pairs of terminals, it would be more interesting to start with a pair of terminals $t, t' \in T$ having a maximum value $\eta_{tt'}$. Each time we add a new column to the restricted master problem, we check if the associated two-terminal vertex separator separates other terminals. Then, we add several columns in the same iteration.

4.4. Branching scheme

As before the branching generates two nodes by imposing either $y(v) = 1$ or $y(v) = 0$ for some fractional variable $y(v)$.

4.5. The Lagrangian bound

Let $\pi = (u, \eta)$ be a dual vector obtained from the linear relaxation of a restricted master problem. Let Z be the optimal value of the pricer. Hence,

$$\sum_{t,t' \in T} \sum_{v \in V} -a^S_{tt'v} u^v_{tt'} + b^S_{tt'} \eta_{tt'} \leq Z \quad \forall S \in \mathcal{S}. \tag{37}$$

Let $\overline{\eta} \in \mathbb{R}^{T \times T}$ be the vector such that $\overline{\eta}_{tt'} = \eta_{tt'} - Z$ for all terminal $t, t' \in T$. Then (37) can be written as

$$\sum_{t,t' \in T} \sum_{v \in V} -a^S_{tt'v} u^v_{tt'} + b^S_{tt'} \overline{\eta}_{tt'} \leq 0 \qquad \forall S \in \mathcal{S}.$$

Consequently, since $\sum_{t,t' \in T} b^S_{tt'} = 1$, $(u, \overline{\eta}, \mu)$ is a feasible solution for the DLMP, and $LB = \sum_{t,t' \in T} \overline{\eta}_{tt'}$ is a lower bound for LMP.

Fig. 8 displays the state of the Lagrangian bound and the upper bound (optimal solution of the restricted master problem) during the column generation phase at the root node.

Let $LR^{Isolating}$ and $LR^{Two-terminal}$ be the linear relaxations associated with the isolating-separator and two-terminal vertex separator formulations, respectively.

Theorem 4.3. *The optimal solution of $LR^{Isolating}$ is greater than or equal to the one of $LR^{Two-terminal}$.*

Proof. It is enough to prove that each feasible solution of $LR^{Isolating}$ is also a feasible solution of $LR^{Two-terminal}$. Note that, every isolating-separator associated with terminal $t \in T$ is a two-terminal vertex separator of t and t' for all $t' \in T \setminus \{t\}$. Let $(x, y) \in \mathbb{R}^{\mathcal{S}} \times \mathbb{R}^V$ be the optimal solution of $LR^{Isolating}$ and $S^t = \{S \in \mathcal{S}^t \mid x^S > 0\} = \{S_1, \ldots, S_{|S^t|}\}$ for all $t \in T$. Let $T = \{t_1, \ldots, t_k\}$. Consider the following two-terminal vertex separators

- $S_l^{t_1 t_j} = S_l$ for all $S_l \in S^{t_1}$, $t_j \in T \setminus \{t_1\}$ such that $1 < j < k$,
- $S_l^{t_i t_j} = S_l$ for all $S_l \in S^{t_i}$, $t_i, t_j \in T$ such that $1 < i < j$,
- $S_l^{t_k t_1} = S_l$ for all $S_l \in S^{t_k}$.

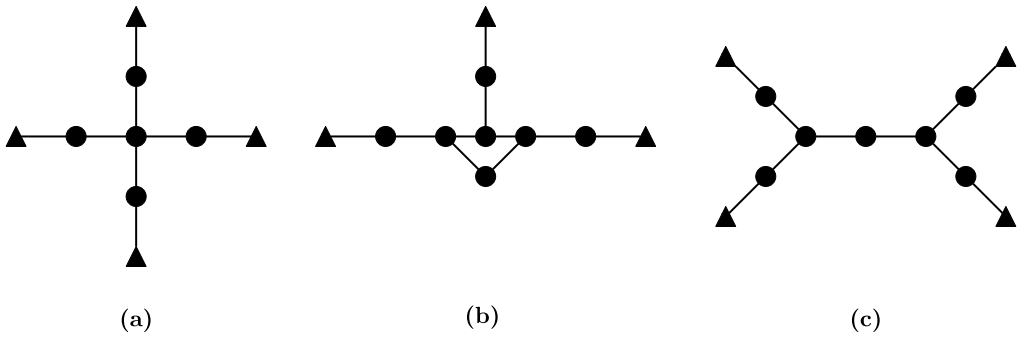

Fig. 9. Examples of star tree, terminal cycle and terminal tree graphs.

Let $\bar{x}$ be the incidence vector of these two-terminal vertex separators and $\bar{y} = y$. We have that $(\bar{x}, \bar{y})$ is a solution of $LR^{Two-terminal}$. Moreover, both relaxations have the same value. ■

5. Branch-and-Cut-and-Price

In this section we propose Branch-and-Cut-and-Price algorithms based on the terminal-set formulation, the isolating-separator and the two-terminal vertex separator formulations. These formulations use a variable vector $y \in \{0, 1\}^V$ such that $y(v) = 1$ if v belongs to the MTV separator and 0 if not. It follows that one can add valid inequalities to these formulations.

In what follows we present the valid inequalities that we will add to the isolating-separator and the two-terminal vertex separator formulations in order to cut the non feasible solutions and to strengthen the linear relaxation. We add three classes of valid inequalities introduced in [11]. We note that adding valid inequalities on y variables will not affect the pricing problems.

The valid inequalities used in the Branch-and-Cut-and-Price algorithms are presented below.

A first class of inequalities is induced by the so-called terminal trees.

A *terminal tree* $H = (V(H) \cup T(H), E(H))$, is a tree such that the terminals of $T(H)$ are the leaves of H. Let $d_H(v)$ be the degree of v in H and $q \in \mathbb{N}$ the number of terminals in $T(H)$. Fig. 9(c) displays a terminal tree with four terminals. If $H = (V(H) \cup T(H), E(H))$ is a terminal tree subgraph of G, then the following terminal tree inequality

$$\sum_{v \in V(H)} (d_H(v) - 1) y_v \geq q - 1. \tag{38}$$

is valid for the MTVS polytope. A heuristic is used to separate this class of inequalities that runs in $O(mlog(m))$ time, described in [11].

The inequalities of the second class are induced by the so-called star trees. A *star tree* $H = (V(H) \cup T(H), E(H))$ of G is a tree such that the terminals of $T(H)$ are the leaves of H and all the other (non-terminal) nodes, different from the root node, are of degree two. Let $q \in \mathbb{N}$ be the number of terminals in $T(H)$. Fig. 9(a) displays a star tree with 4 terminals. If $H = (V(H) \cup T(H), E(H))$ is a star tree subgraph of G with root v_r, then the following star tree inequality

$$y(V(H) \setminus \{v_r\}) + (q - 1) y_{v_r} \geq q - 1 \tag{39}$$

is valid for the MTVS polytope. The class is a subclass of terminal tree inequalities. Moreover, it has a separation algorithm different from the one proposed for the terminal tree cuts that runs in $O(k^2 n(n + m)log(n))$ time [11].

A third class of inequalities is induced by the so-called terminal cycles.

A *terminal cycle* $J = (V(J) \cup T(J), E(J))$ is given by a cycle C and q disjoint paths between a vertex in C and a terminal in T. Fig. 9(b) displays a terminal cycle with three terminals. If $J = (V(J) \cup T(J), E(J))$ is a terminal cycle subgraph of G, then the following terminal cycle inequality

$$y(V(J)) \geq \lceil \frac{|q|}{2} \rceil \tag{40}$$

is valid for the MTVS polytope. A heuristic is used to separate this class of inequalities that runs in $O(kn^2(n + m)log(m))$ time, described in [11].

6. Computational results

Based upon the previous theoretical results, we have developed a Branch-and-Price algorithm for each extended formulation presented in Sections 2–4. Branch-and-Cut-and-Price algorithms are developed for the isolating-separator and

Table 1
Results with the Branch-and-Price algorithm of the terminal-set formulation on DIMACS instances.

Instance	n	Den	k	Cols	Gap	No	Opt	Heur	CPU
anna	138	0.11	5	15 252	38	2	–	28	–
			7	17 588	1	6	25	35	2942.97
			9	14 513	2	10	32	46	1048.37
david	87	0.22	5	13 871	1	5	19	20	2390.71
			7	3 967	4	4	24	30	115.91
			9	13 249	6	78	29	31	1376.27
DSJC1000.1	1000	0.20	5	1 481	–	1	–	200	–
			7	1 533	–	1	–	300	–
			9	1 559	–	1	–	400	–
DSJC125.1	125	0.10	5	5 041	–	1	–	28	–
			7	6 384	–	1	–	42	–
			9	6 042	4	4	–	56	–
DSJR500.1	500	0.06	5	1 721	–	1	–	100	–
			7	2 160	–	1	–	150	–
			9	2 121	–	1	–	200	–
games120	120	0.18	5	5 346	–	1	–	24	–
			7	6 722	–	1	–	36	–
			9	8 547	0	1	46	48	2903.35
huck	74	0.23	5	10 226	0	1	11	13	431.90
			7	9 304	10	11	17	20	228.94
			9	4 752	7	16	23	28	64.70
inithx.i.1	864	0.05	5	1 655	–	1	–	109	–
			7	1 775	–	1	–	159	–
			9	1 963	–	1	–	208	–
inithx.i.2	645	0.07	5	1 456	–	1	–	111	–
			7	1 836	–	1	–	172	–
			9	1 969	–	1	–	229	–
inithx.i.3	621	0.07	5	1 547	–	1	–	115	–
			7	1 756	–	1	–	176	–
			9	1 990	–	1	–	236	–
jean	80	0.17	5	7 937	0	1	12	16	270.63
			7	5 072	2	7	19	23	86.81
			9	2 691	3	5	21	25	17.92
miles1000	128	0.80	5	1 974	–	1	–	28	–
			7	2 215	–	1	–	42	–
			9	2 253	–	1	–	56	–
mulsol.i.1	197	0.21	5	2 048	–	1	–	33	–
			7	2 193	–	1	–	46	–
			9	2 511	–	1	–	59	–
mulsol.i.2	188	0.22	5	2 229	–	1	–	39	–
			7	2 422	–	1	–	57	–
			9	2 808	–	1	–	77	–
mulsol.i.3	184	0.24	5	2185	–	1	–	40	–
			7	2 376	–	1	–	59	–
			9	2 813	–	1	–	78	–
myciel6	95	0.17	5	5 571	0	1	20	20	1850.77
			7	5 928	0	1	30	30	1185.75
			9	12 605	7	13	40	40	1619.49
myciel7	191	0.13	5	2 822	–	1	–	40	–
			7	2 845	–	1	–	60	–
			9	3 513	–	1	–	80	–
queen12_12	144	0.51	5	2 135	–	1	–	32	–
			7	2 357	–	1	–	48	–
			9	2 782	–	1	–	64	–
queen13_13	169	0.47	5	1 825	–	1	–	36	–
			7	1 843	–	1	–	54	–
			9	2 294	–	1	–	72	–
queen14_14	196	0.44	5	1 562	–	1	–	40	–
			7	1 807	–	1	–	60	–
			9	1 952	–	1	–	80	–

the two-terminal vertex separator formulations in order to efficiently solve the multi-terminal vertex separator problem. After a numerical comparison, only the branching scheme on y variables is considered, in this paper, for the terminal-set formulation. We use two kinds of instances, the DIMACS graph coloring instances [1] and random instances generated using boost graph library [2]. The terminals are new nodes added to the graphs. Each terminal is randomly connected to at least 2 vertices and to at most a given $deg\_T \in \mathbb{N}$ vertices. In the Tables below, deg_T is fixed in relation with the size of the graph, *i.e.,* the higher the size of the graph is, the higher deg_T is. The edges incident to the terminals are added

Table 2
Results with the Branch-and-Price algorithm of the terminal-set formulation on random instances.

Instance	n	Den	k	Cols	Gap	No	Opt	Heur	CPU
r_100	100	0.46	5	1871	0	1	20	20	816.57
			7	1758	0	1	30	30	1028.25
			9	3233	–	1	–	40	–
r_200	200	0.46	5	1421	–	1	–	40	–
			7	1510	–	1	–	60	–
			9	1560	–	1	–	80	–
r_300	300	0.46	5	1201	–	1	–	60	–
			7	1260	–	1	–	90	–
			9	1381	–	1	–	120	–
r_400	400	0.45	5	1136	–	1	–	80	–
			7	1241	–	1	–	120	–
			9	1328	–	1	–	160	–
r_300	500	0.45	5	1061	–	1	–	100	–
			7	1159	–	1	–	150	–
			9	1311	–	1	–	200	–
r_600	600	0.45	5	1161	–	1	–	120	–
			7	1203	–	1	–	180	–
			9	1267	–	1	–	240	–
r_500	700	0.45	5	1166	–	1	–	140	–
			7	1240	–	1	–	210	–
			9	1277	–	1	–	280	–
r_800	800	0.45	5	1201	–	1	–	160	–
			7	1249	–	1	–	240	–
			9	1305	–	1	–	320	–
r_900	900	0.45	5	1246	–	1	–	180	–
			7	1293	–	1	–	270	–
			9	1324	–	1	–	360	–
r_1000	1000	0.45	5	1311	–	1	–	200	–
			7	1344	–	1	–	300	–
			9	1379	–	1	–	400	–

Table 3
Results with the Branch-and-Cut-and-Price algorithm of the terminal-set formulation on DIMACS instances.

Instance	n	Den	k	Cols	ST	TC	TT	Gap	No	Opt	Heur	CPU
anna	138	0.11	5	13 616	356	63	1664	38	2	–	28	–
			7	17 489	442	26	1032	1	6	25	35	2912.74
			9	12 952	624	32	846	2	7	32	46	1609.97
david	87	0.22	5	10 701	126	0	128	1	3	19	20	1640.55
			7	3 861	190	14	213	4	4	24	30	95.56
			9	3 637	222	0	249	6	8	29	31	93.80
DSJC1000.1	1000	0.20	5	1 476	0	0	0	–	1	–	200	–
			7	1 533	0	0	0	–	1	–	300	–
			9	1 586	0	0	0	–	1	–	400	–
DSJC125.1	125	0.10	5	5 201	0	0	0	–	1	–	28	–
			7	6 524	0	0	0	–	1	–	42	–
			9	6 128	495	284	522	4	4	–	56	–
DSJR500.1	500	0.06	5	1 776	0	0	0	–	1	–	100	–
			7	2 083	0	0	0	–	1	–	150	–
			9	2 040	0	0	0	–	1	–	200	–
games120	120	0.18	5	5 776	0	0	0	–	1	–	24	–
			7	6 197	0	0	0	–	1	–	36	–
			9	8 500	386	0	386	–	1	–	46	–
huck	74	0.23	5	11 744	18	0	30	0	1	11	13	525.19
			7	12 942	751	195	976	10	11	17	20	908.30
			9	4 217	159	10	249	7	14	23	28	71.89
inithx.i.1	864	0.05	5	1 660	0	0	0	–	1	–	109	–
			7	1 782	0	0	0	–	1	–	159	–
			9	2 008	0	0	0	–	1	–	208	–
inithx.i.2	645	0.07	5	1 446	0	0	0	–	1	–	111	–
			7	1 822	0	0	0	–	1	–	172	–
			9	1 978	0	0	0	–	1	–	229	–
inithx.i.3	621	0.07	5	1 542	0	0	0	–	1	–	115	–
			7	1 749	0	0	0	–	1	–	176	–
			9	1 990	0	0	0	–	1	–	236	–
jean	80	0.17	5	8 401	718	43	1120	0	1	12	16	557.94

(*continued on next page*)

Table 3 (*continued*).

Instance	n	Den	k	Cols	ST	TC	TT	Gap	No	Opt	Heur	CPU
			7	4 264	315	2	441	2	7	19	23	80.17
			9	2 741	55	10	79	3	5	21	25	21.18
Miles1000	128	0.80	5	1 979	0	0	0	–	1	–	28	–
			7	2 341	0	0	0	–	1	–	42	–
			9	2 271	0	0	0	–	1	–	56	–
Mulsol.i.1	197	0.21	5	2 003	0	0	0	–	1	–	33	–
			7	2 158	0	0	0	–	1	–	46	–
			9	2 493	0	0	0	–	1	–	59	–
Mulsol.i.2	188	0.22	5	2 229	0	0	0	–	1	–	39	–
			7	2 436	0	0	0	–	1	–	57	–
			9	2 808	0	0	0	–	1	–	77	–
Mulsol.i.3	184	0.24	5	2 190	0	0	0	–	1	–	40	–
			7	2 369	0	0	0	–	1	–	59	–
			9	2 822	0	0	0	–	1	–	78	–
Myciel6	95	0.17	5	5 571	0	0	0	0	1	20	20	1859.86
			7	5 928	0	0	0	0	1	30	30	1192.29
			9	12 483	314	111	408	7	13	40	40	2233.89
Myciel7	191	0.13	5	2 827	0	0	0	–	1	–	40	–
			7	2 852	0	0	0	–	1	–	60	–
			9	3 513	0	0	0	–	1	–	80	–
Queen12_12	144	0.51	5	2 150	0	0	0	–	1	–	32	–
			7	2 357	0	0	0	–	1	–	48	–
			9	2 728	0	0	0	–	1	–	64	–
Queen13_13	169	0.47	5	1 840	0	0	0	–	1	–	36	–
			7	1 836	0	0	0	–	1	–	54	–
			9	2 330	0	0	0	–	1	–	72	–
Queen14_14	196	0.44	5	1 572	0	0	0	–	1	–	40	–
			7	1 800	0	0	0	–	1	–	60	–
			9	1 943	0	0	0	–	1	–	80	–

Table 4
Results with the Branch-and-Cut-and-Price algorithm of the terminal-set formulation on random instances.

Instance	n	Den	k	Cols	ST	TC	TT	Gap	No	Opt	Heur	CPU
r_100	100	0.46	5	1871	0	0	0	0	1	20	20	815.07
			7	1758	0	0	0	0	1	30	30	1018.51
			9	3215	0	0	0	–	1	–	40	–
r_200	200	0.46	5	1461	0	0	0	–	1	–	40	–
			7	1545	0	0	0	–	1	–	60	–
			9	1569	0	0	0	–	1	–	80	–
r_300	300	0.46	5	1196	0	0	0	–	1	–	60	–
			7	1253	0	0	0	–	1	–	90	–
			9	1381	0	0	0	–	1	–	120	–
r_400	400	0.45	5	1131	0	0	0	–	1	–	80	–
			7	1234	0	0	0	–	1	–	120	–
			9	1337	0	0	0	–	1	–	160	–
r_500	500	0.45	5	1061	0	0	0	–	1	–	100	–
			7	1166	0	0	0	–	1	–	150	–
			9	1302	0	0	0	–	1	–	200	–
r_600	600	0.45	5	1161	0	0	0	–	1	–	120	–
			7	1203	0	0	0	–	1	–	180	–
			9	1267	0	0	0	–	1	–	240	–
r_700	700	0.45	5	1166	0	0	0	–	1	–	140	–
			7	1240	0	0	0	–	1	–	210	–
			9	1277	0	0	0	–	1	–	280	–
r_800	800	0.45	5	1201	0	0	0	–	1	–	160	–
			7	1249	0	0	0	–	1	–	240	–
			9	1305	0	0	0	–	1	–	320	–
r_900	900	0.45	5	1251	0	0	0	–	1	–	180	–
			7	1293	0	0	0	–	1	–	270	–
			9	1324	0	0	0	–	1	–	360	–
r_1000	1000	0.45	5	1306	0	0	0	–	1	–	200	–
			7	1344	0	0	0	–	1	–	300	–
			9	1388	0	0	0	–	1	–	400	–

in such a way that they respect Hypotheses 1–5 stated in Section 1. The instances are available on [3]. The primal bound is the value associated with any multi-terminal vertex separator of G. We use the heuristic in Algorithm 2 to solve the MTVSP for generating a good primal bound.

Table 5
Results with the Branch-and-Price algorithm of the isolating-separator formulation on DIMACS instances.

Instance	n	Den	k	Cols	Gap	No	Opt	Heur	CPU
anna	138	0.11	5	52 690	29	3872	–	22	–
			7	46 454	45	2804	32	35	–
			9	45 501	64	3908	45	46	–
david	87	0.22	5	6 836	52	835	19	20	25.68
			7	15 777	45	1289	24	30	171.37
			9	57 523	41	7140	29	31	3435.23
DSJC1000.1	1000	0.20	5	1 263	60	351	200	200	35.96
			7	8 238	71	8572	–	300	–
			9	12 118	78	3987	–	400	–
DSJC125.1	125	0.10	5	5 186	60	225	28	28	24.50
			7	40 842	71	666	–	42	–
			9	31 485	78	498	–	56	–
DSJR500.1	500	0.06	5	11 772	60	80	100	100	–
			7	12 459	52	124	150	150	–
			9	4 670	53	92	–	200	–
games120	120	0.18	5	8 683	53	141	23	24	59.27
			7	40 875	67	537	–	35	–
			9	35 024	78	471	48	48	–
huck	74	0.23	5	162	29	17	11	13	0.06
			7	421	31	112	17	20	0.42
			9	500	35	240	23	28	0.94
inithx.i.1	864	0.05	5	10 072	57	1585	109	109	386.87
			7	29 762	66	1130	159	159	–
			9	26 054	67	2464	–	201	–
inithx.i.2	645	0.07	5	32 786	49	3758	–	106	–
			7	25 779	35	4839	–	137	–
			9	22 070	32	4089	181	229	–
inithx.i.3	621	0.07	5	31 621	52	3938	–	111	–
			7	24 577	29	8399	–	134	–
			9	20 776	33	9363	–	181	–
jean	80	0.17	5	656	20	150	12	16	0.46
			7	9 128	36	1287	19	23	46.34
			9	2 893	31	872	21	25	8.83
miles1000	128	0.80	5	334	60	67	28	28	0.62
			7	1 117	71	473	42	42	5.76
			9	5 954	78	1387	56	56	80.25
mulsol.i.1	197	0.21	5	355	57	47	33	33	0.38
			7	848	67	157	46	46	2.08
			9	2 026	74	1927	59	59	32.71
mulsol.i.2	188	0.22	5	510	59	71	39	39	0.60
			7	3 569	70	1241	57	57	26.92
			9	29 370	56	12491	68	77	2574.64
mulsol.i.3	184	0.24	5	485	60	67	40	40	0.57
			7	2 932	62	1165	56	59	19.04
			9	26 439	50	13395	66	78	2506.98
myciel6	95	0.17	5	203	60	29	20	20	0.07
			7	941	71	77	30	30	0.88
			9	12 986	78	379	40	40	181.59
myciel7	191	0.13	5	543	60	59	40	40	0.56
			7	13 099	71	389	60	60	323.29
			9	31 991	78	455	–	80	–
queen12_12	144	0.51	5	267	60	47	32	32	0.31
			7	1 127	71	247	48	48	3.62
			9	10 988	78	1113	64	64	248.29
queen13_13	169	0.47	5	317	60	53	36	36	0.48
			7	1 561	71	333	54	54	7.36
			9	14 193	78	1421	72	72	471.86
queen14_14	196	0.44	5	378	60	59	40	40	0.80
			7	1 637	71	395	60	60	11.76
			9	16 291	78	1787	80	80	733.82

In the column generation phases, we stop generating variables, either when there is no column to add, or when the Lagrangian bound is equal to the objective value.

For the Branch-and-Cut-and-Price algorithms, a cutting phase is considered, using the valid inequalities (38), (39) and (40) at the end of each column generation phase, if the gap between the linear relaxation value and the best primal bound is greater than or equal to 1. We perform the separation of all cuts at each iteration. We use the separation algorithms presented in [11]. The Branching tree is managed by our algorithms using a depth first search. All flow problems are

Table 6
Results with the Branch-and-Price algorithm of the isolating-separator formulation on random instances.

Instance	n	Den	k	Cols	Gap	No	Opt	Heur	CPU
r_100	100	0.46	5	161	60	29	20	20	0.08
			7	199	71	49	30	30	0.21
			9	264	78	69	40	40	0.32
r_200	200	0.46	5	267	60	59	40	40	0.61
			7	353	71	99	60	60	1.74
			9	431	78	139	80	80	3.05
r_300	300	0.46	5	372	60	89	60	60	2.10
			7	451	71	149	90	90	5.24
			9	522	78	209	120	120	8.89
r_400	400	0.45	5	456	60	119	80	80	3.96
			7	535	71	199	120	120	11.22
			9	609	78	279	160	160	17.08
r_500	500	0.45	5	583	60	149	100	100	8.27
			7	624	71	249	150	150	21.33
			9	730	78	349	200	200	40.22
r_600	600	0.45	5	664	60	179	120	120	32.46
			7	692	71	299	180	180	41.97
			9	796	78	419	240	240	76.27
r_700	700	0.45	5	764	60	209	140	140	61.52
			7	790	71	349	210	210	52.30
			9	851	78	489	280	280	73.20
r_800	800	0.45	5	869	60	239	160	160	35.54
			7	876	71	399	240	240	107.89
			9	923	78	559	320	320	108.79
r_900	900	0.45	5	952	60	269	180	180	40.02
			7	946	71	449	270	270	111.06
			9	1042	78	629	360	360	181.27
r_1000	1000	0.45	5	1037	60	299	200	200	163.15
			7	1067	71	499	300	300	500.30
			9	1152	78	699	400	400	323.08

solved using Lemon library [4], and the linear programs are solved using Cplex [5]. The numerical experiments were done on an Intel Core Processor 2.39 GHz $\times 24$ with 20Gb RAM, running under Linux 64 bits. The maximum CPU run time has been fixed to 1 hour.

In the different Tables given below, we have the following entries.

n : the number of vertices in V.
Den : the density of the graph.
k : the number of terminal in T.
Cols : the number of columns added during the Branch-and-Price or the Branch-and-Cut-and-Price algorithms.
ST : the number of the star tree inequalities generated.
TC : the number of the terminal cycle inequalities generated.
TT : the number of the terminal tree inequalities generated.
Gap %: the relative error between the best upper and lower bound obtained at the root node of the branching tree.
No : the number of nodes in the branching tree.
Opt : the optimal objective value.
Heur : the cardinality of the MTV separator given by the heuristic in Algorithm 2.
CPU : the CPU time in seconds, given by the Branch-and-Price or the Branch-and-Cut-and-Price algorithm.

Tables 1–12 concern the Dimacs and random instances which are induced by graphs having up to 1000 nodes, and whose number of terminals is fixed to 5, 7 and 9. The symbol "-" in the tables means that the corresponding instance has not been solved in the time limit. Tables 1 and 2 report results obtained by the Branch-and-Price algorithm with the terminal-set formulation for the DIMACS and the random instances, respectively. As it appears from the tables, the algorithm could not solve most of the instances in the time limit. Only instances of less than 150 nodes could be solved, and the CPU time is high. We also notice that the number of generated columns is high for both types of instances, more than 5000 in most of the solved instances. We remark that the instance "games120" has been solved with 9 terminals but not with 5 and 7 terminals. Moreover, on "huck", the higher is the number of terminals, the faster is the algorithm.

Tables 3 and 4 report results obtained by the Branch-and-Cut-and-Price algorithm with the terminal-set formulation for the DIMACS and the random instances, respectively. The results obtained from this algorithm are similar to those of the Branch-and-Price one. This behavior is explained by the fact that the cuts are only separated when the gap is greater than or equal to 1.0. Since, in most of the instances, the column generation did not manage to decrease enough the linear relaxation within the time limit, no cut has been generated.

Table 7
Results with the Branch-and-Cut-and-Price algorithm of the isolating-separator formulation on DIMACS instances.

Instance	n	Den	k	Cols	ST	TC	TT	Gap	No	Opt	Heur	CPU
anna	138	0.11	5	2 126	149	0	180	8	116	21	28	11.70
			7	2 608	109	0	213	2	161	25	35	18.79
			9	23 947	172	0	1703	6	2202	32	46	1447.35
david	87	0.22	5	376	81	0	102	9	14	19	20	0.71
			7	2 074	156	0	412	16	159	24	30	9.86
			9	7 940	102	0	1145	20	637	29	31	102.28
DSJC1000.1	1000	0.20	5	1 479	318	74	463	1	5	200	200	136.45
			7	2 862	1359	393	1728	60	6	–	300	–
			9	3 469	1390	310	1893	70	76	–	400	–
DSJC125.1	125	0.10	5	492	43	30	58	0	1	28	28	1.18
			7	421	149	23	232	0	1	42	42	2.83
			9	35 821	2123	4	1127	19	666	–	56	–
DSJR500.1	500	0.06	5	2 855	769	26	1015	0	1	100	100	438.42
			7	4 116	1696	9	803	5	16	150	150	–
			9	4 543	1500	0	10	19	12	200	200	–
games120	120	0.18	5	283	83	0	95	0	1	23	24	0.50
			7	359	138	0	210	0	1	34	36	2.03
			9	19 315	762	0	1266	14	330	46	48	1533.08
Huck	74	0.23	5	135	13	0	36	3	8	11	13	0.07
			7	257	11	0	75	10	63	17	20	0.37
			9	423	20	0	167	11	48	23	28	0.98
inithx.i.1	864	0.05	5	1 853	345	0	456	53	67	109	109	48.97
			7	14 371	1451	0	597	21	649	–	158	–
			9	11 022	2400	0	619	13	356	–	181	–
inithx.i.2	645	0.07	5	2 699	775	0	136	2	44	90	111	85.79
			7	13 585	2385	0	146	14	662	–	128	–
			9	11 290	2030	0	234	18	547	–	164	–
inithx.i.3	621	0.07	5	2 761	942	0	115	1	75	91	115	98.11
			7	13 741	3451	0	434	5	413	117	176	–
			9	15 198	1756	0	287	19	1490	164	236	–
jean	80	0.17	5	156	12	0	33	0	1	12	16	0.08
			7	542	38	0	103	9	58	19	23	0.98
			9	1 135	13	0	183	13	262	21	25	3.07
Miles1000	128	0.80	5	186	20	0	31	0	1	28	28	0.20
			7	237	58	0	85	5	3	42	42	0.70
			9	430	71	0	155	12	15	56	56	2.43
Mulsol.i.1	197	0.21	5	340	36	0	50	37	41	33	33	0.67
			7	392	50	0	84	27	37	46	46	1.16
			9	717	72	0	159	38	61	59	59	4.10
Mulsol.i.2	188	0.22	5	283	58	0	78	0	1	39	39	0.41
			7	785	166	0	216	15	63	57	57	4.89
			9	2 099	179	0	264	11	393	68	77	22.47
Mulsol.i.3	184	0.24	5	275	63	0	79	7	15	40	40	0.50
			7	608	135	0	225	10	88	56	59	3.74
			9	1 030	116	0	191	6	101	66	78	8.09
Myciel6	95	0.17	5	235	24	3	30	0	1	20	20	0.15
			7	319	75	0	133	5	3	30	30	0.63
			9	2 626	257	1	451	20	73	40	40	18.91
Myciel7	191	0.13	5	514	67	11	87	0	1	40	40	1.41
			7	509	169	6	261	0	1	60	60	5.70
			9	24 606	1771	4	1113	15	415	80	80	–
Queen12_12	144	0.51	5	224	24	0	35	0	1	32	32	0.26
			7	308	75	0	123	0	1	48	48	1.19
			9	416	110	0	234	2	3	64	64	2.63
Queen13_13	169	0.47	5	242	31	0	45	0	1	36	36	0.34
			7	337	82	0	138	0	1	54	54	1.84
			9	440	123	0	237	0	1	72	72	3.61
Queen14_14	196	0.44	5	291	36	0	51	0	1	40	40	0.61
			7	402	95	0	151	0	1	60	60	2.78
			9	449	133	0	236	0	1	80	80	4.50

The results obtained by the Branch-and-Price algorithm of the isolating-separator formulation are shown in Tables 5 and 6. Here, the algorithm could solve 77% of the instances in the time limit and 50% of the instances in less than 60 seconds. The Gap is around 60% in most of the instances. This may be explained by the fact that the linear relaxation is weak. Also, the number of nodes in the branching tree is not less than 100 in most of the cases. However, it increases with the size of the instance. Contrary to the terminal-set formulation, in most of the instances, the higher is the number of terminals, slower is the algorithm.

Table 8
Results with the Branch-and-Cut-and-Price algorithm of the isolating-separator formulation on random instances.

Instance	n	Den	k	Cols	ST	TC	TT	Gap	No	Opt	Heur	CPU
r_100	100	0.46	5	126	11	0	15	0	1	20	20	0.05
			7	160	23	0	43	0	1	30	30	0.19
			9	211	43	0	84	8	7	40	40	0.49
r_200	200	0.46	5	255	29	0	37	51	3	40	40	0.56
			7	306	52	0	82	21	15	60	60	1.99
			9	430	113	1	212	0	1	80	80	5.23
r_300	300	0.46	5	360	36	1	49	48	65	60	60	3.40
			7	482	86	0	136	55	25	90	90	9.65
			9	843	187	1	353	55	33	120	120	32.84
r_400	400	0.45	5	503	61	0	74	56	85	80	80	9.45
			7	629	96	0	160	64	165	120	120	26.09
			9	1319	265	2	496	56	145	160	160	109.59
r_500	500	0.45	5	574	77	1	99	55	139	100	100	25.35
			7	796	137	0	217	64	273	150	150	67.17
			9	1856	371	0	722	71	407	200	200	357.97
r_600	600	0.45	5	700	81	1	110	57	159	120	120	46.07
			7	931	181	0	334	66	481	180	180	185.90
			9	1881	458	0	899	70	779	240	240	762.09
r_700	700	0.45	5	785	97	1	133	58	247	140	140	98.15
			7	996	228	0	356	62	573	210	210	373.21
			9	2065	553	2	1037	69	925	280	280	1382.35
r_800	800	0.45	5	874	124	3	164	57	311	160	160	157.91
			7	1106	258	2	423	67	763	240	240	686.76
			9	2302	675	3	1291	69	1385	320	320	2790.30
r_900	900	0.45	5	1016	144	2	208	55	321	180	180	396.10
			7	1190	299	1	463	66	853	270	270	1079.91
			9	1817	690	2	1290	73	1750	360	360	–
r_1000	1000	0.45	5	1080	160	0	227	59	413	200	200	508.66
			7	1320	259	2	406	68	721	300	300	1663.08
			9	1438	428	0	909	73	1471	400	400	–

Table 9
Results with the Branch-and-Price algorithm of the two-terminal vertex separator formulation on DIMACS instances.

Instance	n	Den	k	Cols	Gap	No	Opt	Heur	CPU
anna	138	0.11	5	42 090	24	3 158	21	28	1542.94
			7	50 270	4	2 738	–	27	–
			9	53 441	59	5 367	–	46	–
david	87	0.22	5	7 403	52	937	19	20	27.08
			7	17 082	45	1 870	24	30	176.49
			9	60 628	41	6 687	29	31	2546.66
DSJC1000.1	1000	0.20	5	7 623	60	2 821	200	200	1441.53
			7	10 270	71	1 548	300	300	–
			9	3 958	78	240	400	400	–
DSJC125.1	125	0.10	5	9 739	60	181	28	28	129.79
			7	29 019	71	276	–	42	–
			9	26 495	78	272	–	56	–
DSJR500.1	500	0.06	5	8 699	60	79	–	100	–
			7	3 556	71	23	–	150	–
			9	3 862	–	1	200	200	–
games120	120	0.18	5	9 046	53	129	23	24	130.03
			7	28 206	67	224	–	35	–
			9	22 127	78	175	48	48	–
huck	74	0.23	5	179	29	17	11	13	0.11
			7	468	31	79	17	20	0.59
			9	1 937	35	256	23	28	6.17
inithx.i.1	864	0.05	5	40 932	–	2 480	109	109	–
			7	18 119	66	671	159	159	–
			9	14 662	73	393	–	208	–
inithx.i.2	645	0.07	5	31 976	54	1 628	109	111	–
			7	12 097	31	123	133	172	–
			9	17 557	55	1 137	202	229	–
inithx.i.3	621	0.07	5	28 392	55	1 705	113	115	–
			7	19 733	38	1 385	143	176	–
			9	7 695	77	40	236	236	–

(*continued on next page*)

Table 9 (*continued*).

Instance	n	Den	k	Cols	Gap	No	Opt	Heur	CPU
jean	80	0.17	5	790	20	146	12	16	0.84
			7	5 533	36	685	19	23	14.77
			9	2 732	31	549	21	25	8.15
miles1000	128	0.80	5	353	60	59	28	28	0.88
			7	2 015	71	425	42	42	15.82
			9	8 545	78	1 427	56	56	210.60
mulsol.i.1	197	0.21	5	353	57	47	33	33	0.61
			7	880	67	147	46	46	3.44
			9	3 711	74	1 907	59	59	61.71
mulsol.i.2	188	0.22	5	549	59	71	39	39	0.95
			7	5 445	70	1 045	57	57	58.17
			9	42 971	72	8 445	–	75	–
mulsol.i.3	184	0.24	5	519	60	65	40	40	0.94
			7	5 821	62	1 449	56	59	77.37
			9	38 032	73	12 063	76	78	–
myciel6	95	0.17	5	236	60	29	20	20	0.11
			7	1 337	71	75	30	30	2.62
			9	11 362	78	395	40	40	147.71
myciel7	191	0.13	5	495	60	59	40	40	0.68
			7	29 237	71	409	60	60	2953.72
			9	25 961	78	349	–	80	–
queen12_12	144	0.51	5	277	60	47	32	32	0.43
			7	1 925	71	221	48	48	10.75
			9	21 630	78	1 199	64	64	3496.65
queen13_13	169	0.47	5	327	60	53	36	36	0.69
			7	3 013	71	305	54	54	32.58
			9	19 903	78	1 017	72	72	–
queen14_14	196	0.44	5	374	60	59	40	40	1.11
			7	3 009	71	341	60	60	40.44
			9	16 459	78	695	–	80	–

Table 10
Results with the Branch-and-Price algorithm of the two-terminal vertex separator formulation on random instances.

Instance	n	Den	k	Cols	Gap	No	Opt	Heur	CPU
r_100	100	0.46	5	167	60	29	20	20	0.10
			7	300	71	49	30	30	00.45
			9	526	78	69	40	40	1.41
r_200	200	0.46	5	295	60	59	40	40	1.18
			7	415	71	99	60	60	3.59
			9	847	78	139	80	80	11.87
r_300	300	0.46	5	405	60	89	60	60	3.67
			7	633	71	149	90	90	14.47
			9	1129	78	209	120	120	41.35
r_400	400	0.45	5	518	60	119	80	80	8.82
			7	828	71	199	120	120	30.75
			9	1386	78	279	160	160	92.18
r_500	500	0.45	5	628	60	149	100	100	18.26
			7	1008	71	249	150	150	61.67
			9	1592	78	349	200	200	158.85
r_600	600	0.45	5	724	60	179	120	120	66.31
			7	1126	71	299	180	180	210.62
			9	1965	78	419	240	240	665.79
r_700	700	0.45	5	855	60	209	140	140	89.45
			7	1350	71	349	210	210	599.57
			9	2217	78	489	280	280	1316.14
r_800	800	0.45	5	948	60	239	160	160	181.15
			7	1514	71	399	240	240	678.95
			9	2498	78	559	320	320	862.90
r_900	900	0.45	5	1080	60	269	180	180	126.82
			7	1632	71	449	270	270	881.48
			9	2840	78	629	360	360	2345.08
r_1000	1000	0.45	5	1220	60	299	200	200	226.10
			7	1784	71	499	300	300	647.77
			9	2161	78	548	–	400	–

Tables 7 and 8 concern results obtained by the Branch-and-Cut-and-Price algorithm based on the isolating-separator formulation. As it appears, the algorithm could solve 84% of the instances and 60% in less than 60 seconds, which is better

Table 11
Results with the Branch-and-Cut-and-Price algorithm of the two-terminal vertex separator formulation on DIMACS instances.

Instance	n	Den	k	Cols	ST	TC	TT	Gap	No	Opt	Heur	CPU
anna	138	0.11	5	2 436	190	0	240	7	191	21	28	12.10
			7	2 599	107	0	245	2	160	25	35	20.88
			9	28 061	206	0	1934	6	2120	32	46	1447.35
david	87	0.22	5	397	73	0	122	13	19	19	20	0.72
			7	1 616	129	0	345	15	175	24	30	8.31
			9	6 683	94	0	1048	17	627	29	31	81.42
DSJC1000.1	1000	0.20	5	1 560	321	78	468	5	101	200	200	235.05
			7	2 665	1241	363	1654	14	18	300	300	–
			9	2 383	989	183	1342	28	13	400	400	–
DSJC125.1	125	0.10	5	262	41	29	65	0	1	28	28	0.52
			7	394	133	30	219	0	1	42	42	3.34
			9	40 953	1689	8	1511	28	1207	56	56	–
DSJR500.1	500	0.06	5	1 671	761	21	1005	0	1	100	100	337.15
			7	6 003	1632	18	1627	6	27	–	150	–
			9	8 693	2175	8	1036	20	102	200	200	–
games120	120	0.18	5	296	80	0	100	0	1	23	24	0.66
			7	713	152	1	233	6	9	34	36	5.02
			9	17 127	595	0	1198	15	315	46	48	784.69
huck	74	0.23	5	135	8	0	25	13	8	11	13	0.08
			7	353	18	0	71	11	56	17	20	0.59
			9	723	30	0	176	10	45	23	28	2.47
inithx.i.1	864	0.05	5	877	0	0	0	0	1	109	109	0.35
			7	10 839	2957	0	584	14	173	153	159	–
			9	9 794	2141	0	608	27	215	198	208	–
inithx.i.2	645	0.07	5	3 403	1076	0	186	0	1	90	111	128.62
			7	11 771	3072	0	142	13	184	128	172	–
			9	14 211	1126	0	329	35	651	201	229	–
inithx.i.3	621	0.07	5	4 262	1640	0	135	1	70	91	115	237.13
			7	14 681	4026	0	436	6	402	127	176	–
			9	10 069	2882	0	264	70	82	–	236	–
jean	80	0.17	5	157	16	0	51	6	8	12	16	0.09
			7	591	28	0	136	11	51	19	23	1.28
			9	1 424	23	0	205	14	183	21	25	5.80
miles1000	128	0.80	5	186	23	0	34	0	1	28	28	0.24
			7	308	52	0	91	6	7	42	42	1.45
			9	599	74	0	178	5	15	56	56	6.07
mulsol.i.1	197	0.21	5	209	0	0	0	0	1	33	33	0.03
			7	440	46	0	106	23	27	46	46	2.09
			9	1 225	102	0	194	26	105	59	59	16.73
mulsol.i.2	188	0.22	5	285	53	0	81	4	5	39	39	0.52
			7	1 178	188	0	275	17	103	57	57	12.98
			9	4 612	248	0	312	8	336	68	77	115.37
mulsol.i.3	184	0.24	5	360	51	0	86	45	59	40	40	1.11
			7	1 112	123	0	234	11	111	56	59	11.31
			9	1 678	215	0	251	6	103	67	78	34.72
myciel6	95	0.17	5	167	22	5	30	36	15	20	20	0.12
			7	365	51	1	107	29	9	30	30	0.97
			9	2 398	141	0	391	21	97	40	40	25.08
myciel7	191	0.13	5	523	66	11	96	35	5	40	40	1.76
			7	576	194	7	309	2	3	60	60	8.50
			9	34 633	1195	6	1262	19	466	–	80	–
queen12_12	144	0.51	5	256	19	0	38	35	3	32	32	0.37
			7	306	66	2	122	0	1	48	48	1.47
			9	522	97	1	226	4	9	64	64	5.06
queen13_13	169	0.47	5	230	32	0	44	0	1	36	36	0.32
			7	356	82	0	146	0	1	54	54	2.45
			9	663	106	0	227	3	11	72	72	9.42
queen14_14	196	0.44	5	313	29	0	55	31	5	40	40	0.71
			7	567	94	0	169	9	15	60	60	6.97
			9	500	124	0	245	3	11	80	80	8.34

than the Branch-and-Price algorithm. The Gap is equal to 0% in 24% of the instances while it is less than or equal to 10% in 45% of instances. This implies that the linear relaxation here is quite strong. This is also clear from the fact that several instances are solved in the root node of the branching tree. Concerning the new valid inequalities used here, as we can see, few terminal-cycle inequalities are generated for most of the instances. However, the terminal tree inequalities appear quite useful for solving the problem. Comparing these results with the ones of Tables 5 and 6 obtained by the Branch-and-Price algorithm of the isolating-separator formulation, we can notice that the added inequalities, in particular the

Table 12
Results with the Branch-and-Cut-and-Price algorithm of the two-terminal vertex separator formulation on random instances.

Instance	n	Den	k	Cols	ST	TC	TT	Gap	No	Opt	Heur	CPU
r_100	100	0.46	5	153	10	0	15	32	11	20	20	0.10
			7	174	18	0	35	4	5	30	30	0.22
			9	240	44	0	83	0	1	40	40	0.78
r_200	200	0.46	5	302	30	0	45	40	43	40	40	1.46
			7	349	59	0	95	28	17	60	60	3.11
			9	622	122	0	234	18	63	80	80	22.19
r_300	300	0.46	5	390	43	1	57	41	63	60	60	4.88
			7	480	103	0	152	34	145	90	90	23.70
			9	878	202	0	358	39	79	120	120	87.13
r_400	400	0.45	5	509	55	1	85	53	107	80	80	13.98
			7	621	120	0	195	47	167	120	120	54.82
			9	1192	323	3	591	49	425	160	160	454.01
r_500	500	0.45	5	573	78	1	108	54	197	100	100	33.26
			7	722	170	0	260	53	455	150	150	171.17
			9	1281	415	1	783	54	819	200	200	1283.63
r_600	600	0.45	5	686	83	1	128	55	243	120	120	47.19
			7	868	203	1	350	56	627	180	180	288.71
			9	1511	493	2	1020	57	1139	240	240	3011.21
r_700	700	0.45	5	779	102	0	151	55	293	140	140	184.30
			7	983	217	0	370	54	605	210	210	569.52
			9	1346	431	0	820	57	1028	280	280	–
r_800	800	0.45	5	879	126	2	180	56	355	160	160	357.02
			7	1117	300	1	472	61	819	240	240	2154.49
			9	1253	348	0	673	63	902	320	320	–
r_900	900	0.45	5	1022	142	0	188	57	335	180	180	896.72
			7	1173	278	2	444	56	767	270	270	2487.06
			9	1275	253	0	526	64	633	–	360	–
r_1000	1000	0.45	5	1097	154	1	222	57	429	200	200	648.34
			7	1279	282	0	483	62	846	–	300	–
			9	1293	164	0	371	64	392	–	400	–

terminal tree inequalities, have significatively strengthened the linear relaxation of the problem. These also permitted to decrease the number of generated columns that do not exceed here 3000 for most of the instances. Moreover, the Branch-and-Cut-and-Price algorithm showed a better performance than the Branch-and-Price algorithm. One can observe that the latter could not solve the instance "anna" with 5, 7, and 9 terminals within the time limit. Moreover, it solved "david" with 9 terminals in 3435.23 s. Whereas, the Branch-and-Cut-and-Price solved the "anna" instances in less than 1448 seconds (the one with 5 and 7 has been solved in less than 20 seconds), and "david" with 9 terminals in 102.28 s.

In Tables 9 and 10 are reported the results obtained by the Branch-and-Price algorithm based on the two-terminal vertex separator formulation. Here the algorithm was able to solve around 72% of DIMACS instances within the time limit of 1 hour and around 46% in less than 2 minutes. However, the Gap is high, around 60% for most of the instances. Thus it also appears that the linear relaxation of the two-terminal vertex separator formulation is not sufficiently strong.

The numerical results in Tables 3, 4, 9 and 10 confirm Theorem 4.3. Indeed, we notice that the gap given by the two-terminal vertex separator formulation is always equal to the gap given by the isolation-separator formulation.

Tables 11 and 12 are related to the Branch-and-Cut-and-Price algorithm based on the two-terminal vertex separator formulation. As it appears, the algorithm could solve 81% of the instances. Moreover, 13 instances have been solved in the root node, which implies that the linear relaxation here is sufficiently strong. One can also observe that the cycle inequalities played virtually no role in the resolution of the problem comparing to the terminal tree and the star tree inequalities, which have shown to be useful. Moreover, comparing with the Branch-and-Price algorithm of the two-terminal vertex separator formulation, it appears that the added inequalities have been very efficient and permitted to considerably reduce both the CPU time and the number of generated columns.

Overall tables, we notice that the heuristic gives the optimal solution in most cases.

7. Conclusion

In this paper, we have proposed three extended integer programming formulations for the MTVSP. For each of the formulations, we have developed a Branch-and-Price algorithm and presented extensive computational results. And for two of the formulations, we have added three families of valid inequalities presented in [11], to develop Branch-and-Cut-and-Price algorithms. For all the formulations, we have shown how to compute the dual bound within the column generation phase and analyzed the complexity of the pricing problem, the branching scheme, and the strategy of the column generation. The computational results have permitted to measure the impact of each formulation and approach. They have shown that the Branch-and-Cut-and-Price algorithms perform better than the Branch-and-Price ones, and than the Branch-and-Cut algorithm [11] for more realistic DIMACS instances. The added cuts have been very effective,

in particular, for the isolating-separator formulation. Big-sized random instances could be solved using this algorithm. However, the Branch-and-Cut algorithm was shown to be more efficient for large random instances.

Further variants of the multi-terminal vertex separator problem are of practical interest and can also be considered. One variant would be an extension of the MTVSP to the balanced case, that is to say when the k elements of the partition, containing the terminals, have balanced cardinalities. Another variant would be the edge version of the MTVSP. Here the problem consists in finding an edge subset whose removal partitions the graph into k disjoint subsets, each containing exactly one terminal [12]. It would be interesting to study the polyhedral structures of these problems and to develop cutting plane based approaches for them.

References

[1] http://www.info.univ-angers.fr/pub/porumbel/graphs/.
[2] http://www.boost.org/doc/libs/1_60_0/libs/graph/.
[3] https://www.lamsade.dauphine.fr/~magnouche/Instances.zip.
[4] https://lemon.cs.elte.hu/trac/lemon.
[5] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.
[6] E. Balas, C.C. de Souza, The vertex separator problem: a polyhedral investigation, Math. Program. 103 (3) (2005) 583–608.
[7] W. Ben-Ameur, M. Didi Biha, On the minimum cut separator problem, Networks 59 (1) (2012) 30–36.
[8] W. Ben-Ameur, M.A.M. Sidi, J. Neto, The k-separator problem, in: COCOON'13: 19th International Computing & Combinatorics Conference, vol. 7936, Springer, 2013, pp. 337–348.
[9] M.D. Biha, M.-j. Meurs, An exact algorithm for solving the vertex separator problem, J. Global Optim. 49 (3) (2011) 425–434.
[10] D. Cornaz, F. Furini, M. Lacroix, E. Malaguti, A.R. Mahjoub, S. Martin, Mathematical formulations for the balanced vertex k-separator problem, in: Control, Decision and Information Technologies (CoDIT), 2014 International Conference on, IEEE, 2014, pp. 176–181.
[11] D. Cornaz, Y. Magnouche, A.R. Mahjoub, S. Martin, The multi-terminal vertex separator problem: Polyhedral analysis and branch-and-cut, Discrete Appl. Math. 256 (2019) 11–37.
[12] E. Dahlhaus, d.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, The complexity of multiterminal cuts, SIAM J. Comput. 23 (4) (1994) 864–894.
[13] G. Desaulniers, J. Desrosiers, M.M. Solomon, Column Generation, vol. 5, Springer Science & Business Media, 2006, p. 358.
[14] N. Garg, V.V. Vazirani, M. Yannakakis, Multiway cuts in directed and node weighted graphs, in: Automata, Languages and Programming, Springer, 1994, pp. 487–498.
[15] X. Ji, J.E. Mitchell, Branch-and-price-and-cut on the clique partitioning problem with minimum clique size requirement, Discrete Optim. 4 (1) (2007) 87–102.
[16] K. Menger, Zur allgemeinen kurventheorie, Fund. Math. 10 (1) (1927) 96–115.
[17] G. Naves, V. Jost, The graphs with the max-mader-flow-min-multiway-cut property, 2011, arXiv preprint arXiv:1101.2061.
[18] C. de Souza, E. Balas, The vertex separator problem: algorithms and computations, Math. Program. 103 (3) (2005) 609–631.