Discrete Applied Mathematics 256 (2019) 11–37

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

The multi-terminal vertex separator problem: Polyhedral analysis and Branch-and-Cut

D. Cornaz [a], 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, optimization et modélisation des systèmes, Université de Lorraine, Metz, France

ARTICLE INFO

Article history:
Received 20 March 2017
Received in revised form 16 September 2018
Accepted 1 October 2018
Available online 30 January 2019

Keywords:
Vertex separator problem
Integer programming
Polytope
Facet
Branch-and-Cut algorithm
Complexity
Separation algorithm

ABSTRACT

In this paper we consider a variant of the k-separator problem. Given a graph $G = (V \cup T, E)$ with $V \cup T$ the set of vertices, where T is a set of k terminals, the multi-terminal vertex separator problem consists in partitioning $V \cup T$ into $k + 1$ subsets $\{S, V_1, \ldots, V_k\}$ such that there is no edge between two different subsets V_i and V_j, each V_i contains exactly one terminal and the size of S is minimum. In this paper, we first show that the problem is NP-hard. Then we give two integer programming formulations for the problem. For one of these formulations, we investigate the related polyhedron and discuss its polyhedral structure. We describe some valid inequalities and characterize when these inequalities define facets. We also derive separation algorithms for these inequalities. Using these results, we develop a Branch-and-Cut algorithm for the problem, along with an extensive computational study.

1. Introduction

Let $G = (V \cup T, E)$ be a simple graph with $V \cup T$ the set of vertices, where T is a set of k distinguished vertices called *terminals*, and E the set of edges. A *multi-terminal vertex separator* in G is a subset of vertices such that the graph induced by $(V \cup T) \setminus S$ consists of k 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. The MTVSP can also be seen as the problem of finding a vertex subset $S \subseteq V$ of minimum size such that each path between each pair of terminals intersects S. Indeed, if S intersects all paths between each pair of terminals, then S is a multi-terminal vertex separator, and the graph induced by $(V \cup T) \setminus S$ may be built so that it consists of k components. Moreover, each component contains exactly one terminal. In the case where $k = 2$, the MTVSP can be solved in polynomial time [1].

The MTVSP has applications in different areas like VLSI conception, linear algebra, connectivity problems and parallel algorithms. It has also applications in network security. Consider for instance a connected graph $G = (V \cup T, E)$, representing a telecommunication network where V represents a set of routers, T a set of customers and an edge between two vertices expresses the possibility of transferring data between each other. Suppose that a cost is associated with each vertex. And we search to set up a monitoring system on routers in order to monitor all data exchanged between customers. The set of routers on which the monitoring system is set up is a multi-terminal vertex separator.

In the literature the MTVSP is also called the *vertex multi-terminal cut problem* and the *node multi-way cuts problem* [2–4]. In [5] Cunningham considers the edge version of the MTVSP that consists, given a graph $G = (V \cup T, E)$, and a cost vector

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

https://doi.org/10.1016/j.dam.2018.10.005
0166-218X/

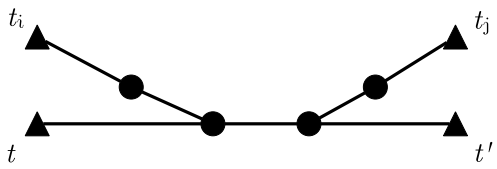

Fig. 1. Minimal terminal path.

$c \in \mathbb{R}^E_+$, in finding a set of edges $E' \subseteq E$ of minimum cost such that $(V \cup T, E \setminus E')$ has $|T| = k$ components and no two terminals are in the same component. He implicitly introduces the MTVSP and shows that it is as hard as the edge version, which is known to be NP-hard.

The MTVSP is a variant of the *vertex k-separator problem* which in turn 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 two subsets V_i and V_j. Many variants of this problem have been considered in the literature [1,6–13]. In [6,11], the authors discuss the vertex separator problem that consists, given a simple graph $G = (V, E)$ and a positive integer $\beta(|V|)$, in partitioning V into three subsets A, B and C such that no vertex in A is adjacent to a vertex in B, $\max\{|A|, |B|\} \leq \beta(|V|)$ and $|C|$ is minimum. They study the facial structure of the associated polytope and develop a Branch-and-Cut algorithm. In [1], another variant of the k-separator problem is considered. Given a simple graph $G = (V, E)$ and two terminals $a, b \in V$, the problem here is to partition V into three subsets A, B and C such that $a \in A$, $b \in B$, there is no edge connecting A and B and the size of the cut induced by C is minimum. It is shown that this problem can be reduced to a minimum cut problem in an auxiliary graph and then can be solved in polynomial time. In [13], the authors give a linear system for the MTVSP and characterize the class of graphs for which the system is totally dual integral. Another variant called the balanced vertex k-separator problem is considered in [10]. Given a graph $G = (V, E)$ and a constant $q \in \mathbb{N}$, the problem consists in partitioning V into $k + 1$ subsets $\{S, V_1, \ldots, V_k\}$ in such a way that $|S|$ is minimum and $|V_i| - |V_j| \leq q$, for all $i, j \in \{1, \ldots, k\}$. The authors propose compact and extended formulations for the problem and develop a Branch-and-Price algorithm. In [7,8] and [12], the authors consider the k-vertex separator problem that consists, given a graph $G = (V, E)$, a constant $k \in \mathbb{N}$ and weights on the vertices, in finding a minimum weight (size) subset of vertices whose removal leads to a graph where the size of each connected component is less than or equal to k. In [7] and [8], the authors show that this problem can be solved in polynomial time for some classes of graphs and present approximation algorithms. In [12], the authors present some approximation algorithms for both the vertex and the edge version of the k-separator problem. They also study the k-Path Transversal problem, where the goal is to remove the minimum number of vertices such that there is no simple path of length k.

In this paper, we show that the MTVSP is NP-hard, we propose two linear programming formulations for the problem and discuss their linear relaxations. We study the properties of the associated polytope, and perform a facial investigation of the basic inequalities. We introduce further valid inequalities and discuss necessary conditions and sufficient conditions for these inequalities to define facets. We also propose a heuristic for the MTVSP problem and analyze their performance guarantee. Several preprocessing operations on the graph are given. Using all these results we develop a Branch-and-Cut algorithm for the multi-terminal vertex separator problem.

The paper is organized as follows. In Section 2 we discuss the complexity aspect of the MTVSP. In Section 3 we give the integer programming formulations for the problem and analyze their linear relaxations. Section 4 is devoted to the polyhedral analysis of the MTVSP and description of some valid inequalities. In Section 5 we discuss some graph reduction operations. In Section 6 we describe separation routines for the inequalities described in Section 4 and develop a Branch-and-Cut algorithm for the MTVSP. Our computational results are presented in Section 7, and finally some concluding remarks are given in Section 8. The rest of this section is devoted to more definitions and notations.

We consider simple graphs. We denote a graph by $G = (V \cup T, E)$, where $V \cup T$ is the set of vertices, T is the set of terminals in G and E is the set of edges. We 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*. If $W \subset V$, we denote by $G[W]$ the subgraph of G induced by W. 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 $N(v)$ called degree of v in G. 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. For $v \in V$, we 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 $x \in \mathbb{R}^{V \cup T}$, we let $x(W) = \sum_{v \in W} x_v$. Given a subgraph H of G, we denote by $V(H)$, $T(H)$ and $E(H)$ its sets of nodes, terminals and edges, respectively. Given an inequality $ax \leq b$, where $a \in \mathbb{R}^{V \cup T}$, the *support graph* of $ax \leq b$ is the subgraph induced by the vertices corresponding to variables having a non-zero coefficient in the inequality. A *path* P is a set of p distinct vertices $v_1, v_2, \ldots, v_p$ such that for all $i \in \{1, \ldots, p-1\}$, $v_i v_{i+1}$ is an edge. Vertices $v_2, \ldots, v_{p-1}$ are called *the internal vertices* of P. Given a path P between two terminals $t, t' \in T$ such that $P \cap T = \{t, t'\}$, the set of internal vertices of P will be called *a terminal path* and denoted by $P_{tt'}$. A terminal path is *minimal* if it does not strictly contain a terminal path. In Fig. 1, terminal path $P_{t_i t_j}$ is not minimal, it contains $P_{tt'}$. (In all figures in this paper, the terminals are represented by triangles.)

For a multi-terminal vertex separator $S \subseteq V$, let $x^S \in \{0, 1\}^V$ be the vector given by $x_v = 1$ if $v \in S$ and $x_v = 0$ otherwise. x^S is called the incidence vector of S.

In what follows we consider the following assumptions.

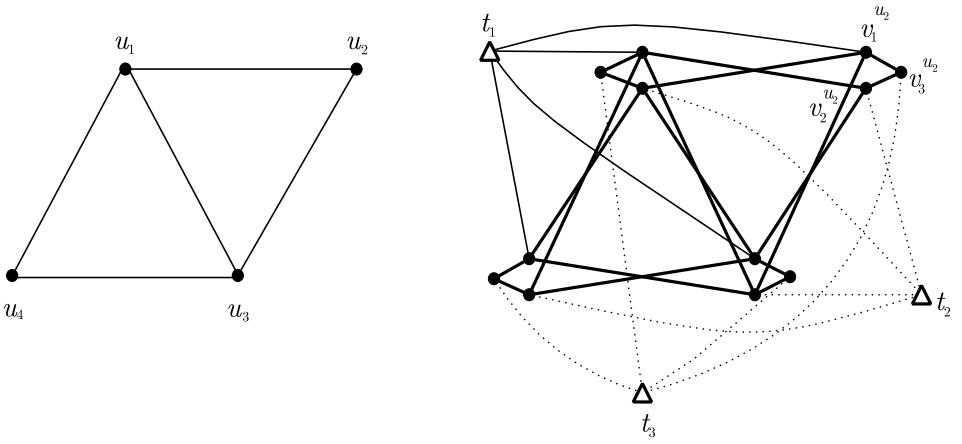

Fig. 2. Transformation from the graph H to the graph G.

Assumption 1- There is no edge between two terminals, otherwise the problem has no solution.
Assumption 2- 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.
Assumption 3- 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, in a transformed directed graph, a minimum weight flow of value 2 between v and an artificial sink connected to all the terminals of T, where the capacities and the weights of the edges are all equal to 1.
Assumption 4- Graph G is connected. Otherwise, we consider the MTVSP on each component of the graph.

Remark that, it is possible to consider an integer weight $w(v)$ on each vertices $v \in V$. In this case, if there exists a vertex v with $w(v) = 0$, then we can add v in the separator and delete it from the graph. If $w(v) \geq 1$ then we can replace v by a clique $K_{w(v)}$ of size $w(v)$ such that each vertex of $K_{w(v)}$ is adjacent to each vertex of $N(v)$. This transformation can be used if the weight is integer and bounded by a constant.

2. Complexity analysis

In [14], Garg et al. show that the MTVSP is NP-hard. In this section we give a simpler proof of this result using a polynomial reduction from the vertex cover problem (VCP). Given a graph $H = (U, E')$, the VCP consists in finding a minimum cardinality subset of vertices $R \subseteq U$ such that each edge of E' is incident to at least one vertex of R. The VCP is a well-known NP-hard problem.

Theorem 2.1. *The MTVSP is NP-hard.*

Proof. Consider the VCP on a graph $H = (U, E')$. Let $G = (V_1 \cup V_2 \cup V_3 \cup T, E)$ be the graph obtained from H as follows

- add three vertices t_1, t_2 and t_3 in T.
- for each vertex $u \in U$, add
  - three vertices v_1^u in V_1, v_2^u in V_2 and v_3^u in V_3.
  - three edges $t_1 v_1^u$, $t_2 v_2^u$ and $t_3 v_3^u$ in E.
  - two edges $v_1^u v_3^u$ and $v_2^u v_3^u$ in E.
- for each edge $uw \in E'$, add two edges $v_1^u v_2^w$ and $v_1^w v_2^u$ in E.

Fig. 2 illustrates the above graph transformation, where the dotted edges are for the clarity of the graph.

Proposition 2.1. *Given a separator S of G and a vertex $u \in U$, either $|\{v_1^u, v_2^u, v_3^u\} \cap S| \geq 2$, $v_3^u \in S$ or both.*

Proof. Let us assume the contrary. We distinguish two cases.

1. $\{v_1^u, v_2^u, v_3^u\} \cap S$ is empty. It then follows that $\{v_2^u, v_3^u\}$ is a terminal path, not intersecting S.
2. $v_3^u \notin S$ and, say, $v_2^u \notin S$. Then $\{v_2^u, v_3^u\}$ is a terminal path between t_2 and t_3 not intersecting S.

The two cases contradict the fact that S is a separator. ■

Let R be a vertex cover in H and S a separator in G. Let R^S be the set of vertices $R^S = \{u \in U : |S \cap \{v_1^u, v_2^u, v_3^u\}| \geq 2\}$, and let $S^R \subseteq V_1 \cup V_2 \cup V_3$ be defined as follows. For each vertex $u \in U$, if $u \in R$ then we add the two vertices v_1^u, v_2^u in S^R, otherwise we add v_3^u in S^R.

Proposition 2.2. *The set R^S is a vertex cover in H and S^R is a separator in G.*

Proof. Suppose that R^S is not a vertex cover in H. Then, there exists $uw \in E'$ such that $R^S \cap \{u, w\} = \emptyset$. From the construction of R^S and Proposition 2.1 it follows that $\{v_1^u, v_2^u, v_1^w, v_2^w\} \cap S = \emptyset$ and $\{v_3^u, v_3^w\} \subseteq S$. As a consequence, $\{v_1^u, v_2^w\}$ is a terminal path, between t_1 and t_2, not intersecting S, and thus S is not a separator, a contradiction. Now, suppose that S^R is not a separator of G. Then, there is a terminal path not intersecting S^R in G. We distinguish two cases.

- There exists an edge $uw \in E'$ such that the terminal path $\{v_1^u, v_2^w\}$ or $\{v_1^w, v_2^u\}$, between t_1 and t_2, does not intersect S^R. From the definition of S^R, it follows that $\{u, w\} \cap R = \emptyset$, a contradiction with the fact that R is a vertex cover of H.
- There exists a vertex $u \in U$ such that the terminal path $\{v_1^u, v_3^u\}$, between t_1 and t_3, or $\{v_2^u, v_3^u\}$, between t_2 and t_3, does not intersect S^R. This implies that either $\{v_1^u, v_3^u\} \cap S^R = \emptyset$ or $\{v_2^u, v_3^u\} \cap S^R = \emptyset$, which is impossible. ■

Proposition 2.3. *If S' is a separator in G with a minimum size, then $S^{R^{S'}}$ has the same size as S'.*

Proof. Since S' is a separator in G with a minimum size, it follows that for all $u \in U$, $|S' \cap \{v_1^u, v_2^u, v_3^u\}| \leq 2$. For otherwise, one can delete one of these node, namely v_3^u, and still having a separator. In consequence, from the definition of $R^{S'}$, we have $|S' \cap \{v_1^u, v_2^u, v_3^u\}| = 2$ for all $u \in R^{S'}$. And for all $u \in U \setminus R^{S'}$, $|S' \cap \{v_1^u, v_2^u, v_3^u\}| = 1$. For otherwise, if $|S' \cap \{v_1^u, v_2^u, v_3^u\}| = \emptyset$, the remaining graph after deleting S' would contain the terminal path v_1^u, v_3^u, contradicting the fact that S' is a separator. $|S'| = 2|R^{S'}| + |U \setminus R^{S'}|$. On the other hand, from the construction of $S^{R^{S'}}$, if $u \in R^{S'}$, then $v_1^u, v_2^u \in S^{R^{S'}}$ and if not, $v_3^u \in S^{R^{S'}}$. Hence $|S^{R^{S'}}| = 2|R^{S'}| + |U \setminus R^{S'}|$, which is equal to $|S'|$. ■

Proposition 2.4. *If S' is a separator in G with a minimum size, then S' is of size $q + |U|$ where q is the size of the vertex cover $R^{S'}$ in H.*

Proof. First note that two nodes among $\{v_1^u, v_2^u, v_3^u\}$ suffice to cut all the terminal paths going through these nodes. Hence as S' is a separator with a minimum size in G, for all $u \in U$, $1 \leq |S' \cap \{v_1^u, v_2^u, v_3^u\}| \leq 2$. By Proposition 2.2, $R^{S'}$ is a vertex cover in H of size q and $S^{R^{S'}}$ is a separator in G. Moreover, from Proposition 2.3, $S^{R^{S'}}$ has the same size as S'. From its construction, the size of $S^{R^{S'}}$ is $2q + (|U| - q) = q + |U|$. Hence, that of S' is $q + |U|$. ■

Proposition 2.5. *If R is a vertex cover in H of minimum size, then S^R is a separator in G of minimum size. And if S is a separator in G of minimum size, then R^S is a vertex cover in H of minimum size.*

Proof. Suppose that R is a vertex cover of minimum size q in H but S^R is not of minimum size in G. Note that, from the construction of S^R, S^R is of size $q + |U|$. Let $S' \subseteq V_1 \cup V_2 \cup V_3$ be a separator in G of minimum size. From Proposition 2.4, $|S'| = q' + |U|$ where q' is the size of $R^{S'}$. Since $|U| + q' < q + |U|$, it follows that $q' < q$ and hence R is not of minimum size in H, a contradiction.

Now, suppose that S is a separator in G of minimum size $q + |U|$ but R^S, which is of size q, is not of minimum size in H. Let $R' \subseteq U$ be a vertex cover in H of minimum size q'. Then, $S^{R'}$ is a separator in G of size $q' + |U|$. Since $q' < q$, it follows that S is not of minimum size in G, a contradiction. ■

From Proposition 2.5, finding a vertex cover in H of minimum size is equivalent to finding a minimum size three terminal vertex separator in G. Since the vertex cover problem is NP-hard, the $3 - TVSP$ so is. And consequently the MTVSP is NP-hard. ■

In [15], authors prove that the vertex cover problem is NP-hard to approximate to within a factor of 1.3606. From Proposition 2.5, we have the following.

Corollary 2.1. *The multi-terminal vertex separator problem is NP-hard to approximate to within a factor of 1.3606.*

3. Formulations

In this section we propose two $0 - 1$ linear formulations for the MTVSP. The first one is a compact formulation with a polynomial number of variables and constraints, and uses double indices. The second has a polynomial number of variables but an exponential number of inequalities.

Table 1
Comparing LP-Relaxations value of F^1 and F^2 with the optimal one.

Instances	k	Z_{F^1}	Z_{F^2}	OPT
DSJR500	10	**32**	18.5	**32**
Games120	10	30.34	18.5	31
Miles250	15	32.18	27.5	35
Myciel6	11	38.34	22.5	40
Myciel7	17	**57**	31	**57**
Queen8_ 12	11	**38**	21.5	**38**
Queen14_ 14	18	**65**	35	**65**
Queen16_ 16	16	**59**	32	**59**

3.1. Double index formulation

Let $x \in \{0,1\}^{(V\cup T)\times T}$ such that

$$x_{vt} = \begin{cases} 1 & \text{if vertex } v \text{ belongs to subset } V_t, \quad \forall v \in V, \forall t \in T, \\ 0 & \text{otherwise.} \end{cases}$$

The MTVSP is equivalent to the following integer linear program F^1,

$$\min \quad |V| - \sum_{v\in V}\sum_{t\in T} x_{vt}$$

$$x_{ut} + \sum_{t'\in T\setminus\{t\}} x_{vt'} \leq 1 \quad \forall uv \in E, \forall t \in T, \tag{1}$$

$$\sum_{t\in T} x_{vt} \leq 1 \quad \forall v \in V, \tag{2}$$

$$x_{tt'} = 0 \quad \forall t \in T, \forall t' \in T \setminus \{t\}, \tag{3}$$

$$x_{tt} = 1 \quad \forall t \in T, \tag{4}$$

$$x_{vt} \in \{0,1\} \quad \forall v \in V, \forall t \in T. \tag{5}$$

For each pair of terminals $\{t, t'\} \subseteq T$, inequalities (1) ensure that there is no edge connecting V_t and $V_{t'}$. Inequalities (2) guarantee that each vertex of V belongs to at most one subset of vertices. Equalities (3) and (4) ensure that each terminal t belongs to exactly one subset of vertices, namely V_t. Note that, by inequalities (1) and Eqs. (4), $x_{vt'} = 0$ for all $t \in T, t' \in T\setminus\{t\}$ and $v \in N(t)$.

3.2. Natural formulation

Let Γ be the set of all terminal paths between the terminals in G. Let $x \in \{0,1\}^V$ such that

$$x_v = \begin{cases} 1 & \text{if vertex } v \text{ belongs to the separator}, \quad \forall v \in V, \\ 0 & \text{otherwise.} \end{cases}$$

The MTVSP is equivalent to the following integer linear program F^2,

$$\min \quad \sum_{v\in V} x_v$$

$$\sum_{v\in P_{tt'}} x_v \geq 1 \quad \forall P_{tt'} \in \Gamma, \tag{6}$$

$$x_v \in \{0,1\} \quad \forall v \in V. \tag{7}$$

Inequalities (6) guarantee that at least one vertex of each terminal path belongs to the separator.

3.3. Comparing the LP-relaxations

We first present a numerical comparison of the LP-Relaxation values of the two above formulations with the optimal value for some DIMACS instances [16]. The columns in Table 1 represent the name of the instance, the optimal value of the LP-Relaxation (Z_{F^1}) of formulation F^1, the optimal value of the LP-Relaxation (Z_{F^2}) of formulation F^2 and the value of the optimal separator, respectively.

It appears from Table 1 that the LP-Relaxation of the first formulation is better than the second one. This is shown below.

Proposition 3.1. *Every feasible solution of the LP-Relaxation of F^1 has a corresponding feasible solution of the LP-Relaxation of F^2 with the same objective value.*

Proof. Let $x \in [0, 1]^{V \times T}$ be a feasible solution of the LP-Relaxation of F^1. Let Z be its objective value. For $v \in V$, let $y \in \mathbb{R}^V$ such that $y_v = 1 - \sum_{t \in T} x_{vt}$. Clearly, by inequalities (2), $y_v \in [0, 1]$. Consider a path P with l edges between two terminals t and t'. Let $P_{tt'}$ be the corresponding terminal path. By summing inequalities (1), associated with the edges of P and t (by respecting the order of nodes in $P_{tt'}$), we obtain the following inequality

$$x_{tt} + \sum_{v \in P_{tt'}} \sum_{t'' \in T} x_{vt''} + \sum_{t'' \in T \setminus \{t\}} x_{t't''} \leq l.$$

As $x_{tt} = 1$ and $\sum_{t'' \in T \setminus \{t\}} x_{t't''} = 1$, it follows that

$$\sum_{v \in P_{tt'}} y_v \geq 1,$$

and thus inequalities (6) are satisfied for y. Moreover, the objective value of y is $\sum_{v \in V} y_v = \sum_{v \in V}(1 - \sum_{t \in T} x_{vt}) = Z$. This ends the proof. ■

As a consequence of Proposition 3.1, we have the following.

Corollary 3.1. $Z_{F^2} \leq Z_{F^1}$.

Note that the first formulation has $(|V| + |T|) \cdot |T|$ variables and the second one has only $|V|$ variables. Also note that inequalities (6), which are exponential in number, can be separated in polynomial time. Since the second formulation has less variables and may use a few number of inequalities of type (6) in the Branch-and-Cut algorithm, we will consider it for our analysis. Preliminary experiments have shown that F^2 is more efficient for big sized instances (For the instance R_1300 in Table 3, F^1 has 13100 variables and uses 3329920 non trivial inequalities, whereas F^2 has only 1300 variables and uses 122 non trivial inequalities in the Branch-and-Cut algorithm).

4. Polyhedral analysis

Let $P(G, T)$ be the convex hull of the solutions of formulation F^2, that is,

$$P(G, T) = conv\{x \in \{0, 1\}^V \mid x \text{ satisfies (6)}\}.$$

In this section, we will discuss $P(G, T)$, we will give its dimension, identify several classes of valid inequalities and describe necessary and sufficient conditions for these inequalities to be facet defining.

4.1. Dimension and trivial inequalities

We first establish the dimension of $P(G, T)$.

Theorem 4.1. *$P(G, T)$ is full dimensional.*

Proof.
We need to exhibit $n + 1$ separators such that their incidence vectors are affinely independent. Let $S_0 = V$. Clearly S_0 is a separator of G. For each $v \in V$, let $S_v = V \setminus \{v\}$. By Assumption (2), v is not adjacent to two terminals. It then follows that S_v is a separator. This constitutes a set of $n + 1$ separators of G. Moreover, their incidence vectors are affinely independent. ■

Now we characterize when inequalities $x_v \leq 1$ and $x_v \geq 0$ for all $v \in V$ define facet of $P(G, T)$.

Theorem 4.2. *For $v \in V$, inequality $x_v \leq 1$ defines a facet of $P(G, T)$.*

Proof. We need to exhibit n separators containing v such that their incidence vectors are affinely independent. Let $S_0 = V$, and for each vertex $u \in V \setminus \{v\}$, let $S_u = V \setminus \{u\}$. Clearly, these sets are separators in G. Moreover, their incidence vectors are affinely independent. ■

Theorem 4.3. *For $v \in V$, inequality $x_v \geq 0$ defines a facet of $P(G, T)$ if and only if v does not belong to a terminal path $P_{tt'}$ containing exactly two internal vertices.*

Proof. ($\Rightarrow$) Suppose there exists a terminal path $P_{tt'}$ containing exactly two internal vertices u and v. Then we have the valid inequalities $x_u + x_v \geq 1$ and $x_u \leq 1$. Inequality $x_v \geq 0$ can be obtained as a linear combination of these inequalities, and cannot define a facet. ($\Leftarrow$) Let $S_0 = V \setminus \{v\}$ and $S_u = S_0 \setminus \{u\}$ for all $u \in V \setminus \{v\}$. Since v does not belong to a terminal path of two vertices, these sets are separators of G. Moreover their incidence vectors are affinely independent. ■

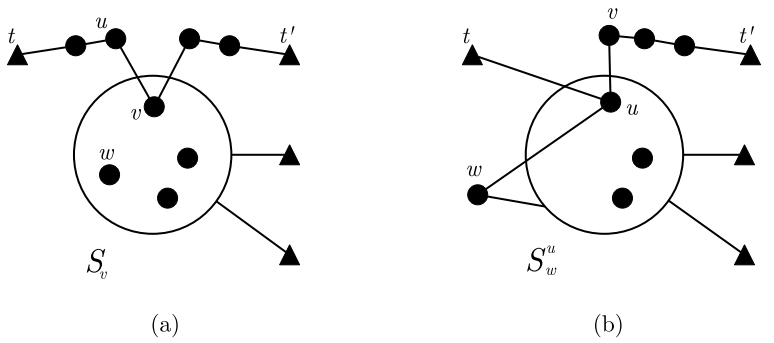

Fig. 3. Two separators S_v and S^u_w.

4.2. Path inequalities

Theorem 4.4. *Inequality (6), associated with a terminal path $P_{tt'}$, defines a facet of $P(G,T)$ if and only if*

(a) *$P_{tt'}$ is minimal.*
(b) *No vertex $v \in V \setminus P_{tt'}$ is adjacent to a terminal $t'' \in T \setminus \{t, t'\}$ and to two vertices of $P_{tt'}$.*

Proof. ($\Rightarrow$)

(a) If $P_{tt'}$ is not minimal, then there exists a nonterminal vertex $v \in P_{tt'}$ such that $P_{tt'} \setminus \{v\}$ contains a terminal path say $P_{t_1t'_1}$. Note that t_1 and t'_1 may coincide with t and t', respectively. Inequality (6) can then be obtained by summing inequality (6) associated with $P_{t_1t'_1}$ and inequalities $x_u \geq 0$ associated with each vertex $u \in P_{tt'} \setminus P_{t_1t'_1}$. Hence, inequalities (6) cannot define a facet.
(b) Suppose $P_{tt'} = \{v_1, \ldots, v_l\}$. Suppose there exists $v \in V \setminus P_{tt'}$ which is adjacent to $t'' \in T \setminus \{t, t'\}$ and to two vertices v_i and v_j of $P_{tt'}$ with $1 \leq i \leq j \leq l$. Let $P_{tt''} = \{v_1, \ldots, v_i, v\}$ and $P_{t't''} = \{v, v_j, \ldots, v_l\}$. Thus we have the following valid inequalities, $x(P_{tt'}) \geq 1, x(P_{tt''}) \geq 1, x(P_{t't''}) \geq 1$ and $x_u \geq 0$ for all $u \in P_{tt'} \setminus (P_{tt''} \cup P_{t't''})$. By summing these inequalities, dividing by 2 and rounding up the right hand side, we obtain the valid inequality $x(P_{tt'}) + x_v \geq 2$. Inequality (6), associated with terminal path $P_{tt'}$, can then be obtained from the above inequality and inequality $x_v \leq 1$, and cannot therefore be facet defining.

($\Leftarrow$) Denote by $ax \geq \alpha$ inequality (6). Let $bx \geq \beta$ be an inequality that defines a facet of $P(G,T)$. Suppose that $\{x \in P(G,T) : ax = \alpha\} \subseteq \{x \in P(G,T) : bx = \beta\}$. We will show that there exists ρ such that $b = \rho a$.

For $v \in P_{tt'}$, let $S_v = (V \setminus P_{tt'}) \cup \{v\}$. Fig. 3.(a) represents the set S_v. By Condition (a), each terminal path either contains v or a vertex from $V \setminus P_{tt'}$. Thus S_v is a separator of G. Moreover, $ax^{S_v} = \alpha$. By symmetry, we have $ax^{S_u} = ax^{S_v}$, and hence $bx^{S_u} = bx^{S_v}$ for every $u, v \in P_{tt'}$. Therefore

$$b(u) = b(v) = \rho \text{ for all } u, v \in P_{tt'} \text{ and some scalar } \rho \in \mathbb{R}.$$

For $w \in V \setminus P_{tt'}$ such that $N(w) \cap (T \setminus \{t, t'\}) = \emptyset$, let $u \in P_{tt'}$ be a vertex such that

- u is adjacent to t if w is adjacent to t',
- u is adjacent to t' if w is adjacent to t,
- u is either adjacent to t or to t', if $N(w) \cap \{t, t'\} = \emptyset$.

Set $S^u_w = S_u \setminus \{w\}$. (Fig. 3.(b) displays the set S^u_w.) Clearly, all paths between t and t' intersect S^u_w. It follows that S^u_w is a separator of G. Moreover, the incidence vector of S^u_w satisfies inequality (6) with equality. Hence $bx^{S_u} = bx^{S^u_w}$, and thus $b(w) = 0$. We then obtain that $b(w) = 0$ for all $w \in V \setminus P_{tt'}$ such that $N(w) \cap (T \setminus \{t, t'\}) = \emptyset$.

For $w \in V \setminus P_{tt'}$ such that $N(w) \cap (T \setminus \{t, t'\}) \neq \emptyset$, let $u \in P_{tt'}$ be a vertex adjacent to w. If there is no vertex in $P_{tt'}$ adjacent to w, then u is arbitrarily chosen in $P_{tt'}$. From Condition (b), w is adjacent to at most one vertex u in $P_{tt'}$. Set $S'^u_w = S_u \setminus \{w\}$. Recall that $S_u = (V \setminus P_{tt'}) \cup \{u\}$. Since u belongs to S'^u_w and by Assumption (2) of Section 1, w is not adjacent to another terminal, we have that S'^u_w is a separator of G.

Moreover, the incidence vector of S'^u_w satisfies inequality (6) with equality. Hence $bx^{S_u} = bx^{S'^u_w}$, and thus $b(w) = 0$. We then obtain that $b(w) = 0$ for all $w \in V \setminus P_{tt'}$ such that $N(w) \cap (T \setminus \{t, t'\}) = \emptyset$.

Therefore, we have that $b = \rho a$, which ends the proof. ■

In what follows we describe further classes of valid inequalities of $P(G,T)$ and discuss their facial structure.

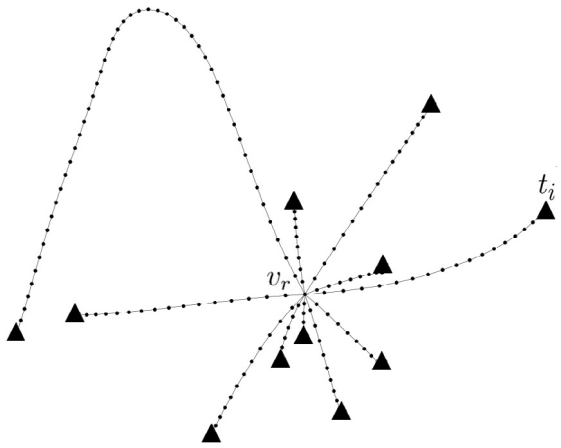

Fig. 4. A Star tree.

4.3. Star tree inequalities

A *star tree* $H = (V(H) \cup T(H), E(H))$ of G is a tree where the leaves are terminal nodes, and all the other nodes of H are nonterminal and, except for the root node, they are of degree two. The star tree H with q terminals $t_1, \ldots, t_q$, can also be seen as the concatenation of q paths P_{t_i}, $i = 1, \ldots, q$ between the root v_r and t_i, $i = 1, \ldots, q$. The paths $P_{t_1}, \ldots, P_{t_q}$ will be called *branches*. A star tree with 2 terminals is nothing but a terminal path. Fig. 4 displays a star tree with 11 branches.

Theorem 4.5. *If $H = (V(H) \cup T(H), E(H))$ is a star tree of G with q terminals and root v_r, then the inequality*

$$x(V(H) \setminus \{v_r\}) + (q-1)x_{v_r} \geq q - 1 \tag{8}$$

is valid for $P(G, T)$.

Proof. Let S be a separator of G. If v_r belongs to S, then $x_{v_r} = 1$ and the inequality is satisfied by x^S. If v_r is not in S, then S must contain at least $q - 1$ nodes of $V(H) \setminus \{v_r\}$ in order to cut all the terminal paths of H. ■

Claim 4.1. *If there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$ and to an internal vertex in each branch of H, then the following inequality*

$$x(V(H) \setminus \{v_r\}) + (q-1)x_{v_r} + x_w \geq q \tag{9}$$

is valid for $P(G, T)$.

Proof. Consider a separator S. As in the proof of Theorem 4.5, we have that either $v_r \in S$ or at least $q - 1$ vertices of $V(H) \setminus \{v_r\}$ belong to S. Therefore, if $w \in S$, then inequality (9) is satisfied by x^S. If this is not the case, then S must contain all the nodes of $N(w) \cap V(H)$. Otherwise, there would exist a terminal path between t' and a terminal in $T(H)$ not intersecting S. As $|N(w) \cap V(H)| \geq q$, inequality (9) is again satisfied. ■

Claim 4.2. *If there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$, and to two different vertices $u_1, u_2 \in P_t$, for some $t \in T(H)$, then the following inequality*

$$x(V(H) \setminus \{v_r\}) + (q-1)x_{v_r} + x_w \geq q \tag{10}$$

is valid for $P(G, T)$.

Proof. Since u_1 and u_2 are different, we can suppose that $u_1 \neq v_r$. Given a separator S of G, S must contain a vertex $v \in P_{tt'} \subseteq P_t \cup \{w\}$, where $P_{tt'}$ is the terminal path between t and t' not containing v_r. If $v_r \in S$, then inequality (10) is satisfied by x^S. If v_r is not in S, then S contains $q - 1$ vertices from $V(H) \setminus \{v_r\}$. Since w is adjacent to t' and to two vertices of P_t, S must contain another vertex from $(V(H) \cup \{w\}) \setminus \{v_r\}$. And hence the inequality (10) is satisfied by x^S. ■

Theorem 4.6. *Given a star tree $H = (V(H) \cup T(H), E(H))$ of G, inequality (8) defines a facet of $P(G, T)$ if and only if the following hold:*

(a) *There is no vertex of $V(H)$ adjacent to a terminal of $T \setminus T(H)$.*
(b) *The subgraph induced by $V(H)$ in G contains no cycle.*

(c) *There is no vertex of* $V \setminus V(H)$ *adjacent to a terminal of* $T \setminus T(H)$*, and to two vertices of the same branch.*
(d) *There is no vertex of* $V \setminus V(H)$ *adjacent to a terminal of* $T \setminus T(H)$ *and to an internal vertex of each branch of* H*.*
(e) *Each branch of* H *has at least one internal vertex.*

Proof. ($\Rightarrow$)

(a) If a vertex $u \in P_t \setminus \{v_r\}$, for some $t \in T(H)$, is adjacent to a terminal $t' \in T \setminus T(H)$, then consider the star tree H' in H with leaf set $T(H') = T(H) \setminus \{t\}$. Inequality (8) associated with H can be obtained from the star tree inequality associated with H', the inequality associated with the terminal path $P_{tt'}$ between t and t' and the trivial inequalities $x_w \geq 0$ for all w of $P_t \setminus P_{tt'}$. If v_r is adjacent to a terminal $t' \in T \setminus T(H)$, then consider the star tree H'' where $T(H'') = T(H) \cup \{t'\}$. Inequality (8), associated with H, can then be obtained by summing the star tree inequality induced by H'' and the trivial inequality $-x_{v_r} \geq -1$. So in both cases, inequality (8) can be obtained as a linear combination of valid inequalities, and it cannot then be facet defining.
(b) Suppose that the subgraph induced by $V(H)$ in G contains a cycle. It follows that there exist two vertices $u, v \in V(H)$ such that $uv \in E \setminus E(H)$. We distinguish two cases.

Case 1. $u \in P_t \setminus \{v_r\}$ and $v \in P_{t'} \setminus \{v_r\}$ for two different terminals $t, t' \in T(H)$. We claim that, the following inequality

$$x(V(H) \setminus \{v_r\}) + (q-2)x_{v_r} \geq q - 1 \tag{11}$$

is valid for $P(G, T)$ and dominates inequality (8) associated with H. Indeed, given a separator S of G, if v_r belongs to S, then S must contain another vertex $w \in P_{tt'}$, where $P_{tt'}$ is the terminal path between t and t' not containing v_r. Then the inequality is satisfied by x^S. If v_r is not in S, then S must contain at least $q-1$ nodes of $V(H) \setminus \{v_r\}$ in order to cut all the terminal paths of H. And the inequality is again satisfied. As inequality (11) dominates inequality (8), the latter cannot be facet defining. It will be shown later, in Section 4.5, that inequality (11) can be obtained as a lifted inequality from the star tree inequality (8).

Case 2. $u, v \in P_t$ for some $t \in T(H)$. Let $P_{uv} \subset P_t$ be the set of the internal vertices of the path between u and v, in the branch P_t. Note that $P_{uv} \neq \emptyset$. Then $((V(H) \setminus P_{uv}) \cup T(H), E(H) \cup \{uv\})$ is also a star tree. Then, the following inequality

$$x(V(H) \setminus (P_{uv} \cup \{v_r\})) + (q-1)x_{v_r} \geq q - 1$$

is a star tree inequality that is valid for $P(G, T)$. However this inequality dominates inequality (8) associated with H. Therefore, the latter cannot be facet defining.

(c) Suppose there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$, and to two different vertices $u_1, u_2 \in P_t$, for some $t \in T(H)$. From Claim 4.2, the following inequality

$$x(V(H) \setminus \{v_r\}) + (q-1)x_{v_r} + x_w \geq q \tag{12}$$

is valid for $P(G, T)$. Hence, inequality (8) can be obtained by summing inequality (12) and $-x_w \geq -1$. Thus inequality (8) cannot be facet defining.
(d) Suppose there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$ and to an internal vertex in each branch of H. From Claim 4.1 the following inequality

$$x(V(H) \setminus \{v_r\}) + (q-1)x_{v_r} + x_w \geq q \tag{13}$$

is valid for $P(G, T)$. Inequality (8) can be obtained by summing inequality (13) and $x(w) \leq 1$. Therefore, inequality (8) cannot be facet defining.
(e) Suppose there exists a branch P_t with no internal vertices. Let H' be the star tree obtained from H by deleting a branch $P_{t'}$ with at least one internal vertex. Inequality (8) can be obtained by summing the star tree inequality associated with H' together with the terminal path inequality associated with $P_t \cup P_{t'}$.

($\Leftarrow$) Suppose that conditions (a)–(e) hold. Denote by $ax \geq \alpha$ inequality (8). Let $bx \geq \beta$ be an inequality that defines a facet of $P(G, T)$. Suppose that $\{x \in P(G, T) : ax = \alpha\} \subseteq \{x \in P(G, T) : bx = \beta\}$. We will prove that there exists ρ such that $b = \rho a$.

By condition (e), it follows that each branch P_t, $t \in T(H)$, contains at least one internal node. For a terminal $t \in T(H)$, a *ring* of $V(H) \setminus P_t$ is a subset of $q-1$ vertices containing exactly one nonterminal vertex from each branch different from P_t. Hence if Q^t is a ring of $V(H) \setminus P_t$ then for all $t' \in T(H) \setminus \{t\}$, $|(P_{t'} \setminus \{v_r, t'\}) \cap Q^t| = 1$, see Fig. 5.(a). Consider two different vertices $u_1, u_2 \in V(H) \setminus \{v_r\}$. Let Q_1, Q_2 be two rings such that $u_1 \in Q_1, u_2 \in Q_2$ and $Q_1 \setminus \{u_1\} = Q_2 \setminus \{u_2\}$. Let $S_1 = (V \setminus V(H)) \cup Q_1$ and $S_2 = (V \setminus V(H)) \cup Q_2$. See Fig. 5.(b) and Fig. 5.(c). By conditions (a) and (b), each terminal path either contains a vertex from each path of Q_i with $i = 1,2$ or it contains a vertex of $V \setminus V(H)$. Hence S_1 and S_2 are two separators of G. Moreover, incidence vectors x^{S_1} and x^{S_2} satisfy inequality (8) with equality. Hence $bx^{S_1} = bx^{S_2}$, implying that $b(u_1) = b(u_2)$. As t, u_1 and u_2 are arbitrarily chosen, we have that

$$b(u) = b(v) = \rho \text{ for all } u, v \in V(H) \setminus \{v_r\} \text{ and a scalar } \rho \in \mathbb{R}.$$

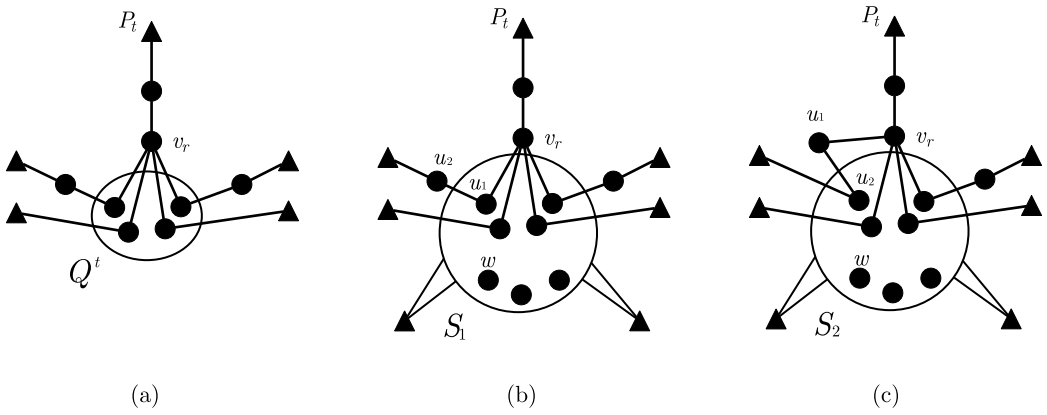

Fig. 5. A ring Q^t and two separators S_1 and S_2.

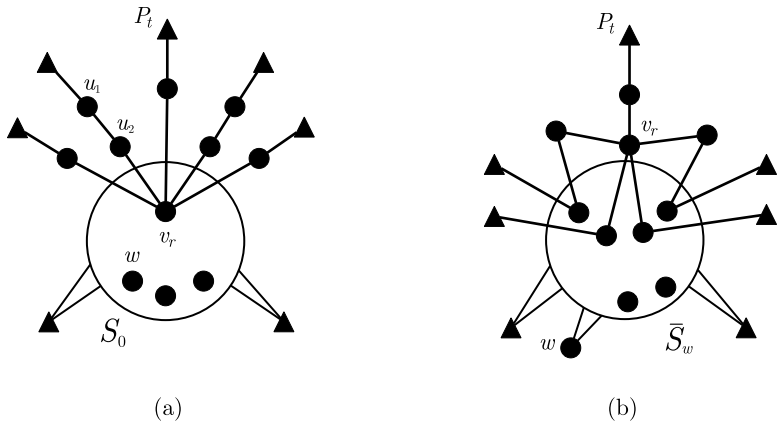

Fig. 6. Two separators S_0 and $\bar{S}_w$.

Set $S_0 = (V \setminus V(H)) \cup \{v_r\}$. By conditions (a) and (b) each terminal path contains v_r or a vertex from $V \setminus V(H)$. Thus, S_0 is a separator of G. See Fig. 6.(a) for an illustration. Also we have $ax^{S_0} = \alpha$. Hence, $bx^{S_1} = bx^{S_0}$, yielding

$$b(v_r) = \sum_{v \in Q_1^t} b(v) = (q-1)\rho.$$

Now consider a vertex $w \in V \setminus V(H)$ adjacent to a terminal in $T \setminus T(H)$. We distinguish two cases.

Case 1. w is not adjacent to the root vertex v_r. By condition (c), w is adjacent to at most one vertex of each branch and by condition (d), there exists at least one branch P_t such that no internal vertex of P_t is adjacent to w. Consider the ring Q of H such that $N(w) \cap V(H) \subseteq Q$ and each vertex of $Q \setminus (N(w) \cap V(H))$ is adjacent to a terminal of $T(H) \setminus \{t\}$. Let $S' = (V \setminus V(H)) \cup Q$ and $\bar{S}_w = S' \setminus \{w\}$. (See Fig. 6.(b) for an illustration). Since Q is a ring, S' is a separator of G, as defined before. Set $\bar{S}_w$ is also a separator of G. Otherwise, since S' is a separator and $N(w) \subseteq S'$, w would be adjacent to two terminals. Contradicting Assumption 2 of Section 1.

Case 2. w is adjacent to the root vertex v_r. Then let $\bar{S}_w = S_0 \setminus \{w\}$. Set $\bar{S}_w$ is a separator of G. Indeed by condition (c), since w is adjacent to v_r, w cannot be adjacent to vertices of $V(H) \setminus \{v_r\}$. Therefore, every terminal path between a terminal of $T \setminus T(H)$ and a terminal of T intersects $\bar{S}_w$

In both cases, $\bar{S}_w$ is a separator in G, and its incidence vector satisfies inequality (8) with equality. Hence $ax^{\bar{S}_w} = ax^{S_0}$, therefore $bx^{\bar{S}_w} = bx^{S_0}$. This implies that $b(w) = 0$, and therefore $b(u) = 0$ for all $u \in V \setminus V(H)$ adjacent to a terminal in $T \setminus T(H)$.

For a vertex $w \in V \setminus V(H)$ not adjacent to a terminal in $T \setminus T(H)$, let Q be a ring of H such that $Q \subseteq N(T(H))$. Let $S'_w = (V \setminus (V(H) \cup \{w\})) \cup Q$. Clearly, S'_w is a separator in G and its incidence vector satisfies inequality (8) with equality. Hence, $ax^{S'_w} = ax^{S_0}$, and therefore $bx^{S'_w} = bx^{S_0}$. In consequence $b(w) = 0$ and then $b(u) = 0$ for all $u \in V \setminus V(H)$ not adjacent

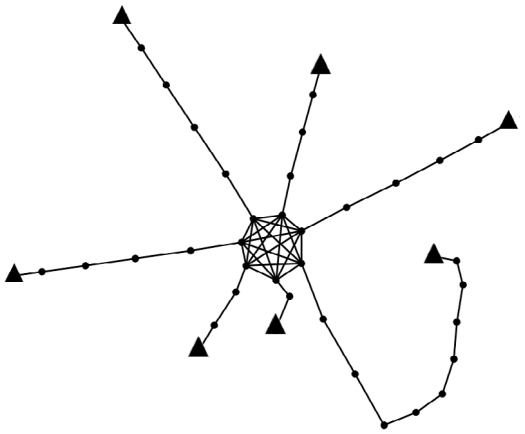

Fig. 7. Clique star of 7 terminals.

to a terminal in $T \setminus T(H)$. Hence, we have that

$$b(u) = 0 \quad \text{for all } u \in V \setminus V(H).$$

Altogether, we have that $b = \rho a$, and the proof is complete. ■

4.4. Clique star inequalities

A graph $Q = (V(Q) \cup T(Q), E(Q))$ is called a *clique star* if Q consists of a clique K_q on q vertices of $V(Q)$, q terminals $t_1, \ldots, t_q$ (that is $T(Q) = \{t_1, \ldots, t_q\}$) and q disjoint paths $P_{t_1}, \ldots P_{t_q}$ such that each path P_{t_i} is between a different vertex of K_q and t_i. The paths P_t are called *branches*. Fig. 7 displays a clique star with 7 branches. In what follows we will suppose that $q \geq 3$, otherwise the clique star is either a terminal path or a branch.

Theorem 4.7. *Given a clique star $Q = (V(Q) \cup T(Q), E(Q))$ of G, the following inequality*

$$x(V(Q)) \geq q - 1 \tag{14}$$

is valid for $P(G, T)$.

Proof. Let S be a separator of G. If $S \cap K_q = \emptyset$, then clearly S must contain at least $q-1$ vertices of $V(Q)$ to cut all the terminal paths of Q. So suppose that $S \cap K_q = \{v_1, \ldots, v_l\}$, $l \leq q-2$. (If $l \geq q-1$, then inequality (14) is satisfied by x^S). Note that each vertex $v_i \in K_q$ is an extremity of branch P_{t_i}. Hence $q - l$ terminals remain to separate. For this at least $q - l - 1$ vertices from $V(Q) \setminus K_q$ are needed to cut all the terminal paths between the remaining terminals. Hence (14) is satisfied by x^S. ■

4.5. Lifted star tree inequalities

In what follows we are going to describe a lifting procedure for the star tree inequalities. This will permit to extend these inequalities to a more general class of valid inequalities.

Let $G = (V \cup T, E)$ be a simple graph. Consider a star tree H of G. Let a *branch clique* $W \subseteq (V(H) \setminus \{v_r\})$ be a clique such that for each pair of vertices $u, v \in W$, u and v belong to different branches of H. Observe that each branch clique yields a clique star in the graph. (See Fig. 8 for an illustration where three branch cliques are displayed).

Theorem 4.8. *Let $\Pi = \{W_1, \ldots, W_s\}$ be a set of vertex disjoint branch cliques in H. Let $\alpha_\Pi = \sum_{i=1}^{s}(|W_i| - 1)$. Then the inequality*

$$x(V(H) \setminus \{v_r\}) + (q - 1 - \alpha_\Pi)x_{v_r} \geq q - 1 \tag{15}$$

is valid for $P(G, T)$.

Proof. Let S be a separator of G. We distinguish two cases. If $v_r \in S$, then for each clique star Q^{W_i} associated with the branch clique W_i, $x^S(V(Q^{W_i})) \geq |W_i| - 1$ by the clique star inequality (14). By summing these inequalities, we obtain that $x^S(\bigcup_{i=1}^{p} V(Q^{W_i})) \geq \alpha_\Pi$. As $\bigcup_{i=1}^{p} V(Q^{W_i}) \subset V(H) \setminus \{v_r\}$, it follows that $x^S(V(H) \setminus \{v_r\}) \geq \alpha_\Pi$. As $v_r \in S$, we have that x^S satisfies inequality (15). If $v_r \notin S$, then $x^S(V(H)) \geq q - 1$ by the star tree inequality. And thus x^S also satisfies inequality (15). ■

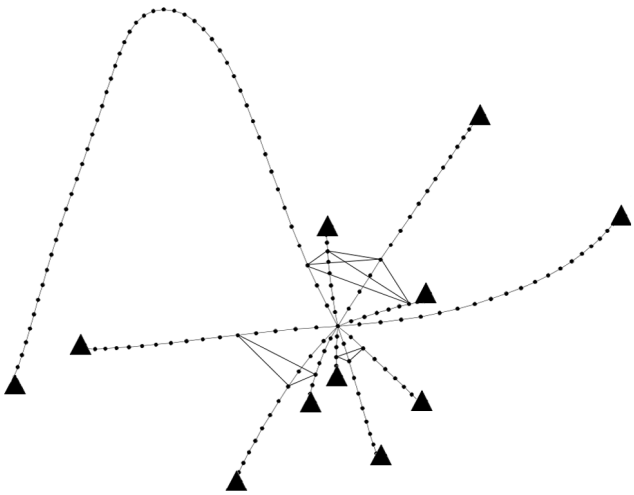

Fig. 8. Star tree with clique branches.

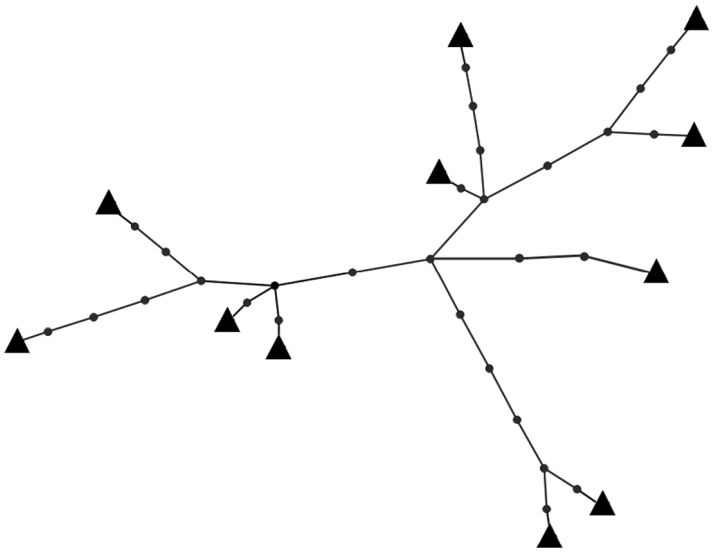

Fig. 9. Terminal tree of 10 terminals.

4.6. Terminal tree inequalities

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. (See Fig. 9 for an illustration). A *leaf branch* P_t of a terminal tree H is a path between a terminal t and a vertex of $V(H)$ of degree greater than or equal to 3, and where all internal vertices are of degree 2. For $v \in V(H)$, let $d_H(v)$ be the degree of v in H. If $|T(H)| \leq 3$, the terminal tree is either a star tree or a terminal path.

Theorem 4.9. *Let $G = (V \cup T, E)$ be a graph and H be a terminal tree of G. Let $q = |T(H)|$. Then the following inequality is valid for $P(G, T)$,*

$$\sum_{v \in V(H)} (d_H(v) - 1)x_v \geq q - 1. \tag{16}$$

Proof. The proof is by induction on q. If $q \leq 3$, inequality (16) is nothing but a star tree inequality (8) associated with H, and then it is valid. So let us suppose $q \geq 3$, and that for each terminal tree with less than q terminals, the associated inequality (16) is valid for $P(G, T)$. Since $q \geq 3$, there must exist a vertex u belonging to two leaf branches P_t and $P_{t'}$. Let us consider the terminal tree H^1 (resp. H^2) obtained from H by removing the vertices of $P_t \setminus \{u\}$ (resp. $P_{t'} \setminus \{u\}$). Thus, H^1 and H^2 have

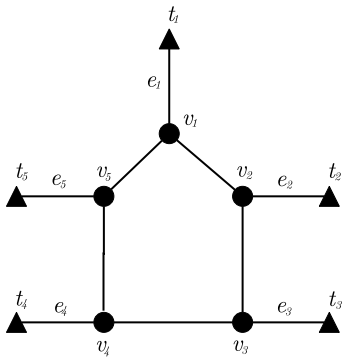

Fig. 10. Terminal cycle of 5 terminals.

each one $q-1$ leaves and $q-1$ terminals. By the induction hypotheses, the terminal tree inequalities (16) associated with H^1 and H^2

$$\sum_{v\in V(H)\setminus(P_t\cup P_{t'})}(d_H(v)-1)x_v+\sum_{v\in P_{t'}\setminus\{u\}}(d_H(v)-1)x_v+(d_H(u)-2)x_u\geq q-2,$$
$$\sum_{v\in V(H)\setminus(P_t\cup P_{t'})}(d_H(v)-1)x_v+\sum_{v\in P_t\setminus\{u\}}(d_H(v)-1)x_v+(d_H(u)-2)x_u\geq q-2.$$

are valid for $P(G,T)$. Observe that $d_H(v)=2$ for all $v\in(P_t\cup P_{t'})\setminus\{u\}$. By summing these inequalities together with the inequality $x(P_t\cup P_{t'})\geq 1$ induced by the terminal path $P_t\cup P_{t'}$, and inequality $x_u\geq 0$, we obtain the inequality

$$\sum_{v\in V(H)\setminus(P_t\cup P_{t'})}2(d_H(v)-1)x_v+\sum_{v\in P_t\cup P_{t'}}2(d_H(v)-1)x_v\geq 2q-3. \tag{17}$$

By dividing by 2 and rounding up the right hand side, we obtain inequality (17). ■

Remark 4.1. If the terminal tree contains some terminals that are not leaves, then the associated inequality (16) is dominated by other terminal tree inequality.

Remark 4.2. The star tree inequalities are a special case of terminal tree inequalities that can be separated in polynomial-time (see Section 6.1) while for the terminal tree inequalities, the complexity of the separation problem is open.

4.7. Lifted terminal tree inequalities

Let R be a terminal tree of G. For a vertex v, let F_v be the union of leaf branches incident to v. Clearly, F_v is a star tree in R. Let Π_{F_v} be a set of vertex disjoint branch cliques in F_v (see, lifted star tree inequalities, Section 4.5).

Theorem 4.10. *The following inequality*

$$\sum_{v\in V(R)}(d_R(v)-1-\alpha_{\Pi_{F_v}})x_v\geq q-1$$

if valid for $P(G,T)$.

Proof. The proof is similar to the one of Theorem 4.8. ■

4.8. Terminal cycle inequalities

A *terminal cycle* $J=(V(J)\cup T(J),E(J))$, where $T(J)$ is a set of q terminals, is a graph given by a cycle of q vertices and q disjoint edges between the vertices of the cycle and the terminals of $T(J)$. We suppose that $|T(J)|\geq 2$. Let $C=V(J)$ be the set of nonterminal vertices of J. (See Fig. 10 displaying a terminal cycle where $q=5$).

Theorem 4.11. *If $J=(V(J)\cup T(J),E(J))$ is a terminal cycle of G, then the following inequality is valid for $P(G,T)$,*

$$x(C)\geq\lceil\frac{q}{2}\rceil. \tag{18}$$

Proof. Let S be a separator in G. Suppose that there exists an edge $uv \in C$ such that $S \cap \{u, v\} = \emptyset$. Since u and v are connected to two different terminals, there would exist a terminal path not intersecting S, a contradiction. So, for each edge $e \in C$, at least one vertex of e belongs to S. Thus the result follows. ■

Theorem 4.12. *Given a terminal cycle $J = (V(J) \cup T(J), E(J))$ of G, with $|T(J)| = q$, inequality (18) defines a facet of $P(G, T)$ if and only if the following hold:*

(a) *q is odd.*
(b) *C is chordless.*
(c) *For $w \in V \setminus C$ adjacent to a terminal, there exists a vertex cover in C of size $\lceil \frac{q}{2} \rceil$ containing C', where $C' = N(w) \cap C$.*
(d) *For all $w \in V \setminus C$ not adjacent to a terminal, there exists a vertex cover of C of size $\lceil \frac{q}{2} \rceil$ containing at least $|C'| - 1$ vertices of C', where $C' = N(w) \cap C$.*

Proof. Let $t_1, \ldots, t_q$ be the terminals of $T(J)$, $v_1, \ldots, v_q$ the nodes of C and $e_i = v_i t_i$, for $i = 1, \ldots, q$. For $i = 1, \ldots, q-1$, let $\overline{P}_{t_i}$ denote the terminal path $(t_i, v_i, v_{i+1}, t_{i+1})$.
($\Rightarrow$)

(a) If q is even, the terminal cycle inequality associated with J can be obtained by summing the terminal path inequalities associated with $\{\overline{P}_{t_1}, \overline{P}_{t_3}, \ldots, \overline{P}_{t_{q-1}}\}$. Thus inequality (18) cannot be facet defining.
(b) Assume that C is odd. If $G[C]$ contains a chord uv, then, J can be decomposed into two terminal cycles J_1 and J_2 with a common edge uv, with q_1 and q_2 terminals, respectively. Suppose, without loss of generality, that J_1 has an odd number of terminals and J_2 has an even one. Also, we may suppose that $v = v_1$ and $v_1, \ldots, v_{q_2}$ are the non terminal nodes of J_2. Let $\mathcal{P}$ be the set of the $\frac{q_2}{2} - 1$ disjoint terminal paths $\{v_2, v_3\}, \{v_4, v_5\}, \ldots, \{v_{q_2-2}, v_{q_2-1}\}$. The terminal cycle inequality associated with J can then be obtained by summing the terminal cycle inequality related to J_1 and the terminal path inequalities related to the terminal paths of $\mathcal{P}$.
(c) Suppose there exists a vertex $w \in V \setminus C$ adjacent to a terminal t' such that each vertex cover of C of size $\lceil \frac{q}{2} \rceil$ contains at most $|C'| - 1$ vertices of C'. Then, each vertex cover of C of size $\lceil \frac{q}{2} \rceil$ does not intersect a one terminal path (t, u, w, t') where $u \in C'$ is not in the vertex cover and t is the terminal adjacent to u. When (18) is saturated by x^S where S is a separator, then S must contain w. Hence (18) cannot define a facet.
(d) Suppose there exists a vertex $w \in V \setminus C$ not adjacent to a terminal, and adjacent to the vertices of a set $C' \subseteq C$, such that each vertex cover of C of size $\lceil \frac{q}{2} \rceil$ contains at most $|C'| - 2$ vertices of C'. Observe that for every two nodes u, u' of C', (t, u, w, u', t') is a terminal path, where t (resp. t') is the terminal adjacent to u (resp. u'). If (18) is saturated by x^S where S is a separator, then S must contain w. Hence (18) cannot define a facet.

($\Leftarrow$) Suppose that all the conditions $(a) - (d)$ are satisfied. Denote by $ax \geq \alpha$ inequality (18) and let $bx \geq \beta$ be an inequality that defines a facet of $P(G, T)$ such that $\{x \in P(G, T) : ax = \alpha\} \subseteq \{x \in P(G, T) : bx = \beta\}$. Since $P(G, T)$ is full dimensional, we need to prove that there exists ρ such that $b = \rho a$.

Let $S_1 = (V \setminus C) \cup \{v_1, v_3, \ldots, v_q\}$ and $S_2 = (S_1 \setminus \{v_1\}) \cup \{v_2\}$. As C is chordless and q is odd, from Assumption 2 of Section 1 we have that S_1 and S_2 are separators of G, and their incidence vectors satisfy (18) with equality. Hence $bx^{S_1} = bx^{S_2} = \beta$, which implies that $b(v_1) = b(v_2)$. By symmetry we obtain that

$$b(u) = b(v) = \rho \text{ for all } u, v \in C, \quad \text{for some } \rho \in \mathbb{R}.$$

Now let $w \in V \setminus C$. Let $C' = N(w) \cap C$. If w is not adjacent to a terminal, by Condition (d), there exists a vertex cover U of C of size $\lceil \frac{q}{2} \rceil$ containing at least $|C'| - 1$ vertices of C'. Let $\overline{S}_w = ((V \setminus C) \cup U) \setminus \{w\}$. As w is not adjacent to a terminal, any terminal path containing w must pass through two vertices of $N(w)$. Since $|N(w) \setminus \overline{S}_w| \leq 1$, $\overline{S}_w$ is a separator of G. The incidence vectors of $\overline{S}_w \cup \{w\}$ and $\overline{S}_w$ both satisfy inequality (18) with equality. Therefore $bx^{\overline{S}_w \cup \{w\}} = bx^{\overline{S}_w}$ and thus $b(w) = 0$. If w is adjacent to a terminal, we can show along the same lines that $b(w) = 0$. Therefore

$$b(v) = 0 \quad \text{for all } v \in V \setminus C.$$

Consequently, we have that $b = \rho a$ and the proof is complete. ■

4.9. Extended terminal cycle inequalities

An *extended terminal cycle* $J = (V(J) \cup T(J), E(J))$ is a general form of a terminal cycle configuration such that each vertex of the cycle C is connected to zero, one or several terminals by means of paths, which will be called *branches* (See Fig. 11 for an illustration). We suppose that $|T(J)| \geq 2$. We note that the internal nodes of the branches are of degree 2. Let $V^1 \subseteq V(J)$ be the set of vertices of degree greater than or equal to 3 and $V^2 \subseteq V(J)$ be the set of vertices of degree equal to 2. For each vertex $v \in V(J)$, let $d_J(v)$ be the degree of v in J. Let $\beta(J) = \sum_{v \in V^1} (d_J(v) - 3)$. Then we have the following

Theorem 4.13. *The inequality*

$$\sum_{v \in V^2} x_v + \sum_{v \in V^1} (d_J(v) - 2) x_v \geq \lceil \frac{|V^1|}{2} \rceil + \beta(J) \tag{19}$$

is valid for $P(G, T)$.

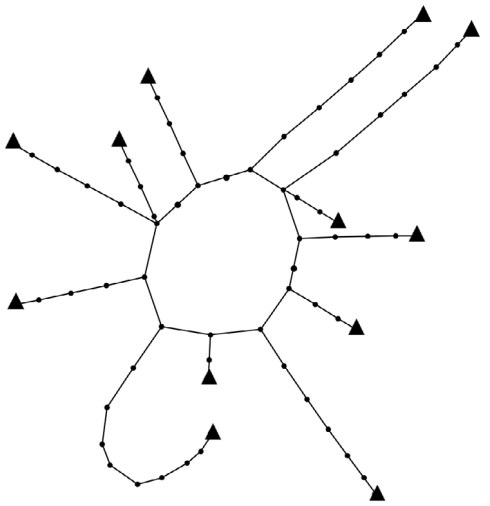

Fig. 11. Extended terminal cycle with $\beta(J) = 2$.

Proof. The proof is by induction on $\beta(J)$. If $\beta(J) = 0$, then the proof is similar to that of Theorem 4.11. So suppose that $\beta(J) \geq 1$, and that inequality (19) is valid for every extended terminal cycle with $\beta(J) \leq s$. We will show that it remains valid when $\beta(J) = s + 1$. So suppose that $\beta(J) = s + 1$. Since $\beta(J) \geq 1$, there must exist two branches P_t and $P_{t'}$ with a common vertex $u \in C$. Let J^1 (resp. J^2) be the extended terminal cycles obtained from J by removing the vertices of $P_t \setminus \{u\}$ (resp. $P_{t'} \setminus \{u\}$). Remark that $\beta(J^1) = \beta(J^2) = s$. From the induction hypothesis, the following inequalities are valid for $P(G, T)$.

$$\sum_{v \in V^2 \setminus P_{t'}} x_v + \sum_{v \in V^1} (d_J(v) - 2)x_v - x_u \geq \lceil \frac{|V^1|}{2} \rceil + \beta(J) - 1,$$

$$\sum_{v \in V^2 \setminus P_t} x_v + \sum_{v \in V^1} (d_J(v) - 2)x_v - x_u \geq \lceil \frac{|V^1|}{2} \rceil + \beta(J) - 1.$$

By summing these inequalities together with $x(P_t \cup P_{t'}) \geq 1$ and $x(u) \geq 0$, dividing the resulting inequality by 2 and rounding up the right hand side, we obtain inequality (19). ■

5. Reduction operations

In this section, we are going to describe some graph reduction operations. These can be used in a preprocessing phase in order to reduce the graph and then accelerate the resolution of the problem.

5.1. Deletion of a subgraph connected to two terminals

Let H be a subgraph of G such that there exist two terminals $t, t' \in T$ adjacent to some vertices in $V(H)$, and no vertex in $V(H) \cup T(H)$ is adjacent to vertices of $(V \cup T) \setminus (V(H) \cup T(H) \cup \{t, t'\})$. See Fig. 12 for an illustration. Let $V^+(H) = V(H) \cup T(H)$.

Lemma 5.1. *Let S^1 (S^2) be a separator of minimum size in graph $G[V^+(H) \cup \{t, t'\}]$ ($G[(V \cup T) \setminus V^+(H)]$). Then $S^1 \cup S^2$ is a minimum size separator in G.*

As a consequence of Lemma 5.1, graph G can be decomposed into $G[V^+(H) \cup \{t, t'\}]$ and $G \setminus H$, and the minimum separator in G can be computed from those of $G[V^+(H) \cup \{t, t'\}]$ and $G \setminus H$.

5.2. Contraction of a subgraph connected to two vertices

Let H be a subgraph of G such that $T(H) = \emptyset$ and there exist two vertices $u, v \in V \setminus V(H)$ adjacent to $V(H)$, and no vertex in $V(H)$ is adjacent to vertices of $(V \cup T) \setminus (V(H) \cup \{u, v\})$. Let G' be the graph obtained from G by deleting the vertices of H and adding an edge between u and v. See Fig. 13 for an illustration.

Lemma 5.2. *A minimum size separator in G' is also a minimum size separator in G.*

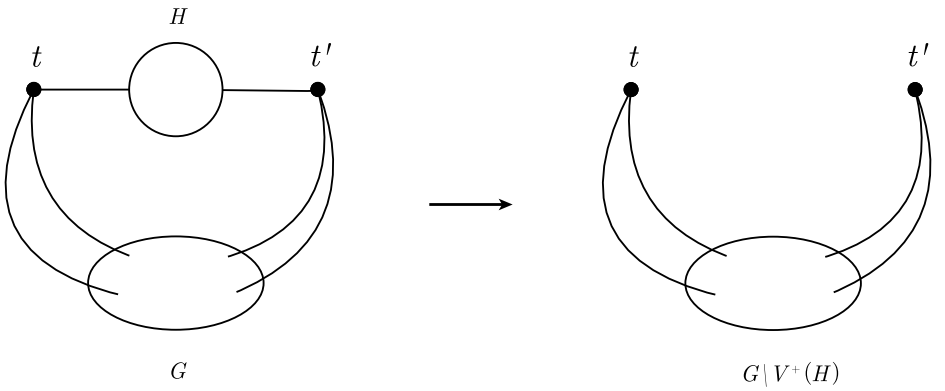

Fig. 12. Deleting a subgraph connected to two terminals.

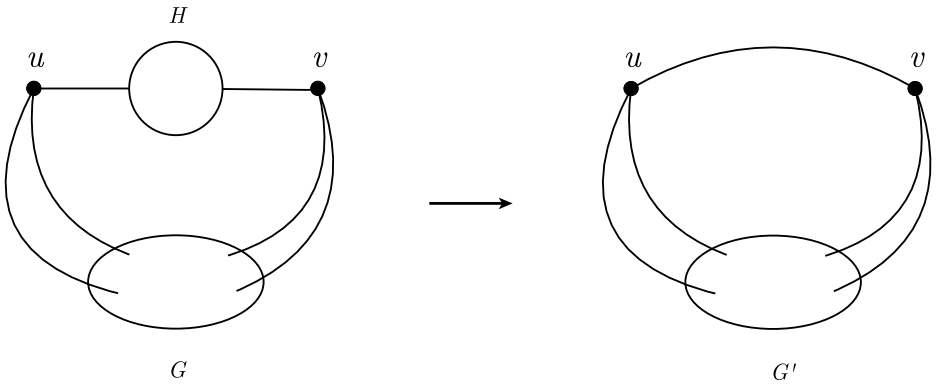

Fig. 13. Contraction of a subgraph connected to two vertices.

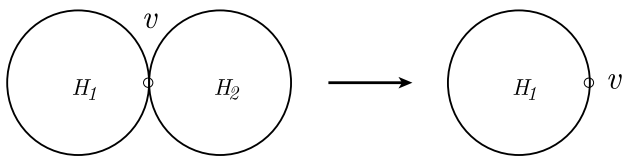

Fig. 14. Deletion of useless components.

5.3. Deletion of useless components

Consider two subgraphs H_1 and H_2 of G such that $V = V(H_1) \cup V(H_2)$, $V(H_1) \cap V(H_2) = \{v\}$ and $T(H_1) = T$. See Fig. 14 for an illustration.

Lemma 5.3. *A minimum size separator in H_1 is also a minimum size separator in G.*

Proof. Remark that there is no terminal path intersecting $V(H_2) \setminus \{v\}$. Thus, no minimal separator intersects $V(H_2) \setminus \{v\}$. ■

It is easily seen that the three operations given by Lemmas 5.1–5.3 can be performed in polynomial time. These operations are used within our Branch-and-Cut algorithm that we present in the next section.

6. Branch-and-Cut algorithm

In this section, we describe a Branch-and-Cut algorithm for the MTVSP. Our aim is to address the algorithmic applications of the theoretical results presented in the previous sections and describe some strategic choices made in order to solve that problem. So, let us assume that we are given a graph $G = (V \cup T, E)$. We suppose that all the operations 5.1–5.3 have been performed in a preprocessing phase, and thus none of these operations can be applied for G.

We now describe the framework of our algorithm. To start the optimization, we consider the linear program given by the terminal path inequalities associated with the terminal paths $P^*_{tt'}$ of minimum length (in terms of number of edges), between each pair of terminals $t, t' \in T$ of graph G together with the trivial inequalities, that is

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ & x(P^*_{tt'}) \geq 1 \quad \forall t, t' \in T, \\ & x_v \leq 1 \quad \forall v \in V, \\ & x_v \geq 0 \quad \forall v \in V. \end{aligned}$$

Note that, for each pair of terminals, there may exist several terminal paths of minimum length (in terms of number of edges) but we add only one terminal path inequality associated with it.

The optimal solution $x^* \in \mathbb{R}^V$ of this relaxation of the MTVSP is feasible for the problem if x^* is an integer vector that satisfies all the terminal path inequalities. Usually, the solution x^* is not feasible for the MTVSP, and thus, in each iteration of the Branch-and-Cut algorithm, it is necessary to generate further inequalities that are valid for the MTVSP but violated by the current solution x^*. For this, one has to solve the so-called separation problem. This consists, given a class of inequalities, in deciding whether the current solution x^* satisfies all the inequalities of this class, and if not, in finding an inequality that is violated by x^*. An algorithm solving this problem is called a separation algorithm. The Branch-and-Cut algorithm uses the inequalities previously described.

We remark that all inequalities are global (i.e. valid for all the Branch-and-Cut tree) and several inequalities may be added at each iteration. Our strategy is to try to detect violated inequalities at each node of the Branch-and-Cut tree in order to obtain the best possible lower bound and thus limit the number of generated nodes.

6.1. Separation procedures

Now, we describe the separation procedures used in our Branch-and-Cut algorithm. We first show that the separation of the terminal path inequalities and the star tree inequalities can be done in polynomial time.

Theorem 6.1. *Inequalities (6) can be separated in polynomial time.*

Proof. Let $x^* \in [0, 1]^V$. For a terminal $t \in T$, add a super terminal t_s to T adjacent to all terminals of $T \setminus \{t\}$. Let $x^*_T \in \mathbb{R}^T$ such that $x^*_T(t_i) = 0$ for all $t_i \in T$. Let $y^* \in [0, 1]^E$ be the vector such that $y^*(uv) = \frac{x^*(u)+x^*(v)}{2}$ for all $uv \in E$ and $u, v \in V$, and $y^*(ut_i) = \frac{x^*(u)+x^*_T(t_i)}{2}$ for all $ut_i \in E$, $u \in V$ and $t_i \in T$. Finding a shortest path, between t and any terminal of $T \setminus \{t\}$, in graph G w.r.t x^* is equivalent to finding a shortest path P_t in graph G w.r.t y^* between t and t_s. Consider the path $\widehat{P}$ among the paths $P_{t'}$, $t' \in T \setminus \{t\}$, of a minimum weight, say $\widehat{z}$. If $\widehat{z} < 1$, then inequality (6), associated with $\widehat{P}$, is violated. Otherwise, all inequalities (6) are satisfied. ■

Since Dijkstra algorithm solves the shortest path problem in $O(|m| + |n + k|log(|n + k|))$ [17], the above algorithm discussed for separating inequalities (6), runs in $O(k(|m| + |n + k|log(|n + k|)))$.

Theorem 6.2. *The star tree inequalities (8) can be separated in polynomial time.*

Proof. We will show that the separation problem for the star tree inequalities can be reduced to a minimum cost flow problem. Let $x^* \in [0, 1]^V$. For a vertex $v \in V$, let $D_v = (V' \cup T', A')$ be the graph obtained from G as follows

1- $T' = T \cup \{t_s\}$, where t_s is a new super terminal.
2- add a vertex v' in V'.
3- for each vertex $u \in V \setminus \{v\}$, add two vertices u_1 and u_2 in V', and the arc (u_1, u_2) in A'.
4- for each edge $uw \in E$ such that $u \neq v$ and $w \neq v$, add two arcs (u_2, w_1) and (w_2, u_1) in A'.
5- for all $u \in N(v) \setminus T$, add an arc (v', u_1) to A', and for all $t \in N(v) \cap T$, add an arc (v', t) in A'.
6- for all $t \in T$, and $u \in N(t) \setminus \{v\}$, add an arc (u_2, t) to A'.
7- for all $t \in T$, add an arc (t, t_s) in A'.

Fig. 15 illustrates the graph transformation. All arcs are given a capacity 1. For all $u \in V \setminus \{v\}$, arc (u_1, u_2) has a weight x^*_u and all other arcs have weight 0.

For an integer value $q \in [3, \ldots, |T|]$, we look for a flow of value q of minimum cost, between v' and t_s in D_v. Let A be the set of arcs used by this flow and z' the cost of this flow. We let $W \subseteq V$ be the set of vertices u of V such that $(u_1, u_2) \in A$. It can be easily seen that if $z' + (q - 1)x^*_{v'} < q - 1$, then the star tree inequality associated with the star tree given by W is violated. ■

The minimum cost flow problem can be solved in $O((n + k)^4CU)$, where U denotes the maximum value among the flow value and arc capacities and C the largest arc cost [18]. The separation of the star tree inequalities thus runs in $O(k(n+k)^4CU)$ time.

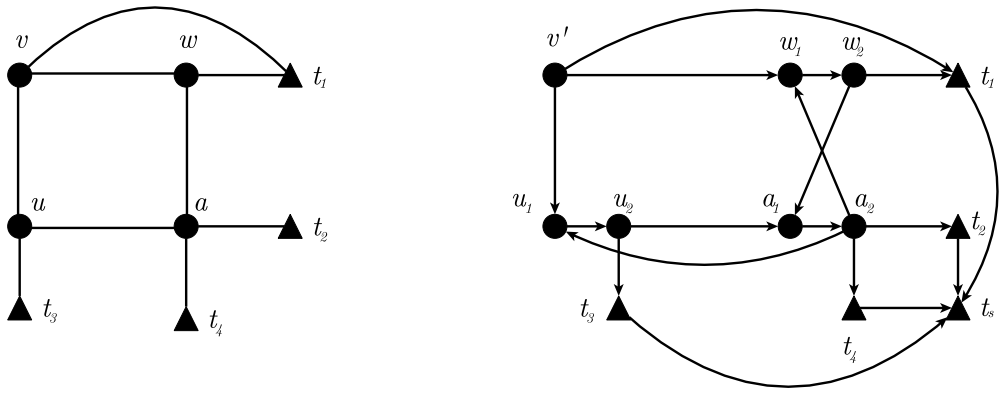

Fig. 15. Transformation graph from G to D_v.

The polynomial algorithm for separating the star tree inequalities, discussed above, may be time consuming. In what follows we devise a heuristic to speed up the separation. The idea of the heuristic is to construct a start tree using shortest paths between a root node and terminals. The heuristic works as follows. We construct a graph G' obtained from G by adding a super terminal t_s adjacent to all terminals of T. Then for each integer $q \in [3, \ldots, k]$ and each vertex $v \in V$, we perform the following procedure.

We set $S = \{v\}$ and the weight of each vertex $u \in V \setminus \{v\}$ to x^*_u and those of $\{t_s, v\} \cup T$ to 0. Then we look for a shortest path P^v between t_s and v. If P^v does not exist, then we stop. Otherwise, we update S, V and T as $S = S \cup (P^v \setminus \{t_s\})$, $V = V \setminus (P^v \setminus \{v\})$ and $T = T \setminus (P^v \setminus \{t_s\})$. If the star tree associated with S has q branches then we stop. Otherwise we look for a new shortest path between t_s and v, and so on.

If set S represents a star tree of G of root vertex v and q branches, and if $x^*(S \setminus (T \cup \{v\})) + (q-1)x^*_v < q-1$, then the star tree inequality associated with S is violated. This heuristic is given in Algorithm 1.

Algorithm 1: Separation heuristic for the star tree inequalities.

Data: $G = (V \cup T, E)$, $v \in V$, $q \in \{3, \ldots, k\}$ and a vector $x^* \in [0, 1]^V$
Result: Star tree inequality violated by x^*

begin
  $S = \{v\}$; $c_v = 0$;
  for $u \in V \setminus \{v\}$ **do**
    $c_u = x^*_u$;
  end
  for $t \in T \cup \{t_s\}$ **do**
    $c_t = 0$;
  end
  for $i = 1, \ldots, q$ **do**
    $P_i \leftarrow$ shortest path in $(V \cup T \cup \{t_s\}, E)$ w.r.t c, between v and t_s;
    if $P_i = \emptyset$ **then**
      Stop algorithm.
    else
      $S \leftarrow S \cup (P_i \setminus \{t_s\})$;
      $V \leftarrow V \setminus (P_i \setminus \{v\})$;
      $T \leftarrow T \setminus (P_i \setminus \{t_s\})$;
    end
  end
  If S is a star tree of root v and q branches then check if the associated star tree inequality is violated by x^* ;
end

Since Dijkstra algorithm solves the shortest path problem in $O((n+k)^2)$ [19], Algorithm 1 runs in $O(k(n+k)^2)$. It follows that the heuristic runs in $O(nk(n+k)^2)$ time.

Now we turn our attention to the separation of the clique star inequalities. This is also performed using a heuristic algorithm. The idea of the heuristic is to generate a set $L(G)$ of cliques in graph G, and for all clique K_q of q vertices, we construct a clique star using shortest paths between each vertex of K_q and terminals.

The heuristic works as follows. We initialize the list $L(G)$ with all cliques in G of size 3. Then, for a fixed number of iterations $l \in \{1, \ldots, k\}$, for each clique $K \in L(G)$ and for all $v \in N(K)$, we check if $K \cup \{v\}$ is a clique and if so, we add it to $L(G)$. Let G' be the graph obtained from G by adding a super terminal t_s adjacent to all terminals of T. During the separation procedure,

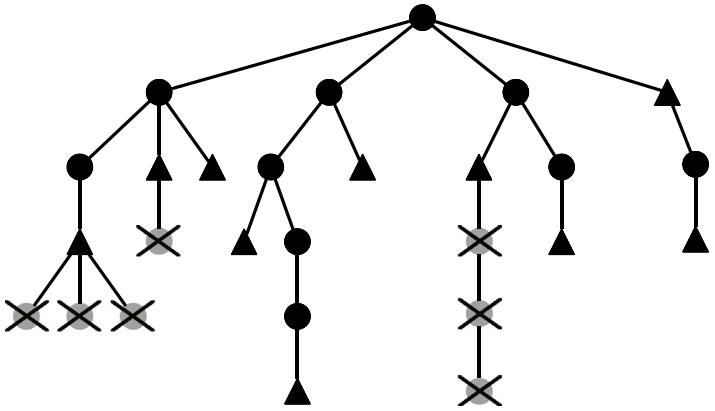

Fig. 16. Deletion of non-terminal leaves.

for each clique $K_q \in L(G)$ of q vertices, we perform the following procedure in G'. We set $S = K_q$ and the weight of each vertex $v \in V$ to x^*_v and those of $\{t_s\} \cup T$ to 0. Then for a vertex $v \in K_q$, we look for a shortest path P^v between t_s and v in $G' \setminus (K_q \setminus \{v\})$. If P^v does not exist, then we stop. Otherwise, we update S, V and T as $S = S \cup (P^v \setminus \{t_s\})$, $V = V \setminus (P^v \setminus \{v\})$ and $T = T \setminus (P^v \setminus \{t_s\})$. If the clique star associated with S has q branches then we stop. Otherwise we look for a new shortest path between t_s and another vertex v' of K_q in $G' \setminus (K_q \setminus \{v'\})$, and so on. If set S represents a clique star of G of clique K_q and q branches, and if $x^*(S) < q - 1$, then the clique star inequality associated with S is violated. This heuristic is given in Algorithm 2.

Algorithm 2: Separation heuristic for the clique star inequalities.

Data: $G = (V \cup T, E)$, a clique $K_q \in L(G)$ of q vertices and a vector $x^* \in [0, 1]^V$
Result: Clique star inequality violated by x^*

begin
  $S = K_q$;
  for $v \in V$ **do**
    $c_v = x^*_v$;
  end
  for $t \in T \cup \{t_s\}$ **do**
    $c_t = 0$;
  end
  for $v \in K_q$ **do**
    $P^v \leftarrow$ shortest path in graph $((V \setminus K_q) \cup \{v\} \cup T \cup \{t_s\}, E)$ w.r.t c, between v and t_s;
    if $P^v = \emptyset$ **then**
      Stop.
    end
    else
      $S \leftarrow S \cup (P^v \setminus \{t_s\})$;
      $V \leftarrow V \setminus (P^v \setminus \{v\})$;
      $T \leftarrow T \setminus (P^v \setminus \{t_s\})$;
    end
  end
  If S is a clique star of clique K_q and q branches then check if the associated clique star inequality is violated by x^* ;
end

Since Dijkstra algorithm solves the shortest path problem in $O((n + k)^2)$ [19], the heuristic runs in $O((n + k)^3)$ time.

Now we discuss the separation of the terminal tree inequalities. For this we devised the following heuristic. Let $x^* \in [0, 1]^V$. Let $y^* \in [0, 1]^E$ be the vector such that $y^*(uv) = \frac{x^*(u)+x^*(v)}{2}$ for all $uv \in E$, $u, v \in V$, and $y^*(ut) = \frac{x^*(u)}{2}$ for all $ut \in E$, $u \in V$ and $t \in T$. We look for a spanning tree R of minimum weight in the graph G w.r.t y^*. Let $\overline{R}$ be the tree obtained from R by deleting the leaf vertices that are not terminals. $\overline{R}$ may have terminals of degree greater than or equal to 2. This means that $\overline{R}$ is not minimal, but the associated terminal tree inequality remains valid for $P(G, T)$. Then we check whether the corresponding terminal tree inequality is violated. For determining the minimum spanning tree we use Kruskal algorithm that runs in $O(mlog(m))$ [19]. Thus our algorithm can be implemented in $O(mlog(m))$ time. The graph in Fig. 16 illustrates the deletion of all non-terminal leaves.

These steps are detailed in Algorithm 3.

Algorithm 3: Separation heuristic for the terminal tree inequalities.

Data: $G = (V \cup T, E)$, $x^* \in [0, 1]^V$
Result: Terminal tree inequality violated by x^*
begin
  Construct $y^* \in \mathbb{R}^E$ from x^* ;
  $S \leftarrow$ spanning tree of minimum weight in G w.r.t y^*;
  while *S has a non terminal vertex of degree* 1 **do**
    $v \leftarrow$ a non terminal vertex of degree 1 in S;
    $S \leftarrow S \setminus \{v\}$;
  end
  Check if the terminal tree inequality induced by S is violated by x^*;
end

We also devised a heuristic for separating the extended terminal cycle inequalities. The idea of the heuristic is to generate a cycle, and construct an extended terminal cycle using shortest paths between each vertex of the cycle and terminals. The heuristic works as follows. First, we look for a cycle C of minimum weight in the graph $G[V]$ w.r.t x^*. This can be done in polynomial time ($|E|$ times the shortest path problem). We then construct a graph G' from G by adding a super terminal t_s adjacent to all terminals of T. We set $S = C$, the weight of each vertex $v \in V$ to x^*_v and those of $\{t_s\} \cup T$ to 0. For each vertex $v \in C$, we look for a shortest path P^v between t_s and v in the graph $G' \setminus (C \setminus \{v\})$. If P^v does not exist, then we stop. If P^v exists note that, as t_s is adjacent to all the terminals and the weights of the terminals and t_s are all equal to 0, P^v can be considered in such a way that it contains only one terminal of T, namely the one in P^v just before t_s. We update S, V and T as $S = S \cup (P^v \setminus \{t_s\})$, $V = V \setminus (P^v \setminus \{v\})$ and $T = T \setminus (P^v \setminus \{t_s\})$. If set S represents an extended terminal cycle of G, let q be the number of its branches. If $x^*(S \setminus T) < \lceil \frac{q}{2} \rceil$, then the extended terminal cycle inequality, associated with the extended terminal cycle induced by S, is violated. This procedure is given in Algorithm 4.

Algorithm 4: Separation heuristic for the extended terminal cycle inequalities.

Data: $G = (V \cup T, E)$, and a vector $x^* \in [0, 1]^V$
Result: Extended terminal cycle inequality violated by x^*
begin
  $C \leftarrow$ Cycle of minimum weight in the graph $G[V]$ w.r.t x^*;
  $S \leftarrow C$;
  for $v \in V$ **do**
    $c_v = x^*_v$;
  end
  for $t \in T \cup \{t_s\}$ **do**
    $c_t = 0$;
  end
  for $v \in C$ **do**
    $P^v \leftarrow$ shortest path in $G' \setminus (C \setminus \{v\})$ w.r.t c, between v and t_s;
    if $P^v = \emptyset$ **then**
      Stop algorithm.
    end
    else
      $S \leftarrow S \cup (P^v \setminus \{t_s\})$;
      $V \leftarrow V \setminus (P^v \setminus \{v\})$;
      $T \leftarrow T \setminus (P^v \setminus \{t_s\})$;
    end
  end
  If S is an extended terminal cycle of cycle C and has $|C|$ branches then check if the associated extended terminal cycle inequality is violated by x^* ;
end

Since Dijkstra algorithm solves the shortest path problem in $O((n + k)^2)$ [19], the heuristic runs in $O(km(n + k)^2)$ time.

6.2. Heuristics and performance guarantee

In [1], the authors show that the multi-terminal vertex separator problem can be solved in polynomial time when the graph has two terminals. In what follows, we will use this result to develop a heuristic with performance guarantee for the problem.

6.2.1. Isolating terminal heuristic

In [20], the authors propose a heuristic for the multi-terminal cut problem (the edge version of the MTVSP). They show that the solution given by the heuristic is of a size guaranteed to be no more than $\frac{2(k-1)}{k}$ times the optimal size. This heuristic can be adapted for the MTVSP. For a terminal $t \in T$, an *isolating terminal set* $S^t \subseteq V$ is a set of vertices intersecting all terminal paths between t and all terminals of $T \setminus \{t\}$. We denote by ST_t the super terminal where all terminals $T \setminus \{t\}$ are merged. The idea of the heuristic is to construct a separator of G, from the union of all minimum isolating terminal sets of G. It works as follows. For each terminal $t \in T$, we compute the separator S^t of minimum size $\omega(S^t)$, intersecting all terminal paths between t and the super terminal ST_t. This can be done in polynomial time [1]. Let S^{t^*} be the separator, among those computed before, of maximum size. Then we return $S = \bigcup_{t \in T \setminus \{t^*\}} S^t$. This heuristic is given in Algorithm 5.

Algorithm 5: Isolating terminal heuristic.

Data: Graph $G = (V \cup T, E)$
Result: Vertex separator
begin
  $S = \emptyset$;
  for $t \in T$ **do**
    Merge all terminals of $T \setminus \{t\}$ into a new terminal t_s;
    $S^t \leftarrow$ The minimum vertex separator of $(V \cup \{t, t_s\}, E)$;
  end
  $t^* \leftarrow \operatorname{argmax}_{t \in T}\{\omega(S^t)\}$;
  $S = \bigcup_{t \in T \setminus \{t^*\}} S^t$;
  return S ;
end

Set S is a separator in G since, for all $t \in T$, S contains a subset of vertices in V intersecting all terminal paths between t and all terminals of $T \setminus \{t\}$.

Theorem 6.3. *The isolating terminal heuristic constructs a multi-terminal vertex separator whose size ω^* is guaranteed to be no more than $(k-1)$ times the optimal size.*

Proof. Let S be the separator given by the isolating terminal heuristic and $S^t \subseteq S$ be the set of vertices of minimum size $\omega(S^t)$, intersecting all terminal paths between t and all terminals of $T \setminus \{t\}$. Let $\overline{S}$ be a separator of minimum size $\overline{\omega}$ in G. For a terminal $t \in T$, let $\overline{S}^t \subseteq \overline{S}$ be a set of vertices having a minimum size $\overline{\omega}^t$, intersecting all terminal paths between t and all terminals in $T \setminus \{t\}$. Clearly, $\omega(S^t) \leq \overline{\omega}^t$. Moreover, each vertex of $\overline{S}$ can belong to each separator $\overline{S}^t$ for each terminal $t \in T$. Thus, $\sum_{t \in T} \overline{\omega}^t \leq k\overline{\omega}$. Therefore

$$\omega^* \leq (k-1)\frac{\sum_{t \in T} \omega(S^t)}{k} \leq \frac{k-1}{k} \sum_{t \in T} \overline{\omega}^t \leq k\frac{k-1}{k}\overline{\omega} \leq (k-1)\overline{\omega}$$

and the result follows. ■

Corollary 6.1. *When $k = 3$, the isolating terminal heuristic is a 2-approximation algorithm for the MTVSP.*

Remark 6.1. The star tree is an example on which the isolating terminal heuristic can give a solution of size equal to (k−1) times the optimal size. Thus the bound given above is tight.

6.2.2. Improved isolating terminal heuristic

This heuristic consists in looking for a separator S by applying the isolating terminal heuristic, and try to pull off one by one each vertex from S, and check if the remaining set is a separator. If not, we put it back in S. It would be more interesting to start by removing the vertices of small degrees. This heuristic is given in Algorithm 6

7. Computational results

The Branch-and-Cut algorithm described in the previous section has been implemented in C++, using CPLEX to manage the Branch-and-Cut tree and also as a lp-solver, and all flow problems are solved using Lemon library [21]. The CPLEX cuts are deactivated. It was tested on an Intel Xeon E312xx machine at 2.39 GHz $\times 1$ with 48GB RAM, running under Linux 64 bits. The maximum CPU run time has been fixed to 4 hours. We use two kinds of instances, the DIMACS graph coloring instances [16] and random graphs, generated using boost graph library [22] in C++. The terminals are new nodes added to the graphs. Each terminal is randomly connected to at least 2 vertices and at most a given $deg\_T \in \mathbb{N}$ vertices of V. The edges incident to the terminals are added respecting the assumptions (3)–(4), given in Section 1. In the tables below, the maximum degree

Algorithm 6: Improved isolating terminal heuristic.

Data: Graph $G = (V \cup T, E)$
Result: Vertex separator $S \subseteq V$
begin
  $S = \emptyset$;
  for $t \in T$ **do**
    Merge all terminals of $T \setminus \{t\}$ into a super terminal TS_t;
    $S^t \leftarrow$ The minimum vertex separator of $(V \cup \{t, t_s\}, E)$;
  end
  $t^* \leftarrow \operatorname{argmax}_{t \in T}\{\omega(S^t)\}$;
  $S = \bigcup_{t \in T \setminus \{t^*\}} S^t$;
  Sort $S = \{v_1, \ldots, v_{|S|}\}$ from the vertex of the smallest degree to the vertex of the largest degree;
  for $(i \in \{1, \ldots, |S|\})$ **do**
    if $S \setminus \{v_i\}$ *is a separator* **then**
      $S \leftarrow S \setminus \{v_i\}$;
    end
  end
  return S;
end

of the terminals is fixed in relation with the size of the graph, *i.e.,* the higher the size of graph is, the higher the maximum degree of terminals is. In all our experiments, we have used the reduction operations described in the previous section to reduce the graph. As the separation algorithms generate less than 10 inequalities per family of inequalities and per iteration, we perform all of them for each separation phase. Moreover, we tested different separation orders, and we noticed that separating all inequalities, at each separation phase, gives the best results. Each instance is given by its name followed by an extension representing the number of nodes of the graph.

In the following tables, we have the following entries

n	:	the number of vertices in V.
m	:	the number of edges in E.
k	:	the number of terminal in T.
d %	:	the density of graphs without terminals.
Tp	:	the number of terminal path inequalities separated.
St	:	the number of star tree inequalities separated.
Cs	:	the number of clique star inequalities separated.
Tt	:	the number of terminal tree inequalities separated.
Etc	:	the number of extended terminal cycle inequalities separated.
Nodes	:	the number of node in the branching tree.
Gap %	:	the relative error between the best upper and lower bound obtained at the root node of the branching tree.
CPU_BC	:	the CPU time given by the Branch-and-Cut algorithm in seconds, on the natural formulation.
CPU_CPLEX	:	the CPU time given by CPLEX in seconds, on the double index formulation.
deg_T	:	the maximum degree of each terminal.
Heur_val	:	the objective value of the solution given by the improved isolating terminal heuristic.
Opt_val	:	the optimal objective value.
CPU_Heur	:	the CPU time of the Heuristic, in seconds.
Heur_Gap %	:	the value of $\frac{\text{Heur\_val} - \text{Opt\_val}}{\text{Opt\_val}}$.

In the first Tables 2 and 3, we compare the Branch-and-Cut algorithm with CPLEX for the compact double index formulation.

Table 2 presents the results obtained for DIMACS instances associated to graphs having up to 500 nodes. The number of terminals is fixed to 6 for graphs with less than 99 nodes and 8 for the others. As we can remark in the table, all instances were solved by the Branch-and-Cut algorithm in less than 33 seconds. The bold values in the CPU_BC column indicate the instances for which the Branch-and-Cut algorithm solves the instance faster than Cplex with the double indices formulation. We notice that in 50% of the instances, the Branch-and-Cut algorithm beats the Cplex. The valid inequalities 4, 7, 16 and 18 have been

Table 2
Results from DIMACS instances.

Instances	n	m	k	Tp	St	Cs	Tt	Etc	Nodes	Gap	CPU_BC	CPU_Cplex
Myciel5	47	258	6	48	189	0	28	0	7	35.20	7.96	0.50
Queen8_8	64	1 477	6	30	22	21	2	9	1	0.00	0.99	0.32
Huck	74	624	6	64	51	3	5	1	23	19.20	3.39	1.51
Jean	80	533	6	69	100	7	1	0	47	28.10	6.94	1.66
David	87	835	6	31	147	25	26	3	26	28.50	8.58	4.24
Myciel6	95	778	6	35	38	0	26	15	1	0.00	1.26	0.31
Queen8_12	96	2 762	6	33	36	35	1	18	1	0.00	1.45	1.41
Queen10_10	100	2 967	8	56	31	32	4	6	1	0.00	**1.55**	2.38
Games120	120	1 307	8	64	377	138	8	18	9	35.40	**16.50**	23.95
DSJC125	125	764	8	105	70	12	64	12	1	0.00	**2.33**	3.87
Miles250	128	804	8	103	127	16	4	2	29	28.10	5.75	0.42
Miles500	128	2 370	8	97	386	62	9	9	6	31.80	16.44	2.41
Miles750	128	4 256	8	63	60	49	13	15	1	0.00	3.62	2.87
Miles1000	128	6 462	8	53	30	24	10	12	1	0.00	**2.03**	6.38
Anna	138	1 022	8	92	344	55	0	1	74	16.60	17.99	0.72
Queen12_12	144	5 224	8	85	20	18	9	9	1	0.00	**1.44**	8.18
Mulsol.i.2	188	3 920	8	116	63	34	2	13	1	0.00	**2.69**	6.32
Myciel7	191	2 387	8	58	27	0	13	13	1	0.00	**1.34**	17.09
Queen14_14	196	8 399	8	56	19	15	12	11	1	0.00	**1.24**	22.52
Mulsol.i.1	197	3 952	8	71	11	5	10	3	1	0.00	**0.58**	5.61
Zeroin.i.3	206	3 576	8	28	47	48	1	2	1	0.00	**2.03**	5.49
Queen16_16	256	12 674	8	88	27	19	5	11	1	0.00	**2.10**	41.55
School1	385	19 129	8	109	16	15	12	5	1	0.00	**1.08**	103.00
DSJR500	500	7 140	8	29	374	331	3	141	4	40.00	32.35	19.80

Table 3
Results from random instances.

Instances	n	m	k	Tp	St	Cs	Tt	Etc	Nodes	Gap	CPU_BC	CPU_Cplex
R_50	50	511	7	48	33	22	31	6	1	0.00	1.24	0.44
R_70	70	993	7	44	18	14	19	4	1	0.00	**0.64**	0.78
R_100	100	1 985	7	45	13	13	12	5	1	0.00	**1.02**	3.07
R_300	300	17 792	7	67	8	7	8	2	1	0.00	**2.31**	55.67
R_600	600	70 673	7	23	10	9	9	5	1	0.00	**4.04**	298.02
R_700	700	96 432	7	48	7	6	5	1	1	0.00	**3.96**	360.79
R_800	800	125 978	10	95	26	23	0	13	1	0.00	**20.98**	1323.90
R_900	900	159 331	10	96	39	36	1	15	1	0.00	**31.31**	1959.60
R_1000	1000	196 771	10	86	35	34	0	16	1	0.00	**34.35**	2395.20
R_1200	1200	283 386	10	92	26	24	0	9	1	0.00	**41.01**	4595.10
R_1300	1300	332 861	10	47	30	30	0	15	1	0.00	**56.91**	5142.00
R_1500	1500	442 444	10	99	33	32	0	16	1	0.00	**92.64**	11735.00
R_1800	1800	637 307	15	110	44	40	40	17	1	0.00	**379.95**	–
R_2000	2000	786 639	15	108	37	32	35	16	1	0.00	**468.78**	–
R_2100	2100	867 430	15	214	34	31	28	15	1	0.00	**641.02**	–
R_2300	2300	1 041 158	15	217	29	28	25	16	1	0.00	**851.49**	–
R_3000	3000	1 770 773	15	215	33	30	19	12	1	0.00	**1698.90**	–
R_3300	3300	2 142 370	15	112	38	37	11	18	1	0.00	**2129.60**	–
R_3800	3800	2 841 805	15	217	28	27	26	13	1	0.00	**2363.80**	–
R_4200	4200	3 472 117	15	222	37	35	30	22	1	0.00	**3856.30**	–
R_4500	4500	3 987 339	15	217	41	40	31	15	1	0.00	**5248.10**	–
R_4800	4800	4 537 794	15	218	31	22	31	10	1	0.00	**4611.80**	–
R_5000	5000	4 922 710	15	110	27	26	27	16	1	0.00	**4291.40**	–

efficient for strengthening the formulation. Indeed 62% of the instances were solved by the Branch-and-Cut algorithm in the first node of the branching tree. We also notice that all the valid inequalities appear in the instances Queen8_8, Queen8_12, Queen10_10, Queen12_12, Queen14_14, Queen15_15, since the underlying graphs are holeless. However, no clique star inequality appears for instances Myciel5, Myciel6 and Myciel7 since they are induced by triangle free graphs. Moreover, we can remark that the Branch-and-Cut algorithm generates a few number of valid inequalities. Only 4 instances over 24 were solved by the Branch-and-Cut algorithm in more than 10 seconds. For these four instances, the number of star tree inequalities is greater than 340, in contrast with the other instances, where the number of star tree inequalities is less than 200.

Table 3 presents the results obtained for random instances on graphs having up to 5000 nodes. The maximum degree of terminals is fixed to 7 for instances with less than 700 nodes, 10 for instances with the number of nodes between 800 and 1500 and to 15 for instances with 1800 nodes and more. As it appears, in 95.6% of instances, the Branch-and-Cut algorithm beats Cplex. Indeed, Cplex was faster than the Branch-and-Cut in only 1 instance (R_50) with a difference of 0.8 s. In the

Table 4
Results from random instances with different density.

Instances	n	m	k	d %	Nodes	Gap	CPU_BC
R_1000	1000	90 682	10	18	1	0.00	15.18
		196 771		39	1	0.00	34.35
		275 346		55	1	0.00	84.88
R_2000	2000	362 335	15	18	1	0.00	172.00
		786 639		39	1	0.00	468.70
		1 100 921		55	1	0.00	1183.00
R_3000	3000	815 760	15	18	1	0.00	622.30
		1770 773		39	1	0.00	1698.00
		2478 823		55	1	0.00	1650.00
R_4000	4000	1450 789	15	18	1	0.00	1010.15
		3150 332		39	1	0.00	2502.08
		4410 131		55	1	0.00	3198.85
R_5000	5000	2267 316	15	18	1	0.00	1904.00
		4922 710		39	1	0.00	4291.00
		6890 000		55	1	0.00	5802.00

Table 5
Results from random instances with different number of terminals.

Instances	n	m	k	Nodes	Gap %	CPU_BC
R_1000	1000	196 756	7	1	0.00	14.35
		196 771	10	1	0.00	34.35
		196 787	13	1	0.00	103.00
R_2000	2000	786 625	12	1	0.00	499.90
		786 639	15	1	0.00	468.70
		786 652	18	1	0.00	990.70
R_3000	3000	1770 758	12	1	0.00	1222.21
		1770 773	15	1	0.00	1698.00
		1770 787	18	1	0.00	2472.00
R_4000	4000	3150 315	12	1	0.00	2274.11
		3150 332	15	1	0.00	2502.08
		3150 347	18	1	0.00	4360.06
R_5000	5000	4922 700	12	1	0.00	3367.00
		4922 710	15	1	0.00	4291.00
		4922 728	18	1	0.00	7919.00

Table 6
Results from random instances with different degree of terminals.

Instances	n	m	k	deg_T	Nodes	Gap %	CPU_BC
R_1000	1000	196 751	10	5	1	0.00	26.93
		196 771		10	1	0.00	34.35
		196 769		15	1	0.00	43.07
R_2000	2000	786 617	15	5	1	0.00	408.50
		786 639		10	1	0.00	468.70
		786 638		15	1	0.00	715.70
R_3000	3000	1770 741	15	5	1	0.00	1446.00
		1770 773		10	1	0.00	1698.00
		1770 766		15	1	0.00	2650.00
R_4000	4015	3150 301	15	5	1	0.00	1676.69
		3150 332		10	1	0.00	2502.08
	4015	3150 322		15	1	0.00	3007.54
R_5000	5000	4922 695	15	5	1	0.00	4015.00
		4922 710		10	1	0.00	4291.00
		4922 714		15	1	0.00	5027.00

other instances, the slack between the CPU_times increases exponentially. For instance, for R_100 the difference is 2.05 s and for R_1300 it is 5085.09 s. Also note that all instances were solved in the root of the branching tree by the Branch-and-Cut algorithm. Indeed, for random instances our valid inequalities are enough for finding the optimal solutions without branching. Cplex, with the Double indices formulation, could not solve the instances with 1800 vertices and more. This can be explained by the out of memory due to the number of constraints in the double indices formulation when the number of edges is high. Also random graphs have no specific structure and this let the instance harder to solve. For example, for instance *R*_1800, the formulation has more than 9.5 millions of constraints. For both DIMACS and random graphs, the number of star tree inequalities, clique star inequalities and terminal tree inequalities is roughly the same.

In Tables 4–6, we vary the density of the graph, the number of terminals and the maximum degree of the terminals.

Table 7
Results from DIMACS instances without the valid inequalities.

Instances	n	m	k	Tp	Nodes	Gap %	CPU_BC
Myciel5	47	258	6	161	44	35.2	0.06
Queen8_8	64	1 477	6	129	30	15.6	0.04
Huck	74	624	6	86	16	19.2	0.01
Jean	80	533	6	139	43	28.1	0.03
David	87	835	6	154	45	28.5	0.04
Myciel6	95	778	6	206	80	36.1	0.08
Queen8_12	96	2 762	6	176	56	38.0	0.08
Queen10_10	100	2 967	8	235	67	38.6	0.11
Games120	120	1 307	8	304	338	35.4	0.57
DSJC125	125	764	8	311	204	39.1	0.31
Miles250	128	804	8	148	57	28.1	0.05
Miles500	128	2 370	8	365	124	31.8	0.22
Miles750	128	4 256	8	372	148	37.5	0.30
Miles1000	128	6 462	8	327	86	37.5	0.21
Anna	138	1 022	8	354	91	16.6	0.13
Queen12_12	144	5 224	8	282	90	36.0	0.18
Mulsol.i.2	188	3 920	8	341	164	37.5	0.26
Myciel7	191	2 387	8	284	121	38.6	0.21
Queen14_14	196	8 399	8	279	89	41.3	0.24
Mulsol.i.1	197	3 952	8	183	56	38.8	0.11
Zeroin.i.3	206	3 576	8	257	65	35.7	0.12
Queen16_16	256	12 674	8	364	142	39.2	0.42
School1	385	19 129	8	359	198	38.8	0.53
DSJR500	500	7140	8	697	2735	40.0	26.52

Table 8
Results from random instances without the valid inequalities.

Instances	n	m	k	Tp	Nodes	Gap %	CPU_BC
R_50	50	511	7	179	48	38.8	0.05
R_70	70	993	7	178	42	39.4	0.05
R_100	100	1 985	7	172	47	38.8	0.06
R_300	300	17 792	7	163	53	39.4	0.43
R_600	600	70 673	7	150	53	39.4	1.16
R_700	700	96 432	7	150	35	38.8	1.31
R_800	800	125 978	10	677	406	42.0	10.55
R_900	900	159 331	10	821	813	43.1	23.00
R_1000	1000	196 771	10	604	463	40.0	17.36
R_1200	1200	283 386	10	670	461	42.0	35.34
R_1300	1300	332 861	10	646	374	42.0	54.95
R_1500	1500	442 444	10	689	657	42.5	90.38
R_1800	1800	637 307	15	1922	3168	44.8	716.00
R_2000	2000	786 639	15	1326	2504	42.6	653.90
R_2100	2100	867 430	15	2172	3716	45.1	1441
R_2300	2300	1041 158	15	1593	2190	44.2	1390
R_3000	3000	1770 773	15	1583	3856	43.3	3570
R_3300	3300	2142 370	15	2055	3498	45.5	5751
R_3800	3800	2841 805	15	1651	2444	44.4	6035
R_4200	4200	3472 117	15	1969	3494	45.0	9521
R_4500	4500	3987 339	15	2049	3753	45.0	1230
R_4800	4800	4537 794	15	1743	2525	44.5	9282
R_5000	5000	4922 710	15	1451	1839	44.7	10037

Table 4 represents the results for random graphs, with different densities, 18%, 39% and 55%. We notice that in almost all the instances, the higher the density is, the higher CPU time is.

Table 5 represents the results for random graphs with different numbers of terminals. In almost all the instances, the higher number of terminals is, the higher the CPU time is. All instances are solved in the root node of the branching tree.

Table 6 represents the results for random graphs with different values of deg_T (We recall that the terminals are randomly connected to at least 2 vertices and at most deg_T vertices of V). We notice that the higher deg_T is, the higher CPU time is. Also, the Branch-and-Cut algorithm could solve all instances in the root node of the branching tree.

Table 7 is for DIMACS graphs and Table 8 is for random graphs both give the results of the Branch-and-Cut algorithm without the additional valid inequalities, *i.e.,* with only the terminal path inequalities. Observe that when the number of vertices is less than 1800, the CPU time is less than the one in Tables 2 and 3. However, the number of nodes in the branching tree and the gap are higher. Indeed, the linear relaxation of the natural formulation is weak. For the instances with at least 1800 vertices, we notice that the CPU time, gap and number of nodes in the branching tree in Table 8 are higher than the

Table 9
Results from DIMACS instances given by the improved isolated terminal heuristic.

Instances	n	m	k	Heur_val	Opt_val	CPU_Heur	Heur_Gap %
Myciel5	47	258	6	17	17	0.00	0.00
Queen8_8	64	1477	6	16	16	0.01	0.00
Huck	74	624	6	14	13	0.00	7.14
Jean	80	533	6	16	16	0.00	0.00
David	87	835	6	14	14	0.00	0.00
Myciel6	95	778	6	18	18	0.00	0.00
Queen8_12	96	2762	6	21	21	0.02	0.00
Queen10_10	100	2967	8	22	22	0.03	0.00
Games120	120	1 307	8	24	24	0.01	0.00
DSJC125	125	764	8	23	23	0.01	0.00
Miles250	128	804	8	16	16	0.01	0.00
Miles500	128	2 370	8	24	22	0.03	8.33
Miles750	128	4 256	8	24	24	0.05	0.00
Miles1000	128	6 462	8	24	24	0.07	0.00
Anna	138	1 022	8	26	21	0.02	19.20
Queen12_12	144	5 224	8	25	25	0.06	0.00
Mulsol.i.2	188	3 920	8	28	28	0.04	0.00
Myciel7	191	2 387	8	22	22	0.03	0.00
Queen14_14	196	8 399	8	23	23	0.10	0.00
Mulsol.i.1	197	3 952	8	18	18	0.04	0.00
Zeroin.i.3	206	3 576	8	21	21	0.04	0.00
Queen16_16	256	12 674	8	28	28	0.15	0.00
School1	385	19 129	8	27	27	0.20	0.00
DSJR500	500	7 140	8	25	25	0.10	0.00

Table 10
Results from random instances given by the improved isolated terminal heuristic.

Instances	n	m	k	Heur_val	Opt_val	CPU_Heur	Heur_Gap %
R_50	50	511	7	18	18	0.00	0.00
R_70	70	993	7	19	19	0.01	0.00
R_100	100	1 985	7	18	18	0.02	0.00
R_300	300	17 792	7	19	19	0.16	0.00
R_600	600	70 673	7	19	19	0.73	0.00
R_700	700	96 432	7	18	18	0.99	0.00
R_800	800	125 978	10	44	44	2.95	0.00
R_900	900	159 331	10	51	51	5.09	0.00
R_1000	1000	196 771	10	45	45	6.03	0.00
R_1200	1200	283 386	10	44	44	9.88	0.00
R_1300	1300	332 861	10	44	44	12.01	0.00
R_1500	1500	442 444	10	47	47	19.49	0.00
R_1800	1800	637 307	15	78	78	71.56	0.00
R_2000	2000	786 639	15	68	68	106.30	0.00
R_2100	2100	867 430	15	82	82	147.90	0.00
R_2300	2300	1041 158	15	70	70	164.00	0.00
R_3000	3000	1770 773	15	75	75	426.70	0.00
R_3300	3300	2142 370	15	80	80	768.80	0.00
R_3800	3800	2841 805	15	72	72	797.70	0.00
R_4200	4200	3472 117	15	81	81	1074.00	0.00
R_4500	4500	3987 339	15	81	81	1283.00	0.00
R_4800	4800	4537 794	15	74	74	1458.00	0.00
R_5000	5000	4922 710	15	67	67	1169.00	0.00

corresponding ones in Table 3. Our inequalities have a great impact for large instances. The size of the branching tree has also been reduced.

Tables 9 and 10 present the results given by the improved isolated terminal heuristic for DIMACS and random instances, respectively. We can remark that the heuristic gives very good results in short time. In 93.6% of the instances, the solution given by the heuristic is optimal. For the other instances, the gap is very small. On the other hand, when the number of terminals is high, the CPU time of the heuristic becomes high. All DIMACS instances were solved in 0.2 s or less.

8. Conclusion

In this paper we have considered the multi-terminal vertex separator problem. We have first shown that the problem is NP-hard. Then we have proposed two integer programming formulations for the problem. For one of them we have identified some valid inequalities and discussed their facial structure. Using this, we have developed a Branch-and-Cut algorithm for

the problem and presented extensive computational results. These show the effectiveness of the valid inequalities used in the algorithm. The computational results have also permitted to measure the importance of our separation heuristics. In particular the heuristic for separating star tree inequalities seemed more efficient than the exact polynomial algorithm. We have also proposed two heuristics for the problem, and analyzed their performance guarantee. The improved isolating terminal heuristic has appeared to be very efficient since it could find the optimal solution for most of the instances.

Now several questions may arise. First, it would be interesting to characterize the graphs for which the terminal path, star tree, clique star, terminal cycle and terminal tree inequalities suffice to describe the multi-terminal vertex separator polytope. Indeed, as it appears from the computational results, using these inequalities, many instances have been solved in the root of the Branch-and-Cut tree. In [13], the authors characterize the class of graph for which the system given by (6) is TDI. It would be also interesting to describe graphs for which the system given by (6) together with classes of valid inequalities among those given in this paper is TDI. Finally, it would be of interest to develop an extended formulation for the problem and to compare a Branch-and-Price algorithm and the algorithm proposed in this paper.

Acknowledgments

We are grateful to the anonymous referees for their constructive comments that permitted to correct some errors introduced in the first version, and improve the presentation of the paper. This research benefited from the support of the FMJH program PGMO and from the support to this program from ADF-THALES-ORANGE-CRITEO.

References

[1] W. Ben-Ameur, M. Didi Biha, On the minimum cut separator problem, Networks 59 (1) (2012) 30–36.
[2] N. Garg, V.V. Vazirani, M. Yannakakis, Multiway cuts in node weighted graphs, J. Algorithms 50 (1) (2004) 49–61.
[3] D. Marx, Parameterized graph separation problems, Theoret. Comput. Sci. 351 (3) (2006) 394–406.
[4] M. Xiao, Simple and improved parameterized algorithms for multiterminal cuts, Theory Comput. Syst. 46 (4) (2010) 723–736.
[5] W.H. Cunningham, The optimal multiterminal cut problem, in: Reliability of Computer and Communication Networks, 1989, pp. 105–120.
[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.-A. Mohamed-Sidi, J. Neto, The k-separator problem, in: Computing and Combinatorics, Springer, 2013, pp. 337–348.
[8] W. Ben-Ameur, M.-A. Mohamed-Sidi, J. Neto, The k-separator problem: polyhedra, complexity and approximation results, J. Comb. Optim. 29 (1) (2015) 276–307.
[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] C. de Souza, E. Balas, The vertex separator problem: algorithms and computations, Math. Program. 103 (3) (2005) 609–631.
[12] E. Lee, Partitioning a graph into small pieces with applications to path transversal, in: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2017, pp. 1546–1558.
[13] G. Naves, V. Jost, The graphs with the max-mader-flow-min-multiway-cut property, 2011, arXiv preprint arXiv:1101.2061.
[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] I. Dinur, S. Safra, On the hardness of approximating minimum vertex cover, Ann. of Math. (2005) 439–485.
[16] http://www.info.univ-angers.fr/pub/porumbel/graphs/.
[17] M. Barbehenn, A note on the complexity of dijkstra's algorithm for graphs with weighted vertices, IEEE Trans. Comput. 47 (2) (1998) 263.
[18] P. Kovács, Minimum-cost flow algorithms: an experimental evaluation, Optim. Methods Softw. 30 (1) (2015) 94–127.
[19] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency. Algorithms and Combinatorics, Volume 24, Springer, 2003.
[20] 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.
[21] https://lemon.cs.elte.hu/trac/lemon.
[22] http://www.boost.org/doc/libs/1_60_0/libs/graph/.