Computers & Operations Research 38 (2011) 1483–1492

Contents lists available at ScienceDirect

Computers & Operations Research

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

An exact approach to the problem of extracting an embedded network matrix

Rosa M.V. Figueiredo [a,b,*], Martine Labbé [c], Cid C. de Souza [d]

[a] Rio de Janeiro State University, Institute of Mathematics and Statistics, 20550-900 Rio de Janeiro-RJ, Brazil
[b] University of Aveiro, Department of Mathematics, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
[c] Université Libre de Bruxelles, Department of Computer Science, CP 210/01, B-1050 Brussels, Belgium
[d] University of Campinas, Institute of Computing, 13083-852 Campinas-SP, Brazil

ARTICLE INFO

Available online 11 January 2011

Keywords:
Network matrix
Signed graph
Facets of polyhedra
Branch-and-cut

ABSTRACT

We study the problem of detecting a maximum embedded network submatrix in a $\{-1,0,+1\}$-matrix. Our aim is to solve the problem to optimality. We introduce a 0–1 integer linear programming formulation for this problem based on its representation over a signed graph. A polyhedral study is presented and a branch-and-cut algorithm is described for finding an optimal solution to the problem. Some computational experiments are carried out over a set of instances available in the literature as well as over a set of random instances.

© 2011 Elsevier Ltd. Open access under the Elsevier OA license.

1. Introduction

The knowledge of a special structure in a matrix defining a linear programming problem can be used to speed up its resolution. It is well known that one of these special structures is a network matrix. A matrix B is called a *network matrix* if the elements of B belong to the set $\{-1,0,+1\}$ and, additionally, if every column of B contains at most one element $+1$ and at most one element -1. Given a row of matrix B, the operation of changing the signs of all non-zero row elements is called a *reflection* of this row. A matrix B is called a *reflected network matrix* if there exists a set of row reflections that transforms matrix B into a network matrix.

In this work, we consider the problem of detecting a maximum embedded reflected network (DMERN). Given a $\{-1,0,+1\}$-matrix A, the DMERN problem consists of finding the maximum number of rows that define a submatrix B of A such that B is a reflected network matrix. Let $\nu(A)$ denote this number. In the remainder of this paper, we assume that A is a $\{-1,0,+1\}$-matrix.

The DMERN problem is known to be NP-hard [5]. Heuristic approaches to solve this problem are of particular interest since, from a practical point of view, we need to extract large embedded networks quickly. In fact, a number of heuristics have been studied in the literature [1,6–9] for the DMERN problem. These heuristics were always evaluated by comparing their relative performance. In addition to not knowing how far the solution obtained lies from the global optimum, this method of evaluation may cause wrong decisions that lead to incorrect conclusions. An absolute evaluation would only be possible if the optimal solution was known for a set of instances. To our knowledge, this paper is the first to present an exact approach to solve the DMERN problem.

In [9], Gulpinar et al. showed that the DMERN problem is closely related to that of balancing subgraphs in signed graphs. A signed graph is a graph whose edges are labeled by signs. The authors proved that the problem of finding a maximum embedded reflected network can be formulated as an optimization problem over the set of all cuts of a graph.

In this work, we propose an integer programming formulation to the DMERN problem based on the results of Gulpinar et al. We also investigate the structure of the polytope associated with the problem. Finally, based on this study, we propose a branch-and-cut method to solve the DMERN problem to optimality.

We now give some notations and definitions to be used throughout the paper. Let $G=(V,E)$ be an undirected graph. For an edge set $B \subseteq E$, let $G[B]$ ($V[B]$) denote the subgraph of G (subset of vertices) induced by B. For a vertex set $S \subseteq V$, let $E[S]=\{(i,j) \in E | i,j \in S\}$ denote the subset of edges induced by S. Also, for a vertex set $S \subseteq V$, we define $\delta(S)=\{(i,j) \in E | i \in S, j \in V \setminus S\}$ and $N(S)=\{j \in V | (i,j) \in \delta(S)\}$. We say that $\delta(S)$ is the *cut* defined by S. A set $K \subseteq V$ is called a *clique* if each pair of vertices in K is joined by an edge. A set $I \subseteq V$ is called a *stable set* if no pair of vertices in I is joined by an edge. Let $\alpha(G)$ denote the cardinality of a maximum stable set of vertices in G. We represent a cycle by its vertex set $C \subseteq V$ and denote by E^C its edge set. A chordless cycle is called a *hole*; an *odd hole* is a hole with an odd number of vertices. Now, consider a function $s: E \to \{+,-\}$ that

* Corresponding author at: University of Aveiro, Department of Mathematics, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal.
E-mail addresses: rosamaria.figueiredo@gmail.com (R.M.V. Figueiredo), mlabbe@ulb.ac.be (M. Labbé), cid@ic.unicamp.br (C.C. de Souza).

0305-0548 © 2011 Elsevier Ltd. Open access under the Elsevier OA license.
doi:10.1016/j.cor.2011.01.003

assigns a sign to each edge in E. An undirected graph G together with a function s is called a *signed graph*. Let $G=(V,E,s)$ denote a signed graph. In this text, a signed graph is allowed to have parallel edges but no loops. Also, we assume that parallel edges always have opposite signs. An edge $e \in E$ is called *negative* if $s(e)=-$ and *positive* if $s(e)=+$. Let E^- and E^+ denote, respectively, the set of negative and positive edges in a signed graph. Notice that, according to the definitions above, $E=E^- \cup E^+$ and $E^- \cap E^+$ is the set of parallel edges in G. We define $G^-=(V,E^-)$. Also, for a vertex set $S \subseteq V$, we define $\delta^-(S)=\delta(S)\cap E^-$, $\delta^+(S)=\delta(S)\cap E^+$, $N^-(S)=\{j \in V | (i,j) \in \delta^-(S)\}$ and $N^+(S)=\{j \in V | (i,j) \in \delta^+(S)\}$. A signed graph $G=(V,E,s)$ is *balanced* if the vertex set V can be partitioned into the sets W and $V\backslash W$ in such a way that $E[W] \cup E[V\backslash W]=E^+$. A set $K \subseteq V$ is called a *negative clique* if each pair of vertices in K is joined by a negative edge. For the sake of conciseness, i stands for $\{i\}$ and we say that an edge (i,j) belongs to cycle C (C contains edge (i,j)), if $(i,j) \in E^C$.

The remainder of the paper is structured as follows. In Section 2, we relate the problem of extracting an embedded network matrix to an optimization problem defined over an associated signed graph. An integer programming formulation based on signed graphs is presented in Section 3. In Section 4, we investigate the polyhedral structure of the polytope associated with the formulation introduced. A branch-and-cut algorithm to solve the DMERN problem is outlined in Section 5. In Section 6, preliminary computational results are reported for test problems available in the literature as well as for a set of random instances. Finally, in Section 7 we present concluding remarks and discuss directions for further investigation.

2. Network matrices and signed graphs

Consider a matrix $A=[a_{ik}]$ with n rows. Two rows of matrix A are said to be in conflict if they both have a $+1$ or they both have a -1 in the same column. A signed graph $G(A)$ can be used to represent existing conflicts in a matrix A [9]. The vertex set of $G(A)$ is defined as $V=\{1,\ldots,n\}$. The set of negative and positive edges of $G(A)$ are defined as follows: an edge $(i,j) \in E^-$ if and only if $a_{ik}=a_{jk} \neq 0$ for some column k of matrix A; an edge $(i,j) \in E^+$ if and only if $a_{ik}=-a_{jk} \neq 0$ for some column k of matrix A. A signed graph G is called a *network graph* if and only if there exists a reflected network matrix A such that $G=G(A)$.

Notice that, an edge (i,j) belongs to E^- if rows i and j are in conflict. Likewise, an edge (i,j) belongs to E^+ if the reflection of either row i or row j will create a conflict in the resulting matrix. Therefore, if an edge (i,j) is a parallel one, i.e., $(i,j) \in E^- \cap E^+$, then rows i and j cannot belong at the same time to a reflected network submatrix of A.

Gulpinar et al. [9] showed that the DMERN problem defined by a matrix A can be solved as a stable set problem defined over the signed graph $G(A)$. This result is stated in the following and Fig. 1 illustrates this.

Theorem 2.1. *For a $\{-1,0,+1\}$-matrix A, we have*

$$\nu(A)=\max_{S \subseteq V}\{\alpha((G(A)^S)^-)\}, \tag{1}$$

where G^S is the graph obtained from G by changing the signs of each edge in the cut $\delta(S)$.

A signed graph based solution approach to the DMERN problem was also described in [9]. It is a heuristic approach that uses a spanning tree of $G(A)$ to construct a vertex set $\overline{S} \in V$ and obtain a lower bound for the value $\nu(A)$, namely, $\alpha((G(A)^{\overline{S}})^-)$. The argument in favor of using a spanning tree in the construction of S is the fact that every signed tree is a network graph [17]. Therefore, this heuristic approach is able to find the optimal solution whenever A is a network matrix.

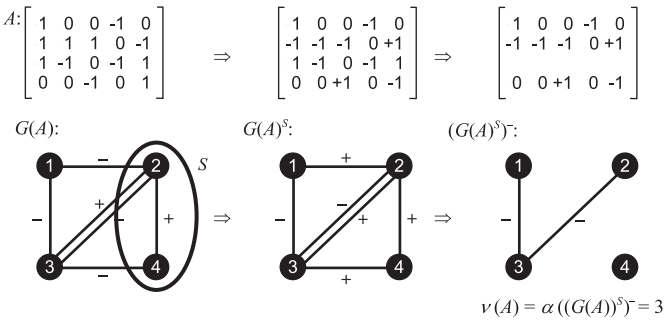

Fig. 1. Existing conflicts in matrix A are represented in signed graph $G(A)$, as well as possible conflicts after row reflections. By applying the Theorem 2.1 we obtain the value of $\nu(A)$. The matrix obtained after reflecting rows 1 and 3 is represented by signed graph $G(A)^S$. The maximum stable set in graph $(G(A)^S)^-$ gives us the set of rows defining the maximum embedded reflected network in A.

Let $\overline{S} \subseteq V$ be an argmax of (1) and let $\overline{I} \subseteq V$ be a stable set such that $\nu(A)=\alpha((G(A)^{\overline{S}})^-)=|\overline{I}|$. The subgraph of $G(A)$ induced by $\overline{I}$ is a maximum network subgraph according to the number of vertices. As a result, the DMERN problem can also be formulated as the problem of finding a maximum network subgraph of $G(A)$.

An equivalent definition of a network graph was also given by Gulpinar et al. in [9]. This is stated in the next theorem.

Theorem 2.2. *Let G be a connected signed graph with no parallel edges. G is a network graph if and only if G is a balanced graph.*

According to the definition of $G(A)$, matrices of different dimensions can yield the same signed graph. Furthermore, huge matrices can define very sparse signed graphs. Thus, signed graphs are the appropriate structure to compare the size of different DMERN problem instances.

In the next section we introduce an integer programming formulation to the DMERN problem in which we explore the relations between network matrices and signed graphs.

3. An integer programming formulation based on signed graphs

Consider a signed graph $G=(V,E,s)$. Our formulation looks for a maximum network subgraph in G which, according to Theorem 2.2, is equivalent to looking for a maximum balanced subgraph in G. It is well known that a signed graph is balanced if and only if it does not contain a parallel edge or a cycle with an odd number of negative edges [4,9,17]. Let $C^o(E)$ be the set of all cycles in G with no parallel edges and with an odd number of negative edges. From now on, a cycle $C \in C^o(E)$ is called an *odd negative cycle*. In this formulation, we use binary decision variables $y \in \{0,1\}^{|E|}$ defined in the following way. For all $i \in V$,

$$y_i=\begin{cases}1 & \text{if vertex } i \in V \text{ belongs to the network subgraph,}\\ 0 & \text{otherwise.}\end{cases}$$

Throughout this text, we use the vector notation $y=(y_i)$, $i \in V$, and the notation $y(V')=\sum_{i \in V'} y_i$ for $V' \subseteq V$. The formulation follows:

$$\begin{aligned}&\text{maximize} && y(V)\\ &\text{subject to} && y_i+y_j \leq 1, \quad \forall\,(i,j) \in E^- \cap E^+, \end{aligned} \tag{2}$$

$$y(C) \leq |C|-1, \quad \forall\, C \in C^o(E), \tag{3}$$

$$y_i \in \{0,1\}, \quad \forall\, i \in V. \tag{4}$$

Let us refer to this formulation as $Y(G,s)$. Consider a parallel edge $(i,j) \in E^- \cap E^+$. Constraints (2) ensure that vertices i and j cannot

belong together to the network subgraph. Constraints (3) forbid cycles with an odd number of negative edges in the subgraph described by variables y. As a consequence, these constraints force variables y to define a network subgraph. Finally, the objective function looks for a maximum network subgraph.

Formulation $Y(G,s)$ has n variables and might have an exponential number of constraints in the worst case. As we have mentioned before, our aim is to solve the DMERN problem to optimality by using a branch-and-cut algorithm, a solution technique based on a linear relaxation of the problem we intend to solve. By dropping the integrality constraints in formulation $Y(G,s)$, we obtain a linear relaxation to the DMERN problem.

4. On the network matrix polytope

The network matrix polytope $P_{G,s}$ associated with formulation $Y(G,s)$ is defined here as

$$P_{G,s} = \text{conv}\{y \in \Re^n | y \text{ satisfies } (2)-(4)\}.$$

Now, we turn our attention to the structure of $P_{G,s}$ in order to describe classes of valid and facet-defining inequalities to be used later as cutting planes in an exact solution approach to the DMERN problem. A partial description of this polytope is presented in [4], including the introduction of some classes of facet-defining inequalities.

Additional notations will be necessary before we can proceed. Given a vertex set $I \subseteq V$, the incidence vector $y^I \in \Re^n$ of I is defined as: $y^I_i = 1$, if $i \in I$, and $y^I_i = 0$, if $i \notin I$. Also, for $i \in V$, let $I_i = \{i\}$.

The first results show that $P_{G,s}$ is full-dimensional and establish which trivial inequalities define facets of $P_{G,s}$, [4].

Theorem 4.1. *The polytope $P_{G,s}$ is full-dimensional, i.e.,* $\dim(P_{G,s}) = n$.

Proof. Since $P_{G,s}$ contains the null vector, it is sufficient to present other n linearly independent solutions $y \in \Re^n$ in $P_{G,s}$. The n linear independent solutions y^{I_i}, $i \in V$, clearly belong to $P_{G,s}$. □

Remark 4.2. The non-negativity of the coefficients in constraints (2) and (3) implies that $a \geq 0$ and $\alpha \geq 0$ for each facet-defining inequality $a^T y \leq \alpha$ that is not a trivial constraint.

Theorem 4.3 (*Trivial inequalities*).

(a) *For all $i \in V$, $y_i \geq 0$ defines a facet of $P_{G,s}$;*
(b) *for all $i \in V$, $y_i \leq 1$ defines a facet of $P_{G,s}$ if and only if* $\delta(i) \cap E^- \cap E^+ = \emptyset$;

Proof.

(a) Consider the $n+1$ affinely independent vectors defined in the proof of Theorem 4.1. The result follows from the fact that only vector y^{I_i} does not belong to the face defined by the inequality $y_i \geq 0$.
(b) Suppose there is a vertex j such that $(i,j) \in E^- \cap E^+$. In this case, the inequality is not facet-defining since constraint (2) dominates the inequality $y_i \leq 1$. Now suppose this is not the case. For a vertex $j \in V \setminus \{i\}$, define $I_{i,j} = \{i,j\}$. The $n-1$ solutions $y^{I_{i,j}}$, $j \in V \setminus \{i\}$ together with the solution y^{I_i} guarantee the result. □

Constraints (2) and (4) in formulation $Y(G,s)$ define the well-known edge formulation of the stable set problem [10] for the subgraph $G[E^- \cap E^+]$. Let $P_{E^- \cap E^+}$ be the polytope associated with this formulation. The next theorem demonstrates that we can use some facets of $P_{E^- \cap E^+}$ in the description of $P_{G,s}$. Consider an inequality $a^T y \leq \alpha$ valid for the polytope $P_{G,s}$. We define the support graph of such inequality to be the graph $G^a = (V^a, E[V^a])$, where $V^a = \{i \in V | a_i \neq 0\}$.

Theorem 4.4. *Let $G = (V,E,s)$ be a signed graph with $E = E^- \cup E^+$ and let $a^T y \leq \alpha$ be a facet-defining inequality for the polytope $P_{E^- \cap E^+}$. The inequality $a^T y \leq \alpha$ is valid for $P_{G,s}$. If, for all $(i,j) \in E[V^a]$, $(i,j) \in E^- \cap E^+$, then $a^T y \leq \alpha$ defines a facet of $P_{G,s}$.*

Proof. The inequality $a^T y \leq \alpha$ is clearly valid for $P_{G,s}$ since $P_{G,s} \subseteq P_{E^- \cap E^+}$. Let $n' = |V[E^- \cap E^+]|$ and let $B_1 \in \Re^{n' \times n'}$ be a matrix composed of n' affinely independent solutions belonging to the facet of $P_{E^- \cap E^+}$ defined by inequality $a^T y \leq \alpha$. We can assume w.l.o.g. that the vertex set V is ordered in such a way that $i \in V[E^- \cap E^+]$ if and only if $1 \leq i \leq n'$. We denote by $b^1, b^2, \ldots, b^{n'}$ the n' rows of matrix B_1. From the properties of the stable set polytope we can assume that B_1 satisfies the following conditions:

(i) row b^1 has entry $b^1_j = 0$ for each vertex $j \notin V^a$;
(ii) for each $k = 2, \ldots, n'$, row b^k satisfies the condition $\sum_{j \notin V^a} b^k_j \leq 1$. When this condition is satisfied with equality, let $v_k \in V[E^- \cap E^+] \setminus V^a$ be the vertex such that $b^k_{v_k} = 1$.

Define a matrix $B_2 \in \Re^{(n-n') \times n'}$ such that all rows of B_2 are equal to b^1. We now argue that the row vectors of matrix,

$$M = \begin{bmatrix} B_1 & 0 \\ B_2 & I \end{bmatrix} \in \Re^{n \times n}$$

are affinely independent and belong to the face of $P_{G,s}$ defined by $a^T y \leq \alpha$. By the definition of matrix M, it is clear that its row vectors are affinely independent. Moreover, remembering the definition of matrices B_1 and B_2, we can conclude that these n vectors satisfy inequality $a^T y \leq \alpha$ with equality. We are left to prove that all of them belong to the polytope $P_{G,s}$, which means that they satisfy constraints (2) and (3). Since constraints (2) belong to the definition of polytope $P_{E^- \cap E^+}$, we can conclude that these constraints are satisfied by the row vectors of matrix M. It now remains to be shown that they also satisfy constraints (3). Let y^k, $k \in \{1, \ldots, n\}$, denote the row vectors of matrix M. First, consider a row vector y^k with $k \in \{1, \ldots, n'\}$, that means $y^k = (b^k, 0)$. According to conditions (i) and (ii), a cycle $\overline{C}$ ($|\overline{C}| \geq 3$) defining a constraint (3) violated by y^k must be composed of at least two vertices $i,j \in V^a$ such that $(i,j) \in E[V^a]$. But this cannot happen since $(i,j) \in E^- \cap E^+$, for all $(i,j) \in E[V^a]$. Now, consider a row vector y^k with $k = n'+1, \ldots, n$. According to the definition of matrix M and since condition (i) is satisfied, we have $y^k = y^I$, where $I = I' \cup \{j\}$ for a $j \in V \setminus V[E^- \cap E^+]$, and I' is a stable set in $G[V^a]$. Again, a cycle $\overline{C}$ ($|\overline{C}| \geq 3$) defining a constraint (3) violated by y^k must be composed of at least two vertices in V^a linked by a negative edge and, for the same reason, this is not possible. □

Clique inequalities [10,11,13] play an important role in the description of the stable set polytope. According to the previous theorem, clique inequalities are always facet defining to $P_{G,s}$.

Corollary 4.5 (*Clique inequality*). *Consider a subset $K \subseteq V$ such that K is a maximal clique in the graph $G[E^- \cap E^+]$. The clique inequality*

$$y(K) \leq 1 \tag{5}$$

defines a facet of $P_{G,s}$.

This result was obtained in [4] by a direct proof. Lifted hole inequalities [11] have also been shown to be very effective in some cutting plane approaches proposed in the literature for the stable set problem [10,13].

Corollary 4.6 (*Lifted hole inequality*). *Consider a subset $H \subseteq V$ such that H is an odd hole in the graph $G[E^- \cap E^+]$ of size $|H| = 2k+1$. The lifted odd hole inequality*

$$y(H) + \sum_{j \in V \setminus H} \beta_j y_j \leq k, \tag{6}$$

is valid for the polytope $P_{G,s}$ where the coefficients β_j are as large as possible while the validity of the inequality is preserved. If the support graph of this inequality satisfies the condition in Theorem 4.4, then the lifted odd hole inequality defines a facet of $P_{G,s}$.

In the literature we can find other classes of facet-defining inequalities for the stable set polytope [16] which can also be used to tighten the network matrix polytope.

Next, we investigate particular substructures of G that give rise to classes of valid and facet-defining inequalities for the polytope $P_{G,s}$. In formulation $Y(G,s)$, constraints (3) are defined over the set of cycles with an odd number of negative edges. The next result establishes the conditions for these valid inequalities to provide facets of $P_{G,s}$.

Theorem 4.7 (*Odd negative cycle inequality*). *Let $C \in C^o(E)$. The inequality*

$$y(C) \leq |C| - 1 \tag{7}$$

defines a facet of $P_{G,s}$ if and only if the following conditions are satisfied:

(i) *C is chordless.*
(ii) *For each vertex $i \in V \setminus C$, $|N^-(i) \cap N^+(i) \cap C| \leq 1$.*
(iii) *For each vertex $i \in V \setminus C$, $G[E[C \cup \{i\}]]$ contains at most two odd negative cycles; if $|N^-(i) \cap N^+(i) \cap C| = 1$, the vertex j such that $N^-(i) \cap N^+(i) \cap C = \{j\}$ is contained in each odd negative cycle in $G[E[C \cup \{i\}]]$.*

Proof. First, we show that there exists at least n linearly independent zero-one solutions satisfying this inequality with equality. For each $i \in C$, define the vertex set $I_i = C \setminus \{i\}$. These $|C|$ incidence vectors satisfy the odd negative cycle inequality with equality and, since C has no chord (condition (i)), they belong to $P_{G,s}$. Now consider a vertex $i \notin C$. Conditions (ii) and (iii) guarantee that there exists a vertex $j \in C$ such that $I_i = C \setminus \{j\} \cup \{i\}$ defines a solution $y^{I_i} \in P_{G,s}$ satisfying the odd negative cycle inequality with equality (see Fig. 2(a)). The n incidence vectors constructed here are affinely independent. Next, we show that conditions (i)–(iii) are necessary conditions. If cycle $C = \{v_1, v_2, \ldots, v_{|C|}\}$ has a chord (v_k, v_l), $1 \leq k < l \leq |C|$, then, C is a composition of two other cycles $C' = \{v_k, \ldots, v_l\}$, $C'' = \{v_l, \ldots, v_{|C|}, v_1, \ldots, v_k\}$, where exactly one of them belongs to $C^o(E)$, w.l.o.g., let $C'' \in C^o(E)$. In this case, the odd negative inequality written for C is obtained by adding the same inequality written for C'' with inequalities $y_i \leq 1$ written for each vertex $i \in C \setminus C''$ (see Fig. 2(b)). Now suppose that condition (ii) is not satisfied for a vertex $i \in V \setminus C$, i.e., there are at least two vertices $j,k \in N^+(i) \cap N^-(i) \cap C$ (see Fig. 2(c)). In this case, each solution satisfying the odd negative cycle inequality written for C with equality also satisfies $y_i \leq 0$ with equality. Finally, assume that condition (iii) is not satisfied. There are two cases: either $G[E[C \cup \{i\}]]$ contains at least three odd negative cycles (see Fig. 2(d)) or there is a vertex $j \in C$ such that $N^-(i) \cap N^+(i) \cap C = \{j\}$ and j is not contained in each odd negative cycle in $G[E[C \cup \{i\}]]$. Again, in these two cases, each solution satisfying the odd negative cycle inequality written for C with equality also satisfies $y_i = 0$. □

In [4], this result was partially proved since condition (ii) and part of condition (iii) are missing. In fact, the authors established the conditions for an odd negative cycle inequality to be facet defining if $G(A)$ has no parallel edges.

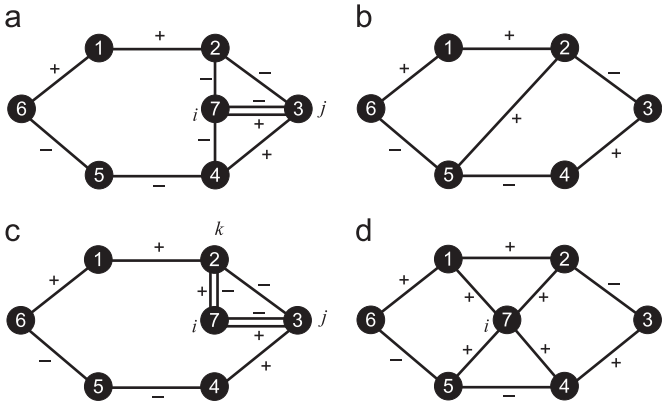

Fig. 2. Consider $C = \{1,2,3,4,5,6\}$. (a) The odd negative cycle C induces a facet-defining inequality of $P_{G,s}$. For vertex $i = 7$, the choice of $j = 3$ makes condition (iii) satisfied. (b) The odd negative cycle C is a composition of cycles $C' = \{2,3,4,5\}$ and $C'' = \{5,6,1,2\}$, where $C'' \in C^o(E)$. The odd negative cycle inequality written to C'' defines a facet of $P_{G,s}$. (c) Since $(2,7),(3,7) \in E^- \cap E^+$, $y(C) = 5$ implies $y_7 = 0$. (d) The graph induced by $C \cup \{7\}$ contains more than two odd negative cycles; $y(C) = 5$ implies $y_7 = 0$.

As it happens for the odd hole inequalities in the definition of the stable set polytope, the conditions for an odd negative cycle inequality to be facet-defining are related to the cycle neighborhood. Thus we can use the same lifting procedure applied to the odd hole inequalities [11] to try to strengthen an odd negative cycle inequality.

The connection between the DMERN problem and the stable set problem starts with Theorem 2.1. This suggests the existence of more similarities between the polytope $P_{G,s}$ and the stable set polytope than those described in Theorem 4.4. Thus we decide to investigate new classes of valid inequalities described over a clique, which is a substructure that plays an important role in the description of the stable set polytope.

In a signed graph a clique with k vertices can occur in different forms depending on the sign of each edge in the clique. In the following results, we present new valid inequalities for $P_{G,s}$ which are related to some different clique structures in $G = (V,E,s)$.

In [4], it is shown that a negative clique induces a facet of $P(G,s)$ if G is a negative graph, i.e., G has no positive edge. Next, we generalize this result for every signed graph.

Theorem 4.8 (*Negative clique inequality*). *Let $K \subseteq V$ be a negative clique in G with $|K| \geq 3$. The inequality*

$$y(K) \leq 2 \tag{8}$$

is valid for $P_{G,s}$. Assume that, for all $i,j \in K$, $(i,j) \notin E^+$, (i.e., there is no parallel edge in clique K). In this case, the inequality is facet-defining if and only if the following conditions are satisfied:

(i) *K is a maximal negative clique in $G[E^-]$.*
(ii) *For each vertex $i \in V \setminus K$, $|N^+(i) \cap K| \leq |K| - 1$.*
(iii) *For each vertex $i \in V \setminus K$, $|N^-(i) \cap N^+(i) \cap K| \leq |K| - 2$.*

Proof. We use induction on the length of clique K for the validity proof. If $|K| = 3$, then K is an odd negative cycle and we can conclude that the inequality is valid. Assume that the inequality is valid for all negative cliques with a given size $p \geq 3$. Now let K be a negative clique with size equal to $p+1$. Consider any three vertices $k,l,m \in K$. The following inequalities are valid:

$$y(K \setminus \{i\}) \leq 2, \quad i = k,l,m,$$

$$y_k + y_l + y_m \leq 2.$$

Adding up these four inequalities and dividing the result by three we obtain

$$y(K) \leq 8/3,$$

and the validity of the negative clique inequality follows from rounding down the right-hand side of the above inequality. In order to prove that this inequality is facet-defining under the previously mentioned conditions, let $F = \{y \in P_{G,s} | y(K) = 2\}$ be the face of $P_{G,s}$ defined by the negative clique inequality written for clique K. We assume that there is an inequality $a^T y \leq \alpha$ valid for $P_{G,s}$ such that $F \subseteq F_a = \{y \in P_{G,s} | a^T y = \alpha\}$ and show that the inequality defining F_a can be written as a positive scalar multiple of the negative clique inequality defining F. Observe that F is a proper face of $P_{G,s}$ since $0 \in P_{G,s} \backslash F$. Consider a vertex $i \in K$. Now, consider $j,k \in K\backslash\{i\}$, $j \neq k$, and define the vertex sets $I^1 = \{i,j\}$ and $I^2 = \{i,k\}$. It is easy to check that the incidence vectors y^{I^1} and y^{I^2} satisfy all the inequalities in formulation $Y(G,s)$, so that these solutions belong to $P_{G,s}$. From solutions y^{I^1} and y^{I^2} in $a^T y = \alpha$ we can conclude that $a_j = a_k$. Repeating this argument for every $k \in K\backslash\{i,j\}$ and given that vertex $i \in K$ is chosen arbitrarily we conclude that $a_i = \gamma$, for all $i \in K$. Now consider a vertex $i \in V\backslash K$. Since we assume conditions (i)–(iii) are satisfied, there exist two vertices $j,k \in K$ such that $j \notin N^-(i)$ and $k \notin N^+(i)$. Define the vertex sets $I^1 = \{j,k\}$ and $I^2 = \{j,k,i\}$. It is not difficult to check that the incidence vectors y^{I^1} and y^{I^2} satisfy all the inequalities in formulation $Y(G,s)$ and, as a consequence, these vectors belong to the network matrix polytope $P_{G,s}$ (see Fig. 3(a)). These solutions lead to $a_i = 0$, for all $i \in V\backslash K$. Next, we argue that (i)–(iii) are necessary conditions. First, assume that the negative clique K is not maximal in $G[E^-]$, i.e., there is a vertex $i \in V\backslash K$ such that $K \cup \{i\}$ is a negative clique. In this case the negative clique inequality written for K is not facet-defining since this inequality is dominated by the same inequality written for $K \cup \{i\}$. Now, assume that there is a vertex $i \notin K$ such that $|N^+(i) \cap K| = |K|$. This implies that, for all $j,k \in K$, there is a cycle $\{i,j,k\} \in C^o(E)$ (see Fig. 3(b)). We observe that each solution $\overline{y}$ satisfying the negative clique inequality with equality has $\overline{y}_j = \overline{y}_k = 1$ for some pair of vertices $j,k \in K$ and, as a consequence, also satisfies $y_i \leq 0$ with equality. Finally, assume that condition (iii) is not satisfied for a vertex $i \in V\backslash K$. Constraints (2) written for each parallel edge (i,j), $j \in N^-(i) \cap N^+(i) \cap K$, imply that each solution satisfying the negative clique inequality with equality also satisfies $y_i \leq 0$ with equality. □

The next result enumerates some cases where the negative clique inequality does not define a facet because of the existence of parallel edges in $E[K]$.

Remark 4.9. Let K be a negative clique. The negative clique inequality written for K does not define a facet of $P_{G,s}$ if,

(i) there exists a vertex $i \in K$ such that $\delta(i) \backslash E[K] \subseteq E^+$;
(ii) there exists a partition K^1, K^2 of K such that $E[K^1] \cup E[K^2] \subseteq E^+$;
(iii) there exist vertex sets $K^1 = K\backslash\{i\}$, $K^2 = K\backslash\{j\}$, for some $i,j \in K$ such that $E[K^1] \cup E[K^2] \subseteq E^+$.

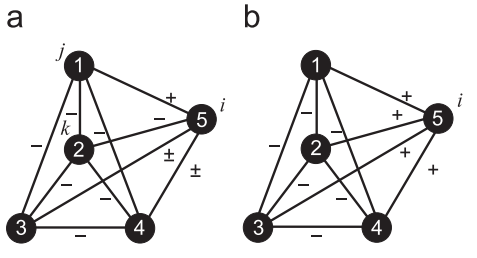

Fig. 3. Consider the negative clique $K = \{1,2,3,4\}$. (a) Conditions in Theorem 4.8 guarantee the existence of vertices j,k such that $\{i,j,k\}$ does not define an odd negative cycle. (b) For any choice of vertices $j,k \in K$ there is an odd negative cycle $C = \{i,j,k\}$.

Proof. In case (i) all the solutions that satisfy the negative clique inequality with equality also satisfy $y_i = 0$. In case (ii) we obtain the negative clique inequality by adding clique inequalities written for K^1 and K^2. Finally, in case (iii) the negative clique inequality is a linear combination of clique inequalities written for K^1 and K^2, and trivial inequalities $y_i \leq 1$ and $y_j \leq 1$. □

Notice that in cases (ii) and (iii) of Remark 4.9, clique inequalities dominate the negative clique inequality. In Section 5, we will see that this fact is taken into consideration in the definition of the separation strategy for the branch-and-cut algorithm.

The next family of inequalities can be seen as a generalization of the clique inequalities described in Theorem 4.8.

Theorem 4.10 (*Partitionable clique inequality*). *Let $K \subseteq V$ be a partitionable clique in G, i.e., a clique with $|K| \geq 3$ and such that there exists a partition $K^1, K^2 \neq \emptyset$ of K satisfying*

(i) $E[K^1] \cup E[K^2] \subseteq E^-$,
(ii) $\delta(K^1, K^2) = \{(i,j) \in E | i \in K^1, j \in K^2\} \subseteq E^+$.

The inequality

$$y(K) \leq 2 \tag{9}$$

is valid for $P_{G,s}$. Assume that, $E[K] \cap E^- \cap E^+ = \emptyset$. The inequality is facet-defining if and only if, K^1 and K^2 are both maximal negative cliques in G.

Proof. The validity proof follows exactly the lines of the validity proof in Theorem 4.8. Notice that, $K\backslash\{i\}$ can be now a negative clique, in which case, we use the validity of inequality (8) for the induction step. The methodology to be used in the facet-defining proof is also the same as in the proof of Theorem 4.8. Using the set of solutions defined in the proof of this theorem, we can conclude that $a_i = \gamma$, for all $i \in K$. Now consider a vertex $i \notin K$. Since K^1 and K^2 are both maximal negative cliques in G, there exist two vertices $j \in K^1$ and $k \in K^2$ such that $(i,j),(i,k) \notin E^-$. Define the vertex sets $I^1 = \{i,j,k\}$ and $I^2 = \{j,k\}$. Since clique K has no parallel edges and set I^1 does not define a negative cycle in G, we have $y^{I^1}, y^{I^2} \in P_{G,s}$. These solutions lead to $a_i = 0$, for all $i \in V\backslash K$. Finally, if either K^1 or K^2 is not a maximal clique, each solution satisfying the partitionable clique inequality with equality also satisfies $y_i \leq 0$ with equality, for some vertex $i \in V\backslash K$. □

The negative clique inequality is a particular case of the partitionable clique inequality where $K^1 = \emptyset$. Another family of clique inequalities is introduced in the following theorem.

Theorem 4.11 (*Cycle negative odd clique inequality*). *Let $K \subseteq V$ be a cycle negative odd clique in G, i.e., a clique such that $|K|$ is odd and there exists a cycle C such that $C = K$, $E^C \subseteq E^-$ and $E[K] \backslash E^C \subseteq E^+$. The inequality*

$$y(K) \leq \max\{\lfloor |K|/2 \rfloor, 3\}$$

is valid for $P_{G,s}$.

Proof. We may assume w.l.o.g. that $K = \{v_0, v_1, \ldots, v_{|K|-1}\}$ and that, for all $i,j \in K$, if $(i,j) = (v_t, v_{(t+1) \bmod |K|})$, for some $t \in \{0, \ldots, |K|-1\}$, then $(i,j) \in E^C$, otherwise $(i,j) \in E[K] \backslash E^C$. First, we prove the validity for the case where $|K| = 5$. Consider the odd negative cycle inequalities written for the cycles $\{v_t, v_{(t+1) \bmod 5}, v_{(t+3) \bmod 5}\}$, $t \in \{0, \ldots, 4\}$ (see Fig. 4(a)). Adding up these five inequalities and dividing the result by three we obtain

$$y(K) \leq 10/3,$$

and the validity follows from rounding down the right-hand side of this last inequality. Now consider the case where $|K| \geq 7$. Adding the partitionable clique inequalities written for the cliques $\{v_t, v_{(t+1) \bmod |K|}, v_{(t+3) \bmod |K|}, v_{(t+4) \bmod |K|}\}$, $t \in \{0, \ldots, |K|-1\}$ (see Fig. 4(b)) and dividing the result by four we obtain the inequality

$$y(K) \leq |K|/2.$$

Again, the validity of the cycle negative odd clique inequality follows from rounding down the right-hand side of the inequality obtained. □

The next result shows that the change of signs in the clique structure defined in Theorem 4.11 gives rise to another class of valid inequalities for $P_{G,s}$.

Theorem 4.12 (*Cycle positive odd clique inequality*). *Let $K \subseteq V$ be a cycle positive odd clique in G, i.e., a clique such that $|K|$ is odd and there exists a cycle C such that $C=K$, $E^C \subseteq E^+$ and $E[K] \backslash E^C \subseteq E^-$. If $|K| = 5$ the inequality*

$$y(K) \leq 3$$

is valid for $P_{G,s}$. For the cases where $|K| \geq 7$ the inequality

$$y(K) \leq 4$$

is valid for $P_{G,s}$.

Proof. We make the same assumptions as in the proof of Theorem 4.11. When $|K| = 5$ the inequality is clearly valid since K also satisfies the conditions in Theorem 4.11 (see Fig. 5(a)). Now consider the case where $|K| \geq 7$. Adding the negative clique inequalities written for the cliques $\{v_t, v_{(t+2) \bmod |K|}, v_{(t+4) \bmod |K|}, \ldots, v_{(t+|K|-3) \bmod |K|}\}$, $t \in \{0, \ldots, |K|-1\}$ (see Fig. 5(b)) and dividing the result by $(|K|-1)/2$ we obtain the inequality

$$y(K) \leq 4|K|/(|K|-1).$$

Finally, rounding down the right-hand side of this inequality finishes the validity proof. □

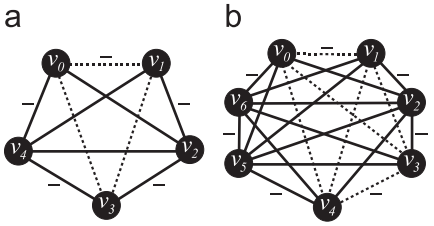

Fig. 4. Cycle negative odd cliques with $|K| = 5$ and 7. Signs of positive edges are omitted. Dotted edges define (a) an odd negative cycle when $t=0$ and (b) a partitionable clique when $t=0$; both of them used in the proof of Theorem 4.11.

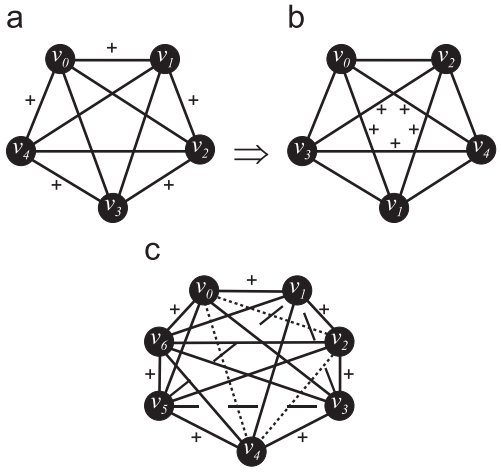

Fig. 5. Cycle positive odd cliques with $|K| = 5$ and 7. Signs of negative edges are omitted. (a) A cycle positive odd clique inequality is equivalent to a cycle negative odd clique inequality if $|K| = 5$. (b) Dotted and dashed edges define, respectively, negative cliques used in the proof of Theorem 4.12 when $t=0$ and 1.

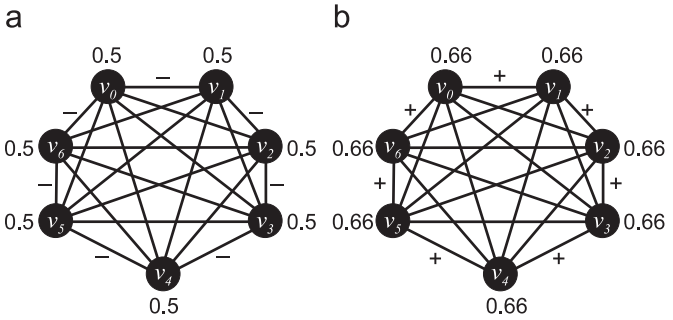

Fig. 6. Fractional solutions to the LP relaxation of formulation $Y(G,s)$ cutted off by cycle (a) negative and (b) positive clique inequalities.

As depicted in Fig. 6, cycle negative (positive) clique inequalities are able to cut off fractional solutions to the LP relaxation of formulation $Y(G,s)$ that cannot be cut by the inequalities introduced so far.

5. A branch-and-cut algorithm

In this section, we describe a simple branch-and-cut algorithm for the DMERN problem. The algorithm has three basic components: the initial formulation, the cut generation and the primal heuristic. We outline these further in the section. Before that, we give some information about the strategies adopted in the branch-and-cut implementation.

After an LP has been solved in the branch-and-cut tree, we check if the solution is integer feasible. If this is not the case, the primal heuristic is executed. The cut generation procedure is then called and all the separation routines described below are executed (a limit of 100 cuts per iteration is set). If no violated inequality is found or if a limit of 10 cut generations rounds is reached, we enter the branching phase of the algorithm. A standard 0–1 branching rule is applied. The same branching priority is assigned to each variable and the branch-and-cut tree is investigated with the best-bound-first strategy.

These strategies were defined after a number of computational experiments has been performed. Concerning the branching strategy, we have also implemented a version of the branching rule proposed in [2]. Although it has been successfully applied to solve the stable set problem [12–14], we obtained better results with the standard 0–1 branching rule.

5.1. Initial formulation

The initial formulation is composed of a set of clique inequalities (5), odd negative cycle inequalities (7), partitionable clique inequalities (9) (remember that negative clique inequalities (8) are a special case thereof) and by all the trivial inequalities.

Using the greedy procedure CLQ1 described in [10], we generate a set of cliques L in graph $G[E^- \cap E^+]$ such that each edge $(i,j) \in E^- \cap E^+$ is contained in at least one clique.

Likewise, we generate a set of odd negative cycles M such that each vertex $i \in V$ is contained in at least one odd negative cycle. For each vertex $i \in V$, we find the minimum odd negative cycle C passing through i with respect to the total number of edges

(the algorithm that finds a minimum odd negative cycle is described in Subsection 5.2). Cycle C is included in M.

We also generate a set of partitionable cliques N. For each negative edge $(i,j) \in E^- \backslash E^+$ that is not contained in any partitionable clique generated so far, let us define $K^1 = \{i,j\}$. Applying the greedy strategy used in CLQ1, we expand K^1 to a maximal negative clique. Define $S = \bigcap_{i \in K^1} N^+(i)$. Once again, we apply the CLQ1 greedy strategy to try to generate a maximal negative clique $K^2 \subseteq S$. The partitionable clique $K^1 \cup K^2$ is included in N.

The initial formulation is then defined as

$$\begin{aligned} \text{maximize} \quad & y(V) \\ \text{subject to} \quad & y(K) \leq 1, \quad \forall\, K \in L, \\ & y(C) \leq |C| - 1, \quad \forall\, C \in M, \\ & y(K) \leq 2, \quad \forall\, K \in N, \\ & 0 \leq y_i \leq 1, \quad \forall\, i \in V. \end{aligned}$$

5.2. Cut generation

The cut generation phase contains two separation routines: an exact separation routine for the odd negative cycle inequalities (7) and a heuristic separation routine for inequalities (5) and (9) defined over cliques. In the following, let $\bar{y} \in \mathbb{R}^n$ be a fractional solution.

Odd negative cycle inequalities: This separation routine is based on a polynomial algorithm described in [3] to solve the separation problem for cut inequalities. We define a weight $p_i = 1 - \bar{y}_i$ for each vertex $i \in V$. Clearly, an odd negative cycle inequality is violated for a cycle $C \in C^o(E)$ if and only if $p(C) < 1$. Thus, the separation problem is solved by looking for a minimum weighted odd negative cycle with respect to the vertex weights $p \in \mathbb{R}^n$. From the given signed graph G we define a new signed graph $L(G)$ with vertices i and i', for every vertex $i \in V$. The sets of positive and negative edges of the new graph are defined as follows: $L(E)^+ = \{(i,j),(i',j') | (i,j) \in E^+ \backslash E^-\}$ and $L(E)^- = \{(i',j),(i,j') | (i,j) \in E^- \backslash E^+\}$. We define the new graph with no parallel edge since odd negative cycles have no parallel edges. Now, for each vertex $i \in V$ we find a shortest path from i to i'. The minimum over all shortest paths we have found is the weight of the minimum weighted odd negative cycle in G. It is easy to check that this procedure takes a polynomial number of iterations to end.

Clique inequalities: This separation routine tries to find violated clique inequalities (5) and violated partitionable clique inequalities (9). A greedy strategy is used, as is usually done for the separation of clique inequalities for the stable set problem [10,13]. Let us define a weight $p_i = \bar{y}_i$ for each vertex $i \in V$. First of all, we try to discover violated clique inequalities by using the greedy procedure CLQ2 proposed in [10]. To avoid the generation of redundant inequalities (see Remark 4.9), we eliminate from the weighted graph any vertex contained in at least one clique that has been found. Then, we try to find a set of violated partitionable clique inequalities in the reduced graph by adapting the CLQ2 procedure. Starting with a vertex of maximum weight in the reduced graph, we try to expand a maximal negative clique K^1. If such a clique is found, we apply the same steps to try to expand a maximal negative clique $K^2 \subseteq \cap_{i \in K^1} N^+(i)$. Finally, we check if $K^1 \cup K^2$ define a violated partitionable clique inequality. We repeat this process recursively until the reduced graph is empty.

Table 1
Results on DMERN instances by using only inequalities in formulation $Y(G,s)$.

Instance				Solution		%Gap		Cuts	B&C Tree	
Name	Cols	n	$\|E\|$	%Gap	Time (s)	InitForm	Root	#C	#Nod	NodB
agg2	223	141	348	39.77	–	91.93	55.90	934014	818717	78
agg3	223	141	348	40.32	–	91.93	55.90	1096324	827264	10
modszk1	625	148	184	0.00	0	6.15	1.02	78	11	5
perold	613	150	97	0.00	0	0.00	0.00	0	1	1
fffff800	780	157	712	0.00	331	16.8	6.0	51677	8719	7244
nesm	1635	190	90	0.00	0	0.00	0.00	0	1	1
pilot.we	1046	202	84	0.00	0	0.00	0.00	0	1	1
pilot.ja	766	205	101	0.00	0	0.25	0.25	0	2	2
pilotnov	839	209	102	0.00	0	0.25	0.25	0	2	2
25fv47	911	224	335	0.00	0	0.94	0.25	50	5	3
bnl1	690	275	220	0.00	0	0.38	0.00	6	1	1
pilot	903	275	268	0.00	75	4.96	3.43	29160	10915	1184
scfxm2	784	282	412	0.00	0	0.79	0.39	10	5	3
woodw	3545	301	24	0.00	0	0.00	0.00	0	1	1
maros	1281	305	366	0.00	1	0.33	0.22	6	5	3
pilot87	1090	339	329	0.00	20	3.10	2.12	4562	1445	207
seba	825	402	27885	138.30	–	184.32	173.52	195154	709	29
ship12s	2187	456	3960	0.00	0	20.00	0.00	548	1	1
sctap2	1410	470	0	0.00	0	0.00	0.00	0	1	1
shell	1476	483	1391	0.00	4	0.00	0.00	0	1	1
cycle	2350	505	663	0.00	0	0.00	0.00	0	1	1
gfrd-pnc	1066	590	809	0.00	4	0.00	0.00	0	1	1
ganges	1481	631	783	0.00	0	0.00	0.00	0	1	1
czprob	3102	719	2900	0.00	13	0.00	0.00	0	1	1
ship12l	5223	828	10032	0.00	3	9.83	0.00	719	1	1
d2q06c	3177	844	1016	0.00	3	0.37	0.16	50	21	14
greenbea	4549	915	1722	0.00	19	0.78	0.22	48	39	39
greenbeb	4542	915	1721	0.00	20	0.78	0.22	48	39	39
stocfor2	1690	1262	1690	0.00	2	1.17	0.00	111	1	1
80bau3b	2348	1374	2348	0.00	106	0.37	0.18	102	61	6
bnl2	1474	1418	1474	0.00	8	0.14	0.08	122	19	10
degen3	1818	1503	50178	59.60	–	88.73	59.91	4404	40	1
dfl001	11536	6019	32364	44.04	–	47.75	44.38	4200	3	3
stocfor3	13190	9632	12480	0.00	97	0.95	0.00	313	1	1

5.3. Primal heuristic

With the information of a fractional LP solution, we construct vertex sets S^1 and S^2 such that $y^{S^1 \cup S^2} \in Y(G,s)$. For this purpose, it is sufficient to generate these vertex sets in such a way that $E(S^1), E(S^1) \subseteq E^+$ and $\delta(S^1,S^2) \subseteq E^-$ (in that case, there will be no odd negative cycle in $E(S^1 \cup S^2)$). We use a rounding heuristic to define S^1 and S^2. First of all, we sort the vertex set V in a non-increasing order $v_1, v_2, \ldots, v_n$ according to the values of $\overline{y} \in \mathbb{R}^n$. We start with $S^1 = S^2 = \emptyset$. At each iteration $1 \leq t \leq n$, we try to insert vertex v_t in either S_1 or S_2 checking, respectively, if $N^-(i) \cap S^1 = \emptyset$ or $N^-(i) \cap S^2 = \emptyset$. In the event of a tie, we choose the set of minimum cardinality.

6. Computational experiments

In this section, we report some computational experiments carried out with formulation $Y(G,s)$ introduced in Section 3 and with the branch-and-cut algorithm described in Section 5.

The branch-and-cut algorithm is coded in C++ on a Sony Vaio Computer with a processor Intel Core 2 Duo of 2.10 GHz and 3 GB of RAM memory. We use Xpress-Optimizer 20.00.21 [15] to implement the components of the enumerative algorithm. The CPU time limit is set to 1 h.

As we have mentioned before, Gulpinar et al. proposed in [9] a heuristic approach based on signed graphs to the solution of the DMERN problem. They reported the computational performance of their method over a set of 44 instances from Netlib (http://www.netlib.org/lp/data/). Before solving each instance, they applied a preprocessing in order to reduce the size of the coefficient matrix and a scaling procedure in order to increase the dimension of the $\{-1,0,+1\}$-matrix. From the set of instances used in [9], 34 instances can be found in http://www.cs.rhul.ac.uk/~zvero/thesis/sga_ref/all_models.tar.gz already preprocessed and scaled. The size of these instances range from the smallest to the largest among the 44 instances. We define this set of instances as our test set.

Computational experiments over the test set showed that the automatic cut generation of Xpress-MP Optimizer is useless to solve these instances. Thus, this strategy is deactivated in the experiments reported next.

Before evaluating the branch-and-cut algorithm described in Section 5, we investigate how strong formulation $Y(G,s)$ is. For that purpose, we try to solve all the instances in the test set by using a simple version of the branch-and-cut algorithm in which only constraints in formulation $Y(G,s)$ are used: initial formulation is composed by constraints (2) and by odd negative cycle inequalities (7) define over M; only odd negative cycle inequalities (7) are considered in the cut generation phase; other implementation strategies (including primal heuristic) are as described in Section 5. Table 1 displays the results obtained as well as some information about the instances.

The first four columns in this table give us information about the instances: the Netlib instance name, the number of columns in the matrix, the number of rows/vertices and the number of edges. The next two columns, "%Gap" and "Time", give us the final gap (between the best solution found and the upper bound, in percentage) and the time (in seconds) spent to solve the instances to optimality ("–" means the instance was not solved within the

Table 2
Results of the branch-and-cut on DMERN instances.

Instance	Solution		Init form			%Gap		Cuts		B&C Tree	
Name	OptVal	Time(s)	#L	#M	#N	InitForm	Root	#C	#K	#Nod	NodB
agg2	**62**	**0**	**0**	**134**	**23**	**0.00**	**0.00**	**3**	**0**	**1**	**1**
agg3	**62**	**0**	**0**	**134**	**23**	**0.00**	**0.00**	**3**	**0**	**1**	**1**
modszk1	**130**	**0**	**0**	**56**	**12**	**0.00**	**0.00**	**12**	**0**	**1**	**1**
perold	143	0	7	13	0	0.00	0.00	0	0	1	1
fffff800	**125**	**0**	**0**	**146**	**234**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
nesm	190	0	0	0	0	0.00	0.00	0	0	1	1
pilot.we	202	0	0	0	0	0.00	0.00	0	0	1	1
pilot.ja	**198**	**0**	**2**	**17**	**1**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
pilotnov	**203**	**0**	**1**	**17**	**1**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
25fv47	**212**	**0**	**0**	**102**	**11**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
bnl1	**261**	**0**	**0**	**44**	**1**	**0.00**	**0.00**	**6**	**0**	**1**	**1**
pilot	**252**	**0**	**0**	**45**	**13**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
scfxm2	**252**	**0**	**6**	**108**	**2**	**0.79**	**0.39**	**10**	**0**	**5**	**3**
woodw	301	0	0	0	0	0.00	0.00	0	0	1	1
maros	**300**	**0**	**0**	**21**	**1**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
pilot87	**306**	**0**	**0**	**91**	**13**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
seba	**140**	**7**	**0**	**330**	**61**	**0.71**	**0.00**	**13**	**4**	**1**	**1**
ship12s	360	0	0	456	0	20.00	0.00	500	0	1	1
sctap2	470	0	0	0	0	0.00	0.00	0	0	1	1
shell	482	4	0	15	0	0.00	0.00	0	0	1	1
cycle	504	0	0	3	0	0.00	0.00	0	0	1	1
gfrd-pnc	522	4	0	326	0	0.00	0.00	0	0	1	1
ganges	534	1	0	303	0	0.00	0.00	0	0	1	1
czprob	718	13	0	4	0	0.00	0.00	0	0	1	1
ship12l	732	3	0	828	0	9.83	0.00	700	0	1	1
d2q06c	**804**	**1**	**6**	**227**	**15**	**0.00**	**0.00**	**13**	**0**	**2**	**2**
greenbea	**890**	**2**	**2**	**148**	**37**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
greenbeb	**890**	**2**	**2**	**148**	**37**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
stocfor2	1106	2	0	784	0	1.17	0.00	125	0	1	1
80bau3b	**1346**	**26**	**0**	**226**	**2**	**0.29**	**0.11**	**43**	**1**	**7**	**4**
bnl2	**1367**	**1**	**0**	**257**	**11**	**0.00**	**0.00**	**0**	**0**	**1**	**1**
degen3	**796**	**110**	**80**	**1502**	**1469**	**0.00**	**0.00**	**19**	**0**	**1**	**1**
dfl001	**3366**	–	**45**	**5998**	**2387**	**35.37**	**34.76**	**3900**	**1**	**3**	**3**
stocfor3	8516	113	0	5836	0	0.95	0.00	313	0	1	1

time limit). Time spent to load an instance and to construct the signed graph is not included. One observes that matrices with a big number of columns can give rise to sparse signed graphs and that the size of a matrix is not necessarily related to the effort required to solve an instance. For a methodology based on a signed graph representation of the problem, the dimension and sparsity of the associated signed graph are factors that indicate the complexity of solving the problem. Instances are presented in the table in increasing order of number of vertices. By using only inequalities in formulation $Y(G,s)$, we were unable to solve five of the instances including the smallest one. Furthermore, the final percentage gap was very big for the unsolved instances.

The meaning of the other columns in this table is as follows: in "%Gap", "InitForm" and "Root" are, respectively, the gap (between the best solution found and the upper bound, in percentage) obtained for the initial formulation and for the last LP solved in the root of the branch-and-cut tree; "Cuts #C" is the total number of odd negative cycle cuts (7) generated; in "B&C Tree", "#Nod" and "NodB" are, respectively, the total number of nodes in the branch-and-cut tree and the node where the best solution was found. One observes that, for some instances, it is necessary to go deep in the branch-and-cut tree to find the optimal solution, even if the percentage gap in the root is very small.

Next, we run the branch-and-cut algorithm described in Section 5 over the test set. Table 2 reports the results obtained. In the solution of 14 instances, we obtain the same results as in Table 1 since no inequality defined for a clique was generated either in the initial formulation or in the cut generation phase. The other 20 instances are displayed in bold in this table. For these instances, except for "scfxm2", at least one of the following improvements was achieved; the instance was now solved to optimality, the percentage gap was smaller or the size of the branch-and-cut tree was reduced.

In Table 2, Column "OptVal" gives us the value of the optimal solution. Whenever the time limit is reached, the column reports the value of the best integer solution found. Multicolumn "Init Form" gives us information about the number of inequalities in the initial formulation: "#L"—clique inequalities (5); "#M"—odd negative cycle inequalities (7); and "#N"—partitionable clique inequalities (9). In multicolumn "Cuts", "#K" is the total number of partitionable clique cuts (9) generated. All other notations in Table 2 are the same as in Table 1.

The branch-and-cut algorithm is able to solve almost all instances in the test set in just a few seconds. Only instance "dfl001" was not solved within the time limit (the final gap was equal to the gap in the root). Most instances were solved to optimality at the root of the branch-and-cut tree. Moreover, the percentage gap of the first formulation was very small for all instances except for instances "ship12s", "ship12l" and "dfl001". Regarding the cut generation phase, we can observe that the number of computed clique inequalities was quite small. This is explained by the fact that most of these instances have no parallel edges. For that reason, at this stage of our work, we have not used separation routines for lifted hole inequalities (6). It is clear that inequalities defined for clique substructures are very important for the solution of instances in the test set. However, few of these inequalities were necessary except for instances "fffff800", "degen3" and "dfl001". This is why we have not yet implemented separation routines for cycle negative/positive clique inequalities described in Theorems 4.11 and 4.12.

In order to better measure the effectiveness of our exact approach, we also provide numerical results for a set of random instances of the DMERN problem. We generated random signed graphs (instead of random matrices) by varying the number of vertices n and the graph density $d=|E|/(n^2-n)$. We considered a set of 48 random signed graphs having n ranging in the set

Table 3
Results on random instances.

Instance		$Y(G,s)$ + Xpress cuts					Branch-and-cut					Branch-and-cut + Xpress cuts				
		Solution		%Gap	B&C Tree		Solution		%Gap	B&C Tree		Solution		%Gap	B&C Tree	
d	n	%Gap	Time (s)	InitForm	#Nod	NodB	%Gap	time (s)	InitForm	#Nod	NodB	%Gap	time (s)	InitForm	#Nod	NodB
0.1	50	0.00	0.00	5.33	16.00	11.67	0.00	0.00	3.26	11.33	7.67	0.00	0.00	2.23	9.67	8.00
	100	0.00	24.67	11.98	935.00	356.67	0.00	23.33	10.22	865.00	386.00	0.00	22.67	7.93	814.33	361.67
	150	11.12	–	24.03	33882.67	7032.67	9.84	–	25.94	33220.33	12280.00	10.48	–	22.85	32940.33	22746.00
	200	29.44	–	40.93	18644.67	6484.67	30.53	–	44.14	19263.67	5908.67	29.46	–	40.01	18672.33	9690.67
0.3	50	0.00	0.33	19.52	161.67	32.33	0.00	0.33	22.06	141.00	28.33	0.00	0.33	15.89	105.67	16.33
	100	1.45	1120.50(1)	49.85	94877.67	5157.00	0.00	1724.33	51.76	86574.33	5173.67	2.78	2328.00(1)	47.80	132925.67	51519.33
	150	46.99	–	80.82	44560.33	21790.33	48.38	–	87.49	38483.00	2545.67	44.54	–	79.16	38002.00	15068.00
	200	97.19	–	120.26	20845.00	5711.67	85.59	–	121.65	15727.00	4043.33	83.84	–	118.33	15159.00	6475.00
0.5	50	0.00	2.00	42.06	403.67	89.00	0.00	1.33	42.11	344.33	119.00	0.00	1.67	35.71	273.67	52.33
	100	0.00	594.67	78.52	27045.67	5695.00	0.00	688.00	80.93	24082.33	1072.67	0.00	653.33	76.50	24359.67	4180.00
	150	62.16	–	114.58	39774.33	16189.33	58.89	–	127.02	24060.33	1057.67	56.43	–	117.56	25817.67	5914.00
	200	138.67	–	169.43	15136.67	427.00	113.67	–	171.22	10016.67	3213.33	111.11	–	165.30	9355.00	3819.33
0.7	50	0.00	2.33	53.01	343.00	114.33	0.00	1.33	55.20	183.00	25.00	0.00	1.67	47.47	153.00	25.00
	100	0.00	353.33	104.79	15013.00	1125.67	0.00	359.67	109.32	8298.33	1613.00	0.00	402.00	102.37	9413.00	3412.67
	150	62.56	–	148.13	36000.00	2408.33	53.92	–	156.54	15725.33	4576.00	51.92	–	150.95	15057.67	4463.67
	200	166.87	–	209.08	8765.33	1136.67	151.01	–	228.33	3292.33	1050.00	139.21	–	212.87	2744.33	794.67
		26		13		12	30		2		24	38		38		12

{50,100,150,200} and having d varying in the set {0.1,0.3,0.5,0.7}. For each combination of n and d values three different signed graphs were generated; each edge having an equal probability to be positive, negative or parallel.

Table 3 reports the experiments with a number of different strategies, namely: a first one, denoted as "$Y(G,s)$ + Xpress cuts", is the simple version of the branch-and-cut algorithm (in which only constraints in formulation $Y(G,s)$ are used) having Xpress-MP Optimizer automatic cuts activated; a second one, denoted as "Branch-and-cut", is our branch-and-cut algorithm (described in Section 5) having Xpress-MP Optimizer automatic cuts deactivated; and a third one, denoted as "Branch-and-cut + Xpress cuts", is our branch-and-cut algorithm having now Xpress-MP Optimizer automatic cuts activated.

The meaning of columns in Table 3 are the same as in the previous tables except for column "d" that reports the graph density. Each line of this table corresponds to the mean value obtained from the three randomly generated instances. The mean value in column "Time (s)" involves only the instances solved to optimality within the time limit; the value in brackets reports the number of instances not solved to optimality. The last line compares the different strategies by presenting some statistics over the 48 random instances. Column "NodB" counts the number of times each strategy finds the better feasible solution; in the event of a tie the winner strategy is the one obtaining the better feasible solution with a less number of branches. The other columns count the number of times each strategy finds the better gap.

One observes that for most instances the strategy "Branch-and-cut + Xpress cuts" exhibited the smallest gaps. Comparing the other two strategies, the "Branch-and-cut" strategy presented the worst gaps for the first formulation but the best final gaps. Moreover, the "Branch-and-cut" strategy was able to solve two instances not solved by the others. As it was expected, in most cases, the "$Y(G,s)$ + Xpress cuts" investigated more nodes in the branch and bound tree. The difference in columns "#Nod" indicates that our separation routines are not extremely time-consuming. We also observe that, in most cases, the strategy "Branch-and-cut" found the better integer solution with less branches. The computational results obtained for the set of random instances confirm the strength of our cuts.

7. Concluding remarks

The DMERN problem is a combinatorial problem with application in the efficient solution of linear programming problems. In [9] this problem was reformulated as a graph problem defined over a signed graph. While the literature proposes various heuristics for the solution of this problem, there does not yet exist, to the best of our knowledge, an efficient exact approach. We attempt to fill this gap in the literature by presenting a branch-and-cut algorithm for the solution of the DMERN problem. Some computational experiments are carried out over a set of instances that can be found in the literature as a test set for the problem and also over a set of random instances. The numerical results reported in Section 6 show that the branch-and-cut algorithm is an efficient approach for the exact solution of the DMERN problem; most of the instances found in the literature were solved to optimality in a few seconds. We believe the instances used in [9] are not appropriate for assessing the quality of a heuristic approach to the problem; indeed more difficult instances should be used. In a forthcoming study, we will present the optimal solution for a set of larger instances. The numerical experience over random instances indicates that, in order to handle the enlarged instances, we first need to develop other separation routines. We also need to implement preprocessing and scaling procedures such as in [8,9].

Acknowledgments

R.M.V. Figueiredo acknowledges CAPES (Brazilian Ministry of Education) for the Pos-Doctoral Grant. Cid C. de Souza acknowledges CNPq (Brazilian Ministry of Science and Technology) for Grants 301732/2007-8 and 473867/2010-9. The authors would also like to thank a referee for constructive comments.

References

[1] Baker BM, Maye PJ. A heuristic for finding embedded network structure in mathematical programmes. European Journal of Operational Research 1993;67(1):52–63.
[2] Balas E, Yu CS. Finding a maximum clique in an arbitrary graph. SIAM Journal on Computing 1986;14(4):1054–68.
[3] Barahona F, Mahjoub AR. On the cut polytope. Mathematical Programming 1986;36:157–73.
[4] Barahona F, Mahjoub AR. Facets of the balanced (acyclic) induced subgraph polytope. Mathematical Programming 1989;45:21–33.
[5] Barthold JJ. A good submatrix is hard to find. Operations Research Letters 1982;1:190–3.
[6] Bixby RE, Fourer R. Finding embedded network rows in linear programs I. Extraction heuristics. Management Science 1988;34(3):342–76.
[7] Brown GG, Wright WG. Automatic identification of embedded network rows in large-scale optimization models. Mathematical Programming 1984;29:41–56.
[8] Gülpinar N, Gutin G, Mitra G, Zverovitch A. Extracting pure network submatrices in linear programs using signed graphs II. Communications of DQM Research Center 2003;5:58–65.
[9] Gülpinar N, Gutin G, Mitra G, Zverovitch A. Extracting pure network submatrices in linear programs using signed graphs. Discrete Applied Mathematics 2004;137:359–72.
[10] Nemhauser GL, Sigismondi G. A strong cutting plane/branch-and-bound algorithm for node packing. Journal of Operational Research Society 1992;43(5):443–57.
[11] Padberg M. On the facial structure of set packing polyhedra. Mathematical Programming 1973;5:199–215.
[12] Rebennack S. Stable set problem: branch & cut algorithms. In: Floudas CA, Pardalos PM, editors. Encyclopedia of optimization. 2nd ed. Springer; 2008. p. 3676–88.
[13] Rebennack S, Oswald M, Theis DO, Seitz H, Reinelt G, Pardalos, PM. A branch and cut solver for the maximum stable set problem. Journal of Combinatorial Optimization, in press, doi:10.1007/s10878-009-9264-3.
[14] Rossi F, Smriglio S. A branch-and-cut algorithm for the maximum cardinality stable set problem. Operations Research Letters 2001;28:63–74.
[15] FICO™ Xpress Optimization Suite. Xpress-Optimizer, Reference manual, 2009. Release 20.00.
[16] Trotter LE. A class of facet producing graphs for vertex packing polyhedra. Discrete Mathematics 1975;12:373–88.
[17] Zaslavsky T. Signed graphs. Discrete Applied Mathematics 1982;4:47–74.