Min (a)cyclic feedback vertex sets and min ones monotone 3-SAT

Irena Rusu

▶ **To cite this version:**

Irena Rusu. Min (a)cyclic feedback vertex sets and min ones monotone 3-SAT. Theoretical Computer Science, 2019, 771, pp.23-38. 10.1016/j.tcs.2018.11.009 . hal-02464917

HAL Id: hal-02464917
https://hal.science/hal-02464917v1

Submitted on 22 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Version of Record: https://www.sciencedirect.com/science/article/pii/S0304397518306807
Manuscript_5523f2f446884cd6183fad45711f1ddc

Min (A)cyclic Feedback Vertex Sets and Min Ones Monotone 3-SAT

Irena Rusu[1]

LS2N, UMR 6004, Université de Nantes, 2 rue de la Houssinière, BP 92208, 44322 Nantes, France

Abstract

In directed graphs, we investigate the problems of finding: 1) a minimum feedback vertex set (also called the FEEDBACK VERTEX SET problem, or MFVS), 2) a feedback vertex set inducing an acyclic graph (also called the VERTEX 2-COLORING WITHOUT MONOCHROMATIC CYCLES problem, or ACYCLIC FVS) and 3) a minimum feedback vertex set inducing an acyclic graph (ACYCLIC MFVS).

We show that these problems are strongly related to (variants of) MONOTONE 3-SAT and MONOTONE NAE 3-SAT, where monotone means that all literals are in positive form. As a consequence, we deduce several NP-completeness results on restricted versions of these problems. In particular, we define the 2-CHOICE version of an optimization problem to be its restriction where the optimum value is known to be either D or $D+1$ for some integer D, and the problem is reduced to decide which of D or $D+1$ is the optimum value. We show that the 2-CHOICE versions of MFVS, ACYCLIC MFVS, MIN ONES MONOTONE 3-SAT and MIN ONES MONOTONE NAE 3-SAT are NP-complete. The two latter problems are the variants of MONOTONE 3-SAT and respectively MONOTONE NAE 3-SAT requiring that the truth assignment minimize the number of variables set to true.

Finally, we propose two classes of directed graphs for which ACYCLIC FVS is polynomially solvable, namely flow reducible graphs (for which MFVS is already known to be polynomially solvable) and C1P-digraphs (defined by an adjacency matrix with the Consecutive Ones Property).

Keywords: feedback vertex set; 3-SAT; minimum ones 3-SAT; NP-completeness; flow reducible graphs

1. Introduction

In this paper, we consider only simple directed graphs (with no loops or multiple arcs). A *feedback vertex set* (abbreviated FVS) of a directed graph (or *digraph*) $G = (V, E)$ is a set $S \subsetneq V$ such that S contains at least one vertex from each cycle of G. Then we say that S *covers* the cycles in G. The term of *cycle cutset* or simply *cutset* is also used in the literature to name S. The FEEDBACK VERTEX SET problem (abbreviated MFVS) requires to find a FVS S of minimum size in G. A *vertex 2-coloring without monochromatic cycle* of G is a coloring of the vertices in V with two colors, such that no cycle of G has all its vertices of the same color. Each of the colors thus defines a set of vertices inducing an acyclic graph, and each of them may therefore be seen as an acyclic FVS of G. The problem of deciding whether an acyclic FVS exists for a given digraph G is classically called VERTEX 2-COLORING WITHOUT MONOCHROMATIC CYCLE. We abbreviate it as ACYCLIC FVS, in order to emphasize its relationship with feedback vertex sets.

[1]Corresponding author. Phone: 033.2.51.12.58.16 Fax: 033.2.51.12.58.12 Email: Irena.Rusu@univ-nantes.fr

© 2018 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

MFVS has applications in path analysis of flowcharts of computer programs [27], deadlock recovery in operating systems [30], constraint satisfaction and Bayesian inference [3]. The NP-completeness of MFVS in directed graphs has been established in [17], and - given the reduction from VERTEX COVER - it implies both the APX-hardness of the problem, and an extension of the results to undirected graphs. The NP-completeness stands even for directed graphs with indegree and outdegree upper bounded by 2, as well as for planar graphs with indegree and outdegree upper bounded by 3 [14]. Several classes of directed graphs for which MFVS is polynomially solvable are known, including reducible flow graphs [27], cyclically reducible graphs [30], quasi-reducible graphs [24] and completely contractible graphs [19]. The best approximation algorithms [26, 12] reach an approximation factor of $O(\min\{log\tau^* loglog\tau^*, logn\, loglogn\})$, where τ^* is the optimal fractional solution in the natural LP relaxation of the problem. In the undirected case, more intensively studied, many classes of graphs admitting polynomial solutions are known, as for instance interval graphs [20], permutation graphs [6] and cocomparability graphs [7]. The best approximation algorithms [2, 4] reach an approximation factor of 2. A review on MFVS may be found in [13].

ACYCLIC FVS has applications in micro-economics, and more particularly in the study of rationality of consumption behavior [10, 23, 22]. The problem has been introduced in [9, 10] together with a proof of NP-completeness in directed graphs. But the problem is NP-complete even in simple directed graphs with no opposite arcs [23] (called *oriented graphs*). A variant requiring that S covers only the cycles of a fixed length k also turns out to be NP-complete for each $k \geq 3$ [18]. Very few particular classes of directed graphs for which the problem becomes polynomial are known: line-graphs of directed graphs [22], oriented planar graphs of maximum degree 4 and oriented outerplanar graphs [23]. In each graph of these classes, an acyclic FVS always exists. Several results not related to our work here also exist on the undirected case but are in general devoted to the VERTEX k-COLORING WITH NO MONOCHROMATIC CYCLE PROBLEM which is a generalization of ACYCLIC FVS (see [23] for more information). By similarity with MFVS, we introduce here the problem ACYCLIC MFVS, which requires to find a minimum size FVS that induces an acyclic graph, if such FVS exist.

The NP-completeness results we prove concern simple directed graphs containing no pair of opposite arcs, *i.e.* oriented graphs. We define 3c-digraphs as a class in which it is sufficient to cover the 3-cycles in order to cover all the cycles. Starting with a set $\mathcal{C}$ of 3-literal clauses, we associate with $\mathcal{C}$ a set $\mathcal{F}$ of 3-literal clauses with literals in positive form, and we associate with $\mathcal{F}$ a *representative graph* denoted $G^{\triangleleft}(\mathcal{F})$. Then we show various relationships between 3-SAT problems and FVS problems, according to the outline below. In this diagram, a *one* is a variable which is set to true.

$\mathcal{C}$	$\rightarrow$	$\mathcal{F}$	$\rightarrow$	$G^{\triangleleft}(\mathcal{F})$
set of 3-literal clauses on variable set X		set of 3-literal clauses with positive literals on variable set U		graph defined by the clauses, in $\mathcal{F}$, whose vertex set is U and which is a 3c-digraph
satisfies 3-SAT	iff	satisfies NOT-ALL-EQUAL 3-SAT		
		satisfies 3-SAT with a truth assignment whose ones define the set $S \subsetneq U$	iff	S is a FVS
		satisfies NOT-ALL-EQUAL 3-SAT with a truth assignment whose ones define the set S	iff	S is an acyclic FVS

Due to these results, MFVS and ACYCLIC MFVS are close to the problems MIN ONES MONOTONE 3-SAT and MIN ONES MONOTONE NOT-ALL-EQUAL 3-SAT. We use these relationships to deduce new hardness results on these problems. In particular, we define the 2-CHOICE version of an optimization problem to be its restriction where the optimum value is known to be either D or

$D+1$ for some integer D, and the problem is reduced to decide which of D or $D+1$ is the optimum value. We show that the 2-CHOICE versions of MFVS, ACYCLIC MFVS, MIN ONES MONOTONE 3-SAT and MIN ONES MONOTONE NAE 3-SAT are NP-complete. In addition, we also show the NP-completeness of ACYCLIC FVS even for 3c-digraphs. The second part of the paper proposes two classes of graphs for which ACYCLIC FVS is polynomially solvable. More particularly, the graphs in these classes always have an acyclic FVS, and it may be found in polynomial time. These are the reducible flow graphs [15] and the class of C1P-digraphs defined by an adjacency matrix with the Consecutive Ones Property.

The paper is organized as follows. In Section 2 we give the main definitions and notations, and precisely state the problems we are interested in. Section 3 presents the properties of the main reduction, and gives a first NP-completeness result. Section 4 contains the other hardness results. The polynomial cases are studied in Section 5. Section 6 is the Conclusion.

2. Definitions and notations

We denote a directed graph as $G = (V, E)$, and its subgraph induced by a set $V' \subseteq V$ as $G[V']$. Given an arc vw from v to w, w is called a *successor* of v, and v is called a *predecessor* of w. The set of successors (resp. predecessors) of v is denoted $N^+(v)$ (resp. $N^-(v)$). Moreover, $N^+[v]$ (resp. $N^-[v]$) is the notation for $N^+(v) \cup \{v\}$ (resp. $N^-(v) \cup \{v\}$). A *3-cycle digraph* (or a *3c-digraph*) is a directed graph in which each cycle has three vertices defining a 3-cycle. (All the cycles used here are directed).

Focusing now on instances of satisfiability problems, let $\mathcal{C}$ be a set of 3-literal clauses over a set of given variables $X = \{x_1, x_2, \ldots, x_n\}$. A literal is in *positive form* if it equals a variable, and in *negative form* otherwise. A truth assignment of the variables in X such that each clause has at least one true literal is called a *standard* truth assignment. When the truth assignment is such that each clause has at least one true and at least one false literal, it is called a *not-all-equal (NAE)* truth assignment. Let $C_r = (l_a^r \vee l_b^r \vee l_c^r)$ be any clause of $\mathcal{C}$, and assume its literals are ordered from left to right. We define the (non-transitive) relation $\lhd$ as:

$$l_a^r \lhd l_b^r, l_b^r \lhd l_c^r \text{ and } l_c^r \lhd l_a^r \tag{1}$$

where we assume that all clauses contribute to the same relation $\lhd$, defined over all literals in $\mathcal{C}$.

Then we define $G^{\lhd}(\mathcal{C})$ to be the *representative graph* of $\mathcal{C}$ associated with the relation $\lhd$, whose vertices are the literals and such that ll' is an arc iff the relation $l \lhd l'$ has been established by at least one clause (note that there are no multiple arcs).

A cycle of $G^{\lhd}(\mathcal{C})$ is called a *strongly 3-covered cycle* if it contains three vertices that are the three literals of some clause in $\mathcal{C}$. These vertices thus induce a 3-cycle. Moreover, say that a set of 3-literal clauses $\mathcal{C}$, in which the order of literals is fixed, is in *strongly 3-covered form* if all the cycles of the graph $G^{\lhd}(\mathcal{C})$ associated with the relation $\lhd$ are strongly 3-covered cycles. Note that not all the sets $\mathcal{C}$ of 3-literal clauses admit a strongly 3-covered form (examples are easy to build).

We present below the list of problems we are interested in. The notation $t_s(\mathcal{C})$ (resp. $t_{NAE}(\mathcal{C})$) represents the minimum number of true variables (or *ones*) in a standard (resp. NAE) truth assignment of $\mathcal{C}$, if such an assignment exists; otherwise, $t_s(\mathcal{C})$ (resp. $t_{NAE}(\mathcal{C})$) equals the number of variables plus 1. Moreover, $mfvs(G)$ (resp. $amfvs(G)$) is the cardinality of a minimum FVS (respectively a minimum acyclic FVS) of G. The *restrictions* mentioned in the list below indicate the names of the subproblems we deal with, and which are explained below.

3-SAT
Input: A set $\mathcal{C}$ of 3-literal clauses over a set of given variables.
Question: Is there a standard truth assignment for the variables?

Restrictions:
M 3-SAT

MIN ONES 3-SAT (MIN1 3-SAT)
Input: A set $\mathcal{C}$ of 3-literal clauses over a set of given variables. A variable k.
Question: Is there a standard truth assignment for the variables with no more than k true variables?

Restrictions:
MIN1-M 3-SAT
2-CHOICE-MIN1-M 3-SAT

NAE 3-SAT
Input: A set $\mathcal{C}$ of 3-literal clauses over a set of given variables.
Question: Is there a NAE truth assignment for the variables?

Restrictions:
M-NAE 3-SAT

MIN ONES NAE 3-SAT (MIN1-NAE 3-SAT)
Input: A set $\mathcal{C}$ of 3-literal clauses over a set of given variables. A variable k.
Question: Is there a NAE truth assignment for the variables with no more than k true variables?

Restrictions:
MIN1-M-NAE 3-SAT
2-CHOICE-MIN1-M-NAE 3-SAT

FEEDBACK VERTEX SET (MFVS)
Input: A directed graph $G = (V, E)$. A positive integer k.
Question: Is it true that $mfvs(G) \leq k$?

Restrictions:
2-CHOICE-MFVS

ACYCLIC FVS
Input: A directed graph $G = (V, E)$.
Question: Is there a FVS $S \subseteq V$ such that $G[S]$ is acyclic?

MIN ACYCLIC FVS (ACYCLIC MFVS)
Input: A directed graph $G = (V, E)$. A positive integer k.
Question: Is it true that $amfvs(G) \leq k$?

Restrictions:
2-CHOICE-ACYCLIC MFVS

For each of the four satisfiability problems above we have the MONOTONE restriction, asking that the clauses in the input be made only of literals in positive form. These restrictions are identified by a supplementary M in the name of the problem (see the first subproblem in the list of restrictions of each 3-SAT problem). The 2-CHOICE restriction is defined for each minimization problem. Here the input is reduced to instances for which the minimized parameter is known to be either D or $D + 1$, for a given integer D, and the question is - as in the initial problem - whether the parameter is at most D (or, equivalently, equal to D, given the hypothesis). The 2-CHOICE restriction of the MIN1-M 3-SAT problem is therefore:

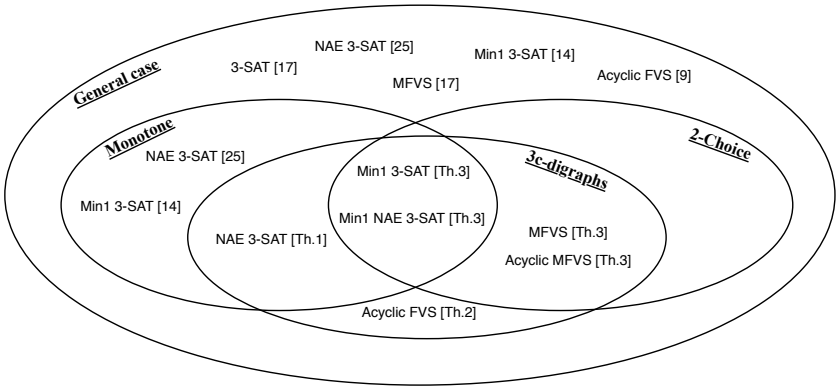

Figure 1: Existing and new hardness results on the problems in our list. The external ellipse contains the problems in their most general form. Each internal ellipse represents a restriction: MONOTONE, 2-CHOICE or 3c-digraphs. Concerning the last restriction, it means that the input graph is a 3c-digraph for FVS problems, respectively that the representative graph is a 3c-digraph for 3-SAT problems. Note that we always use the names of the initial (general) problems, the restrictions being deduced from inclusions into one or several ellipses.

2-CHOICE MIN1-M 3-SAT
Input: A set $\mathcal{C}$ of 3-literal clauses, all in positive form, over a set of given variables. An integer D such that $D \leq t_s(\mathcal{C}) \leq D+1$.
Question: Is it true that $t_s(\mathcal{C}) \leq D$? (equivalently, is it true that $t_s(\mathcal{C}) = D$?)

The 2-CHOICE restrictions for the other minimization problems are similarly defined.

FIG. 1 indicates known NP-completeness results about the problems in our list, as well as the six NP-completeness results we prove here (the six problems in the ellipse representing the restriction to 3c-digraphs). The NP-completeness of 3-SAT is due to Karp [17], whereas that of NAE 3-SAT and M-NAE 3-SAT results from general theorems proved by Schaefer [25]. The problem M 3-SAT is obviously polynomial, but MIN1-M 3-SAT is NP-complete (implying that Min1 3-SAT is NP-complete), since it is the same as 3-HITTING SET and VERTEX COVER for 3-Uniform Hypergraphs [14]. As indicated above, MFVS and ACYCLIC FVS are also NP-complete.

3. From 3-SAT to M-NAE 3-SAT

Our proofs are based on a classical sequence of reductions from 3-SAT to M-NAE 3-SAT, and on the good properties of the representative graph $G^{\lhd}(\mathcal{F})$ of the resulting set of clauses $\mathcal{F}$.

Let $\mathcal{C}$ be a set of m 3-literal clauses, whose literals may be in positive or negative form, on the variable set $X = \{x_1, x_2, \ldots, x_n\}$. We assume w.l.o.g. that no clause of 3-SAT contains the same variable twice (regardless to the positive or negative form). Otherwise, such clauses may be either removed (in case both forms are present) or replaced with equivalent suitable clauses ($(l \vee l \vee l')$ for instance may be replaced with $(l \vee l' \vee u)$ and $(l \vee l' \vee \overline{u})$ where u is a new variable). Then in each clause $C_r = (l_i \vee l_j \vee l_k)$ we assume $i < j < k$ and $l_u \in \{x_u, \overline{x_u}\}$, for all $u \in \{i, j, k\}$. When $\mathcal{C}$ is considered as an instance of 3-SAT, each clause $C_r = (l_i \vee l_j \vee l_k)$ is successively replaced with sets of clauses, as in Table 1, so as to successively reduce 3-SAT to NAE 4-SAT, to NAE 3-SAT and to M-NAE 3-SAT. Note that NAE 4-SAT is stated similarly to NAE 3-SAT except that the clauses have four literals.

These reductions are explained as follows. In the first one (3-SAT to NAE 4-SAT), each variable y_u is related with x_u in the sense that x_u is assigned the value true in the 3-SAT instance iff the truth

Table 1: Successive reductions from 3-SAT to M-NAE 3-SAT. Explanations are given in the text.

Reduction	**New variable set**	**Set of clauses replacing clause** $C_r = (l_i \vee l_j \vee l_k)$	**Precisions**
3-SAT to NAE 4-SAT	$U_1 = \{y_1, y_2, \ldots, y_n, z\}$	$(h_i \vee h_j \vee h_k \vee z)$	$h_u = y_u$ if $l_u = x_u$ $h_u = \overline{y_u}$ if $l_u = \overline{x_u}$
NAE 4-SAT to NAE 3-SAT	$U_2 = \{y_1, y_2, \ldots, y_n, z,$ $w_1, w_2, \ldots, w_m\}$	$(h_i \vee h_j \vee w_r), (\overline{w_r} \vee h_k \vee z)$	
NAE 3-SAT to M-NAE 3-SAT	$U_3 = \cup_{g \in U_2 \setminus \{z\}} \{\alpha_g, \beta_g,$ $a_g, b_g, c_g\} \cup \{z\}$	(Basic clauses) $(\gamma_i \vee \gamma_j \vee \alpha_{w_r}), (\beta_{w_r} \vee \gamma_k \vee z)$, (Consistency clauses) $\cup_{g \in U_2 \setminus \{z\}} \{(\alpha_g \vee \beta_g \vee a_g)$, $(\alpha_g \vee \beta_g \vee b_g), (\alpha_g \vee \beta_g \vee c_g)$, $(a_g \vee b_g \vee c_g)\}$	$\gamma_u = \alpha_{y_u}$ if $h_u = y_u$ $\gamma_u = \beta_{y_u}$ if $h_u = \overline{y_u}$ One occurrence. (do not duplicate for each clause)

assignment in NAE 4-SAT is such that $y_u \neq z$. Equivalently, l_u is true in the 3-SAT instance iff $h_u \neq z$ in the NAE 4-SAT instance. Furthermore, a new variable w is added for each clause when the transition from 4-literal clauses to 3-literal clauses is performed, making that each 4-literal clause is replaced with an equivalent pair of 3-literal clauses. Finally, in order to eliminate literals in negative form in the reduction from NAE 3-SAT to M-NAE 3-SAT, each variable $g \in U_2 \setminus \{z\}$ (since z is already present only in positive form) is replaced with two variables α_g and β_g, respectively representing its literals in positive and negative form. Then, each clause from NAE 3-SAT is replaced with its corresponding clause, a literal in positive (resp. negative) form being replaced with its corresponding α_g (resp. β_g) literal in positive form. The consistency clauses added for each variable guarantee that each pair of variables α_g and β_g must have different values in a NAE truth assignment.

The final set $\mathcal{F}$ of 3-literal clauses (whose literals are all in positive form) associated to the initial set $\mathcal{C}$ of 3-literal clauses is then:

$$\mathcal{F} = \cup_{C_r=(l_i \vee l_j \vee l_k), C_r \in \mathcal{C}} \{\underbrace{(\gamma_i \vee \gamma_j \vee \alpha_{w_r})}_{F_r}, \underbrace{(\beta_{w_r} \vee \gamma_k \vee z)}_{F'_r}\} \cup$$
$$\cup_{g \in U_2 \setminus \{z\}} \{\underbrace{(\alpha_g \vee \beta_g \vee a_g)}_{F_{1g}}, \underbrace{(\alpha_g \vee \beta_g \vee b_g)}_{F_{2g}}, \underbrace{(\alpha_g \vee \beta_g \vee c_g)}_{F_{3g}}, \underbrace{(a_g \vee b_g \vee c_g)}_{F_{4g}}\} \quad (2)$$

where each γ_u is either α_{y_u} or β_{y_u} ($1 \leq u \leq n$) according to the rules in Table 1. In order to fix the terminology, the four consistency clauses associated with a variable g in $U_2 \setminus \{z\}$ are denoted F_{1g}, F_{2g}, F_{3g}, F_{4g} from left to right, whereas the two basic clauses $(\gamma_i \vee \gamma_j \vee \alpha_{w_r})$ and $(\beta_{w_r} \vee \gamma_k \vee z)$ for some fixed $r \in \{1, 2, \ldots, m\}$ are respectively denoted F_r and F'_r. We also denote by $N = |U_3|$ the total number of variables in $\mathcal{F}$, which therefore satisfies $N = 5(n + m) + 1$.

The set $\mathcal{F}$ of clauses resulting from $\mathcal{C}$ in this way is said to be the *M-NAE version* of $\mathcal{C}$. The successive equivalences between problems in Table 1 (with their corresponding inputs) are not very difficult to show (see [28] or [21] for a proof) and imply that:

Proposition 1. *3-SAT with input $\mathcal{C}$ has a solution iff M-NAE 3-SAT with input $\mathcal{F}$ has a solution.*

Or, equivalently, $\mathcal{C}$ has a standard truth assignment iff $\mathcal{F}$ has a NAE truth assignment. Now, we fix for each clause of $\mathcal{F}$ the order of literals used in Equation (2) (from left to right, in each clause). Define the relation $\lhd$ according to Equation(1) and consider the representative graph $G^{\lhd}(\mathcal{F})$. Note that, since $\mathcal{F}$ has only literals in positive form, the set of vertices of $G^{\lhd}(\mathcal{F})$ is U_3.

Proposition 2. *Let $\mathcal{C}$ be any set of 3-literal clauses on a variable set, and let $\mathcal{F}$ be its M-NAE version. The representative graph $G^{\lhd}(\mathcal{F})$ is oriented, and each of its cycles is strongly 3-covered.*

Proof. Recall that $i < j < k$ in each initial clause $C_r \in \mathcal{C}$. We prove several affirmations, which lead to the conclusion.

(A1) *$G^{\lhd}(\mathcal{F})$ contains no pair of opposite arcs and no loop.*

As the variables involved in each clause of $\mathcal{C}$ (and thus of $\mathcal{F}$) are distinct, $G^{\lhd}(\mathcal{F})$ has no loop. Furthermore - due to the local role of a_g, b_g, c_g - the only arcs defined by the consistency clauses that could share both endpoints with other arcs are $\alpha_g\beta_g$. But no basic clause contains literals with the same index g (by the same previous assumption), therefore the arc $\alpha_g\beta_g$ cannot have an opposite arc. Focusing now exclusively on arcs defined by basic clauses, note that basic clauses too contain literals with a local role - namely $\alpha_{w_r}, \beta_{w_r}$ - and therefore only the arcs $\gamma_i\gamma_j$ and $\gamma_k z$ could possibly have opposite arcs. But for the former one this is impossible by the assumption that $i < j$ in each clause, whereas the latter is impossible since the only arcs with source z have destination β_{w_r} ($r = 1, 2, \ldots, m$) and these literals are distinct from literals γ_u ($u = 1, 2, \ldots, n$).

(A2) *Assume $Q = q_1 q_2 \ldots q_d$ is a shortest cycle of $G^{\lhd}(\mathcal{F})$ which is not strongly 3-covered. Then we have*

(i) Q cannot contain any of the vertices a_g, b_g or c_g ($g \in U_2 \setminus \{z\}$).

(ii) the arcs of Q possibly defined by consistency clauses are $\alpha_g\beta_g$, $g \in U_2 \setminus \{z\}$.

(iii) Q cannot contain z.

To show *(i)*, we notice that the (arcless) subgraph of $G^{\lhd}(\mathcal{F})$ induced by a_g, b_g and c_g, for a fixed g, has ingoing arcs only from β_g and outgoing arcs only to α_g, implying that the cycle should necessarily contain α_g and β_g, additionally to a_g, b_g or c_g. Therefore the three literals of a clause among F_{1g}, F_{2g} and F_{3g} would be contained in Q. This is in contradiction with the assumption that Q is not strongly 3-covered. Affirmation *(ii)* is an easy consequence of *(i)*. Finally, if Affirmation *(iii)* was false, then the successor of z would be some β_{w_t} (arc defined by clause F'_t) since the only successors of z are of this form and thus - taking into account that Q cannot contain all literals of F'_t- we would have that the next literal along the cycle is a_{w_t} or b_{w_t} or c_{w_t} (from the consistency clauses F_{1t}, F_{2t} or F_{3t}) thus contradicting *(i)*.

(A3) *If a cycle of $G^{\lhd}(\mathcal{F})$ contains at least two distinct literals $\gamma_e \in \{\alpha_e, \beta_e\}$ and $\gamma_f \in \{\alpha_f, \beta_f\}$ with $e, f \in \{1, 2, \ldots, n\}$, then it is a strongly 3-covered cycle.*

Note that, by Affirmation (A1), such a cycle has at least three vertices. By contradiction, assume that $Q = q_1 q_2 \ldots q_d$ ($d \geq 3$) is a shortest cycle of $G^{\lhd}(\mathcal{F})$ which is not a strongly 3-covered cycle and contains $\gamma_e \in \{\alpha_e, \beta_e\}$ and $\gamma_f \in \{\alpha_f, \beta_f\}$ with $\gamma_e \neq \gamma_f$. In the case where vertices γ_e and γ_f with $e \neq f$ exist on Q, choose γ_e and γ_f such that $e < f$ and the path P from γ_f to γ_e along the cycle Q is as short as possible. Then P contains no other vertex $\gamma_h \in \{\alpha_h, \beta_h\}$, $h \in \{1, 2, \ldots, n\}$, since otherwise γ_e or γ_f would have been differently chosen. Moreover, P has at least two arcs, otherwise an arc $\gamma_f\gamma_e$ would exist with $f > e$ and no clause in $\mathcal{F}$ allows to build such an arc, because of the convention that $i < j < k$ in each clause of $\mathcal{C}$. In the opposite case, *i.e.* when only two vertices γ_e and γ_f with $e = f$ exist on Q, we necessarily have that $\{\gamma_e, \gamma_f\} = \{\alpha_e, \beta_e\}$ and $\alpha_e\beta_e$ is an arc of Q, otherwise Q is not as short as possible. Then we denote P the path from β_e (denoted

γ_f for homogenization reasons) to α_e (denoted γ_e). We have again that P contains no other vertex $\gamma_h \in \{\alpha_h, \beta_h\}$, and P contains at least two arcs (since $d \geq 3$).

Thus in all cases we have a subpath P of Q with at least two arcs, joining γ_f to γ_e, such that $e \leq f$ and there is no other γ_h, $h = 1, 2, \ldots, n$, on P. We may assume that $\gamma_f = q_p$ and $\gamma_e = q_s$, with $p < s$. Then, recalling that P has at least two arcs, we deduce that $q_{p+1} = \alpha_{w_r}$ (due to some basic clause $F_r = (\gamma_i \vee \gamma_f \vee \alpha_{w_r})$ with an appropriate i) or $q_{p+1} = z$ (due to some basic clause $F'_s = (\beta_{w_s} \vee \gamma_f \vee z)$). The latter case is impossible by Affirmation (A2.*iii*). In the former case ($q_{p+1} = \alpha_{w_r}$), we deduce that $q_{p+2} = \beta_{w_r}$ (the only other successor of α_{w_r} is γ_i from the same clause F_r, but then Q would be a strongly 3-covered cycle) and according to (A2.*ii*) q_{p+3} cannot be defined by a consistency clause indexed w_r. Therefore $q_{p+3} = \gamma_k$ from the clause $F'_r = (\beta_{w_r} \vee \gamma_k \vee z)$ and therefore $\gamma_k = \gamma_e$ (since there is no other literal of this form on the path from γ_f to γ_e). As F_r and F'_r express the initial 3-SAT clause $(l_i \vee l_f \vee l_k)$, we deduce $i < f < k$. With $k = e$ we then have $f < e$ and this contradicts our choice above.

(A4) *Every cycle in $G^{\triangleleft}(\mathcal{F})$ is a strongly 3-covered cycle.*

By contradiction, assume that $Q = q_1 q_2 \ldots q_d$ ($d \geq 3$, by (A1)) is a shortest cycle of $G^{\triangleleft}(\mathcal{F})$ which is not a strongly 3-covered cycle. By affirmation (A2.*iii*), z does not belong to Q, and thus the basic clauses F'_r can only contribute to Q with arcs of type $\beta_{w_r}\gamma_k$, for some r and k. By affirmation (A3) which guarantees that at most one vertex γ_h belongs to Q, we deduce that basic clauses F_s can only contribute with arcs of type $\gamma_j\alpha_{w_s}$ or $\alpha_{w_s}\gamma_i$, for some i, j, s. We also have by (A3) that exactly 0 or 2 arcs, among all arcs of these three types - namely $\beta_{w_r}\gamma_k$, $\gamma_j\alpha_{w_s}$ and $\alpha_{w_s}\gamma_i$, for all possible r, i, j, k, s - exist in Q, since 0 or 1 occurrence of a literal of type γ_e is possible on Q. In the case 0 arc is accepted, then no arc from basic clauses is admitted on Q and therefore Q cannot exist (since consistency clauses form only strongly 3-covered cycles). In the case two arcs are accepted, exactly one vertex γ_e exists in Q. One of the two arcs incident with it is $\gamma_e\alpha_{w_u}$ (for some clause $F_u = (\gamma_i \vee \gamma_e \vee \alpha_{w_u})$) since this is the only type of admitted arcs outgoing from γ_e. The arc ingoing to γ_e is either $\beta_{w_v}\gamma_e$ (for some clause $F'_v = (\beta_{w_v} \vee \gamma_e \vee z)$) or $\alpha_{w_z}\gamma_e$ (for some clause $F_z = (\gamma_e \vee \gamma_j \vee \alpha_{w_z})$). All the other arcs are from consistency clauses. The successor of α_{w_u} is then necessarily β_{w_u} and Affirmation (A2.*ii*) implies we have to use an arc outgoing from β_{w_u} and defined by a basic clause. The only solution avoiding to use a third arc from basic clauses is that the predecessor of γ_e on Q is β_{w_v} and $v = u$ (so that $\beta_{w_u} = \beta_{w_v}$). But then γ_e appears both in F_u and in F'_u, and this is impossible since F_u and F'_u correspond to the same initial 3-SAT clause, which has three distinct literals. □

As a consequence, $G^{\triangleleft}(\mathcal{F})$ is a 3c-digraph on the set of variables U_3, with strongly 3-covered cycles. Let us give the full name STRONGLY 3-COVERED MONOTONE NAE 3-SAT to the problem M-NAE 3-SAT reduced to instances where the set of clauses admits a strongly 3-covered form. Then, the representative graph of the input is a 3c-digraph. We have:

Theorem 1. STRONGLY 3-COVERED MONOTONE NAE 3-SAT *is NP-complete.*

Proof. The problem is in NP since it is a particular case of M-NAE 3-SAT. The reduction is from 3-SAT, by associating to a set $\mathcal{C}$ of 3-literal clauses its M-NAE version $\mathcal{F}$. The set $\mathcal{F}$ is in strongly 3-covered form, by Proposition 2. By Proposition 1, the theorem follows. □

4. Hardness results

This section first presents results showing the relationships between solutions of the various 3-SAT problems and the FVS problems (Subsections 4.1 and 4.2). Then these results are used to deduce the aforementioned NP-completeness results (Subsection 4.3).

4.1. Properties of standard truth assignments

Proposition 3. *Let S be a set of variables from U_3. Then S is the set of true variables in a standard truth assignment of $\mathcal{F}$ iff S is a FVS of $G^{\lhd}(\mathcal{F})$ or $S = U_3$.*

Proof. By Proposition 2, the set S of true variables in a standard truth assignment for $\mathcal{F}$ allows us to cover all the cycles in $G^{\lhd}(\mathcal{F})$. If $S \neq U_3$, we have that S is a FVS. Conversely, if S is a FVS of G or $S = U_3$, S covers all the 3-cycles and thus contains at least one literal in each clause. Thus the truth assignment that sets to true all the variables in S is a standard truth assignment for $\mathcal{F}$. □

Remark 1. *Since $\mathcal{F}$ has only literals in positive form, a standard truth assignment for $\mathcal{F}$ always exists.*

Lemma 1. *Consider a standard truth assignment of $\mathcal{F}$ with a minimum number of true variables. Then:*

i) for each $g \in U_2 \setminus \{z\}$ at least one of α_g and β_g is true.

ii) for each $g \in U_2 \setminus \{z\}$, exactly one of the variables a_g, b_g and c_g is true.

Therefore $\frac{2(N-1)}{5} \leq t_s(\mathcal{F}) \leq \frac{2(N-1)}{5} + 1$.

Proof. Let $ta()$ be a standard truth assignment of $\mathcal{F}$ with a minimum number of true variables.

Then no clause of type $(a_g \vee b_g \vee c_g)$, for some g, contains more than one true literal. If, by contradiction, such a clause has all its variable set to true, then we can define a new truth assignment $tb()$ as a variant of $ta()$ where variables b_g and c_g are set to false and α_g is set to true (if it is not already true). Then $tb()$ has less true variables than $ta()$, a contradiction. And in the case where there are exactly two true literals in the clause $(a_g \vee b_g \vee c_g)$, the false literal implies that at least one of the variables α_g, β_g needs to be true so as to satisfy all the clauses F_{1g}, F_{2g} and F_{3g}. But then one of the two true literals in $(a_g \vee b_g \vee c_g)$ may be set to false, implying that $ta()$ does not have a minimum number of true literals, a contradiction.

Then $ta()$ sets exactly one of the variables a_g, b_g, c_g to true, resulting into $\frac{N-1}{5}$ true variables. Moreover, at least one of the variables α_g, β_g needs to be true so as to satisfy all the clauses F_{1g}, F_{2g} and F_{3g}. This yields $\frac{N-1}{5}$ supplementary true variables, and fixes the lower bound of $\frac{2(N-1)}{5}$. In order to show the upper bound, consider the truth assignment that defines z and, for all $g \in U_2 \setminus \{z\}$, variables α_g, a_g as true. All the other variables are false. This is a standard truth assignment for $\mathcal{F}$ with $\frac{2(N-1)}{5} + 1$ true variables. □

Lemma 2. $\frac{2(N-1)}{5} \leq mfvs(G^{\lhd}(\mathcal{F})) = t_s(\mathcal{F}) \leq \frac{2(N-1)}{5} + 1$.

Proof. By Proposition 3, a minimum FVS S corresponds to the minimum set of true variables in a standard truth assignment of $\mathcal{F}$, and viceversa. Therefore $mfvs(G^{\lhd}(\mathcal{F})) = t_s(\mathcal{F})$. By Lemma 1, $t_s(\mathcal{F})$ has the required bounds. □

4.2. Properties of NAE truth assignments

Let $G = (V, E)$ be an oriented graph, and S an acyclic FVS of it. We start this section with a result which allows us to deal simultaneously with the acyclicity of $G[V \setminus S]$ and of $G[S]$. Say that a graph is an *LR-digraph* if there exists a linear order $\prec$ on the vertices of G such that, for each vertex v, its successors are either all before v (*i.e.* on its left side) or all after v (on its right side) in the order $\prec$. Vertex v is called a *left vertex* in the former case, and a *right vertex* in the latter case. The order is called an *LR-order*. The LR-order is *non-trivial* if it admits at least one right vertex and at least one left vertex. An LR-order may be modified so as to move all the right vertices towards left (without changing their relative order) and all the left vertices towards right (again, without changing their relative order) after all the right vertices. The result is still an LR-order, that we call a *standard LR-order*. The subgraph induced by each type of vertices is acyclic, since all the arcs outgoing from a right (resp. left) vertex are oriented towards right (resp. left).

The following proposition is now easy:

Proposition 4. *G has a non-empty acyclic FVS S iff G admits a non-trivial LR-order whose set of right vertices is S.*

Proof. If G has an acyclic FVS S, then $S \neq V$ and a topological order of $G[S]$ followed by a reversed topological order of $G[V \setminus S]$ yields a non-trivial standard LR-order. Conversely, let $\prec$ be a non-trivial LR-order and S be the set of right vertices. Then $S \neq V$ and $G[S]$ is acyclic, since all the arcs in S are oriented from left to right. The graph $G[V \setminus S]$ is also acyclic, since $V \setminus S$ is the set of left vertices and all arcs are oriented from right to left. □

As a consequence, it is equivalent to look for an acyclic FVS in G, and to show that G is an LR-digraph. In order to be as close as possible to the problems we defined, and which have been defined in previous works, we state the results in terms of acyclic FVS. However, in the proofs we merely look for an LR-order of G.

Proposition 5. *Let S be a set of variables from U_3. Then S is the set of true variables in a NAE truth assignment of $\mathcal{F}$ iff S is an acyclic FVS of $G^{\triangleleft}(\mathcal{F})$.*

Proof. Recall that $G^{\triangleleft}(\mathcal{F})$ has vertex set U_3 of cardinality N, and has only strongly 3-covered cycles, by Proposition 2. An arbitrary clause of $\mathcal{F}$ is denoted $(u_i^r \vee u_j^r \vee u_k^r)$ meaning that literal u_a^r is the occurrence of variable u_a from U_3 (necessarily in positive form) in the r-th clause of $\mathcal{F}$. By definition, in each clause, the order $\triangleleft$ defining the strongly 3-covered form is given by the order u_i^r, u_j^r, u_k^r of the variables in the clause.

$\Rightarrow$: Consider a NAE truth assignment for the variables in U_3, whose true variables form the set S. For each arc $u_a u_c$ of $G^{\triangleleft}(\mathcal{F})$ define the relation $u_a \blacktriangleleft u_c$ (intuitively: u_a before u_c) if u_a is true, and $u_c \blacktriangleleft u_a$ if u_a is false. Now, build the directed graph Γ with vertex set $U_3^\Gamma = \{u_1^\Gamma, \ldots, u_N^\Gamma\}$ and arc set $\{u_e^\Gamma u_f^\Gamma | u_e \blacktriangleleft u_f\}$ (do not make use of transitivity). This graph collects the precedence relations defined by $\blacktriangleleft$, and which require that true variables be placed before their successors in $G^{\triangleleft}(\mathcal{F})$ and that false variables be placed after their successors in $G^{\triangleleft}(\mathcal{F})$. More precisely:

$$u_e \blacktriangleleft u_f \text{ iff either literal } u_e \text{ is true and } u_e u_f \text{ is an arc of } G^{\triangleleft}(\mathcal{F}) \text{ or literal } u_f \text{ is false and } u_f u_e \text{ is an arc of } G^{\triangleleft}(\mathcal{F}). \quad (3)$$

We show that Γ is acyclic, so that any topological order of its vertex results into the sought LR-order $\prec$, in which the true variables (*i.e.* those in S) are right vertices. By contradiction, assume Γ is not acyclic.

(B1) *Let* $q_1^\Gamma q_2^\Gamma \dots q_d^\Gamma$ *be a cycle in* Γ. *Then* $q_1 q_2 \dots q_d$ *or* $q_d q_{d-1} \dots q_1$ *is a cycle in* $G^\blacktriangleleft(\mathcal{F})$.

Note first that $d \geq 3$, since otherwise the two arcs of a cycle of length two in Γ would be defined by two arcs in $G^\blacktriangleleft(\mathcal{F})$ between the two same vertices, and $G^\blacktriangleleft(\mathcal{F})$ has at most one arc between two given vertices (Proposition 2). We make the convention that $q_{d+1} = q_1$, $q_0 = q_d$ and similarly for the vertices in Γ. The reasoning is again by contradiction. Assume first that $q_1 q_2$ is an arc of $G^\blacktriangleleft(\mathcal{F})$. Since by contradiction $q_1 q_2 \dots q_d$ is not a cycle in $G^\blacktriangleleft(\mathcal{F})$, let h be the smallest index such that $q_{h+1} q_h$ is an arc of $G^\blacktriangleleft(\mathcal{F})$. Then there must also exist an index $l \geq h+1$ such that $q_l q_{l-1}$ and $q_l q_{l+1}$ are arcs of $G^\blacktriangleleft(\mathcal{F})$. Now, in Γ the orientations of the two arcs with endpoint q_l^Γ are defined by the truth value of q_l, and these arcs are both ingoing to q_l^Γ (if q_l is false) or both outgoing from q_l^Γ (if q_l is true). In both cases, we deduce that $q_1^\Gamma q_2^\Gamma \dots q_d^\Gamma$ is not a cycle in Γ, contradicting the hypothesis. The reasoning is similar in the case where $q_2 q_1$ is an arc of $G^\blacktriangleleft(\mathcal{F})$.

(B2) *No cycle* $q_1 q_2 \dots q_d$ *in* $G^\blacktriangleleft(\mathcal{F})$ *induces a cycle in* Γ.

Every cycle in $G^\blacktriangleleft(\mathcal{F})$ is a strongly 3-covered cycle, since $\mathcal{F}$ is in strongly 3-covered form (Proposition 2). Among the three literals of the same clause present in $q_1 q_2 \dots q_d$, at least one (say q_a) is true and another one (say q_b) is false, according to the truth assignment. Then the arcs of Γ with endpoints $q_a^\Gamma, q_{a+1}^\Gamma$ respectively $q_b^\Gamma, q_{b+1}^\Gamma$ have opposite orientations, and the cycle of $G^\blacktriangleleft(\mathcal{F})$ does not define a cycle in Γ.

(B3) Γ *has no cycle and any topological order of* Γ *is an LR-order of* $G^\blacktriangleleft(\mathcal{F})$.

By (B1) a cycle in Γ could only be defined by a cycle in $G^\blacktriangleleft(\mathcal{F})$. However, by (B2) cycles in $G^\blacktriangleleft(\mathcal{F})$ cannot define cycles in Γ. So Γ is acyclic. Any topological order $\prec$ of Γ extends the relation $\blacktriangleleft$, and therefore satisfies:

$$u_a \prec u_b \text{ and there is an arc with endpoints } u_a, u_b \text{ in } G^\blacktriangleleft(\mathcal{F}) \text{ iff}$$
$$\text{either literal } u_a \text{ is true and } u_a u_b \text{ is an arc of } G^\blacktriangleleft(\mathcal{F}) \text{ or literal } u_b \text{ is false and } u_b u_a \text{ is an arc of } G^\blacktriangleleft(\mathcal{F}). \tag{4}$$

Now, let u_c be a vertex of $G^\blacktriangleleft(\mathcal{F})$. Then either literal u_c is true and for each of its successors u_d we have that (by (4)) $u_c \prec u_d$ so that u_c is a right vertex in $G^\blacktriangleleft(\mathcal{F})$; or literal u_c is false and for each of its successors u_d we have that (by (4)) $u_d \prec u_c$, so that u_c is a left vertex of $G^\blacktriangleleft(\mathcal{F})$.

$\Leftarrow$: Assume now that $G^\blacktriangleleft(\mathcal{F})$ admits an acyclic FVS S. Then $S \neq \emptyset$ since $G^\blacktriangleleft(\mathcal{F})$ has cycles. By Proposition 4, $G^\blacktriangleleft(\mathcal{F})$ admits an LR-order $\prec$ in which the right vertices are defined by S. Assign the value true to every variable that defines a right vertex in $G^\blacktriangleleft(\mathcal{F})$ and the value false to every variable that defines a left vertex in $G^\blacktriangleleft(\mathcal{F})$.

Every clause $(u_i^r \vee u_j^r \vee u_k^r)$ of $\mathcal{F}$ defines a cycle in $G^\blacktriangleleft(\mathcal{F})$ with three vertices u_i, u_j and u_k. In the LR-order $\prec$ of $G^\blacktriangleleft(\mathcal{F})$, at least one of these three vertices is a left vertex (the largest according to $\prec$) and at least one of them is a right vertex (the lowest according to $\prec$). Thus by assigning the value true to all the vertices in S (*i.e.* the right vertices of $G^\blacktriangleleft(\mathcal{F})$), we define a NAE truth assignment for $\mathcal{F}$ whose true variables are exactly those in S. $\square$

Recall that $t_{NAE}(\mathcal{F})$ denotes the minimum number of true variables in a NAE truth assignment of $\mathcal{F}$, if such an assignment exists.

Lemma 3. *Each NAE truth assignment for $\mathcal{F}$ (if such an assignment exists) with minimum number of true variables satisfies:*

i) for each $g \in U_2 \setminus \{z\}$ exactly one of α_g and β_g is true.

ii) for each $g \in U_2 \setminus \{z\}$, exactly one of the variables a_g, b_g and c_g is true.

Therefore $\frac{2(N-1)}{5} \leq t_{NAE}(\mathcal{F}) \leq \frac{2(N-1)}{5} + 1$, with the minimum (resp. maximum) reached iff z is false (resp. is true).

Proof. Affirmation *i)* is due to the consistency constraints and the NAE requirements. For Affirmation *ii)*, clause $(a_g \vee b_g \vee c_g)$ and NAE requirements imply that at least one and at most two of the variables are true. Moreover, if for instance a_g and b_g are true in a NAE truth assignment, modifying b_g to false yields another NAE truth assignment with smaller number of true variables, a contradiction.

By affirmations *i)* and *ii)*, a NAE truth assignment with a minimum number of true variables defines as true at least $\frac{2(N-1)}{5}$ variables, namely one variable among each pair α_g, β_g and one variable among each triple a_g, b_g, c_g. The unique remaining variable in U_3, the variable set of $\mathcal{F}$, is z. The lemma follows. □

Lemma 4. $\frac{2(N-1)}{5} \leq amfvs(G^{\triangleleft}(\mathcal{F})) = t_{NAE}(\mathcal{F}) \leq \frac{2(N-1)}{5} + 1$.

Proof. By Proposition 5, a minimum acyclic FVS S corresponds to the minimum set of true variables in a NAE truth assignment of $\mathcal{F}$, and viceversa. Therefore $amfvs(G^{\triangleleft}(\mathcal{F})) = t_{NAE}(\mathcal{F})$. By Lemma 3, $t_{NAE}(\mathcal{F})$ has the required bounds. □

4.3. NP-completeness results

The results in the previous section allow us to easily deduce the hardness of ACYCLIC FVS for 3c-digraphs:

Theorem 2. ACYCLIC FVS *is NP-complete even for 3c-digraphs.*

Proof. The problem is in NP, since testing the acyclicity of a graph is done in polynomial time. Then the theorem is proved by reduction from 3-SAT. Given a set $\mathcal{C}$ of 3-literal clauses for 3-SAT, its M-NAE version $\mathcal{F}$ satisfies Proposition 1, *i.e.* 3-SAT with input $\mathcal{C}$ has a solution iff M-NAE 3-SAT with input $\mathcal{F}$ has a solution. The last affirmation holds, by Proposition 5, iff $G^{\triangleleft}(\mathcal{F})$ has an acyclic FVS. Moreover, by Proposition 2, $G^{\triangleleft}(\mathcal{F})$ is a 3c-digraph, and the theorem is proved. □

The following lemma will allow us to prove the hardness results for the 2-CHOICE restrictions we defined.

Lemma 5. *The following affirmations are equivalent:*

i) $t_{NAE}(\mathcal{F}) = \frac{2(N-1)}{5}$

ii) $t_s(\mathcal{F}) = \frac{2(N-1)}{5}$

iii) $mfvs(G^{\triangleleft}(\mathcal{F})) = \frac{2(N-1)}{5}$

iv) $amfvs(G^{\lhd}(\mathcal{F})) = \frac{2(N-1)}{5}$

Proof. By Lemma 2, $mfvs(G^{\lhd}(\mathcal{F})) = t_s(\mathcal{F})$ and thus Affirmations *ii)* and *iii)* are equivalent. Similarly, by Lemma 4, Affirmations *i)* and *iv)* are equivalent. It remains to show that *i)* and *ii)* are equivalent.

$i) \Rightarrow ii)$: By Lemma 1, the hypothesis that $t_{NAE}(\mathcal{F}) = \frac{2(N-1)}{5}$ and since each NAE truth assignment is also a standard truth assignment, we have that $\frac{2(N-1)}{5} \leq t_s(F) \leq t_{NAE}(\mathcal{F}) = \frac{2(N-1)}{5}$ and thus $t_s(\mathcal{F}) = \frac{2(N-1)}{5}$.

$ii) \Rightarrow i)$: Since $t_s(\mathcal{F}) = \frac{2(N-1)}{5}$, there exists a standard truth assignment $ta()$ of $\mathcal{F}$ in which exactly one of α_g, β_g and exactly one of a_g, b_g, c_g, for each $g \in U_2 \setminus \{z\}$, is true (by Lemma 1 *i)* and *ii)*). Moreover z is false in this standard truth assignment, since the number of aforementioned true variables is already $\frac{2(N-1)}{5}$. We now show that we can modify $ta()$ in order to obtain a NAE truth assignment containing the same number of true variables. To this end, notice first that if $\mathcal{F}$ contains clauses with three positive literals, these clauses cannot be among clauses $F_{1g}, F_{2g}, F_{3g}, F_{4g}$ and F'_r since all of them contain at least one negative literal. Then define the truth assignment $tb()$ for $\mathcal{F}$ as a variant of $ta()$ where, for each r such that all literals are true in the clause F_r, α_{w_r} is set to false and β_{w_r} is set to true. Then $tb()$ is a NAE truth assignment with the same number of true literals as $ta()$, and thus by Lemma 3 $tb(\mathcal{F})$ has minimum number of true literals, thus gives the value of $t_{NAE}(\mathcal{F})$. □

We are now ready to prove the result concerning the 2-CHOICE variants we defined in Section 2. Let us give the full name STRONGLY 3-COVERED 2-CHOICE-MIN1-M 3-SAT to the problem 2-CHOICE-MIN1-M 3-SAT where the input is restricted to clauses $\mathcal{C}$ admitting a 3-strongly dominated form, and similarly for 2-CHOICE-MIN1-M-NAE 3-SAT. Then the representative graph of the set of clauses in the input is a 3c-digraph.

Theorem 3. *The following problems are NP-complete:*

- STRONGLY 3-COVERED 2-CHOICE-MIN1-M 3-SAT
- STRONGLY 3-COVERED 2-CHOICE-MIN1-M-NAE 3-SAT
- 2-CHOICE MFVS *on 3c-digraphs*
- 2-CHOICE ACYCLIC MFVS *on 3c-digraphs.*

Proof. All problems belong to NP, since they are restrictions of problems belonging to NP.

The reduction is done from NAE 3-SAT, which is NP-complete. Let the input of NAE 3-SAT be a set $\mathcal{C}$ of 3-literal clauses over the variable set $X = \{x_1, x_2, \ldots, x_n\}$. As we did above for the instances of 3-SAT (which are the same as those for 3-SAT), we build the M-NAE version $\mathcal{F}$ of $\mathcal{C}$. Then we have:

(C1) NAE 3-SAT *with input* $\mathcal{C}$ *has a solution iff* M-NAE 3-SAT *with input* $\mathcal{F}$ *has a solution where* z *is false.*

Indeed, if NAE 3-SAT with input $\mathcal{C}$ has a solution then each clause $(l_i \vee l_j \vee l_k)$ has at least one true literal (say l_p) and at least one false literal (say l_q). Following the reductions in Table 1, recall the

interpretation of the literals h_u: l_u is true in the initial instance iff $h_u \neq z$ in the NAE 4-SAT instance. Then, if z is false, h_p is true and h_q is false, implying that the clause $(h_i \vee h_j \vee h_k \vee z)$ of NAE 4-SAT is true. The other reductions successively transform this clause into equivalent sets clauses, implying that there is a solution of M-NAE 3-SAT where z is false. Conversely, if z is false in the solution of M-NAE 3-SAT for $\mathcal{F}$, then by the equivalence of the clauses at each successive reduction (taken backwards in Table 1) we deduce that in each clause $(h_i \vee h_j \vee h_k \vee z)$ of NAE 4-SAT some h_p must be true whereas some h_q must be false. The relationship between l_u and h_u then implies that l_p is true and l_q is false, therefore NAE 3-SAT with input $\mathcal{C}$ also has a solution.

(C2) M-NAE 3-SAT *with input $\mathcal{F}$ has a solution where z is false iff* $t_{NAE}(\mathcal{F}) = \frac{2(N-1)}{5}$.

This is an easy consequence of Lemma 3.

(C3) $t_{NAE}(\mathcal{F}) = \frac{2(N-1)}{5}$ *iff* STRONGLY 3-COVERED 2-CHOICE-MIN1-M-NAE 3-SAT *with input $\mathcal{F}$ and $D = \frac{2(N-1)}{5}$ has a positive answer.*

We first notice that $\mathcal{F}$ satisfies the hypothesis of STRONGLY 3-COVERED 2-CHOICE MIN1-M-NAE 3-SAT. Indeed $\mathcal{F}$ has only strongly 3-covered cycles by Proposition 2, and $\frac{2(N-1)}{5} \leq t_{NAE}(\mathcal{F}) \leq \frac{2(N-1)}{5} + 1$ by Lemma 3. The affirmation is then immediate, by the definition of the problem and the bounds we have on $t_{NAE}(\mathcal{F})$.

From the equivalences proved in (C1), (C2) and (C3), we deduce that STRONGLY 3-COVERED 2-CHOICE-MIN1-M-NAE 3-SAT is NP-complete. To deduce that the three other problems are NP-complete, the same approach is used. By Lemma 5, (C2) holds even when in its second affirmation $t_{NAE}()$ is replaced with $t_s()$, or $mfvs()$ or $amfvs()$. And (C3) holds when 1) $t_{NAE}()$ is replaced with $t_s()$, or $mfvs()$ or $amfvs()$, and 2) STRONGLY 3-COVERED 2-CHOICE MIN1-M-NAE 3-SAT is respectively replaced with STRONGLY 3-COVERED 2-CHOICE MIN1-M 3-SAT, or 2-CHOICE MFVS on 3c-digraphs, or 2-CHOICE ACYCLIC MFVS on 3c-digraphs. □

5. Classes for which *Acyclic FVS* is polynomially solvable

According to Proposition 4, solving ACYCLIC FVS on a particular class of directed graphs is equivalent to testing in polynomial time, for the graphs in the class, whether an LR-order exists.

In this section, we present two classes of LR-digraphs for which the LR-order always exists and may be found in polynomial time. ACYCLIC FVS is therefore solvable in polynomial time, and a solution is computable in polynomial time.

5.1. C1P-digraphs

A directed simple graph $G = (V, E)$ is called a C1P-digraph if its adjacency matrix has the Consecutive Ones Property for Rows (C1PR). That means the columns of the matrix can be reordered such that in each row of the matrix the ones are consecutive. An order of the columns that correctly positions the ones in each row is called a *good order*. Then we easily have:

Proposition 6. *C1P-digraphs are exactly the LR-digraphs admitting an LR-order positioning the successors of each vertex on consecutive places.*

Proof. Let G be a C1P-digraph and A be its adjacency matrix. Let A^{C1R} be the matrix obtained from A by reordering the columns according to the good order, and then reordering the rows according to the good order. Note that the row reordering does not change the C1R property realized by the column reordering. Then A^{C1R} still defines G but with the vertices ordered according to the good order. In this order, the successors of each vertex v (represented by the ones on the row corresponding to v) are consecutive. Moreover, since G is loopless, the element on the main diagonal of A^{C1R} is 0 in each row, therefore the successors of v are either all before or all after the diagonal. Then v is a left vertex if the ones on its row are before the diagonal, and a right vertex otherwise.

Conversely, it is straightforward that each LR-digraph admitting an LR-order positioning the successors of each vertex on consecutive places is a C1P-digraph. □

Note that each order positioning the successors of each vertex on consecutive places necessarily is an LR-order. To find such an order, we take advantage of the numerous studies on the C1P property (see [11] for a survey). They show that a good order may be obtained in linear time for such a graph, using for instance the algorithm in [5].

C1P-digraphs may be defined in terms of intersections of intervals, as follows. Following [8], consider a family V of ordered pairs of intervals (S_v, T_v), $v = 1, 2, \ldots, n$, on the real line. The *intersection digraph* of the family V is the digraph with vertex set $\{1, 2, \ldots, n\}$ and whose arcset is defined by all the ordered pairs vw such that $S_v \cap T_w \neq \emptyset$. An *interval digraph* is any intersection digraph of a family of ordered pairs of intervals. When the intervals T_v are singletons, the interval digraph is called an *interval-point digraph*. Then we have:

Proposition 7 ([8])**.** *A digraph D is an interval-point digraph iff its adjacency matrix has the Consecutive Ones Property for rows.*

As C1P-digraphs are simple digraphs, we deduce that C1P-digraphs are exactly the loopless interval-point digraphs, *i.e.* those for which the ordered pairs of intervals $(S_v, \{t_v\})$ satisfy $t_v \notin S_v$ (so that vv is never an arc).

5.2. Reducible flow graphs

A directed graph $G = (V, E)$ is a *flow graph* if there exists a vertex s such that any vertex in $V \setminus \{s\}$ is reachable by a path from s. The flow graph is then denoted (G, s). Flow graphs are used to model program flows and are therefore important in practice [27]. Among the well-known subclasses of flow graphs, rooted DAGs and series-parallel directed graphs are very well studied examples. We consider here the class of reducible flow graphs, introduced in [15] and studied for instance in [16, 29, 1]. This class is one of the (not so many) classes of directed graphs for which MFVS may be solved in polynomial time [27]. However, the FVS with minimum size returned by the algorithm in [27] may not induce an acyclic graph, as shown by the graph in Fig. 5.2 (which is inspired from an example proposed in [27]).

Most of the definitions we give below follow those in [29]. Let (G, s), with $G = (V, E)$, be a flow graph. We assume G is a simple directed graph, *i.e.* it does not have loops or multiple arcs. A *preorder numbering* of G is an assignment $po()$ of numbers to the vertices of G according to a depth-first search traversal of G: $po(s) = 1$ and $po(v) > po(w)$ iff v is visited after w. Notice that we do not require consecutive $po()$ values. The directed tree $T = (V, E(T))$ defined by the traversal is called a *depth-first spanning tree* (or *DFST*) of G. The existence of a (directed) path in T from v to w is denoted $v \xrightarrow{*} w$. Then the arc set $E(G)$ is partitioned into:

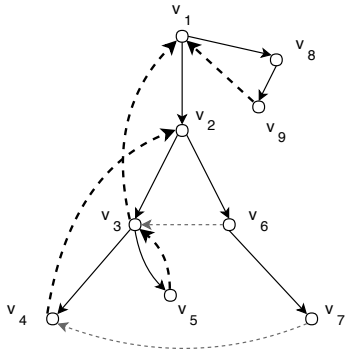

Figure 2: Example of a graph for which the minimum FVS computed by the algorithm in [27] induces a cycle. Plain black, bold dashed and dotted arrows respectively represent tree, cycle and cross arcs. The minimum FVS returned by the algorithm in [27] is formed by the vertices v_1, v_2 and v_3, which induce a cycle.

- *tree arcs* vw, satisfying $vw \in E(T)$.
- *forward arcs* vw, satisfying $vw \notin E(T)$ and $v \xrightarrow{*} w$.
- *cycle arcs* vw, satisfying $w \xrightarrow{*} v$.
- *cross arcs* vw, such that none of $v \xrightarrow{*} w$ and $w \xrightarrow{*} v$ holds, and $po(w) < po(v)$.

Several equivalent definitions of reducible flow graphs are proposed [16]. We chose to say that a flow graph (G, s) is *reducible* if its acyclic subgraph induced by the union of tree, forward and cross arcs is invariable when T changes, *i.e.* has the same set of arcs for any DFST T rooted at s. Reducible flow graphs have a lot of nice properties, that we need for our proof and that we recall below. We use the notation (G, s, T) for a flow graph (G, s) provided with a DFST T. The associated preorder numbering is denoted $po()$.

5.2.1. Properties of reducible flow graphs

For this first property, say that a vertex w *dominates* a vertex v if $w \neq v$ and each path from s to v in G contains w.

Lemma 6 ([16])**.** *Let (G, s, T) be a flow graph. Then (G, s) is reducible iff w dominates v for any cycle arc vw.*

The second property confirms an intuition issued from the definition:

Lemma 7 ([16])**.** *The cycle arcs of a reducible flow graph (G, s) are invariable for any DFST T rooted at s.*

Let w be a vertex of the flow graph (G, s, T). Then define $C(w) = \{v \mid vw \text{ is a cycle arc}\}$ and $P(w) = \{v \mid \exists z \in C(w)$ such that there is a path from v to z which avoids $w\}$. If $C(w)$ is non-empty and $w \neq s$, then w is called a *head*. This definition is correct, because of Lemma 7. Vertex s is not called a head even if $C(w)$ is not empty since it has a special role, as root of T. Furthermore, say that a vertex w satisfies the *T-path property* if for all $v \in P(w)$, $w \xrightarrow{*} v$.

Lemma 8 ([29])**.** *The flow graph (G, s, T) is reducible iff each head w satisfies the T-path property.*

Notice that s trivially satisfies the T-path property, by the definition of T. Let (G, s) be a flow graph and v, w with $v \neq s$ be two of its vertices. *Collapsing v into w* in G is the operation that "shrinks" v and w into a single vertex called w, removing loops and redundant arcs. More formally, arcs are added from w to each vertex in $N^+(v) \setminus N^+[w]$ and from each vertex in $N^-(v) \setminus N^-[w]$ to w. Then v and its incident arcs are removed. *Collapsing a set $S \subseteq V \setminus \{s\}$ into w* in G is the operation that successively collapses each $v \in S$ into w. Given two vertices v, w, a *reduction* collapses v into w if $N^-(v) = \{w\}$ and $v \neq s$.

Lemma 9 ([16]). *A flow graph (G, s) is reducible iff it can be transformed into the graph $(\{s\}, \emptyset)$ by a series of reductions.*

In order to show that reducible flow graphs are LR-digraphs, we need to explain the algorithm in [29] for recognizing reducible flow graphs and constructing the sequence of reductions (according to Lemma 9).

5.2.2. Tarjan's reducibility algorithm

The basic idea of the algorithm is issued from Lemma 8: (G, s) is reducible iff each head w has the T-path property. The algorithm proposed in [29] uses this idea in an optimized way, by successively collapsing sets of vertices, so that subsequent tests are easier to perform. This is possible since the collapsed graph inherits useful properties and constructions of the initial graph:

Lemma 10 ([29]). *Let (G, s, T) be a flow graph and w_1 be its head with largest $po()$ value. Let G' be the graph resulting from G by collapsing $P(w_1)$ into w_1. Then:*

i) (G', s) is a flow graph.

ii) For each arc $v'u'$ in G', either $v'u'$ is an arc of G, or $v' = w_1$ and there is a vertex u in G such that uu' is an arc of G and $w_1 \xrightarrow{*} u$ in T ($v'u'$ in G' then corresponds *to the arcs $v'u'$ of G in the first case, resp. to the arc uu' of G in the second case).*

iii) The subgraph T' of G' whose arcs correspond to arcs in T is a DFST of G', with the preorder numbering given by the $po()$ values (restricted to the vertices in G').

iv) Cycle, forward, cross arcs of G' respectively correspond to cycle, forward, cross arcs of G.

v) If $C'(w)$ and $P'(w)$ are defined in (G', s, T') similarly to (G, s, T), then the heads of G' are the same as the heads of G except w_1. Moreover, for each head w of G', w has the T'-path property in G' iff w has the T-path property in G.

In the Reducibility algorithm (Algorithm 1), heads w are ordered in decreasing order of their $po()$ value (step 1). For each w in this order, the algorithm tests the T-path property in the *current graph* using the sets $P^*(w)$ of vertices (step 5 to 10), collapses $P^*(w)$ into w and continues with the next head. In order to recall the head a vertex v is collapsed into, parameter $hn(v)$ is defined to be $po(w)$ for all $v \in P^*(w)$ (step 8). In order to homogenize the presentation, we define $hn(v) = 1$ for all $v \neq s$ that belong to no $P^*(w)$ and we denote $P^*(s) = \{v \in V \mid hn(v) = 1\}$. Then $P^*(s)$ is collapsed into s (step 13), and thus when the graph is reducible, in the end the resulting graph is $(\{s\}, \emptyset)$. Note that the reductions are not detailed in this algorithm, but collapsing $P^*(w)$ into w at each step means that a sequence of reductions is able to successively collapse each v from $P^*(w)$ into w, as explained later.

Algorithm 1 Reducibility algorithm [29]

Input: A flow graph (G, s).

Output: Answer "No" if (G, s) is not reducible; otherwise, answer "Yes" and for each vertex $v \neq s$ the value $hn(v)$.

```
1: Let w_1, w_2, ..., w_k be the heads of G, ordered by decreasing value of po()
2: for v ∈ V \ {s} do hn(v) = 1 endfor
3: G^{w_0} ← G                                   //w_0 does not exist, this is a simple notation
4: for i = 1, 2 ..., k do
5:    Let P*(w_i) be defined as P(w_i) in the current graph G^{w_{i-1}}
6:    for v ∈ P*(w_i) do
7:       if not w_i →* v then return "No" endif
8:       hn(v) ← po(w_i)
9:    end for
10:   Collapse P*(w_i) into w_i, and call G^{w_i} the resulting graph
11: end for
12: Let P*(s) = {v ∈ V \ {s} | hn(v) = 1}
13: Collapse P*(s) into s
14: Return "Yes"
```

Remark 2. *Sets $C^*(w)$ and $P^*(w)$ are subsets of V, although they are defined in a modified graph. Moreover, sets $P^*(w)$ are disjoint.*

Example 1. Consider the flow graph in Fig. 3. The four heads w_1, w_2, w_3 and w_4 are indicated on the figure. Algorithm Reducibility computes $P^*(w_1) = \{e, f\}$, $P^*(w_2) = \{c, d\}$, $P^*(w_3) = \{w_1, w_2\}$, $P^*(w_4) = \{a, b, w_3\}$ and $P^*(s) = \{w_4\}$. The values $hn()$ are computed on this basis, and are indicated on the figure (second integer in the triple associated with each vertex).

The following relationship may be established between $P^*(w)$ and $P(w)$:

Lemma 11. *Let (G, s, T) be a reducible graph. Then $v \in P^*(w_i)$ iff $v \in P(w_i)$ and $po(w_i) = \max\{po(w_j) \mid v \in P(w_j)\}$. Moreover, the vertices w_j such that $v \in P(w_j)$ appear on the path from s to v, in decreasing order of their value j.*

Proof. We first show that if $v \in P^*(w_i)$ then $v \in P(w_i)$. Since $v \in P^*(w_i)$, there exists a path P in $G^{w_{i-1}}$ avoiding w_i which joins v to a vertex $z \in C^*(w_i)$. By Lemma 10*ii)* applied to $w_{i-1}, w_{i-2}, \ldots, w_1$, each arc of P may be successively replaced by paths in $G^{w_{i-2}}, G^{w_{i-3}}, \ldots, G^{w_1}, G$ so as to obtain a path in G from v to z. Moreover, in Lemma 10*ii)*, the path in T from w_1 to u cannot contain any w_i, $i > 1$, since then the head w_1 would not have the largest value $po()$. This implies that the resulting path in G also avoids w_i, and thus $v \in P(w_i)$.

We are now ready to prove the lemma.

$\Leftarrow$: By Lemma 10*v*) w_i remains a head in $G^{w_1}, \ldots, G^{w_{i-1}}$. Moreover, by the property we just proved, v cannot belong to $P^*(w_1), \ldots, P^*(w_{i-1})$, therefore v is a vertex of $G^{w_{i-1}}$. Now, $v \in P(w_i)$ implies the existence of a path P in G avoiding w_i and joining v to some z in $C(w_i)$. The resulting paths in $G^{w_1}, \ldots, G^{w_{i-1}}$ also avoid w_i (since no vertex is collapsed into w_i) and join v to z (or the vertex z collapses into). But then $v \in P^*(w_i)$ since $C^*(w_i)$ contains z (or the vertex z collapses into), as the arc from z (or the vertex it collapses into) to w_i remains a cycle arc (Lemma 10*iv*)).

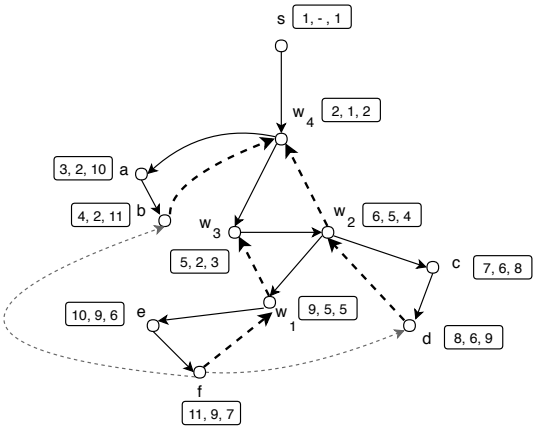

Figure 3: An example of flow graph. The three integer values next to each vertex respectively represent its $po()$, $hn()$ and $sn()$ numbers (the definitions are given in the text when we need them). As before, plain black, bold dashed and dotted arrows respectively represent tree, cycle and cross arcs.

$\Rightarrow$: By the property at the beginning of the proof, we have that $v \in P(w_i)$. If by contradiction $po(w_i) \neq \max\{po(w_j) \mid v \in P(w_j)\}$, then let w_k be the head reaching the maximum value. According to the $\Leftarrow$ part of the lemma, $v \in P^*(w_k)$. This is in contradiction with Remark 2 indicating that the sets $P^*(w_j)$ are disjoint.

To finish the proof, by Lemma 8, the property $w_j \xrightarrow{*} v$ holds for all j such that $v \in P(w_j)$. Since there is a unique path from s to v in T, all these heads w_j are on this path. As $po(w_j) > po(w_l)$ iff $j < l$ by definition, it follows that the order of the heads w_j on the path from s to v is the decreasing order of the index j. □

Once Algorithm Reducibility is applied, the reduction order is computed as follows. First, perform a preorder traversal of T (already built) by considering the children of each vertex in decreasing order of their $po()$ numbers. This new preorder traversal assigns new numbers $sn(v)$ to the vertices v such that [29]:

$$sn(v) < sn(w) \text{ for an arc } vw \text{ iff } vw \text{ is a tree, forward or cross arc.} \tag{5}$$

Then the reduction order α is established using the couples of values $(hn(v), sn(v))$, for each $v \in V$, as follows: vertex v appears before vertex t in the reduction order iff either $hn(v) > hn(t)$, or $hn(v) = hn(t)$ and $sn(v) < sn(t)$. Equivalently, all vertices collapsed into w_1 are before all the vertices collapsed into w_2 and so on. Vertices collapsed into s (those with $hn(v) = 1$) are at the end of the reduction sequence. Moreover, vertices collapsed into the same vertex w_i are ordered according to their increasing value $sn()$. Then, each vertex represents the reduction with its father in the tree T' resulting from the previous reductions. This father is exactly w such that $v \in P^*(w)$.

Example 2. On the example in Fig. 3, the resulting order α gives the sequence of vertices: $e, f, c, d, w_2, w_1, w_3, a, b, w_4, s$.

Remark 3. *In the following, we call Reducibility+ the algorithm obtained from Reducibility by adding the necessary instructions allowing it to output the heads w_i ordered by decreasing value of $po()$, as well as the sets $P^*(s)$ and, for all i, $P^*(w_i)$.*

5.2.3. Our algorithm for finding an LR-order

A cycle arc vw of (G, s) is called *solved* if v is collapsed into w (i.e. if $hn(v) = po(w)$) by Algorithm Reducibility, and *unsolved* otherwise. In the latter case, w is also a head, and there exists a path in T from w to $hn(v)$ (since the cycle arcs from v to w and to $hn(v)$ means they are both on the path from s to v; by definition, $hn(v)$ is the one with higher $po()$ number, so the lower one).

The LR-ordering algorithm (Algorithm 2) attempts to affect the vertices of G either to the set R (right vertices) or to the set L (left vertices) following the idea that for each w which is either a head or s, the vertices in $P^*(w)$, on the one hand, and w, on the other hand, should be in different parts of the partition (R, L). Instead of working with vertices, the algorithm starts by working with *blocks*, that are sets of vertices. The blocks are the singletons containing one vertex w each (where w is a head or s) and the non-empty sets $P^*(w) \setminus W$ (so that each vertex of G belongs to a unique block). The algorithm thus builds (steps 3 to 5) the *undirected* graph B whose vertices are the blocks and whose edges join vertices that must be in different parts (R, L). This graph turns out to be a tree (see Lemma 12), and thus admits a partition in two sets of blocks R' and L' with no internal edges, computed in step 6. Once the partition is found, R (resp. L) collects all the vertices in some block of R' (resp. of L'). The graphs $G[R]$ and $G[L]$ turn out to be acyclic (Lemma 13 below), and thus each of them is an acyclic FVS for G. In order to find an LR-order of G, it is sufficient to order the vertices in each part according to the topological order, to reverse the order found for $G[L]$, and to concatenate the two resulting sequences of vertices (steps 9 to 11).

Example 3. On the example in Fig. 3, the boxes are $\{w_1\}$, $\{w_2\}$, $\{w_3\}$, $\{w_4\}$, $\{e, f\}$(=$P^*(w_1) \setminus W$), $\{c, d\}$(=$P^*(w_2) \setminus W$) and $\{a, b\}$(=$P^*(w_4) \setminus W$). The set $P^*(w_3) \setminus W$ is empty, and is not a vertex of B. The edges of B are therefore joining $\{w_1\}$ to $\{e, f\}$ (due to w_1 and $P^*(w_1)$), $\{w_2\}$ to $\{c, d\}$ (due to w_2), $\{w_3\}$ to $\{w_1\}$ and $\{w_2\}$ (due to w_3), $\{w_4\}$ with $\{w_3\}$ and $\{a, b\}$ (due to w_4), and $\{s\}$ to $\{w_4\}$ (due to s). The partition (R', L') of B is thus:

$R' = \{\{w_1\}, \{w_2\}, \{w_4\}\}$

$L' = \{\{e, f\}, \{c, d\}, \{w_3\}, \{a, b\}, \{s\}\}$

or viceversa. Then $R = \{w_1, w_2, w_4\}$ and $G[R]$ has only two arcs, from w_2 to the other vertices, meaning that one can choose for instance the topological order R^* given by w_2, w_1, w_4. Similarly, $L = \{e, f, c, d, w_3, a, b, s\}$, with arcs ab, ef, cd, fb, fd. The topological order on $G[R]$ may be chosen to be, for instance, $s, e,\ f, a, b, c,\ d, w_3$ which yields, after a complete reversal, L^* : w_3, d, c, b, a, f, e, s. The LR-order U is then $w_2, w_1, w_4,\ w_3, d,\ c, b, a, f, e, s$, with the three first vertices being right vertices and the remaining ones being left vertices.

Note that, in order to avoid confusions, we always use the term *box* to designate the vertices of B, and we reserve the term *vertex* for the vertices of G. The edges of the undirected graph B then join each box $\{w\}$ (corresponding to some $w \in W$) with the boxes $\{w'\}$ whose unique vertex belongs to $P^*(w)$, as well as to the box $P^*(w) \setminus W$ containing the other elements in $P^*(w)$.

Theorem 4. *Algorithm LR-ordering computes in polynomial time an LR-order of a reducible flow graph* (G, s).

Proof. It is clear that the algorithm runs in polynomial time, since all the operations it performs are polynomial. The proof is organized in three lemmas, showing that the behavior of the algorithm is the one we expected in our previous explanations.

Lemma 12. *The (undirected) graph* B *defined in steps 3-5 of Algorithm LR-ordering is a tree.*

Algorithm 2 LR-ordering algorithm

Input: A reducible flow graph (G, s, T).

Output: A sequence U of the vertices in G, defining an LR-order of V.

1: Apply Algorithm Reducibility+ on (G, s, T)
2: $W \leftarrow \{w_1, \ldots, w_k, s\}$
3: $V' \leftarrow \cup_{w \in W}\{\{w\}, P^*(w) \setminus W\}$ // if $P^*(w) \setminus W = \emptyset$, do not use it
4: $E' \leftarrow \{\{w\}X \mid w \in W, X = P^*(w) \setminus W$ or $X = \{w'\}$ with $w' \in W \cap P^*(w)\}$ //edges, not arcs
5: $B \leftarrow (V', E')$
6: $(R', L') \leftarrow$ partition of V' such that each of $B[R'], B[L']$ is edgeless
7: $R \leftarrow \{x \in V \mid x$ belongs to a box in $R'\}$
8: $L \leftarrow \{w \in V \mid w$ belongs to a box in $L'\}$
9: $R^* \leftarrow$ a topological order of the vertices in $G[R]$
10: $L^* \leftarrow$ a reversed topological order of the vertices in $G[L]$
11: $U \leftarrow$ concatenate R^* and L^* in this order
12: Return U

Proof. Each box X in B but $\{s\}$ is collapsed exactly once in Algorithm Reducibility, since by definition each box X is a subset of some $P^*(w_{i(X)})$. According to the definition of E', the edges of B are exactly the pairs $\{w_{i(X)}\}X$. Then each box X but $\{s\}$ has exactly one father $w_{i(X)}$, and B is a tree. □

.

Lemma 13. *The graphs $G[R]$ and $G[L]$ are acyclic.*

Proof. The proof is similar for $G[R]$ and $G[L]$. We therefore present it only for G_R. Let xy be an arc of $G[R]$.

(D1) *If xy is a cycle arc then $sn(x) > sn(y)$ and $hn(x) > hn(y)$. Otherwise, $sn(x) < sn(y)$ and $hn(x) \geq hn(y)$.*

Case 1. Consider first the case where y collapses into a head $w \neq s$. Then $y \in P^*(w)$ and thus by Lemma 11 we have $y \in P(w)$. Let $wb_1 \ldots b_i(= y) \ldots b_l(= v)$ be the cycle of G obtained by concatenating the path from w to y in T (see Lemma 8), a path avoiding w that joins y with some $v \in C(w)$ and the arc vw. We show that w dominates x in G. Indeed, if a path P_1 from s to x avoiding w existed, then the path $P = P_1 b_i(= y) b_{i+1} \ldots b_l(= v)$ would be a path from s to v avoiding w, a contradiction to Lemma 6. Then w dominates x, and thus w is on the path P' in T joining s to x. Then x collapses into one of the vertices in P' (by Lemma 11). If it collapses into one of the vertices z between w (non-included) and x, then $hn(x) > hn(y)$ since $po(z) > po(w)$. Otherwise, x belongs to $P^*(w)$ since there is a path from x to $v(= b_l)$, namely $xb_i(= y)b_{i+1} \ldots b_l(= v)$, and w satisfies $po(w) = \max\{po(w) \mid x \in P(w)\}$ (see Lemma 11). In this case, $hn(x) = hn(y)$. Thus in all cases $hn(x) \geq hn(y)$. However, the equality cannot occur when xy is a cycle arc. Indeed, if xy is a cycle arc, then y is on the path in T from s to x (by the definition of a cycle arc) and moreover xy is unsolved. Then x cannot collapse into y and must collapse into a vertex w' situated on the path from y to x in T. But then $hn(x) > hn(y)$ since $po(w') > po(w)$. Property (5) of $sn()$ finishes the proof in this case.

Case 2. In this case, y collapses into s, thus necessarily $hn(y) = 1$ and thus $hn(x) \geq hn(y)$. Again, for an unsolved cycle arc xy the equality cannot occur since then x must collapse into some w which necessarily has larger $po()$ than s, and thus has strictly larger $hn()$. According to property (5), $sn(x) < sn(y)$ for an arc xy iff xy is a tree, forward or cross arc and we are done.

(D2) *$G[R]$ cannot contain a cycle $A = a_1 a_2 \dots a_t$.*

By (D1), if such a cycle exists, arcs $a_j a_{j+1}$ ($j = 1, \dots, t-1$) imply that $hn(a_1) \geq hn(a_2) \geq \dots \geq hn(a_t)$. If there is at least one strict inequality, we deduce $hn(a_1) > hn(a_t)$ and thus the arc $a_t a_1$ does not satisfy (D1), a contradiction. Therefore, none of the arcs $a_j a_{j+1}$ ($j = 1, \dots, t$), where by convention $a_{t+1} = a_1$, is a cycle arc, as cycle arcs satisfy $hn(a_j) > hn(a_{j+1})$. But then again by (D1) we have that $sn(a_1) < sn(a_2) < \dots < sn(a_t)$, and the arc $a_t a_1$ does not satisfy (D1). □

Lemma 14. *The sequence U gives a standard LR-order of G.*

Proof. This is obvious now, since $G[R]$ and $G[L]$ are acyclic, and U is built using their topological orders.

□

Theorem 4 is now proved. □

Remark 4. *On the example in Fig. 5.2, Algorithm LR-ordering finds $R = \{v_1, v_3, v_4, v_6, v_7\}$ and $L = \{v_2, v_5, v_8, v_9\}$, each of which induce an acyclic graph. Note that none of the acyclic FVS R and L solves* ACYCLIC MFVS*, since a minimum size acyclic FVS with three vertices exists (take for instance $\{v_2, v_5, v_8\}$).*

6. Conclusion

In this paper, we investigated feedback vertex set problems, both from the viewpoint of their hardness and by proposing easier particular cases. We have shown close relationships between these problems, in their standard or acyclic variant, and the 3-SAT problems, in their standard or NAE variant. As a result, we showed the NP-completeness of ACYCLIC FVS even on the class of 3c-digraphs. We have also shown close relationships between the minimal variants of the aforementioned problems, and deduced that even the choice between two proposed values that are one unit far from each other is NP-hard. And this holds even on the class of 3c-digraphs (which are the input graphs in the case of FVS problems, and the representative graphs of the clauses provided with an order of the literals, in the case of 3-SAT problems).

Many questions remain open however. Is the class of 3c-digraphs a hard case for other NP-complete problems? If so, what structural properties justify it? Can we extend the NP-hardness of the 2-CHOICE variants we have studied to smaller classes of graphs? Are the LR-digraphs an important or useful class of graphs, for which - for instance - other NP-hard problems than ACYCLIC FVS have polynomial solutions? To start with, is it possible to solve MFVS or ACYCLIC MFVS in polynomial time on LR-digraphs? Or are there many other classes of graphs (*e.g.* the cyclically-reducible, the quasi-reducible graphs or the completely contractible graphs that we cited in the Introduction) that are subclasses of LR-digraphs?

References

[1] Aho, A. V., & Ullman, J. D. (1976). Node listings for reducible flow graphs. *Journal of Computer and System Sciences*, *13*, 286–299.

[2] Bafna, V., Berman, P., & Fujito, T. (1995). Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs. In *International Symposium on Algorithms and Computation* (pp. 142–151). Springer.

[3] Bar-Yehuda, R., Geiger, D., Naor, J., & Roth, R. M. (1998). Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM Journal on Computing*, *27*, 942–959.

[4] Becker, A., & Geiger, D. (1996). Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, *83*, 167–188.

[5] Booth, K. S., & Lueker, G. S. (1976). Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, *13*, 335–379.

[6] Brandstädt, A., & Kratsch, D. (1985). On the restriction of some NP-complete graph problems to permutation graphs. In *International Conference on Fundamentals of Computation Theory* (pp. 53–62). Springer.

[7] Coorg, S. R., & Rangan, C. P. (1995). Feedback vertex set on cocomparability graphs. *Networks*, *26*, 101–111.

[8] Das, S., Sen, M., Roy, A., & West, D. B. (1989). Interval digraphs: An analogue of interval graphs. *Journal of Graph Theory*, *13*, 189–202.

[9] Deb, R. (2008). Acyclic partitioning problem is NP-complete for $k = 2$. *Private communication, Yale University, United States*, .

[10] Deb, R. (2008). An efficient nonparametric test of the collective household model. *unpublished, available at http://ssrn.com/abstract=1107246*, .

[11] Dom, M. (2009). Algorithmic aspects of the consecutive-ones property. *Bulletin of the European Association for Theoretical Computer Science*, *98*, 27–59.

[12] Even, G., Naor, J. S., Schieber, B., & Sudan, M. (1998). Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, *20*, 151–174.

[13] Festa, P., Pardalos, P. M., & Resende, M. G. (1999). Feedback set problems. In *Handbook of combinatorial optimization* (pp. 209–258). Springer.

[14] Garey, M. R., & Johnson, D. S. (2002). *Computers and intractability*. W.H. Freeman New York.

[15] Hecht, M. S., & Ullman, J. D. (1972). Flow graph reducibility. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing* STOC '72 (pp. 238–250).

[16] Hecht, M. S., & Ullman, J. D. (1974). Characterizations of reducible flow graphs. *Journal of ACM*, *21*, 367–375.

[17] Karp, R. M. (1972). Reducibility among combinatorial problems. In R. Miller, & J. Thatcher (Eds.), *Complexity of Computer Computations* (pp. 85–103). Plenum Press.

[18] Karpiski, M. (2017). Vertex 2-coloring without monochromatic cycles of fixed size is NP-complete. *Theoretical Computer Science*, *659*, 88 – 94.

[19] Levy, H., & Low, D. W. (1988). A contraction algorithm for finding small cycle cutsets. *Journal of algorithms*, *9*, 470–493.

[20] Lu, C. L., & Tang, C. Y. (1997). A linear-time algorithm for the weighted feedback vertex problem on interval graphs. *Information Processing Letters*, *61*, 107–111.

[21] Moore, C., & Mertens, S. (2011). *The nature of computation*. OUP Oxford.

[22] Nobibon, F. T., Cherchye, L., De Rock, B., Sabbe, J., & Spieksma, F. C. (2011). Heuristics for deciding collectively rational consumption behavior. *Computational Economics*, *38*, 173–204.

[23] Nobibon, F. T., Hurkens, C. A. J., Leus, R., & Spieksma, F. C. R. (2012). Coloring graphs using two colors while avoiding monochromatic cycles. *INFORMS Journal on Computing*, *24*, 485–499.

[24] Rosen, B. K. (1982). Robust linear algorithms for cutsets. *Journal of Algorithms*, *3*, 205–217.

[25] Schaefer, T. J. (1978). The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing* STOC '78 (pp. 216–226). New York, NY, USA: ACM.

[26] Seymour, P. D. (1995). Packing directed circuits fractionally. *Combinatorica*, *15*, 281–288.

[27] Shamir, A. (1979). A linear time algorithm for finding minimum cutsets in reducible graphs related databases. *SIAM Journal on Computing*, *8*, 645–655.

[28] Steurer, D. (). http://www.cs.cornell.edu/courses/cs4820/2014sp/notes/reduction-maxcut.pdf.

[29] Tarjan, R. E. (1974). Testing flow graph reducibility. *Journal of Computer and System Sciences*, *9*, 355 – 365.

[30] Wang, C.-C., Lloyd, E. L., & Soffa, M. L. (1985). Feedback vertex sets and cyclically reducible graphs. *Journal of ACM*, *32*, 296–313.