

Journal Pre-proof

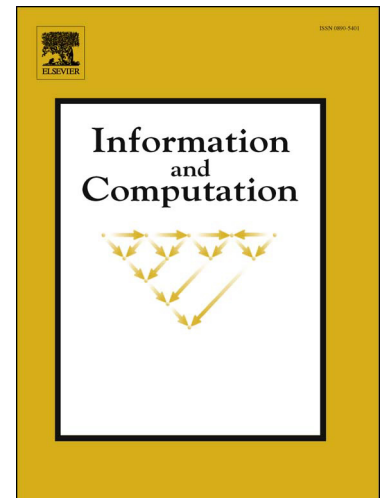
Generalising the maximum independent set algorithm via Boolean networks

Maximilien Gadouleau and David C. Kutner

PII: S0890-5401(25)00002-1
DOI: <https://doi.org/10.1016/j.ic.2025.105266>
Reference: YINCO 105266

To appear in: *Information and Computation*

Received date: 26 March 2024
Revised date: 10 January 2025
Accepted date: 10 January 2025



Please cite this article as: M. Gadouleau and D.C. Kutner, Generalising the maximum independent set algorithm via Boolean networks, *Information and Computation*, 105266, doi: <https://doi.org/10.1016/j.ic.2025.105266>.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2025 Published by Elsevier.

Generalising the maximum independent set algorithm via Boolean networks

Maximilien Gadouleau and David C. Kutner*

January 16, 2025

Abstract

A simple greedy algorithm to find a maximal independent set (MIS) in a graph starts with the empty set and visits every vertex, adding it to the set if and only if none of its neighbours are already in the set. In this paper, we consider the generalisation of this so-called MIS algorithm by allowing it to start with any set of vertices and we prove the hardness of many decision problems related to this generalisation. Our results are based on two main strategies. Firstly, we view the MIS algorithm as a sequential update of a Boolean network, which we shall refer to as the MIS network, according to a permutation of the vertex set. The set of fixed points of the MIS network corresponds to the set of MIS of the graph. Our generalisation then consists in allowing to start from any configuration and to follow a sequential update given by a word of vertices. Secondly, we introduce the concept of a constituency of a graph, that is a set of vertices that is dominated by an independent set. Deciding whether a set of vertices is a constituency is **NP**-complete; decision problems related to the MIS algorithm will be reduced from the CONSTITUENCY problem or its variants. In this paper, we first consider universal configurations, i.e. those that can reach all maximal independent sets; deciding whether a configuration is universal is **coNP**-complete. Second, we consider so-called fixing words, that allow to reach a MIS regardless of the starting configuration, and fixing permutations, which we call permises; deciding whether a permutation is fixing is **coNP**-complete. Third, we consider permissible graphs, i.e. those graphs that have a permis. We construct large classes of permissible and non-permissible graphs, notably near-comparability graphs which may be of independent interest; deciding whether a graph is permissible is **coNP**-hard. Finally, we generalise the MIS algorithm to digraphs. In this case, the algorithm uses the so-called kernel network, whose fixed points are the kernels of the digraph. Deciding whether the kernel network of a given digraph is fixable is **coNP**-hard, even for digraphs that have a kernel. As an alternative, we introduce two further Boolean networks, namely the independent and the dominating networks, whose sets of fixed points contain all kernels. Unlike the kernel network, those networks are always fixable and their fixing word problem is in **P**.

1 Introduction

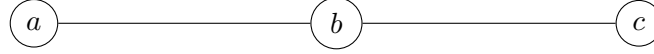
The MIS algorithm A simple greedy algorithm to find a maximal independent set (MIS) in a graph starts with the empty set and visits every vertex, adding it to the set if and only if none of its neighbours are already in the set. We shall refer to it as the MIS algorithm. Because the MIS algorithm always terminates in a maximal independent set, it has been the subject of a stream of work (see [9] and references therein).

A core feature of the classical MIS algorithm is that the starting set of vertices is the empty set. However, the seminal observation of this paper is that this constraint can be lifted. Indeed, starting from any set of vertices and visiting each vertex once, removing a vertex if one of its neighbours already appears in the set, one always terminates at an independent set. Moreover, starting from any independent set and visiting each vertex once, one always terminates at a MIS. Thus, iterating over the vertex set twice is sufficient to obtain a MIS from any starting set of vertices.

*Department of Computer Science, Durham University, Durham, UK. {m.r.gadouleau,david.c.kutner}@durham.ac.uk

As such, the scope of this paper is the generalisation of the MIS algorithm, where one can start with any (not necessarily independent) set of vertices, and one can visit vertices in any order with possible repetitions. Of course, some sequences of vertices will guarantee reaching a MIS from any starting configuration – we shall call those fixing words, while others will not, as in Example 1.1 below.

Example 1.1. *Consider the path on three vertices:*



This graph has two MIS, namely $\{a, c\}$ and $\{b\}$. Starting at the empty set, the MIS algorithm terminates at $\{b\}$ if the sequence begins with b (i.e. $w = bac$ or $w = bca$) or at $\{a, c\}$ otherwise (i.e. $w \in \{abc, acb, cab, cba\}$).

For this graph, abc is not a fixing word: if one starts from the set $\{b, c\}$, then one terminates at $\{c\}$. However, acb is a fixing word: if the starting set contains b , then one terminates at $\{b\}$; otherwise one terminates at $\{a, c\}$.

Finally, for this graph, the words $w = abcab$ and $w = acbacb$ are fixing words, as are any words of the form $w = w^1w^2$, where w^1 and w^2 are permutations of the vertex set.

Contributions for graphs Below we give a summary of our contributions for the MIS algorithm on graphs.

When starting from the empty set, the MIS algorithm is able to reach any possible maximal independent set (if the algorithm goes through the MIS first, then it would terminate with that MIS). However, the empty set is not the only set with that property: the full set of vertices also allows that (this time, if the algorithm finishes with a MIS). In Theorem 4.1, we prove that deciding whether a set of vertices can reach every MIS is **coNP**-complete.

As we showed in Example 1.1, though iterating over the whole set of vertices twice is always sufficient to reach a MIS, it is not always necessary. Consequently, we ask: what are the sequences of vertices which always reach a MIS, regardless of the starting set of vertices? In Theorem 5.1, we prove that deciding whether a sequence offers that guarantee is **coNP**-complete.

Since the MIS algorithm visits each vertex exactly once, we also consider permutations of vertices that are guaranteed to reach a MIS; we call those permises. In Theorem 5.11, we prove that deciding whether a permutation of vertices is a permis is **coNP**-complete. A graph that admits a permis is called permissible. Not all graphs are permissible; the smallest non-permissible graph is the heptagon. We exhibit large classes of permissible and non-permissible graphs. In particular, we introduce near-comparability graphs and classify them in Theorem 6.2; they naturally generalise comparability graphs and can be recognised in polynomial time. We prove that near-comparability graphs are permissible in Proposition 6.1. In Theorem 6.13, we prove that deciding whether a graph is permissible is **coNP**-hard. There is no obvious candidate for a no-certificate, so it may be that the problem is not actually in **coNP**.

In some situations, one can skip some vertices and still guarantee a MIS is reached. For instance, in the complete graph, one can simply update all but one vertex and still reach a maximal independent set, from any starting configuration. We prove in Theorem 5.8 that deciding whether a given set of vertices can be skipped is **coNP**-complete. We also prove in Theorem 5.9 that deciding whether *any* vertices can be skipped is **coNP**-complete.

Boolean networks Our main tool is Boolean networks. A configuration on a graph $G = (V, E)$ is $x \in \{0, 1\}^V$, i.e. the assignment of a Boolean state to every vertex of the graph. A Boolean network is a mapping $F : \{0, 1\}^V \rightarrow \{0, 1\}^V$ that acts on the set of configurations. Boolean networks are used to model networks of interacting entities. As such, it is natural to consider a scenario wherein the different entities update their state at different times. This gives rise to the notion of sequential (or asynchronous) updates, by updating the state of one vertex at a time; a word w then gives the order in which those vertices are updated (with repeats allowed in general). Since the original works by Kauffman [21] and Thomas [29], asynchronous updates have been widely studied, both in terms of modelling purposes and of dynamical analysis (see [11, 4] and references therein). The problem

of whether a Boolean network converges (sequentially) goes back to the seminal result by Robert on acyclic interaction graphs [27]; further results include [19, 18, 24]. Recently, [5] introduced the concept of a fixing word: a word w such that updating vertices according to w will always lead to a fixed point, regardless of the initial configuration. Fixing words are a natural feature of Boolean networks, for two main reasons. Firstly, almost all networks with a fixed point, and hence a positive asymptotic proportion of all networks, have fixing words [10]. Secondly, large classes of Boolean networks, including monotone networks and networks with acyclic interaction graphs, have short fixing words (of length at most cubic in $|V|$) [5, 15].

We refer to the Boolean network where the update function is the conjunction of all the negated variables in the neighbourhood of a vertex as the MIS network on the graph, i.e. $M : \{0, 1\}^V \rightarrow \{0, 1\}^V$ with $M(x)_v = \bigwedge_{u \sim v} \neg x_u$ for all $v \in V$. The MIS network was highlighted in [26, 3], where the fixed points of different conjunctive networks on (directed) graphs are studied. In particular, [26] shows that the set of fixed points of the MIS network is the set of (configurations whose supports are) maximal independent sets of the graph. It is further shown in [3] that for square-free graphs, the MIS network is the conjunctive network that maximises the number of fixed points.

The MIS algorithm can be interpreted in terms of Boolean networks as follows: starting with the all-zero configuration x , update one vertex v at a time according to the update function $M(x)_v = \bigwedge_{u \sim v} \neg x_u$. Once all vertices have been updated, we obtain the final configuration y where the set of ones is a maximal independent set, regardless of the order in which the vertices have been updated. As such, fixing words of the MIS network correspond to sequences of vertices that guarantee that the MIS algorithm terminates for any starting set of vertices. The seminal observation of this paper is that for any permutation w , the word ww is a short fixing word (of length $2|V|$).

Self-stabilization and distributed computing Some of our results may be applied in the context of distributed computing. Similar to Boolean networks, distributed algorithms produce an output through local, independent updates of nodes in a fixed topology. Asynchronous models for distributed computing do not assume a bound on message delay [2], making them less relevant here. Consequently, we focus on synchronous models, in which time is discrete and all nodes perform a SEND-RECEIVE-UPDATE loop synchronously at each time step. The algorithm executed at all nodes is identical, and the state of some node at time t depends only on the state of (all nodes in) its inclusive neighbourhood at time $t-1$. Some key differences with the Boolean Network setting are worth emphasizing. For example, in standard models for synchronous distributed computation: nodes each have an (unbounded-size) internal state, and may solve arbitrarily hard problems during their UPDATE; messages sent may differ from the sender's state; nodes may choose not to SEND anything at all; and updates occur synchronously.

The problem of finding a MIS has been a focus of much study in this setting, including in the LOCAL [17], CONGEST [16] and Beeping models [1, 8]. LOCAL is characterized by its unrestricted message size, whereas CONGEST limits messages to $O(\log |V|)$ bits per outgoing edge. The Beeping model is a significant restriction, in which nodes can communicate only via beeps (which are indistinguishable) and silence [13]. We refer the interested reader to [12] for a more complete treatment of this model's variants, which also includes a discussion of the distributed MIS problem in Sections 4.5 and 6.2.

An algorithm or procedure is said to be “self-stabilizing” if it is guaranteed to reach a legitimate state regardless of its initial state, and additionally will never reach an illegitimate state from a legitimate state [14, 28]. This notion has been integrated into the design of distributed algorithms [22] and is explicitly identified as a feature of the Beeping MIS algorithm given in [1].

Our results do not directly apply to these models; in particular, we assume (and sometimes exploit) asynchronous and instantaneous updates. That is, each vertex's local update is immediately “visible” to all its neighbours. This differs from standard models of distributed computing, which generally incorporate some transmission delay (which is typically one unit in synchronous models, and controlled by an adversary in asynchronous models).

To emulate the MIS network M studied in the present work, it would then be sufficient for a distributed model to support: asynchronous updates (scheduled by an adversary or a helper) and instantaneous transmission. We call the minimum length of time within which each node updates at

least once a *phase*. In the adversarial setting, our seminal observation translates to the fact that this protocol necessarily reaches a MIS within two phases from any starting configuration and self-stabilizes. In the helpful setting, a *Permis* is an update schedule which guarantees self-stabilization within a single phase. By Theorem 5.1, it is **coNP**-complete to determine whether some update schedule satisfies this property; by Theorem 5.11, the problem remains **coNP**-complete even if the schedule is guaranteed to contain every node exactly once; and by Theorem 6.13 it is **coNP**-hard to determine whether any such update schedule exists at all for the given network. If the helpful scheduler is limited to some subset of nodes, Theorem 5.8 means it is **coNP**-hard to determine whether there is a self-stabilizing schedule which uses only that subset of nodes. Furthermore, Theorem 5.9 entails that deciding whether there exists any such schedule using $n - 1$ nodes (even allowing repetitions) is **coNP**-hard.

Constituencies The main tool for the hardness results in this paper is that of a constituency. A constituency is a set of vertices of a graph that is dominated by an independent set, i.e. S is a constituency if there exists an independent set I such that $S \subseteq N(I)$. We believe that the constituency problem is of independent interest for a couple of reasons. Firstly, this is a natural definition for a set of vertices, but to the best of the authors' knowledge, it has not been considered in the literature yet. Secondly, the CONSTITUENCY problem asks whether a set S is a constituency. Unlike problems like CLIQUE, INDEPENDENT SET or VERTEX COVER, the CONSTITUENCY problem does not rely on an integer parameter. Nonetheless, CONSTITUENCY is **NP**-complete, while the problem of deciding whether a set is a clique (or independent set, or vertex cover) is clearly in **P**. As such, CONSTITUENCY provides a natural intractable graph problem whose input does not include an integer. We heavily use CONSTITUENCY and its variants in our hardness proofs, and we believe that this problem could be used more broadly for reductions in the wider graph theory community.

Extension to digraphs In this paper, we further generalise the MIS algorithm by applying it to the digraph case. In this case, a vertex is added to the set if and only if none of its in-neighbours are already in the set. The expected outcome is a kernel, i.e. a dominating independent set (equivalent to a maximal independent set if the digraph is a graph). Unfortunately, not all digraphs have a kernel: odd directed cycles provide an intuitive example of kernel-less digraphs. In fact, deciding whether a digraph has a kernel is **NP**-complete (see [7], p. 119).

Nonetheless, the algorithm for digraphs now corresponds to sequential updates of the kernel network, with $K(x)_v = \bigwedge_{u \rightarrow v} \neg x_u$. Again, the set of fixed points of the kernel network is the set of (configurations whose supports are) kernels [26]. As a side note, the kernel network has been heavily used in logic and philosophy. Indeed, Yablo discovered the first non-self-referential paradox in [30]. The construction for this paradox implicitly applies the fact that the kernel network on a transitive tournament on \mathbb{N} has no fixed point. The study of acyclic digraphs that admit a paradox is continued further in [25], where the kernel network is referred to as an \mathcal{F} -system.

Some digraphs have a kernel and yet the corresponding kernel network does not have a fixing word. More strongly, in Theorem 7.4 we show that deciding whether the kernel network has a fixing word is **coNP**-hard, and we show in Theorem 7.5 that the result holds even for a restricted class of oriented digraphs that have a kernel.

We then consider two other Boolean networks, that are fixable for any digraph. Firstly, the independent network is given by $I(x)_v = x_v \wedge \bigwedge_{u \rightarrow v} \neg x_u$; its set of fixed points consists of the independent sets of the digraph. We classify the fixing words of the independent network in Proposition 7.7 and in Corollary 7.8 we prove that deciding whether a word fixes the independent network is in **P**. Secondly, the dominating network is given by $D(x)_v = x_v \vee \bigwedge_{u \rightarrow v} \neg x_u$; its set of fixed points consists of the dominating sets of the digraph. Similarly, we classify the fixing words of the dominating network in Proposition 7.10 and in Corollary 7.11 we prove that deciding whether a word fixes the dominating network is in **P**.

Outline The rest of the paper is organised as follows. Some necessary background is given in Section 2. Constituencies and districts are introduced in Section 3, where some decision problems based on those are proved to be **NP**- or **coNP**-complete. The configurations that allow to reach any possible

maximal independent set are determined in Section 4. Fixing words, fixing sets, and permises for the MIS network are studied in Section 5. Classes of permissible and non-permissible graphs are given in Section 6. The extension to digraphs is carried out in Section 7, where we first consider fixing words of the kernel network and then study the independent and dominating networks instead. Finally, some conclusions and possible avenues for future work are given in Section 8.

2 Preliminaries

2.1 Graphs and digraphs

Most of our contributions (Sections 3 to 6) will focus on (undirected) graphs. However, when we extend our focus to directed graphs, we shall view graphs as natural special cases of digraphs. As such, we give the background on graphs and digraphs in its full generality, i.e. for digraphs first, and then we make some notes about the special case of graphs.

By digraph, we mean an irreflexive directed graph, i.e. $G = (V, E)$ where $E \subseteq V^2 \setminus \{(v, v) : v \in V\}$. We use the notation $u \rightarrow v$ to mean that $(u, v) \in E$. We say an edge $(u, v) \in E$ is **symmetric** if (v, u) is also an edge, and **oriented** otherwise. We will sometimes emphasize that (u, v) is symmetric by writing it uv instead. For a vertex v , the open in-neighbourhood, closed in-neighbourhood, open out-neighbourhood and closed out-neighbourhood of the vertex v are respectively defined as

$$\begin{aligned} N^{\text{in}}(v) &= \{u \in V : u \rightarrow v\}, & N^{\text{in}}[v] &= N^{\text{in}}(v) \cup \{v\}, \\ N^{\text{out}}(v) &= \{u \in V : v \rightarrow u\}, & N^{\text{out}}[v] &= N^{\text{out}}(v) \cup \{v\}. \end{aligned}$$

All of those are generalised to sets of vertices, e.g. $N^{\text{in}}(S) = \bigcup_{s \in S} N^{\text{in}}(s)$. Clearly, all notations above should reflect the dependence on the digraph G , e.g. $N^{\text{in}}(v; G)$; we shall omit that dependence on any notation when the digraph is clear from the context.

For a digraph $G = (V, E)$ and set of vertices $S \subseteq V$, we call the digraph $(S, \{(u, v) : (u, v) \in E \wedge \{u, v\} \subseteq S\})$ the **induced subgraph** on S , which we denote $G[S]$. We denote $G - S$ the digraph $G[V \setminus S]$. A **path** is a sequence of edges $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k$ where all vertices are distinct; a **cycle** in a digraph is a sequence of edges $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow v_1$ where only the first and the last vertices are equal. A digraph is **strong** if for all vertices u and v , there is a path from u to v . A **strong component** of G is a subset of vertices S such that $G[S]$ is strong, but $G[T]$ is not strong for all $T \supsetneq S$. A digraph is **acyclic** if it has no cycles. An acyclic digraph has a **topological order**, whereby $u \rightarrow v$ only if $u \leq v$. For instance, the digraph where each vertex is a strong component of G and $C \rightarrow C'$ if and only if $u \rightarrow u'$ for some $u \in C$, $u' \in C'$ is acyclic. If $C \rightarrow C'$ in that digraph, we say that C is a **parent component** of C' ; a strong component without any parent is called an **initial component**.

We say that two vertices u and v are **closed twins** if $N^{\text{in}}[u] = N^{\text{in}}[v]$. Accordingly, we say that the vertex m is a **benjamin** of G if there is no vertex v with $N^{\text{in}}[v] \subset N^{\text{in}}[m]$. We denote the set of benjamins of G by $B(G)$ and the corresponding induced subgraph by $G_B = G[B(G)]$. We say that a set of vertices S is **tethered** if there is an edge st between any $s \in S$ and any $t \in T = N(S) \setminus S$.

An **out-tree** is a digraph where all edges are oriented, one vertex has no in-neighbours (the so-called root), and all other vertices have exactly one in-neighbour each. A **spanning out-forest** of a digraph G rooted at a set $S \subseteq V$ is a collection of out-trees, each rooted at a different vertex of S , such that each out-tree is a subgraph of G and each vertex of G appears in exactly one out-tree. We shall use the following simple fact about spanning out-forests.

Lemma 2.1. *If G is strong, then for any nonempty $S \subseteq V$, G has a spanning out-forest rooted at S .*

Proof. Let $S = \{s_1, \dots, s_k\}$. For any $u \in V$, let $s_{n(u)}$ denote the nearest vertex in S from u with the smallest index. More formally, let $d(a, b)$ denote the length of a shortest path from a to b , then we define $s_{n(u)}$ such that $d(s_{n(u)}, u) < d(s_l, u)$ for all $1 \leq l \leq n(u)$ and $d(s_{n(u)}, u) \leq d(s_m, u)$ for all $n(u) \leq m \leq k$. For all $1 \leq l \leq k$, let $T_l = \{u \in V : n(u) = l\}$. It is clear that if v is on a shortest path from $s_{n(u)}$ to u , then $n(v) = n(u)$ (in particular, $n(s_l) = l$). Therefore, each T_l has a spanning out-forest rooted at s_l . The union of all the T_l trees forms the desired spanning out-forest rooted at S . \square

A digraph is **undirected** if all its edges are symmetric; which we shall simply call a **graph**. We then denote $N(v) = N^{\text{in}}(v) = N^{\text{out}}(v)$, which we call the **neighbourhood** of v . A strong graph is called **connected**, and the (initial) strong components of a graph are called its **connected components**. In a graph, if $u \rightarrow v$, then $v \rightarrow u$, which we shall denote by $u \sim v$. Lemma 2.1 applied to graphs is given below – a **spanning forest** of a graph G rooted at a set $S \subseteq V$ is a collection of trees, each rooted at a different vertex of S , such that each tree is a subgraph of G and each vertex of G appears in exactly one tree.

Corollary 2.2. *If G is connected, then for any nonempty $S \subseteq V$, G has a spanning forest rooted at S .*

2.2 Boolean networks

A **configuration** on a digraph $G = (V, E)$ is $x \in \{0, 1\}^V = (x_v : v \in V)$, where $x_v \in \{0, 1\}$ is the state of the vertex v for all v . We denote $\mathbf{1}(x) = \{v \in V : x_v = 1\}$ and $\mathbf{0}(x) = \{v \in V : x_v = 0\}$. Conversely, for any set of vertices $S \subseteq V$, the **characteristic vector** of S is the configuration $x = \chi(S)$ such that $\mathbf{1}(x) = S$. For any set of vertices $S \subseteq V$, we denote $x_S = (x_v : v \in S)$. We denote the all-zero (all-one, respectively) configuration by 0 (by 1 , respectively), regardless of its length.

We consider the following kinds of sets of vertices of, and accordingly configurations on, a digraph G :

1. An **independent set** I is a set such that $(i, j) \notin E$ for all $i, j \in I$. (In other words, $N^{\text{out}}(I) \cap I = \emptyset$.) Every digraph G has an independent set, namely the empty set \emptyset . The collection of characteristic vectors of independent sets of G is denoted by $\mathbf{I}(G)$.
2. A **dominating set** D is a set such that for every vertex $v \in V$, either $v \in D$ or there exists $u \in D$ such that $(u, v) \in E$. (In other words, $N^{\text{out}}(D) \cup D = V$.) Every digraph G has an dominating set, namely V . The collection of characteristic vectors of dominating sets of G is denoted by $\mathbf{D}(G)$.
3. A **kernel** K is a dominating independent set. (In other words, $N^{\text{out}}(K) \uplus K = V$.) Not all digraphs have a kernel, for instance the directed cycle \vec{C}_n (with vertex set \mathbb{Z}_n and edges $(v, v+1)$ for all $v \in \mathbb{Z}_n$) does not have a kernel whenever $n \geq 3$ is odd. The collection of characteristic vectors of kernels of G is denoted by $\mathbf{K}(G)$.
4. If G is a graph, then a kernel is a **maximal independent set** of G , i.e. an independent set I such that there is no independent set $J \supset I$. Every graph has a maximal independent set. In order to highlight this special case of particular importance to this paper, the collection of characteristic vectors of maximal independent sets of G is denoted by $\mathbf{M}(G)$.

Let $w = w_1 \dots w_l \in V^*$ be a sequence of vertices, or briefly a **word**. For any $a, b \in \{1, \dots, l\}$, we denote $w_{a:b} = w_a \dots w_b$ if $a \leq b$ and $w_{a:b}$ is the empty sequence if $a > b$. We also denote by $[w] = \{u \in V : \exists j \ w_j = u\}$ the set of vertices that w visits. For any $S \subseteq V$, the subword of w induced by S , denoted by $w[S]$, is obtained by deleting all the entries in w that do not belong to S ; alternatively, it is the longest subword of w such that $[w[S]] \subseteq S$. A **permutation** of V is a word $w = w_{1:n}$ such that $[w] = V$ and $w_a \neq w_b$ for all $a \neq b$.

A **Boolean network** is a mapping $F : \{0, 1\}^V \rightarrow \{0, 1\}^V$. For any Boolean network F and any $v \in V$, the update of the state of vertex v is represented by the network $F^v : \{0, 1\}^V \rightarrow \{0, 1\}^V$ where $F^v(x)_v = F(x)_v$ and $F^v(x)_u = x_u$ for all other vertices u . We extend this notation to words as follows: if $w = w_1 \dots w_l$ then

$$F^w = F^{w_l} \circ \dots \circ F^{w_2} \circ F^{w_1}.$$

Unless otherwise specified, we let x be the initial configuration, $w = w_1 \dots w_l$ be a word, $y = F^w(x)$ be the final configuration, and for all $0 \leq a \leq l$, $y^a = F^{w_{1:a}}(x)$ be an intermediate configuration, so that $x = y^0$ and $y = y^l$.

If there is a word w such that $y = F^w(x)$, we say that y is **reachable** from x , and we write $x \mapsto_F y$. For any two configurations x and y , we denote $\Delta(x, y) = \{v \in V : x_v \neq y_v\}$. An **F-geodesic** from x to

y is a word w such that $y = F^w(x)$, $[w] = \Delta(x, y)$ and $w_a \neq w_b$ for all $a \neq b$, i.e. w visits every vertex v where x and y differ exactly once, and does not visit any other vertex. If there exists a geodesic from x to y , we denote it by $x \xrightarrow{\text{geo}}_F y$.

The set of **fixed points** of F is $\text{Fix}(F) = \{x \in \{0, 1\}^V : F(x) = x\}$. It is clear that $x \in \text{Fix}(F)$ if and only if $F^w(x) = x$ for any word w , i.e. a “parallel” fixed point is also a “sequential” fixed point. The word w is a **fixing word** for F [5] (and we say that w **fixes** F) if for all x , $F^w(x) \in \text{Fix}(F)$ (see [5] for some examples of fixing words). A Boolean network is **fixable** if it has a fixing word.

3 Constituencies and districts

In this section, we introduce two kinds of sets of vertices, namely constituencies and districts, and we determine the complexity of some decision problems related to them. Even though both concepts will be useful to the sequel of this paper (an intuition behind the role of constituencies is given in the introduction of Section 5), we believe that the concept of constituency in particular is a natural property and is interesting to the wider graph theory community.

3.1 Constituencies

Let $G = (V, E)$ be a graph. A subset S of V is a **constituency** of G if there exists an independent set I such that $S \subseteq N(I)$ (note that this requires that $S \cap I = \emptyset$). The following are equivalent for a set of vertices $S \subseteq V$ (the proof is easy and hence omitted):

1. S is a constituency of G , i.e. there exists an independent set I of G such that $S \subseteq N(I)$;
2. $V \setminus S$ contains a maximal independent set of G ;
3. there exists a maximal independent set M of G such that $M \cap S = \emptyset$;

A **non-constituency** is a set of vertices that is not a constituency. The CONSTITUENCY (NON-CONSTITUENCY, respectively) problem asks, given a graph G and set S , whether S is a constituency (a non-constituency, respectively) of G .

CONSTITUENCY

Input: A graph $G = (V, E)$ and a set of vertices $S \subseteq V$.

Question: Is S a constituency of G ?

NON-CONSTITUENCY

Input: A graph $G = (V, E)$ and a set of vertices $S \subseteq V$.

Question: Is S a non-constituency of G ?

Theorem 3.1. CONSTITUENCY is **NP**-complete.

Proof. Membership of **NP** is known: the yes-certificate is an independent set I such that $S \subseteq N(I)$.

The hardness proof is by reduction from SET COVER, which is **NP**-complete [20]. In SET COVER, the input is a finite set of elements $X = \{x_1, \dots, x_n\}$, a collection $C = \{C_1, C_2, \dots, C_m\}$ of subsets of X , and an integer k . The question is whether there exists a subset $Y \subseteq C$ of cardinality at most k such that $\cup_{C_i \in Y} C_i = X$.

We first construct the graph G on $n + mk$ vertices. G consists of: vertices $Q_j = \{q_j^1, \dots, q_j^k\}$, for each $j \in [m]$; vertices v_i for each $i \in [n]$; edges from each vertex in Q_j to v_i , whenever $x_i \in C_j$; edges connecting $\{q_1^l, q_2^l, \dots, q_m^l\}$ in a clique, for each $l \in [k]$. Let the target set $S = \{v_1, \dots, v_n\}$. This concludes our construction; an illustrative example is shown in Figure 1.

We now show that if (X, C, k) is a yes-instance of SET COVER, then (G, S) is a yes-instance of CONSTITUENCY. Let $Y \subseteq C$ be a set cover of X of cardinality at most k . We obtain the set I as follows:

$$I = \{q_j^a : C_j \text{ is the } a\text{th element of } Y\}.$$

Note that every vertex in I exists in G since Y has cardinality at most k (if $|Y| = k$ then the last subset to appear in Y is its k th element exactly). Further, I is an independent set, since by construction

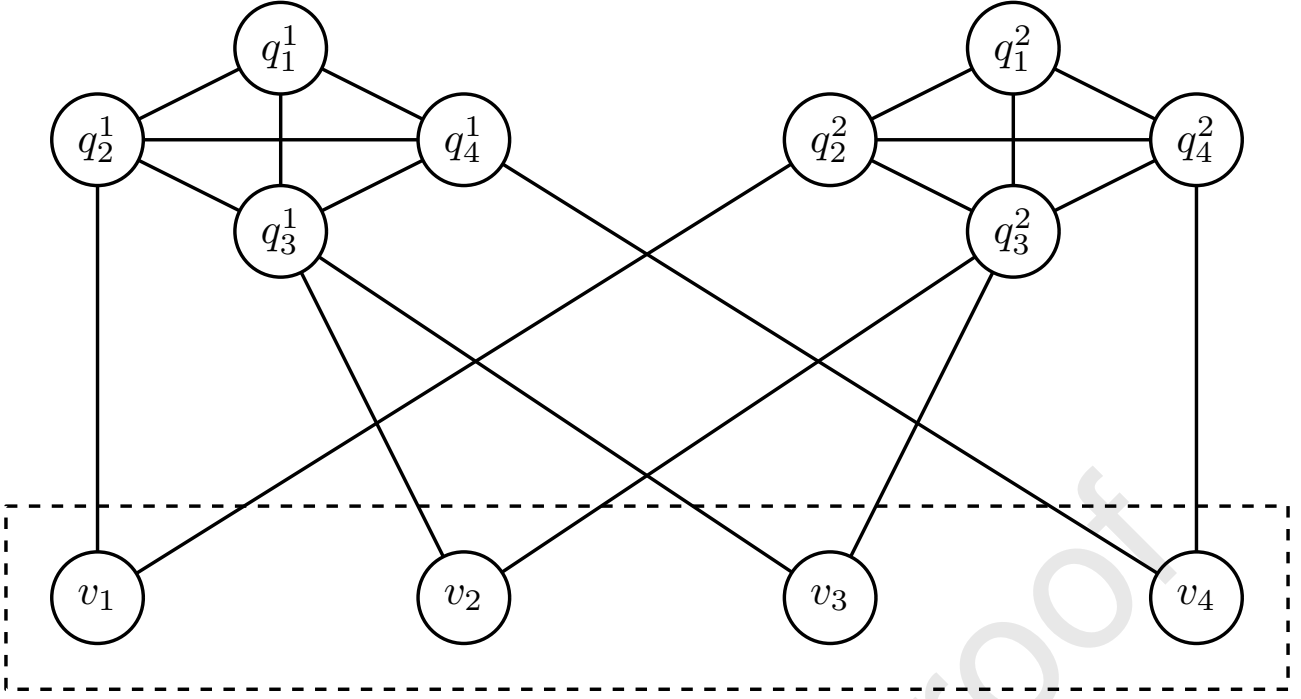


Figure 1: Illustration of the reduction from SET COVER to CONSTITUENCY (the set S is the vertices in the dashed box). Here the SET COVER instance has $C_1 = \emptyset$, $C_2 = \{x_1\}$, $C_3 = \{x_2, x_3\}$, $C_4 = \{x_4\}$, with $k = 2$. Observe that both the SET COVER instance and the CONSTITUENCY instance are no-instances.

every vertex q_j^a is adjacent to some other vertex q_l^b if and only if $a = b$. Lastly, every vertex $v_i \in Y$ is incident to some vertex in I ; for any i , $\exists j : v_i \in C_j$. Then necessarily $\exists a : q_j^a \in I$, and by construction (v_i, q_j^a) is an edge in G .

Conversely, if (G, S) is a yes-instance of CONSTITUENCY then (X, C, k) is a yes-instance of SET COVER. Let I be an independent set in G which dominates S . By construction of G , I has cardinality at most k . Suppose otherwise, for contradiction - then by the pigeon-hole principle there is some clique C_j such that $|C_j \cap I| \geq 2$, contradicting that I is an independent set. We obtain the set Y of cardinality $|I|$ as follows:

$$Y = \{C_j : \exists a \text{ such that } q_j^a \in I\}.$$

We now show Y is a set cover of X . For each $i \in [n]$, v_i must be adjacent to some vertex in I ; denote this vertex q_j^a - now by construction x_i is in the set C_j , and $C_j \in Y$. □

Corollary 3.2. NON-CONSTITUENCY is **coNP**-complete.

We make four remarks about constituencies. Let G be a graph, S be a subset of its vertices, and $T = N(S) \setminus S$. First, if $G - S$ has an isolated vertex t , then S is a constituency of G if and only if $S \setminus N(t)$ is a constituency of $G - t$. Second, whether S is a constituency of G is independent of the edges in $G[S]$. As such, we can (and shall) reduce ourselves to either of two canonical types of instances (G, S) of CONSTITUENCY (and of course, of NON-CONSTITUENCY as well):

Complete type: $G[S]$ is complete and $G - S$ has no isolated vertices.

Empty type: $G[S]$ is empty and $G - S$ has no isolated vertices.

Third, S is a constituency of G if and only if S is a constituency of $G[S \cup T]$. Therefore, we could reduce ourselves to the case where $V = S \cup N(S)$; however, this assumption shall be unnecessary in our subsequent proofs and as such we shall not use it. Fourth, if S is a constituency of G then every subset of S is also a constituency of G .

3.2 Districts

A subset T of vertices of a graph G is a **district** of G if there exists $v \in V \setminus T$ such that $T \cap N(v)$ is a constituency of $G - v$. A **non-district** is a set of vertices that is not a district. The DISTRICT (NON-DISTRICT, respectively) decision problem asks, given a graph G and a set T , whether T is a district (a non-district, respectively) of G .

DISTRICT

Input: A graph $G = (V, E)$ and a set of vertices $T \subseteq V$.

Question: Is T a district of G ?

NON-DISTRICT

Input: A graph $G = (V, E)$ and a set of vertices $T \subseteq V$.

Question: Is T a non-district of G ?

Theorem 3.3. DISTRICT is **NP**-complete.

Proof. Membership of **NP** is known: the yes-certificate is a vertex v and a set of vertices I such that $v \notin I \cup T$, I is an independent set, and $T \cap N(v) \subseteq N(I)$.

The hardness proof is by reduction from CONSTITUENCY, which is **NP**-complete, as proved in Theorem 3.1. Let (G, S) be an instance of CONSTITUENCY, and construct the instance (\hat{G}, \hat{S}) of DISTRICT as follows.

Let $G = (V, E)$ and denote $T = V \setminus S$. Then consider a copy $T' = \{t' : t \in T\}$ of T and an additional vertex $\hat{v} \notin V \cup T'$. Let $\hat{G} = (\hat{V}, \hat{E})$ with $\hat{V} = V \cup T' \cup \{\hat{v}\}$ and $\hat{E} = E \cup \{tt' : t \in T\} \cup \{s\hat{v} : s \in S\}$, and $\hat{S} = S \cup T'$. This construction is illustrated in Figure 2.

We only need to prove that S is a constituency of G if and only if \hat{S} is a district of \hat{G} . Firstly, if S is a constituency of G , then there exists an independent set I of G such that $S \subseteq N(I; G)$. Then $\hat{S} \cap N(\hat{v}; \hat{G}) = S$ is contained in $N(I; \hat{G} - \hat{v})$, thus \hat{S} is a district of \hat{G} .

Conversely, if \hat{S} is a district of \hat{G} , then there exists $u \in \hat{V} \setminus \hat{S}$ such that $\hat{S} \cap N(u; \hat{G})$ is a constituency of $\hat{G} - u$. Then either $u = \hat{v}$ or $u \in T$. Suppose $u = t \in T$, then $t' \in \hat{S}$ is an isolated vertex of $G - t$, hence $\hat{S} \cap N(t; \hat{G})$ is not a constituency of $\hat{G} - t$. Therefore, $u = \hat{v}$ and there exists an independent set \hat{I} of $\hat{G} - \hat{v}$ such that $\hat{S} \cap N(\hat{v}; \hat{G}) = S$ is contained in $N(\hat{I}; \hat{G})$. Since $S \subseteq V$ and $N(S; \hat{G} - \hat{v}) \subseteq V$, we obtain $S \subseteq N(\hat{I} \cap V; \hat{G} - \hat{v}) \cap V = N(\hat{I} \cap V; G)$, where $I = \hat{I} \cap V$ is an independent set of G . Thus, S is a constituency of G . □

Corollary 3.4. NON-DISTRICT is **coNP**-complete.

If T is a district of G , then any subset of T is also a district of G . Therefore, any superset of a non-district is also a non-district. Furthermore, every graph G has a trivial non-district, namely V . The NON-TRIVIAL NON-DISTRICT problem asks whether G has any other non-district. We provide some illustrative instances in Figure 3. We need only consider sets W with $|W| = n - 1$. For C_4 and C_3 , we can by symmetry assume $W = V \setminus \{a\}$, and then for C_4 $\{c\}$ is an independent set which dominates $N(a) \cap W$, whereas for the C_3 there are no vertices outside $N(a) \cap W$ and hence $\{b, c\}$ is a non-district. Similarly, for P_3 , $W = V \setminus \{b\}$ is a non-trivial non-district.

NON-TRIVIAL NON-DISTRICT

Input: A graph $G = (V, E)$.

Question: Does there exist a non-district $S \neq V$ of G ?

Theorem 3.5. NON-TRIVIAL NON-DISTRICT is **coNP**-complete.

Proof. Since any superset of a non-district is also a non-district, G has a non-district $S \neq V$ if and only if there exists $v \in V$ such that $V \setminus \{v\}$ is a non-district of G . Therefore, NON-TRIVIAL NON-DISTRICT is in **coNP**, where the no-certificate is a collection $(I_v : v \in V)$ such that I_v is an independent set of $G - v$ and $N(v) \subseteq N(I_v)$ for all v .

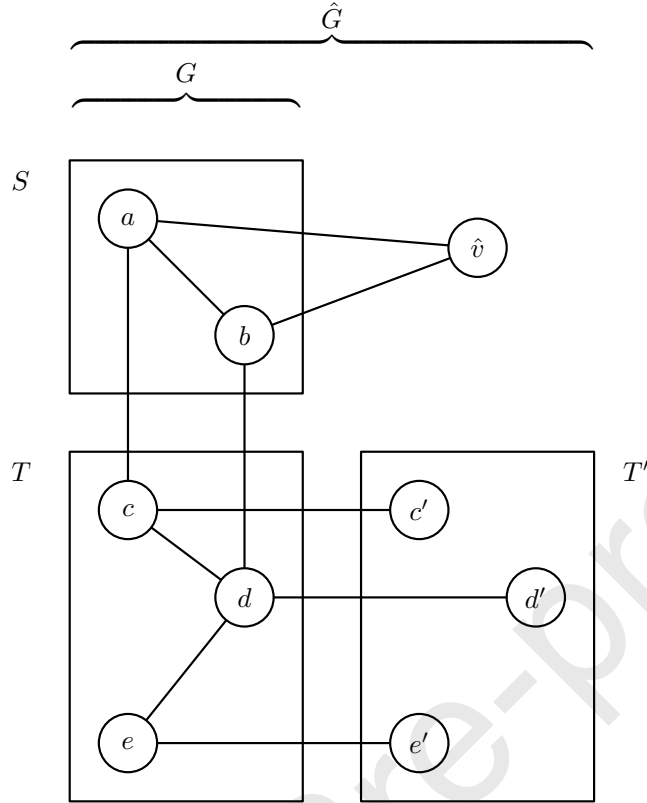


Figure 2: Example reduction from a no-instance of CONSTITUENCY (G, S) to the corresponding no-instance of DISTRICT (\hat{G}, \hat{S}) , with $\hat{S} := S \cup T'$.

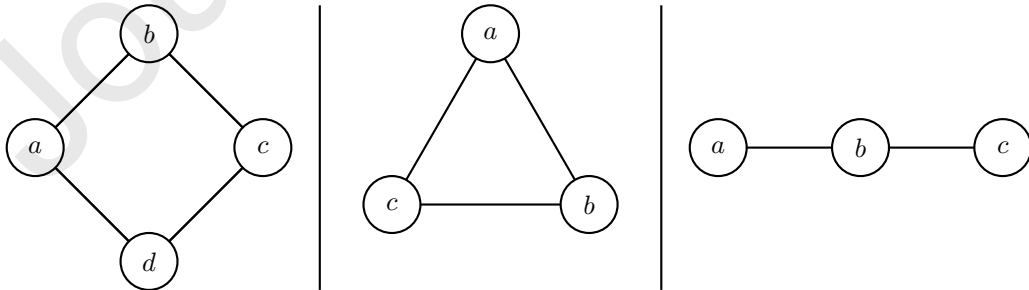


Figure 3: Some example instances of the NON-TRIVIAL NON-DISTRICT problem. C_4 (left) is a no-instance, whereas C_3 (center) and P_3 (right) are yes-instances.

The hardness proof is by reduction from NON-CONSTITUENCY, which is **coNP**-complete by Corollary 3.2. Let $(G = (V, E), S)$ be an instance of NON-CONSTITUENCY of complete type (i.e. where S is a clique in G) and denote $T = V \setminus S$. Let $V' = \{v' : v \in V\}$ be a copy of V , $T'' = \{t'' : t \in T\}$ be a second copy of T , and σ'' and \hat{v} be two additional vertices. For any $A \subseteq V$, we denote $A' = \{a' : a \in A\}$. Let $\hat{G} = (\hat{V}, \hat{E})$ with $\hat{V} = V \cup V' \cup T'' \cup \{\sigma'', \hat{v}\}$ and $\hat{E} = E \cup \{vv' : v \in V\} \cup \{\hat{v}s, \hat{v}s', s'\sigma'' : s \in S\} \cup \{t''\bar{t}'', t''\sigma'' : t, \bar{t} \in T\}$. This is illustrated in Figure 4.

We first show that $W_a = V \setminus \{a\}$ is a district of \hat{G} for all $a \neq \hat{v}$ (note that $W_a \cap N(a; \hat{G}) = N(a; \hat{G})$). Necessarily one of the following holds.

- $a = s \in S$.
Then $N(s; \hat{G}) = \{\hat{v}, s'\} \cup N(s; G)$ is dominated by the independent set $\{\sigma''\} \cup N(s; G)'$.
- $a = s' \in S'$.
Then $N(s'; \hat{G}) = \{\hat{v}, s, \sigma''\}$ is dominated by the independent set $\{\bar{s}, s'\}$ where $\bar{s} \in S \setminus \{s\}$ (and so necessarily $s\bar{s} \in E$).
- $a = \sigma''$.
Then $N(\sigma''; \hat{G}) = S' \cup T''$ is dominated by the independent set $\{\hat{v}\} \cup T''$.
- $a = t \in T$.
Then $N(t; \hat{G}) = t' \cup N(t; G)$ is dominated by the independent set $\{t''\} \cup N(t; G)'$ (or alternatively $\{t'', \hat{v}\}$).
- $a = t' \in T'$.
Then $N(t'; \hat{G}) = \{t, t''\}$ is dominated by the independent set $\{\bar{t}, \bar{t}''\}$ where $t\bar{t} \in E$. (Recall $G - S$ has no isolated vertices in a CONSTITUENCY instance of complete type.)
- $a = t'' \in T''$.
Then $N(t''; \hat{G}) = \{t', \sigma''\} \cup (T'' \setminus \{t''\})$ is dominated by the independent set $\{t\} \cup (V' \setminus \{t'\}) \cup S'$.

We now show that $W_{\hat{v}}$ is a district of \hat{G} if and only if S is a constituency of G . We remark that $W_{\hat{v}} \cap N(\hat{v}; \hat{G}) = S \cup S'$. If $W_{\hat{v}}$ is a district of \hat{G} , then $S \subseteq N(I \setminus \{\hat{v}\}; \hat{G})$ for some independent set I . Therefore $S \subseteq N(I \cap T; G)$, i.e. S is a constituency of G . Conversely, if S is a constituency of G , say $S \subseteq N(I; G)$ for some independent set I of G , then $I \cup \{\sigma''\}$ is an independent set of \hat{G} such that $S \cup S' \subseteq N(I \cup \{\sigma''\}; \hat{G})$, i.e. $W_{\hat{v}}$ is a district of \hat{G} . □

4 Reachability of the MIS network

4.1 The MIS network

By identifying a configuration $x \in \{0, 1\}^V$ with its support $\mathbf{1}(x)$, one can interpret the MIS algorithm as sequential updates of a particular Boolean network. The **MIS network** on G , denoted as $\mathbf{M}(G)$ or simply \mathbf{M} when the graph is clear from the context, is defined by

$$\begin{aligned} \mathbf{M}(x)_v &= \begin{cases} 0 & \text{if } \exists u \in N(v) : x_u = 1 \\ 1 & \text{if } \forall u \in N(v) : x_u = 0 \end{cases} \\ &= \bigwedge_{u \sim v} \neg x_u, \end{aligned}$$

with $\mathbf{M}(x)_v = 1$ if $N(v) = \emptyset$. We then have $\text{Fix}(\mathbf{M}(G)) = \mathbf{M}(G)$ [26, 3].

The MIS algorithm then begins with the all-zero configuration, updates the state of every vertex in order, and leads to a configuration whose support is a maximal independent set. In the language of Boolean networks:

- $x = 0$;

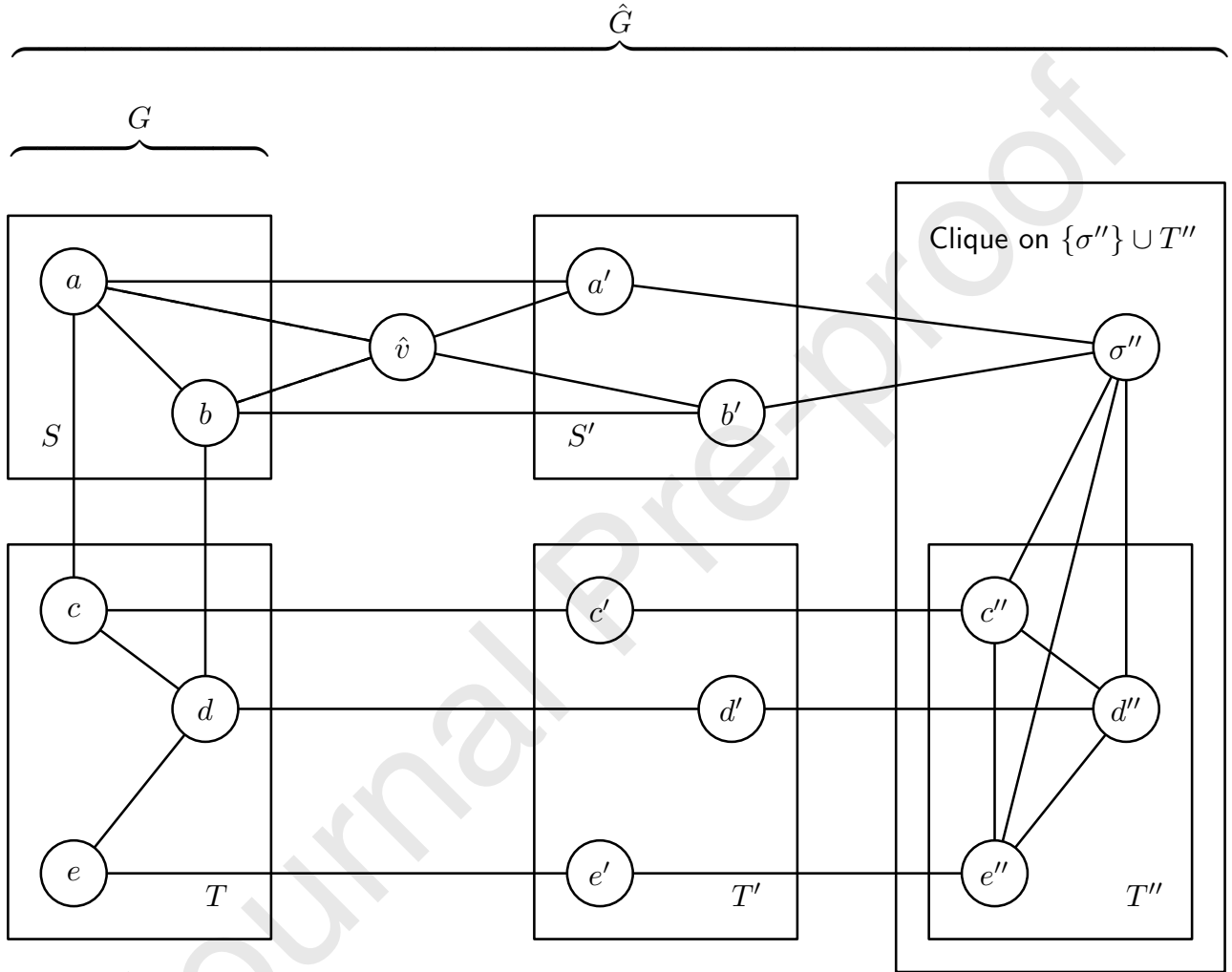


Figure 4: Illustration of the reduction from NON-CONSTITUENCY to NON-TRIVIAL NON-DISTRICT.

- w is a permutation of V ;
- $y = M^w(x) \in M(G)$.

The pivotal role of constituencies for the MIS network can be explained by the equivalence below (its proof is easy and hence omitted). For a set of vertices $S \subseteq V$, S is a constituency of G if and only if there exists a fixed point $y \in M(G)$ such that $y_S = 0$.

4.2 Universal configurations

In this paper, we are interested in removing the constraint on the initial configuration x . This in turn will lead to constraints on the word w , as we shall see in the sequel. For now, in this section, we are interested in initial configurations x that can lead to any MIS y .

Say a configuration x is **F-universal** if every fixed point of F is reachable from x , i.e. $x \mapsto_F z$ for all $z \in \text{Fix}(F)$. Clearly, the all-zero configuration is $M(G)$ -universal, as one can reach any MIS from the empty set. In fact, those fixed points can be reached by a geodesic. We now classify the universal configurations for the MIS network, which actually also allow to reach all fixed points by a geodesic. Since the classification is based on constituencies, the problem of deciding whether a configuration is universal is **coNP**-complete.

M-UNIVERSAL CONFIGURATION

Input: A graph G and a configuration x .

Question: Is x an $M(G)$ -universal configuration?

Theorem 4.1. **M-UNIVERSAL CONFIGURATION** is **coNP**-complete.

We first characterise the configurations y that are reachable from a given configuration x . For any configuration x on G , we denote the collection of connected components of $G[\mathbf{1}(x)]$ as $\mathcal{C}(x)$. Before giving the full statement of the result, we provide some intuition. Suppose y is reachable from x ; we show that y must satisfy two conditions. First, y cannot “create an edge”: if $[w]$ intersects an edge of $G[\mathbf{1}(x)]$, then it will destroy it. Therefore, any edge in $G[\mathbf{1}(y)]$ must be an (untouched) edge of $G[\mathbf{1}(x)]$. Second, y cannot “empty out” a connected component: in order to update a vertex v from $x_v = y_v^{a-1} = 1$ to $y_v = y_v^a = 0$, there must be a neighbour a of v such that $y_v^{a-1} = 1$. Therefore, for any $C \in \mathcal{C}$, $y_C \neq 0$.

Proposition 4.2 then shows that these two conditions are indeed sufficient for reachability, and in fact for reachability by a geodesic.

Proposition 4.2 (Reachability for the MIS network). *Let G be a graph and x, y be two configurations on G . The following are equivalent:*

1. $x \mapsto_M y$;
2. $x \xrightarrow{\text{geo}}_M y$;
3. every edge in $G[\mathbf{1}(y)]$ is an edge in $G[\mathbf{1}(x)]$ and $y_C \neq 0$ for any $C \in \mathcal{C}(x)$.

Lemma 4.3. *Let x be a configuration on G , w be a word, and $y = M^w(x)$. If uv is an edge in $G[\mathbf{1}(y)]$, then $[w] \cap \{u, v\} = \emptyset$.*

Proof. Suppose v is the last updated in $\{u, v\}$, say $v = w_{a+1}$ while $w_b \notin \{u, v\}$ for all $b > a + 1$. Then $y_u^a = 1$ and hence $y_v = y_v^{a+1} = M(y^a)_v = 0$, which is the desired contradiction. \square

Proof of Proposition 4.2. Suppose that $x \mapsto_M y$. It follows from Lemma 4.3 that every edge in $G[\mathbf{1}(y)]$ is an edge in $G[\mathbf{1}(x)]$. We prove that $y_C \neq 0$ for any $C \in \mathcal{C}(x)$. Suppose $y_C = 0$ for some $C \in \mathcal{C}(x)$ with $w = w_{1:l}$ but $y_C^{l-1} \neq 0$. Then $w_l \in C$, $y_{C \setminus \{w_l\}}^{l-1} = 0$, and $y_{w_l}^{l-1} = 1$. Since $M(y^{l-1})_{w_l} = 0$, there exists u such that $u \sim w_l$ and $y_u^{l-1} = 1$.

Claim 1. $u \in [w_{1:l-1}]$.

Proof. Firstly, since $y_{C \setminus \{w_l\}}^{l-1} = 0$ and $y_u^{l-1} = 1$, we have $u \notin C$. Secondly, since $u \in N(w_l) \setminus C$ while $N(w_l) \cap \mathbf{1}(x) \subseteq C$, we have $u \in \mathbf{0}(x)$. Thirdly, since $x_u = 0$ and $y_u^{l-1} = 1$, we must have $u \in [w_{1:l-1}]$. \square

Finally, uw_l is an edge in $G[\mathbf{1}(y^{l-1})]$ with $[w_{1:l-1}] \cap \{u, w_l\} \neq \emptyset$, which contradicts Lemma 4.3.

Conversely, suppose that every edge in $G[\mathbf{1}(y)]$ is an edge in $G[\mathbf{1}(x)]$ and $y_C \neq 0$ for any $C \in \mathcal{C}(x)$. We first describe a word w and we then prove that w is a geodesic from x to y . The word w is constructed in four steps as follows.

1. Let w^0 be any permutation of $\mathbf{1}(y) \cap \mathbf{0}(x)$.
2. For any $C \in \mathcal{C}(x)$, the word w^C is constructed as follows. By Corollary 2.2, let T be a spanning forest of C rooted at $D = \mathbf{1}(y) \cap C$, then w^C is a traversal of the spanning forest from leaves towards roots, skipping the roots. More formally, $w^C = t_1 \dots t_k$ where $\{t_1, \dots, t_k\} = C \setminus D$ and if t_i is a parent of t_j in T , then $i > j$.
3. Let w^1 be a concatenation (in any order) of w^C for every $C \in \mathcal{C}(x)$. More formally, let $\mathcal{C}(x) = \{C_1, \dots, C_m\}$ then $w^1 = w^{C_1} \dots w^{C_m}$.
4. Let $w = w^1 w^0$.

We now verify that w is a geodesic from x to y , i.e. that $[w] = \Delta(x, y)$ and $y = M^w(x)$. Firstly,

$$\begin{aligned} [w^0] &= \mathbf{1}(y) \cap \mathbf{0}(x), \\ [w^1] &= \bigcup_{C \in \mathcal{C}(x)} [w^C] = \bigcup_{C \in \mathcal{C}(x)} C \cap \mathbf{0}(y) = \mathbf{0}(y) \cap \mathbf{1}(x), \\ [w] &= [w^0] \cup [w^1] = \Delta(x, y). \end{aligned}$$

Secondly, we prove that $M^w(x)_{[w^1]} = 0$ while $M^w(x)_{[w^0]} = 1$. Let $C \in \mathcal{C}$ and $w^C = t_1 \dots t_k$. By induction on $1 \leq j \leq k$, we have $M^w(x)_{t_j} = M^{w^C}(x)_{t_j} = 0$ since t_j has a parent $t_i \in C$ which will only be updated after t_j . This shows that $M^w(x)_{[w^1]} = 0$. Moreover, let $w^0 = v_1 \dots v_l$. Suppose $M^w(x)_{v_i} = 0$, then let $z = M^{w^1}(x)$ then there exists u such that $u \sim v_i$ and $z_u = 1$. We derive a contradiction from a case analysis on u .

1. Case 1: $y_u = 1$.
Then uv_i is an edge in $G[\mathbf{1}(y)]$, hence it is an edge in $G[\mathbf{1}(x)]$ so that $x_{v_i} = 1$, which is a contradiction.
2. Case 2: $y_u = 0$ and $x_u = 0$.
Then $u \notin \Delta(x, y) = [w]$ hence $z_u = x_u = 0$, which is a contradiction.
3. Case 3: $y_u = 0$ and $x_u = 1$.
Then $u \in [w^1]$ and hence $z_u = y_u = 0$, which is the desired contradiction.

Therefore $M^w(x)_{[w^0]} = 1$. \square

Corollary 4.4. *The configuration x is $M(G)$ -universal if and only if every $C \in \mathcal{C}(x)$ is a non-constituency of G .*

Proof. If $C \in \mathcal{C}(x)$ is a constituency of G , then there exists a fixed point $z \in M(G)$ with $z_C = 0$, which is not reachable from x by Proposition 4.2. Conversely, if every $C \in \mathcal{C}(x)$ is a non-constituency of G , then for any $z \in M(G)$ we have $z_C \neq 0$ for all C , and hence z is reachable from x . \square

In particular, the all-zero and all-one configurations are M -universal for all graphs.

Proof of Theorem 4.1. Membership of **coNP** is known: the no-certificate is a fixed point $z \in M(G)$ that is not reachable from x ; checking that certificate is by finding $C \in \mathcal{C}(x)$ such that $z_C = 0$.

We prove **coNP**-hardness by reduction from NON-CONSTITUENCY. If (G, S) is an instance of NON-CONSTITUENCY of complete type, then let $x = \chi(S)$ so that $\mathcal{C}(x) = \{S\}$. By Corollary 4.4, x is universal if and only if S is a non-constituency of G . \square

Another consequence of Proposition 4.2 is that any initial configuration can reach a fixed point via a geodesic.

Corollary 4.5. *For any configuration x , there exists $y \in M(G)$ such that $x \xrightarrow{\text{geo}}_M y$.*

Proof. Choose a vertex v_C for every $C \in \mathcal{C}(x)$, then $I = \{v_C : C \in \mathcal{C}(x)\}$ is an independent set. Let M be a maximal independent set that contains I , then $y = \chi(M) \in M(G)$ satisfies Property 3 of Proposition 4.2 and hence is reachable from x by a geodesic. \square

5 Words fixing the MIS network

We now focus on words fixing the MIS network. As we shall prove later, every graph G has a fixing word. Whether a word w fixes the MIS network does not only depend on the set $[w]$ of vertices it visits. Indeed, as seen in Example 1.1 for the graph P_3 , the word $w = abc$ does not fix M , while $w = acb$ does fix M . We define **FIXING WORD** to be the decision problem asking, for an instance (G, w) , whether w fixes $M(G)$.

FIXING WORD

Input: A graph $G = (V, E)$ and a word w .

Question: Does w fix $M(G)$?

Theorem 5.1. *FIXING WORD is **coNP**-complete.*

FIXING WORD is in **coNP**; the certificate being a configuration x such that $M^w(x) \notin M(G)$. We shall prove that **FIXING WORD** is **coNP**-complete, even when restricted to permutations, in Section 5.3. As such, we omit the proof of Theorem 5.1.

5.1 Prefixing and suffixing words

The seminal observation is that if G is a graph, and w is a permutation of V , then ww fixes $M(G)$: for any initial configuration x , $M^w(x) \in I(G)$; then for any $y \in I(G)$, $M^w(y) \in M(G)$. We shall not prove this claim now, as we will prove stronger results in the sequel (see Propositions 5.3 and 5.5).

Following the seminal observation above, we say that w^p **prefixes** $M(G)$ if $M^{w^p}(x) \in I(G)$ for all $x \in \{0, 1\}^V$, and that w^s **suffixes** $M(G)$ if $M^{w^s}(y) \in M(G)$ for all $y \in I(G)$. In that case, for any word ω , $w^p\omega$ also prefixes $M(G)$ and ωw^s also suffixes $M(G)$. Clearly, if $w = w^p w^s$, where w^p prefixes $M(G)$ and w^s suffixes $M(G)$, then w fixes $M(G)$. We can be more general, as shown below.

Proposition 5.2. *If $w = w_{1:l}$ where $w_{1:a}$ prefixes $M(G)$, $w_{b:l}$ suffixes $M(G)$, and $[w_{b:a}]$ is an independent set of G for some $0 \leq a, b \leq l$, then w fixes $M(G)$.*

Proof. First, suppose $a < b$, so that $w = w_1 \dots w_a \dots w_b \dots w_l$. As mentioned above, $w^p = w_{1:b-1}$ prefixes $M(G)$ and $w^s = w_{b:w_l}$ suffixes $M(G)$, hence $w = w^p w^s$ fixes $M(G)$.

Second, suppose $a \geq b$, so that $w = w_1 \dots w_b \dots w_a \dots w_l$. It is easily seen that for any two non-adjacent vertices u and v , $M^{vv} = M^v$ and $M^{uv} = M^{vu}$. As such,

$$M^w = M^{w_1 \dots w_b \dots w_a \dots w_l} = M^{w_1 \dots w_b w_b \dots w_a w_a \dots w_l} = M^{w_1 \dots w_b \dots w_a w_b \dots w_a \dots w_l},$$

and again if we let $w^p = w_{1:a}$ and $w^s = w_{b:l}$, we have $M^w = M^{w^p w^s}$, hence w fixes $M(G)$. \square

We now characterise the words that prefix (or suffix) the MIS network. Interestingly, those properties depend only on $[w]$. Also, while deciding whether a word prefixes the MIS network is computationally tractable, deciding whether a word suffixes the MIS network is computationally hard as it is based on the **NON-DISTRICT** problem.

Proposition 5.3. *Let G be a graph. Then the word w prefixes $M(G)$ if and only if $[w]$ is a vertex cover of G .*

Proof. Suppose $[w]$ is a vertex cover of G and that $y = M^w(x) \notin I(G)$, i.e. $y_{uv} = 11$ for some edge uv of G . Without loss, let the last update in $\{u, v\}$ be v , i.e. there exists a such that $w_{a+1} = v$ and $w_b \notin \{u, v\}$ for all $b > a + 1$. We obtain $y_u^a = y_u = 1$ hence $y_v = y_v^{a+1} = 0$, which is the desired contradiction.

Conversely, if $[w]$ is not a vertex cover, then there is an edge $uv \in E$ such that $[w] \cap \{u, v\} = \emptyset$. Therefore, if we take $x = \chi(\{u, v\})$ then $x_{uv} = 11$ and we have $y_{uv} = 11$ as well. \square

PREFIXING WORD

Input: A graph $G = (V, E)$ and a word w .

Question: Does w prefix $M(G)$?

Corollary 5.4. PREFIXING WORD is in **P**.

Proposition 5.5. Let G be a graph. Then the word w suffixes $M(G)$ if and only if $[w]$ is a non-district of G .

Proof. Suppose $[w]$ is a district of G , i.e. there exists an independent set I and a vertex $v \notin [w]$ such that $W = [w] \cap N(v)$ is dominated by I . Let $x = \chi(I)$ (in particular, $x_v = 0$), and let $y = M^w(x)$. Then for any $u \in W$, u has a neighbour in I , hence $y_u = 0$; thus $y_{N[v]} = 0$ and w does not suffix M .

Conversely, suppose w does not suffix $M(G)$, i.e. there exists $x \in I(G)$ and $v \in V$ such that $y = M^w(x)$ with $y_{N[v]} = 0$. By Proposition 4.2, $y \in I(G)$ and $y \geq x$, hence $x_{N[v]} = 0$. Let $W = [w] \cap N(v)$ and $I = \mathbf{1}(y) \cap N(W)$; we note that I is an independent set. For each $u \in W$, we have $y_u = 0$ hence there exists $i \in V$ such that $y_i = 1$ and $u \in N(i)$, and hence $i \in I$. Therefore, $W \subseteq N(I)$ and W is a constituency of $G - v$, i.e. $[w]$ is a district of G . \square

SUFFIXING WORD

Input: A graph $G = (V, E)$ and a word w .

Question: Does w suffix $M(G)$?

Corollary 5.6. SUFFIXING WORD is **coNP**-complete.

Proof. This immediately follows from Theorem 3.3. \square

5.2 Fixing sets

Some graphs have fixing words that do not visit all vertices. For instance, if $G = K_n$ is the complete graph with vertices v_1, \dots, v_n , then it is easily shown that $w = v_1 \dots v_{n-1}$ is a fixing word for the MIS network. In general, we say that a set S of vertices of G is a **fixing set** of G if there exists a word w with $[w] = S$ that fixes $M(G)$.

We first characterise the fixing sets of graphs. Interestingly, those are the same sets S such that ww is a fixing word of $M(G)$ for any permutation w of S .

Proposition 5.7. Let S be a subset of vertices of G . The following are equivalent.

1. S is a fixing set of $M(G)$, i.e. there exists a fixing word w of $M(G)$ with $[w] = S$.
2. For all words w^p, w^s such that $[w^p] = [w^s] = S$, the word $w^p w^s$ fixes $M(G)$.
3. S is a vertex cover and a non-district.

Proof. $1 \implies 3$. Since w prefixes $M(G)$, $S = [w]$ is a vertex cover by Proposition 5.3; similarly, since w suffixes $M(G)$, $S = [w]$ is a non-district by Proposition 5.5.

$3 \implies 2$. Since S is a vertex cover, then by Proposition 5.3 w^p prefixes $M(G)$; similarly, by Proposition 5.5 w^s suffixes $M(G)$. Therefore, $w^p w^s$ fixes $M(G)$.

$2 \implies 1$. Trivial. \square

The FIXING SET problem asks, given a graph G and a set of vertices S , if S is a fixing set of G . In other words, it asks whether the vertices outside of S can be skipped by some fixing word.

FIXING SET

Input: A graph $G = (V, E)$ and a set $S \subseteq V$.

Question: Is S a fixing set of $M(G)$?

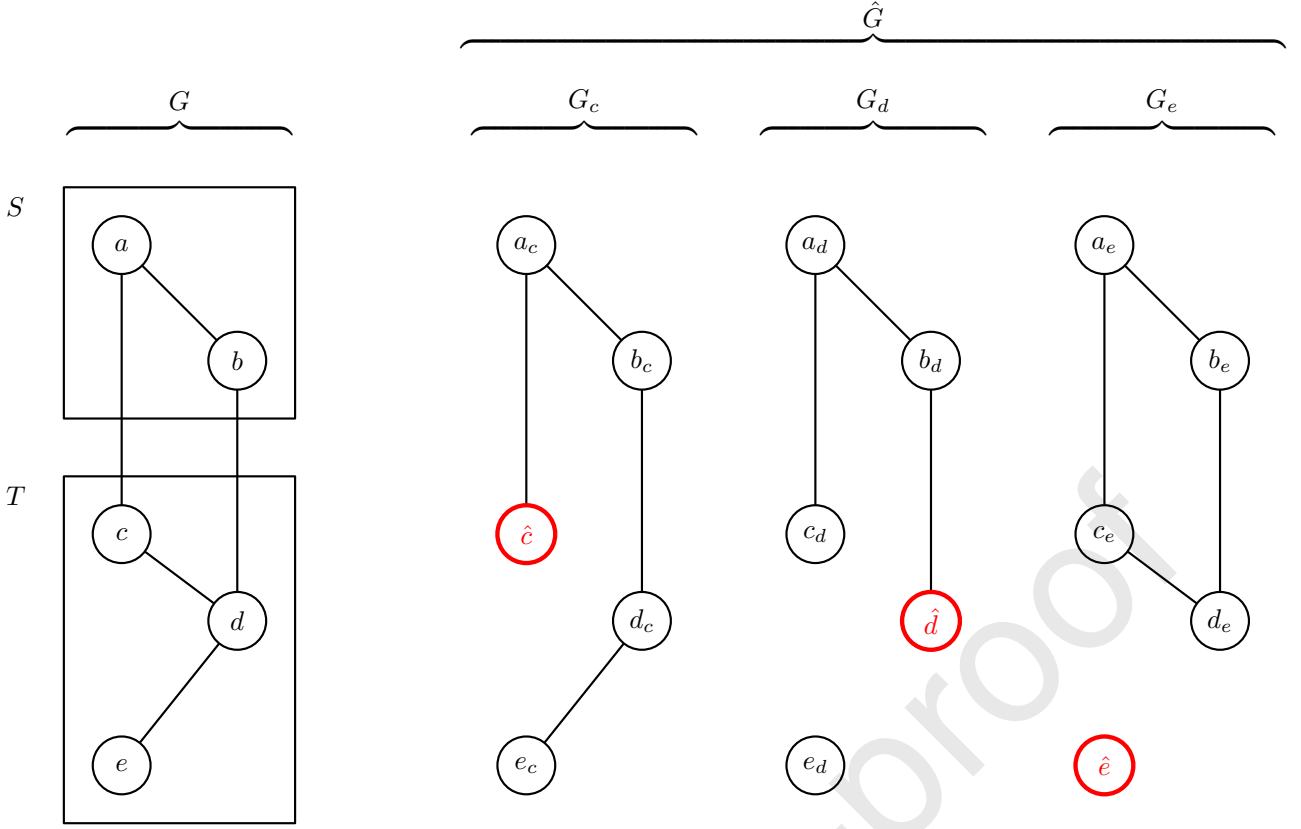


Figure 5: Example reduction from a no-instance of NON-DISTRICT (G, S) to the corresponding no-instance of FIXING SET (\hat{G}, \hat{S}) , with $\hat{S} = V_c \cup V_d \cup V_e$.

Theorem 5.8. FIXING SET is **coNP**-complete.

Proof. Membership of **coNP** is known: the no-certificate is a permutation w of S and an initial configuration $x \in \{0, 1\}^V$ such that $M^{ww}(x) \notin M(G)$ (by Proposition 5.7).

The hardness proof is by reduction from NON-DISTRICT, which is **coNP**-complete, as proved in Theorem 3.3. Let (G, S) be an instance of NON-DISTRICT, and construct the instance (\hat{G}, \hat{S}) of FIXING SET as follows.

Let $G = (V, E)$ and $T = V \setminus S$. For any $t \in T$, let $G_t = (V_t \cup \{\hat{t}\}, E_t)$ be the graph defined as follows: $V_t = \{u_t : u \in V \setminus t\}$ is a copy of all the vertices apart from t , which is replaced by a new vertex $\hat{t} \notin V_t$, and $E_t = \{a_t b_t : ab \in E, a, b \neq t\} \cup \{s_t \hat{t} : st \in E, s \in S\}$ is obtained by removing the edges between \hat{t} and the rest of T . Then G is the disjoint union of all those graphs, i.e. $G = \bigcup_{t \in T} G_t$, while $\hat{S} = \bigcup_{t \in T} V_t$. For the sake of simplicity, we shall use the notation $A_t = \{u_t : u \in A\}$ for all $A \subseteq V \setminus \{t\}$. Our construction is illustrated in Figure 5.

By construction, $\hat{G} - \hat{S}$ is the empty graph on $\{\hat{t} : t \in T\}$, hence \hat{S} is a vertex cover of \hat{G} . All we need to show is that \hat{S} is a non-district of \hat{G} if and only if S is a non-district of G . We have that \hat{S} is a district of \hat{G} if and only if there exists \hat{t} and an independent set \hat{I} of $\hat{G} - \hat{t}$ such that $W = \hat{S} \cap N(\hat{t}; \hat{G}) = (S \cap N(t; G))_t$ is contained in $N(\hat{I}; \hat{G})$. We have $\hat{I} \cap V_t = I_t$ for some independent set I of G . Since $W \subseteq V_t$ and $N(W; \hat{G} - \hat{t}) \subseteq V_t$, we have $W \subseteq N(\hat{I} \cap V_t; \hat{G}) \cap V_t = N(I; G)_t$, which is equivalent to S being a district of G . \square

Clearly, if S is a fixing set of $M(G)$, then every superset of S is also a fixing set. Moreover, every graph G has a trivial fixing set, namely V . The NON-TRIVIAL FIXING SET asks whether G has any other fixing set. Equivalently, it asks whether any vertex can be skipped by a fixing word.

NON-TRIVIAL FIXING SET

Input: A graph G .

Question: Does there exist a fixing set $S \neq V$ of G ?

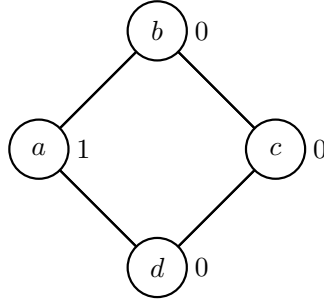


Figure 6: C_4 is a no-instance of the NON-TRIVIAL NON-DISTRICT problem, and hence also a no-instance of NON-TRIVIAL FIXING SET. For any word w with $[w] = \{b, c, d\}$, $M^w(1000) = 1000 \notin M(C_4)$. By symmetry, no set of three vertices is a fixing set for $M(C_4)$.

Theorem 5.9. NON-TRIVIAL FIXING SET is **coNP**-complete.

Proof. We prove that G has a non-trivial fixing set if and only if it has a non-trivial non-district. If G has a non-trivial fixing set, then there exists $S \neq V$ which is a vertex cover and a non-district of G , hence S is a non-trivial non-district of G . Conversely, if G has a non-trivial non-district, then there exists v such that $S = V \setminus \{v\}$ is a non-district of G , in which case S is also a vertex cover, and hence a non-trivial fixing set of G .

The **coNP**-completeness of NON-TRIVIAL FIXING SET then follows Theorem 3.5. The connection between the two problems is illustrated in Figure 6. \square

5.3 Permisses

The MIS algorithm doesn't use any word w to update the state of each vertex, but instead restricts itself to w being a permutation of V . As such, we now focus on permutations and we call a permutation of V that fixes $M(G)$ a **permis** of G . The PERMIS decision problem is equivalent to the FIXING WORD problem, restricted to permutations.

PERMIS

Input: An undirected graph $G = (V, E)$ and a permutation w of V .

Question: Is w a permis of G ?

Let w be a permutation of V , then w naturally induces a linear order on V , whereby $w_i \succ w_j$ whenever $i < j$, i.e. w_i is updated before w_j . Then consider the orientation of G induced by w : $G^w = (V, E^w)$ with $E^w = \{(u, v) : uv \in E, u \succ v\}$. We see that G^w is acyclic, and that conversely any acyclic orientation of G is given by some G^w . A simple application of [6, Theorem 1] shows that if w, w' are two permutations of V such that $G^w = G^{w'}$, then w is a permis if and only if w' is a permis.

We say that the vertex v is **covered** by w if for every $x \in \{0, 1\}^V$, $y_{N[v]} \neq 0$, where $y = M^w(x)$. Thus, w is a permis if and only if w covers all vertices.

COVERED VERTEX

Input: A graph $G = (V, E)$, a permutation w of V and a vertex $v \in V$.

Question: Is v covered by w ?

We now give a sufficient condition for a vertex to be covered. Let G be a graph, H be an orientation of G , and let t and v be vertices of G . We say t is **transitive** if for all $a, b \in V$, $t \rightarrow a \rightarrow b$ implies $t \rightarrow b$ in G^w . We say v is **near-transitive** if there exists a transitive vertex t such that $N[t; G] \subseteq N[v; G]$.

Lemma 5.10. If v is a near-transitive vertex of G^w , then v is covered by w .

Proof. First, suppose $v = t$ is transitive. Let x such that $y_{N[t; G]} = 0$. We shall repeatedly use the fact that for any vertex u , if $y_{N^{\text{in}}[u; G^w]} = 0$, then there exists $u' \in N^{\text{out}}(u; G^w) \cap \mathbf{1}(x)$, i.e. $x_{u'} = 1$ and $u \rightarrow u'$ (u is updated after u').

Since $y_{N^{\text{in}}[t;G^w]} = 0$, there exists $a \in N^{\text{out}}(t;G^w) \cap \mathbf{1}(x)$. Without loss let a be the last vertex of this kind to be updated: if $a' \neq a$ satisfies $a' \in N^{\text{out}}(t;G^w) \cap \mathbf{1}(x)$, then $a' \rightarrow a$. Again, since $x_a = 1$, we have $y_{N^{\text{in}}(a;G^w)} = 0$; and since $y_a = 0$ as well, there exists $b \in N^{\text{out}}(a;G^w) \cap \mathbf{1}(x)$. Since $t \rightarrow a \rightarrow b$ and by transitivity of t , we obtain $t \rightarrow b$, but then $b \in N^{\text{out}}(t;G^w) \cap \mathbf{1}(x)$ and hence $b \rightarrow a$, which is the desired contradiction.

Second, suppose that t is transitive (and hence, as shown above, covered) and that $N[t;G] \subseteq N[v;G]$. For all x we have $y_{N[t;G]} \neq 0$, and hence $y_{N[v;G]} \neq 0$, thus v is also covered by w . \square

Theorem 5.11. *PERMIS is coNP-complete.*

Proof. Membership of **coNP** is known: the no-certificate is a configuration x such that $y = M^w(x) \notin M(G)$.

The hardness proof is by reduction from NON-CONSTITUENCY, which is **coNP**-complete by Corollary 3.2. Let (G, S) be an instance of NON-CONSTITUENCY of empty type and construct the instance (\hat{G}, w) of PERMIS as follows. Let $T = V \setminus S$ and $T' = \{t' : t \in T\}$ be a copy of T . Then let \hat{G} be the graph with vertex set $\hat{V} = \{v, a, b\} \cup V \cup T'$, and with edges $\hat{E} = E \cup \{sv : s \in S\} \cup \{va, ab\} \cup \{tt' : t \in T\}$. Let w be a permutation of \hat{V} such that $v \succ a \succ b \succ T \succ T' \succ S$. This is illustrated in Figure 7.

We claim that w is a permis of \hat{G} if and only if S is not a constituency of G . Firstly, the vertices in $S \cup T' \cup \{b\}$ are all transitive and hence the vertices in $T \cup \{a\}$ are near-transitive. Therefore, w is a permis if and only if v is covered. We prove that v is covered if and only if S is not a constituency of G .

If S is a constituency of G , then let $I \subseteq T$ be a maximal independent set of G (and hence an independent set of \hat{G} as well) such that $S \subseteq N(I)$. Let $x = \chi(I \cup \{a, b\})$. Then $y_v = 0$ (because $x_a = 1$), $y_a = 0$ (because $x_b = 1$), $y_I = 1$ and $y_S = 0$ (because for any vertex u , if $x_u = 1$ and $x_{N(u)} = 0$, then $y_u = 1$ and $y_{N(u)} = 0$). Thus $y_{N[v]} = 0$.

Conversely, if $y_{N[v]} = 0$, then for any $s \in S$, $y_{N(s)} \neq 0$. Since $y_v = 0$, there is $t \in T$ such that $ts \in E$ and $y_t = 1$. Therefore, the set $\mathbf{1}(y) \cap T$ is an independent set that dominates S , i.e. S is a constituency of G . \square

The proof of Theorem 5.11 also settles the complexity of COVERED VERTEX.

Theorem 5.12. *COVERED VERTEX is coNP-complete.*

6 Permissible and non-permissible graphs

We say that G is **permissible** if it has a permis. As we shall see, not all graphs are permissible. In this subsection, we exhibit permissible and non-permissible graphs, and we prove that deciding whether a graph is permissible is computationally hard.

We classified (non-)permissible graphs by computer search, using **nauty's geng** utility (<https://doi.org/10.1016/j.jsc.2013.09.003>) to exhaustively generate connected graphs up to 9 vertices. Full results are available at <https://github.com/dave-ck/MISMax/>. Here are some highlights. Of 273194 connected graphs on at most nine vertices, only 432 are non-permissible; the heptagon C_7 , 13 8-vertex graphs (including the perfect graph shown in Figure 8), and 418 9-vertex graphs. The Petersen graph is also non-permissible.

We prove the graph in Figure 8 is perfect as follows. First note that four vertices in the graph have degree 3 and four vertices in the graph have degree 5. The absence of an induced C_5 can be verified manually. There is no induced C_7 : any subgraph on seven vertices includes at least one vertex formerly of degree 5 hence of degree at least 4 in the induced subgraph. Similarly, there is no induced $\overline{C_7}$; any subgraph on seven vertices includes at least one vertex formerly of degree 3 and hence of degree at most 3 in the induced subgraph ($\overline{C_7}$ is 4-regular).

6.1 Permissible graphs

We now exhibit large classes of permissible graphs.

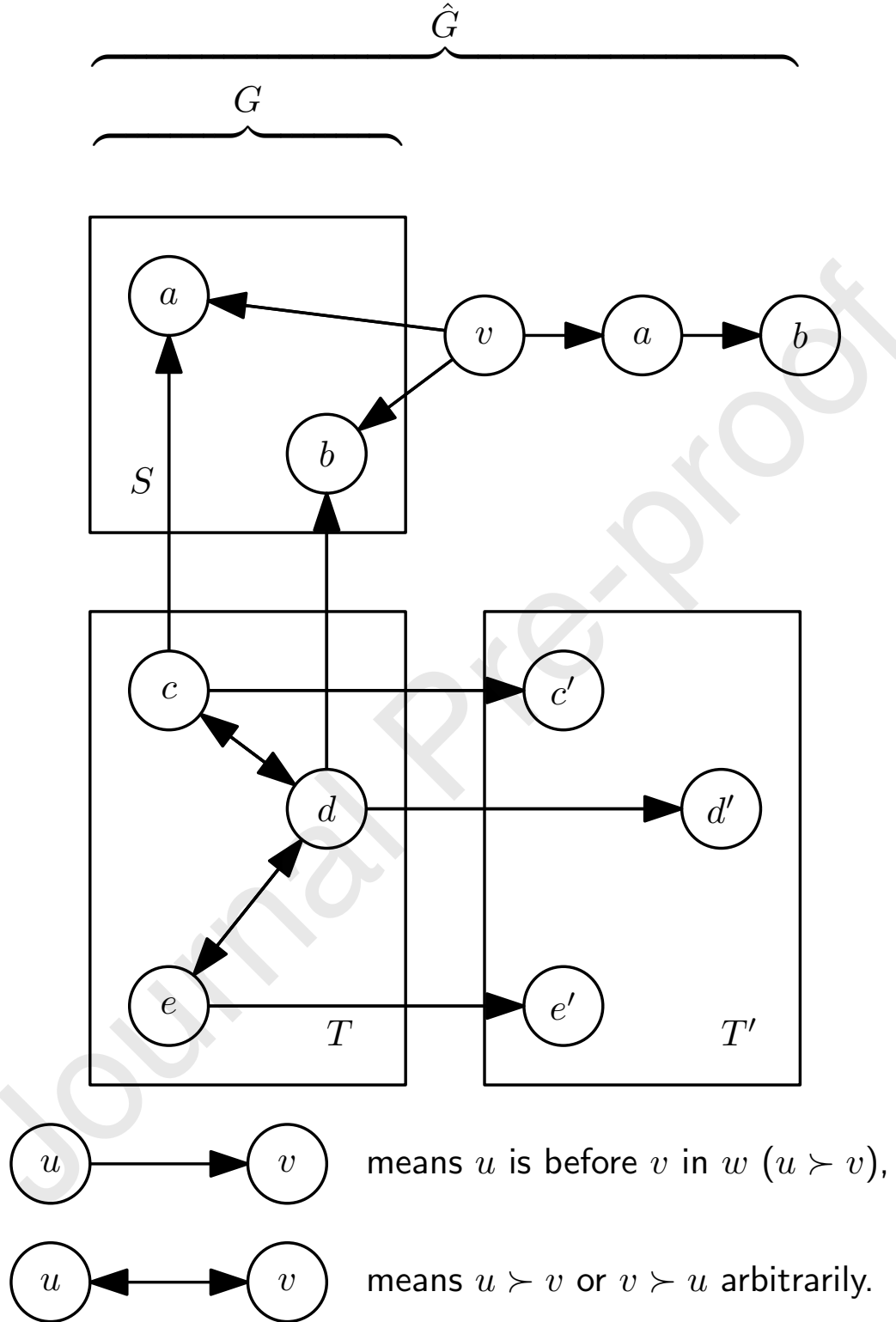


Figure 7: Illustration of the reduction from NON-CONSTITUENCY to PERMIS.

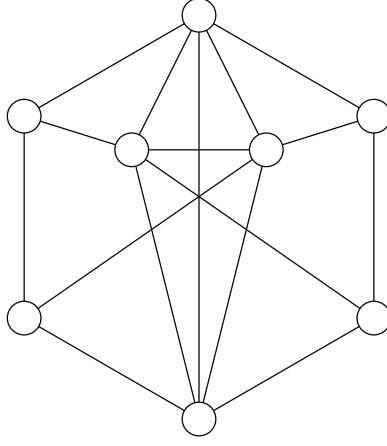


Figure 8: An 8-vertex perfect graph with no permis.

An orientation of G is **transitive** (**near-transitive**, respectively) if all the vertices are transitive (near-transitive, respectively). Any transitive orientation is necessarily acyclic. A graph that admits a transitive orientation is called a **comparability graph**. The following are comparability graphs: complete graphs, bipartite graphs, permutation graphs, cographs, and interval graphs. Accordingly, we say that a graph that admits a near-transitive orientation is a **near-comparability graph**. Lemma 5.10 immediately yields the permissibility of near-comparability graphs.

Proposition 6.1. *All near-comparability graphs are permissible.*

We now give a characterisation of near-comparability graphs below. Recall that the vertex m is a benjamin of G if there is no vertex v with $N^{\text{in}}[v] \subset N^{\text{in}}[m]$ and that G_B is the subgraph of G induced by its benjamins.

Theorem 6.2. *Let G be a graph. The following are equivalent:*

1. G is a near-comparability graph, i.e. it admits a near-transitive orientation;
2. G admits a near-transitive acyclic orientation;
3. G_B is a comparability graph.

Proof. 1 \implies 2. Let H be a near-transitive orientation of G . Let T be the set of transitive vertices of H . Then T is disjoint from all the cycles in H (for if $t \in T$ belongs to the cycle $t \rightarrow v_1 \rightarrow \dots \rightarrow v_k \rightarrow t$, we must have $t \rightarrow v_k$ by transitivity), and in particular $H[T]$ is acyclic. Construct any acyclic orientation of G , say G^w , such that $G^w[T] = H[T]$ and $V \setminus T \succ T$. Note such an orientation can be constructed efficiently (for example by a greedy algorithm). Then any vertex in T is still transitive in G^w , and hence G^w is near-transitive.

2 \implies 3. Let G be a near-comparability graph. In order to prove that G_B is a comparability graph, we need to consider the graph $\langle G \rangle$ obtained by only keeping one twin out of every set of twins. More formally, for any $v \in V(G)$, let $\langle v \rangle = \{u \in V : N[u] = N[v]\}$ be the equivalence class of v . We further denote $\langle V \rangle = \{\langle v \rangle : v \in V\}$, $\langle E \rangle = \{\langle v \rangle \langle v' \rangle : vv' \in E\}$. Then $\langle G \rangle = (\langle V \rangle, \langle E \rangle)$.

Say that a graph is **closed twin-free** if there are no closed twins, i.e. $N[u] \neq N[v]$ for all $u \neq v \in V$.

Claim 2. $\langle G \rangle$ is closed twin-free.

Proof. Suppose $N[\langle u \rangle; \langle G \rangle] = N[\langle v \rangle; \langle G \rangle]$. Then $N[u; G] = \bigcup_{\langle a \rangle \in N[\langle u \rangle; \langle G \rangle]} \langle a \rangle = N[v; G]$, hence u and v are closed twins in G and $\langle u \rangle = \langle v \rangle$. \square

Claim 3. $\langle G \rangle$ is a near-comparability graph.

Proof. Suppose G^w is a near-transitive acyclic orientation of G . Then for every equivalence class $c \in \langle V \rangle$, $G^w[c]$ is a transitive tournament with a unique source. Consider the orientation H of $\langle G \rangle$ naturally induced by w , i.e. $\langle u \rangle \rightarrow \langle v \rangle$ in $\langle G \rangle$ if and only if $u \rightarrow v$, where u and v are the unique sources of $G^w[\langle u \rangle]$ and $G^w[\langle v \rangle]$, respectively. We now prove that all vertices of $\langle G \rangle$ are near-transitive in H . Firstly, if t is transitive in G^w , then $\langle t \rangle$ is transitive in H . Secondly, if v satisfies $N[v; G] \supseteq N[t; G]$ for some transitive t , then $N[\langle v \rangle; \langle G \rangle] \supseteq N[\langle t \rangle; \langle G \rangle]$ and hence $\langle v \rangle$ is near-transitive in H . \square

Claim 4. $\langle G \rangle_B$ is a comparability graph.

Proof. Let $\langle G \rangle^w$ be a near-transitive acyclic orientation of $\langle G \rangle$ and let $B = B(\langle G \rangle)$. If $m \in B$, then there exists a transitive vertex t such that $N[m] \supseteq N[t]$. Since $m \in B$ we have $N[m] = N[t]$ and since $\langle G \rangle$ is closed twin-free we obtain $m = t$, i.e. m is transitive in $\langle G \rangle^w$. Therefore, m is also transitive in $\langle G \rangle^w[B] = \langle G \rangle_B^{w[B]}$. Thus, $\langle G \rangle_B^{w[B]}$ is a transitive orientation of $\langle G \rangle_B$. \square

Claim 5. G_B is a comparability graph.

Proof. We first remark that $\langle u \rangle \in B(\langle G \rangle)$ if and only if $u \in B(G)$. Now, let $\langle G \rangle_B^{w'}$ be a transitive orientation, then consider the orientation G_B^w of G_B as follows. First, fix an arbitrary order of every equivalence class $\langle v \rangle$. Second, orient $u \rightarrow v$ if $\langle u \rangle \rightarrow \langle v \rangle$ in $\langle G \rangle_B^{w'}$.

We now verify that this orientation is transitive. If $u \rightarrow a \rightarrow b$ in G_B^w , then $\langle u \rangle \rightarrow \langle a \rangle \rightarrow \langle b \rangle$ in $\langle G \rangle_B^{w'}$, hence $\langle u \rangle \rightarrow \langle b \rangle$ in $\langle G \rangle_B^{w'}$, and finally $u \rightarrow b$ in G_B^w . \square

3 \implies 1. Construct the orientation of G as follows. First, use the transitive orientation on G_B . Second, orient every edge $v \rightarrow m$ where $v \notin B(G)$ and $m \in B(G)$. Third, use any orientation on $G - B(G)$. Then the vertices in $B(G)$ remain transitive, and for any $v \notin B(G)$, there exists $m \in B(G)$ such that $N[m; G] \subseteq N[v; G]$, i.e. v is near-transitive. \square

Recognising comparability graphs can be done in polynomial time; see [23] and references therein. In fact, the algorithm in [23] not only decides whether a graph is a comparability graph, but it also produces a transitive orientation if such exists. In view of Theorem 6.2, applying that algorithm to G_B not only decides whether a graph is a near-comparability graph, but it also produces a near-transitive orientation (see the proof of Claim 5) if one exists.

Any induced subgraph of a comparability graph is a comparability graph. However, as we shall prove below, any graph is the induced subgraph of some near-comparability graph. Thus, Proposition 6.1 shows that every graph is the induced subgraph of a permissible graph. This entails that the class of permissible graphs is not hereditary, i.e. it is impossible to characterize permissible graphs by some forbidden induced subgraphs.

Corollary 6.3. *For every graph G , there exists a near-transitive (and hence permissible) graph H such that G is an induced subgraph of H .*

Proof. Let $G = (V, E)$ and construct the graph $\hat{G} = (\hat{V}, \hat{E})$ so that $G = \hat{G}[V]$ as follows. Let $V' = \{v' : v \in V\}$ be a copy of V , $\hat{V} = V \cup V'$, and $\hat{E} = E \cup \{vv' : v \in V\}$. Then \hat{G} is a near-comparability graph: let w be a permutation of \hat{V} where $V \succ V'$, then \hat{G}^w is a near-transitive acyclic orientation of \hat{G} . By Proposition 6.1, H is permissible. \square

We now introduce an operation on graphs, that we call **graph composition**, that preserves permissibility. Let H be an n -vertex graph, G_1, \dots, G_n other graphs, then the composition $H(G_1, \dots, G_n)$ is obtained by replacing each vertex h of H by the graph G_h , and whenever $hh' \in E(H)$, adding all edges between G_h and $G_{h'}$. More formally, we have

$$\begin{aligned} V(G) &= \{v_h : h \in V(H), v \in V(G_h)\}, \\ E(G) &= \{v_h v_{h'} : hh' \in E(H), v \in V(G_h), v' \in V(G_{h'})\} \cup \{v_h v_h' : h \in V(H), vv' \in E(G_h)\}. \end{aligned}$$

This is illustrated in Figure 9.

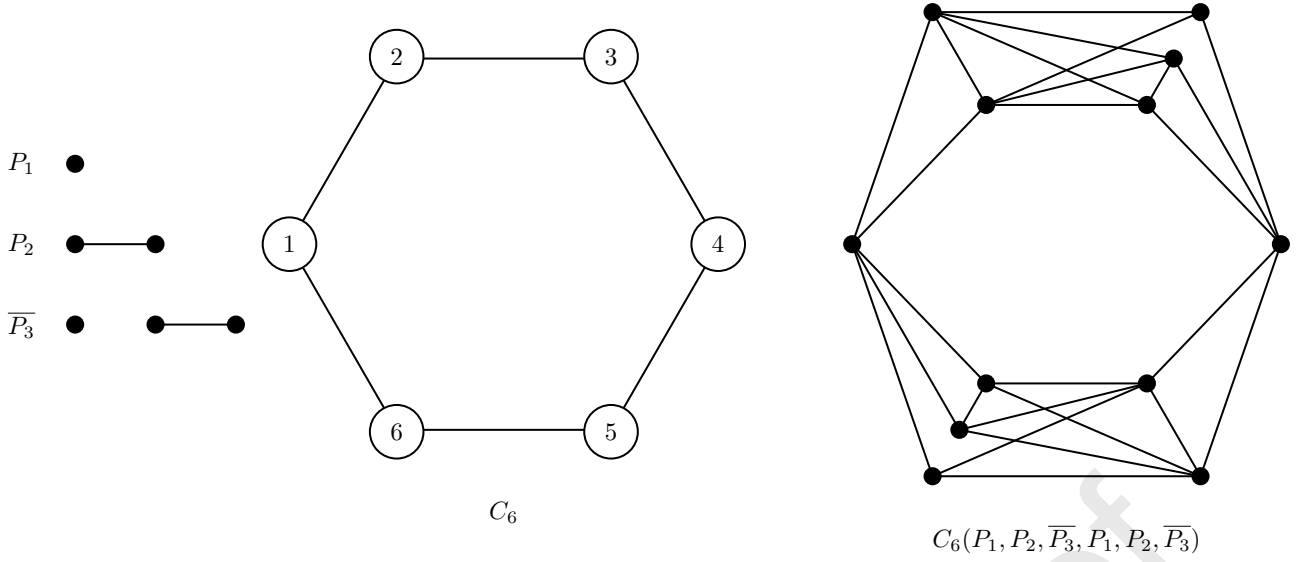


Figure 9: Illustration of the composition operation.

This construction includes for instance the disjoint union of two graphs: $G_1 \cup G_2 = \bar{K}_2(G_1, G_2)$ and the join of two graphs: $K_2(G_1, G_2)$. In the special case where only a single vertex b is replaced by the graph G_b , we use the notation

$$H(b, G_b) = H(K_1, \dots, K_1, G_b, K_1, \dots, K_1).$$

This special case includes adding an open twin (a new vertex b' with $N(b') = N(b)$): $H(b, \bar{K}_2)$ and adding a closed twin ($N[b'] = N[b]$): $H(b, K_2)$. In fact, any composition can be obtained by repeatedly replacing a single vertex.

Lemma 6.4. *Let $G = H(G_1, \dots, G_n)$ be a graph composition, where $V(H) = \{1, \dots, n\}$. For all $0 \leq i \leq n$, let G^i be defined as $G^0 = H$ and $G^i = G^{i-1}(i, G_i)$. Then $G^n = G$.*

Proof. The proof is by induction on $0 \leq i \leq n$. Let $I = \{1, \dots, i\}$ and $J = \{i+1, \dots, n\}$. We prove that

$$\begin{aligned} V(G^i) &= J \cup \{v_a : a \in I, v \in V(G_a)\}, \\ E(G^i) &= E(H[J]) \cup \{v_a v'_a : a \in I, vv' \in E(G_a)\} \\ &\quad \cup \{v_a v'_b : ab \in E(H[I])\} \cup \{hv_a : h \in J, a \in I, ha \in E(H)\}. \end{aligned}$$

This is clear for $i = 0$, and for $i \geq 1$ this is easily verified from the recurrence property:

$$\begin{aligned} V(G^i) &= (V(G^{i-1}) \setminus \{i\}) \cup \{v_i : v \in V(G_i)\} \\ E(G^i) &= E(G^{i-1} - i) \cup \{v_i v'_i : vv' \in E(G_i)\} \cup \{hv_i : v \in V(G_i), hi \in E(H)\}. \end{aligned}$$

The details are omitted. \square

Proposition 6.5. *If $G = H(G_1, \dots, G_n)$, where each of H, G_1, \dots, G_n is permissible, then G is permissible.*

Proof. According to Lemma 6.4, we only need to prove the case where $G = H(b, G_b)$, where the vertices are sorted according to a permis $\hat{w} = \hat{w}_{1:b-1} \hat{w}_b \hat{w}_{b+1:n}$ of H . We denote the vertex set of G_b as V_b , and we let w^b be a permis of G_b . Then we claim that $w = \hat{w}_{1:b-1} w^b \hat{w}_{b+1:n}$ is a permis of G .

For any configuration x of G , let \hat{x} be the configuration of H such that $\hat{x}_u = x_u$ for all $u \neq \hat{w}_b$ and $\hat{x}_{\hat{w}_b} = \bigvee_{v \in V_b} x_v$. We then prove that $y \in M(G)$ by considering the three main steps of w .

- Step 1: before the update of G_b ($\hat{w}_{1:b-1}$).

In Step 1, the initial configuration is x and the final configuration is $\alpha = y^{b-1}$. It is easy to show that for any $1 \leq a < b$, we have $M^{w_{1:a}}(x; G)_{G-V_b} = M^{\hat{w}_{1:a}}(\hat{x}; H)_{H-\hat{w}_b}$. We obtain $\hat{\alpha} = M^{\hat{w}_{1:b-1}}(\hat{x}; H)$.

- Step 2: update of G_b (w^b).

In Step 2, the initial configuration is $\alpha = y^{b-1}$ and the final configuration is $\beta = y^{b-1+|V_b|}$. Note that V_b is a tethered set of G , so let $T = N(V_b; G) \setminus V_b$. If $\alpha_T \neq 0$, then the whole of G_b will be updated to 0: $\beta_{V_b} = 0$. Otherwise, it is as if G_b is isolated from the rest of the graph and $\beta_{V_b} = M^{w^b}(\alpha_{V_b}; G_b)$. In either case, we have $\hat{\beta} = M^{\hat{w}_b}(\hat{\alpha}; H)$.

- Step 3: after the update of G_b ($\hat{w}_{b+1:n}$).

In Step 3, the initial configuration is $\beta = y^{b-1+|V_b|}$ and the final configuration is y . Again, we have for all $b < a \leq n$, $M^{w_{b+1:a}}(\beta; G)_{G-V_b} = M^{\hat{w}_{b+1:a}}(\hat{\beta}; H)_{H-\hat{w}_b}$. We obtain $\hat{y} = M^{\hat{w}_{b+1:n}}(\hat{\beta}; H)$.

We obtain

$$\begin{aligned} \hat{y} &= M^{\hat{w}_{b+1:n}}(\hat{\beta}; H) \\ &= M^{\hat{w}_{b+1:n}}(M^{\hat{w}_b}(\hat{\alpha}; H); H) \\ &= M^{\hat{w}_{b+1:n}}(M^{\hat{w}_b}(M^{\hat{w}_{1:b-1}}(\hat{x}; H); H); H) \\ &= M^{\hat{w}}(\hat{x}; H). \end{aligned}$$

We can now prove that $y_{N[v;G]} \neq 0$ for every vertex v of G . First, if v is not a vertex of G_b , then $\hat{y}_{N[v;H]} \neq 0$, and hence $y_{N[v;G]} \neq 0$. Second, if $v = u_b$ is a vertex of G_b , then we need to consider two cases. Either $\hat{y}_b = 0$, in which case there exists $a \in N(b; H) \subseteq N(u_b; G)$ with $\hat{y}_a = y_a \neq 0$; or $\hat{y}_b = 1$, in which case $y_{V_b} = M^{w^b}(x_{V_b}; G_b) \in M(G_b)$ and in particular $y_{N[u_b;G]} \neq 0$. \square

6.2 Non-permissible graphs

We now exhibit classes of non-permissible graphs. As mentioned earlier, the smallest non-permissible graph is the heptagon; in fact, any odd hole with at least seven vertices is non-permissible.

Proposition 6.6. *For all $k \geq 3$, the odd hole C_{2k+1} is not permissible.*

Proof. Let w be a permutation of the vertex set of $G = C_{2k+1}$. We shall prove that if w is a permis there cannot be two consecutive arcs in G^w with the same direction; this shows that the direction of arcs must alternate, which is impossible because there is an odd number of arcs in the cycle. We do this by a case analysis on the arcs preceding those two consecutive arcs.

We consider six vertices a to f , where the first two arcs $a \leftarrow b \leftarrow c$ are in the same direction. The first case is where $c \leftarrow d$; in that case, if $x_{abc} = 111$, then $y_{bcd} = 000$ and hence c is not covered. This is shown in Figure 10, along with the other three cases. \square

We now give two ways to construct larger non-permissible graphs.

Recall that a set of vertices S is tethered if there is an edge st between any $s \in S$ and any $t \in T = N(S) \setminus S$.

Proposition 6.7. *Let G be a graph. If G has a tethered set of vertices S such that $G[S]$ has no permis, then G has no permis.*

Proof. Let w be any permutation of V and $\hat{w} = w[S]$. Let \hat{x} be a configuration of $G[S]$ which is not fixed by \hat{w} : $M^{\hat{w}}(\hat{x}; G[S]) \notin M(G[S])$. We first note that $\hat{x} \neq 0$ and that for all $0 \leq a \leq |\hat{w}|$, $M^{\hat{w}_{1:a}}(\hat{x}; G[S]) \neq 0$.

Let $T = N(S) \setminus S$ and $U = V \setminus (S \cup T)$ and $x = (x_S = \hat{x}, x_T = 0, x_U)$, where x_U is any partial configuration. We prove by induction on $0 \leq b \leq |w|$ that

$$y^b := M^{w_{1:b}}(x; G) = \left(y_S^b = M^{\hat{w}_{1:b'}}(\hat{x}; G[S]), y_T^b = 0, y_U^b \right),$$

where b' is defined by $[\hat{w}_{1:b'}] = S \cap [w_{1:b}]$. The base case $b = 0$ is clear. Suppose it holds for $b - 1$.

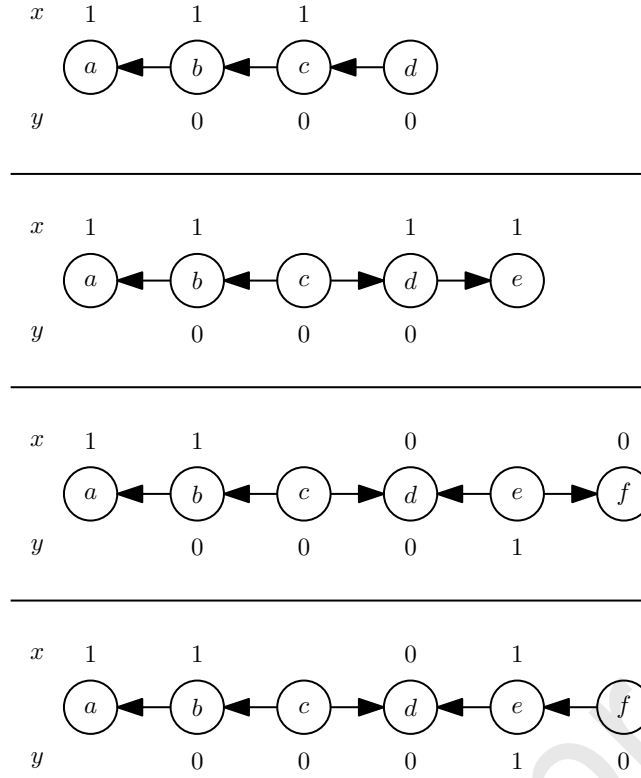


Figure 10: Illustration of Proposition 6.6.

- Case 1: $w_b \in S$.

Then $b' = (b - 1)' + 1$ and $w_b = \hat{w}_{b'}$. Since $y_T^{b-1} = 0$, we have

$$y_{w_b}^b = M(y^{b-1}; G)_{w_b} = M(y_S^{b-1}; G[S])_{w_b} = M(M^{\hat{w}_{1:b'-1}}(\hat{x}; G[S]); G[S])_{\hat{w}_{b'}} = M^{\hat{w}_{1:b'}}(\hat{x}; G[S])_{w_b},$$

and hence $y_S^b = M^{\hat{w}_{1:b'}}(\hat{x}; G[S])$.

- Case 2: $w_b \in T$.

Then $b' = (b - 1)'$. Since $y_S^{b-1} \neq 0$, we have $M(y^{b-1}; G)_{w_b} = 0$ and hence $y_T^b = 0$.

- Case 3: $w_b \in U$.

This case is trivial.

For $b = |w|$ we obtain $y = M^w(x; G) = (M^{\hat{w}}(\hat{x}; G[S]), 0, y_U)$, for which $y_S \notin M(G[S])$, and hence $y \notin M(G)$. \square

Propositions 6.6 and 6.7 yield perhaps the second simplest class of non-permissible graphs. The **wheel graph** is $W_{n+1} = K_2(C_n, K_1)$.

Corollary 6.8. *For all $k \geq 3$, the wheel graph W_{2k+2} is not permissible.*

Clearly, a graph is permissible if and only if all its connected components are permissible. In particular, for any G , the union $H = G \cup C_7$ is not permissible, but is disconnected. An interesting consequence of Proposition 6.7 is that permissibility of a connected graph cannot be decided by focusing on an induced subgraph, even if the latter has all but seven vertices of the original graph. Indeed, for any graph G , the join $H' = K_2(C_7, G)$ is not permissible, since the heptagon is tethered in H' .

Corollary 6.9. *Let G be a graph. Then there exists a connected non-permissible graph H' such that G is an induced subgraph of H' .*

Second, and unsurprisingly, we can construct larger non-permissible graphs by using a constituency.

Proposition 6.10. *Let G be a graph. If there exists $A \subseteq V$ such that $G[A]$ is not permissible and $S = N(A) \setminus A$ is a constituency of $G - A$, then G is not permissible.*

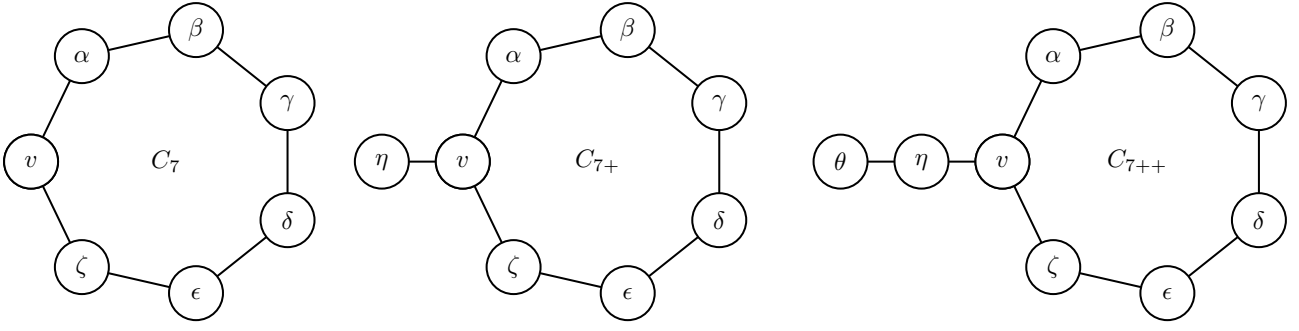


Figure 11: Graphs C_7 (non-permissible), C_{7+} (permissible), and C_{7++} (non-permissible).

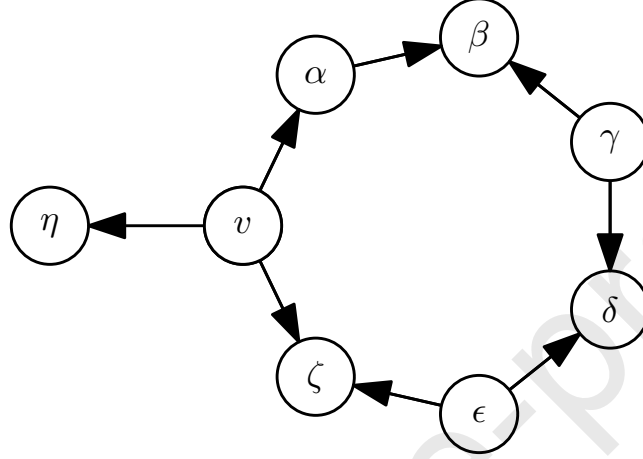


Figure 12: The permis of C_{7+} .

Proof. Let I be an independent set of $G - A$ such that $S \subseteq N(I)$. Let w be a permutation of V and $w[A]$ be the permutation of A induced by w . Then let $\dot{x} \in \{0, 1\}^A$ be a configuration such that $\dot{y} = M^{w[A]}(\dot{x}) \notin M(G[A])$. Then consider $x \in \{0, 1\}^V$ such that: $x_I = 1$, $x_A = \dot{x}$, $x_{V-A-I} = 0$. We then have $y_I = 1$, $y_S = 0$, and hence $y_A = \dot{y}$. Since $\dot{y} \notin M(G[A])$ and $y_S = 0$, we obtain that $y \notin M(G)$, i.e. w is not a permis of G . \square

For all $k \geq 3$ the odd hole C_{2k+1} is non-permissible. Consider the graph C_{2k+1+} by adding a vertex of degree one to C_{2k+1} ; as we shall see later it is permissible. Now add another vertex of degree one to the tail of C_{2k+1+} to obtain C_{2k+1++} . The graphs C_7 , C_{7+} and C_{7++} are illustrated in Figure 11. In C_{7++} , the vertex $S = \{\eta\}$ is a constituency and is the neighbourhood of the heptagon; therefore C_{7++} is not permissible. Obviously, this reasoning applies to all larger C_{2k+1++} as well.

Corollary 6.11. *For all $k \geq 3$, C_{2k+1++} is not permissible.*

6.3 The PERMISSIBLE decision problem

We now prove that deciding whether a graph is permissible is computationally hard.

PERMISSIBLE

Input: A graph G .

Question: Is G permissible?

As mentioned above, C_{7+} is permissible; the proof of Lemma 6.12 below can easily be generalised to show that C_{2k+1+} is permissible for all $k \geq 3$.

Lemma 6.12. *Any w such that C_{7+}^w is given on Figure 12 is a permis of C_{7+} .*

Proof. The vertices β , δ , ζ , and η are all transitive, and v is near-transitive, and hence these are covered. We only need to show that α , γ and ϵ are covered.

For α , suppose $y_{v\alpha\beta} = 000$. We then have the following chain of implications:

$$\begin{aligned} (y_\alpha = 0 \wedge y_\beta = 0) \wedge (\alpha \rightarrow \beta \leftarrow \gamma) &\implies y_\gamma = 1 \\ (y_\gamma = 1) \wedge (\gamma \rightarrow \beta) &\implies x_\beta = 0 \\ (x_\beta = 0 \wedge y_v = 0) \wedge (v \rightarrow \alpha \rightarrow \beta) &\implies y_\alpha = 1, \end{aligned}$$

which is the desired contradiction.

For γ , suppose $y_{\beta\gamma\delta} = 000$. We then have the following chain of implications:

$$\begin{aligned} (y_\beta = 0 \wedge y_\gamma = 0) \wedge (\alpha \rightarrow \beta \leftarrow \gamma) &\implies y_\alpha = 1 \\ (y_\alpha = 1) \wedge (\alpha \rightarrow \beta) &\implies x_\beta = 0 \\ (y_\gamma = 0 \wedge y_\delta = 0) \wedge (\gamma \rightarrow \delta \leftarrow \epsilon) &\implies y_\epsilon = 1 \\ (y_\epsilon = 1) \wedge (\delta \leftarrow \epsilon) &\implies x_\delta = 0 \\ (x_\beta = 0 \wedge x_\delta = 0) \wedge (\beta \leftarrow \gamma \rightarrow \delta) &\implies y_\gamma = 1, \end{aligned}$$

which is the desired contradiction.

For ϵ , suppose $y_{\delta\epsilon\zeta} = 000$. We then have the following chain of implications:

$$\begin{aligned} (y_\delta = 0 \wedge y_\epsilon = 0) \wedge (\gamma \rightarrow \delta \leftarrow \epsilon) &\implies y_\gamma = 1 \\ (y_\gamma = 1) \wedge (\gamma \rightarrow \delta) &\implies x_\delta = 0 \\ (y_\epsilon = 0 \wedge y_\zeta = 0) \wedge (\epsilon \rightarrow \zeta \leftarrow v) &\implies y_v = 1 \\ (y_v = 1) \wedge (\zeta \leftarrow v) &\implies x_\zeta = 0 \\ (x_\delta = 0 \wedge x_\zeta = 0) \wedge (\delta \leftarrow \epsilon \rightarrow \zeta) &\implies y_\epsilon = 1, \end{aligned}$$

which is the desired contradiction. □

Theorem 6.13. PERMISSIBLE is **coNP-hard**.

Proof. Reduction from NON-CONSTITUENCY. Let (G, S) be an instance of NON-CONSTITUENCY of empty type, and let $T = V \setminus S$. Let T' be a copy of T . Let A be another set of 7 vertices inducing a heptagon and let $v \in A$. Then consider the instance $\hat{G} = (\hat{V}, \hat{E})$ of PERMISSIBLE with

$$\begin{aligned} \hat{V} &= V \cup T' \cup A \\ \hat{E} &= E \cup \{tt' : t \in T\} \cup C_7(A) \cup \{sv : s \in S\}. \end{aligned}$$

This construction is illustrated in the left hand side of Figure 13.

We claim that (G, S) is a yes-instance of NON-CONSTITUENCY if and only if \hat{G} is a yes-instance of PERMISSIBLE. First, if S is a constituency of G , then S is a constituency of $\hat{G} - A$, and hence \hat{G} is not permissible by Proposition 6.10. Second, if S is not a constituency of G , let ω be the permis of C_{7+} given in Lemma 6.12 and consider the orientation \hat{G}^ω such that $T' \cup S$ are all sinks and $\hat{G}^\omega[A \cup \{s\}] = C_{7+}^\omega$ for all $s \in S$. (The edges of $\hat{G}[T]$ can be oriented arbitrarily.) This is illustrated in the right hand side of Figure 13.

The vertices in $T \cup T' \cup S$ are all near-transitive, and hence covered by w . We now prove that the vertices in $B = A \setminus \{v\}$ are also covered by w . Let $\dot{x} \in \{0, 1\}^{A \cup \{\eta\}}$ be given by $\dot{x}_a = x_a$ for all $a \in A$ and $\dot{x}_\eta = \bigwedge_{s \in S} \neg x_s$, and let $\dot{y} \in \{0, 1\}^{A \cup \{\eta\}} = \mathbf{M}^\omega(\dot{x}; C_{7+})$. Since $A \succ V \setminus A$ in \hat{G}^ω , we obtain $y_A = \mathbf{M}^w(x; \hat{G})_A = \mathbf{M}^\omega(\dot{x}; C_{7+})_A = \dot{y}_A$. For all $u \in B$, we have $N[u; \hat{G}] = N[u; C_{7+}]$ and since ω is a permis, $y_{N[u; \hat{G}]} = \dot{y}_{N[u; C_{7+}]} \neq 0$, thus u is covered by w . Therefore, the only vertex in contention is v . Suppose $y_v = 0$ and let $I = \mathbf{1}(y) \cap T$. Since I is an independent set of G and S is not a constituency of G , there exists $s \in S$ outside of the neighbourhood of I . Thus, $y_{N(s)} = 0$ and hence $y_s = 1$, which yields $y_{N[v]} \neq 0$. Thus v is covered, and w is a permis of \hat{G} . □

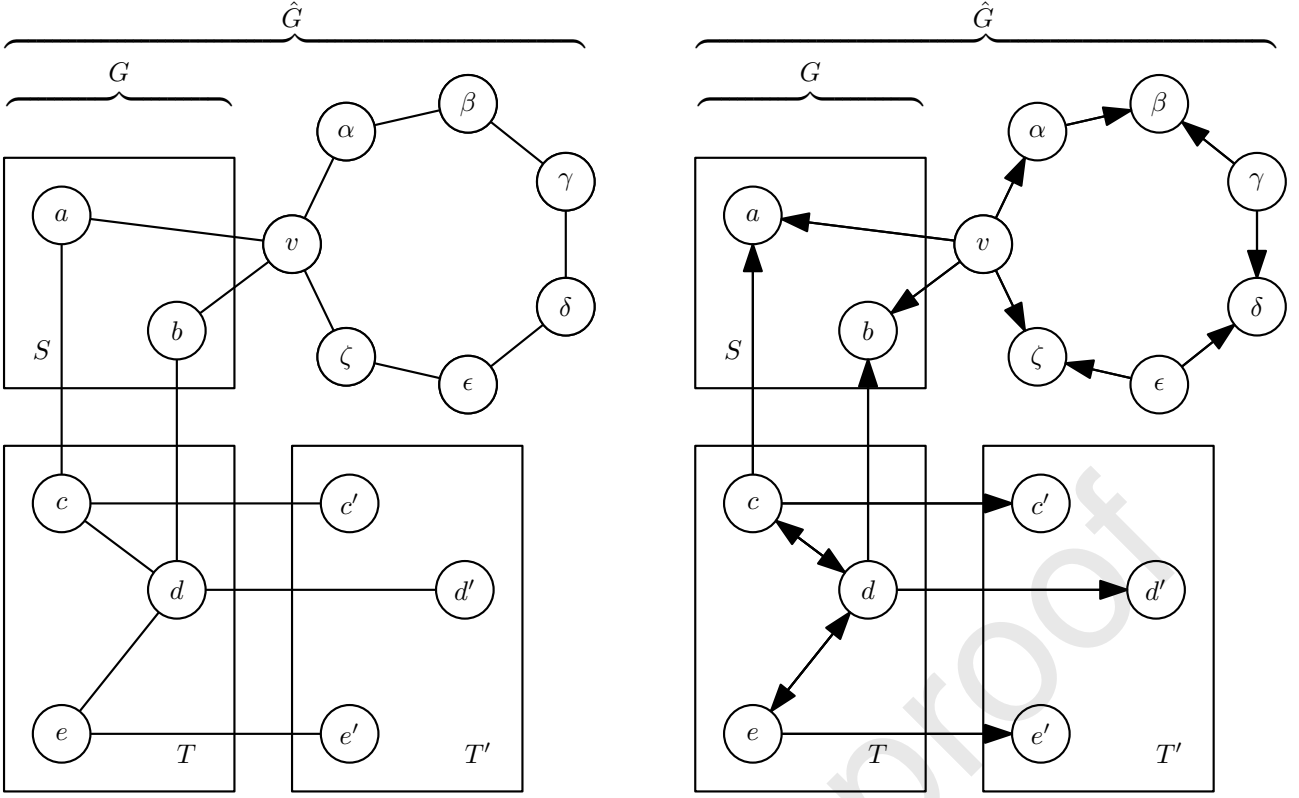


Figure 13: Illustration of the reduction from NON-CONSTITUENCY to PERMISSIBLE.

7 Extension to kernels in digraphs

The MIS network can be easily extended to digraphs as follows. The **kernel network**, denoted by $K(G)$, is defined by

$$K(x)_v = \bigwedge_{u \rightarrow v} \neg x_u,$$

where $K(x)_v = 1$ if $N^{\text{in}}(v) = \emptyset$. We then have $\text{Fix}(K(G)) = K(G)$.

7.1 Fixability of the kernel network

Henceforth we consider the fixability of the kernel network. We saw that \vec{C}_{2k+1} has no kernel, and hence it is not fixable. However, the existence of kernels does not guarantee fixability, as we shall prove in Proposition 7.1.

First, we give some sufficient conditions for a digraph to have a kernel and yet to not be fixable. Generalising the definition from the undirected case, we say a set of vertices S is **tethered** if there is an undirected edge st between any $s \in S$ and any $t \in T = N^{\text{in}}(S) \setminus S$. Note that necessarily $T \subseteq N^{\text{out}}(S)$ but possibly $N^{\text{out}}(S) \setminus T \neq \emptyset$.

Proposition 7.1. *If G has a tethered set S such that $K(G[S])$ is not fixable, then $K(G)$ is not fixable.*

We need two lemmas before giving the proof of Proposition 7.1.

Lemma 7.2. *If $x \in \{0, 1\}^V \neq 0$ and $x \mapsto_K y$, then $y \neq 0$.*

Proof. Suppose, for the sake of contradiction, that $y = K^w(x) = 0$ with $w = w_{1:l}$, while $y^{l-1} \neq 0$. Then $y_{N^{\text{in}}(w_l)}^{l-1} = 0$, hence $y_{w_l} = K(y^{l-1})_{w_l} = 1$, which is the desired contradiction. \square

Lemma 7.3. *If $K(G)$ is not fixable, then for any word w , there exists a nonzero configuration x of G such that $K^w(x) \notin K(G)$.*

Proof. For the sake of contradiction, suppose $K^w(x) \in K(G)$ for all nonzero x . We remark that $K(0)_{w_1} = 1$ hence $K^{w_1}(0) \neq 0$ and by Lemma 7.2, $K^w(0) \neq 0$. Therefore, $K^{ww}(0) \in K(G)$, and by hypothesis $K^{ww}(x) \in K(G)$ for any nonzero x , which shows that ww fixes $K(G)$. \square

Proof of Proposition 7.1. Partition the vertex set of G into three parts: S , $T = N^{\text{in}}(S) \setminus S$, and $U = V \setminus (S \cup T)$.

Let $w \in V^*$ be a word, then by Lemma 7.3 there exists a nonzero configuration \hat{x} of $G[S]$ such that $K^{w[S]}(\hat{x}; G[S]) \notin K(G[S])$. Let x be a configuration of G such that $x_S = \hat{x} \neq 0$ and $x_T = 0$. By induction on $0 \leq a \leq l$, we prove that $y_S^a \neq 0$ and $y_T^a = 0$. The claim is clear for $a = 0$, hence suppose it holds for $a - 1$. We consider three cases.

- Case 1: $w_a \in S$.
Since $y_T^{a-1} = 0$, we have $y_S^a = K^{w_a}(y^{a-1}; G)_S = K^{w_a}(y_S^{a-1}; G[S]) \neq 0$ by Lemma 7.2. Also, $y_T^a = y_T^{a-1} = 0$.
- Case 2: $w_a \in T$.
Since $y_S^{a-1} \neq 0$, we have $K(y^{a-1}; G)_{w_a} = 0$. Also, $y_S^a = y_S^{a-1} \neq 0$.
- Case 3: $w_a \in U$.
Then $y_S^a = y_S^{a-1} \neq 0$ and $y_T^a = y_T^{a-1} = 0$.

Therefore $y_S = K^{w[S]}(x_S; G[S])$ and hence $y_S \notin K(G[S])$. Since $y_T = 0$, this implies that $y \notin K(G)$. \square

In particular, if G has a tethered set S with $G[S] = \vec{C}_{2k+1}$ and $G - S$ is a graph, then G has a kernel (namely every maximal independent set of $G - S$) but $K(G)$ is not fixable.

We are now interested in the computational complexity of deciding whether the kernel network is fixable.

FIXABLE

Input: A digraph G .

Question: Is $K(G)$ fixable?

Theorem 7.4. **FIXABLE** is **coNP-hard**.

Proof. The proof is by reduction from **TAUTOLOGY**, which is **coNP-hard**. Let ϕ be a DNF with set of variables A , set of literals $B = \{\alpha, \neg\alpha : \alpha \in A\}$, and set of clauses Γ , so that ϕ can be expressed as $\phi = \bigvee_{\gamma \in \Gamma} \bigwedge_{\beta \in B_\gamma} \beta$ with $B_\gamma \subseteq B$ for all $\gamma \in \Gamma$. We construct a graph that has a vertex per literal, then builds a NOR-gate circuit for ϕ , and finally attaches the output of this circuit to a \vec{C}_3 . Intuitively, a 0 output of the circuit (which occurs whenever ϕ is not a tautology) will isolate the \vec{C}_3 , which is not fixable, while a 1 output will guarantee convergence.

Let $G = (V, E)$ with

$$V = B \cup \Gamma \cup \{\neg\phi, \phi, a, b, c\},$$

$$E = \{\alpha\neg\alpha : \alpha \in A\} \cup \{\beta \rightarrow \gamma : \neg\beta \in \gamma, \gamma \in \Gamma\} \cup \{\gamma \rightarrow \neg\phi : \gamma \in \Gamma\} \cup \{\neg\phi \rightarrow \phi \rightarrow a \rightarrow b \rightarrow c \rightarrow a\}.$$

This is illustrated in Figure 14.

If ϕ is a tautology, then let $w = w^B w^\Gamma \neg\phi\phi abc$, where w^B and w^Γ are any permutations of B and Γ , respectively. For each α , let $\alpha' = \{\alpha, \neg\alpha\}$. Then $G[\alpha']$ is a complete graph, and has $w[\alpha']$ as a permis. Therefore, at the end of w^B , we have $y_{\alpha'} = y_{\alpha'}^{|B|} \in \{01, 10\}$. Let $z \in \{0, 1\}^A = y_A^{|B|}$. The graph G then induces a circuit for ϕ , so that $y_\gamma = y_\gamma^{|B|+|\Gamma|} = \bigwedge_{\beta \in \gamma} \neg y_{\neg\beta}^{|B|} = \gamma(z)$ and $y_{\neg\phi} = \neg\phi(z)$ and eventually $y_\phi = \phi(z) = 1$. Finally, we obtain $y_{abc} = 010$, and it is easily verified that $y \in K(G)$.

If ϕ is not a tautology, let w be any word and construct the configuration x as follows. Let $z \in \{0, 1\}^A$ such that $\phi(z) = 0$ then let $x_\alpha = z_\alpha$, $x_{\neg\alpha} = \neg z_\alpha$, $x_\gamma = \gamma(z)$, $x_{\neg\phi} = \neg\phi(z)$, $x_\phi = \phi(z) = 0$ and x_{abc} can be chosen arbitrarily, since in a \vec{C}_3 every configuration is such that $K^{w[abc]}(x_{abc}) \notin K(G[abc])$. By construction, updating the value of any vertex outside of the \vec{C}_3 will not change anything, i.e. $y = K^{w[abc]}(x; G)$. Now, since $x_\phi = y_\phi = 0$, we have $K^{w[abc]}(x; G)_{abc} = K^{w[abc]}(x; G[abc]) \notin K(G[abc])$, and hence $y \notin K(G)$. \square

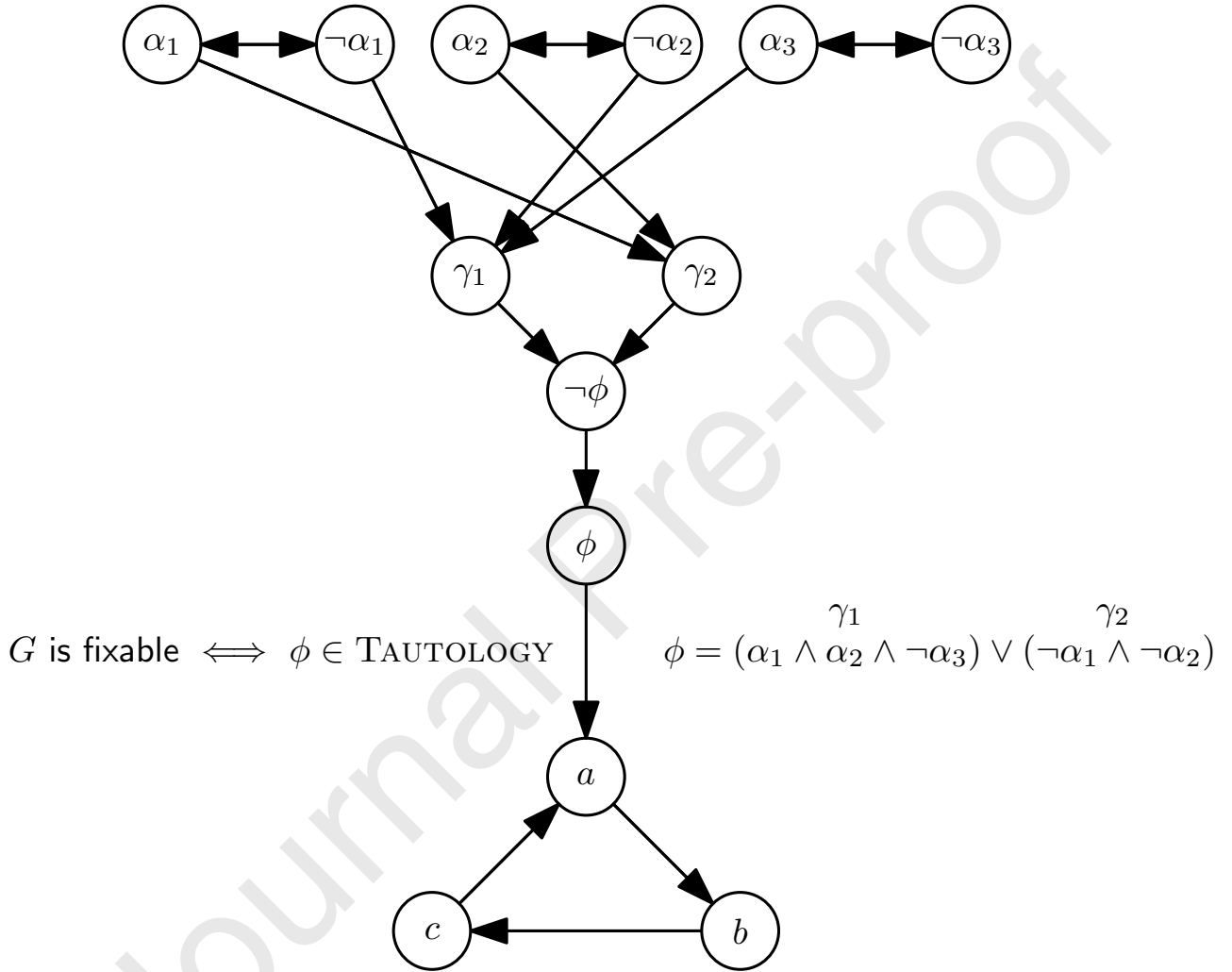


Figure 14: Illustration of the reduction from TAUTOLOGY to FIXABLE.

We now refine Theorem 7.4 by restricting ourselves to a very specific class of digraphs. For any $\epsilon > 0$, say a digraph G is ϵ -**fixable** if there is a word w that fixes at least a fraction $1 - \epsilon$ of configurations, i.e. $|\{x \in \{0, 1\}^V : K^w(x) \in K(G)\}| \geq (1 - \epsilon)2^n$.

Theorem 7.5. *For any $\epsilon > 0$, **FIXABLE** is **coNP**-hard for oriented ϵ -fixable digraphs of maximum in- and out-degree 2.*

Proof. The proof is by reduction from 3-TAUTOLOGY restricted to expressions where each literal appears at most twice, which is **coNP**-hard. We use the notation introduced in the proof of Theorem 7.4.

Let ϕ be a 3-TAUTOLOGY instance where each literal appears at most twice. Let Ω be an additional set of $\lceil -\log_2 \epsilon \rceil$ variables, and let

$$\psi = \phi \vee \bigvee \Omega.$$

Let G be the graph constructed in the proof of Theorem 7.4 for the expression ψ . Now replace each K_2 corresponding to a variable by a \tilde{C}_4 as follows. Let u be a variable in $A \cup \Omega$ and let $u' = \{u, \neg u, \dot{u}, \neg \dot{u}\}$ induce the cycle $u \rightarrow \neg u \rightarrow \dot{u} \rightarrow \neg \dot{u} \rightarrow u$. We finally adapt the NOR-gate circuit of ψ to have fan-in 2. The new graph \hat{G} is illustrated in Figure 15.

By construction, \hat{G} is oriented. Since each literal appears at most twice in ψ , the out-degree of each vertex in \tilde{C}_4 is at most 2; it is easily verified that the in-degree at out-degree of each vertex is then at most 2. Also, as before, \hat{G} is fixable if and only if ψ is a tautology, which in turn occurs if and only if ϕ is a tautology.

We now prove that the new graph \hat{G} is ϵ -fixable by exhibiting a word w that fixes all configurations with $x_\omega = 1$ for some $\omega \in \Omega$. Let $u \in A \cup \Omega$, then the word $w^u = \neg u \dot{u} \neg \dot{u}$ fixes $\hat{G}[u']$ without updating the vertex u . Then let $w' = (w^u : u \in A \cup \Omega)$ be the concatenation of all the w^u words (in any order). Let w'' be a word that follows the circuit in topological order, and finally let $w = w'w''abc$. Let x be a configuration such that $x_\omega = 1$ for some $\omega \in \Omega$. Note that because we chose $|\Omega| \geq -\log_2 \epsilon$ at least $1 - \epsilon$ of all configurations satisfy this property. We have $\psi(x_{A \cup \Omega}) = \psi(y_{A \cup \Omega}) = 1$ and hence $y_\psi = 1$ and $y_{abc} = 010$. Thus $y \in K(\hat{G})$. \square

7.2 The independent network

Since the kernel network is not fixable in general, and since it is not even tractable to decide whether the kernel network of a particular digraph is fixable, we now introduce and study two Boolean networks that are always fixable, and whose sets of fixed points contain all kernels.

The **independent network** on G , denoted by $I(G)$, is defined by

$$I(x)_v = x_v \wedge \bigwedge_{u \rightarrow v} \neg x_u,$$

with $I(x)_v = x_v$ if $N^{\text{in}}(v) = \emptyset$. We then have $\text{Fix}(I(G)) = I(G)$. Moreover, as we shall prove later, any permutation of V is a fixing word of $I(G)$.

We first settle the reachability problem for the independent network. We shall tacitly use the simple fact that if $x \mapsto_1 y$, then $y \leq x$. We characterise the configurations y that are reachable from a given initial configuration x . Generalising the undirected case, for any configuration $x \in \{0, 1\}^V$, we let $\mathcal{C}(x)$ denote the collection of initial strong components of $G[\mathbf{1}(x)]$.

Proposition 7.6 (Reachability for the independent network). *Let $x, y \in \{0, 1\}^V$ be two configurations. The following are equivalent:*

1. $x \mapsto_1 y$;
2. $x \xrightarrow{\text{geo}}_1 y$;
3. $y \leq x$ and $y_C \neq 0$ for any $C \in \mathcal{C}(x)$.

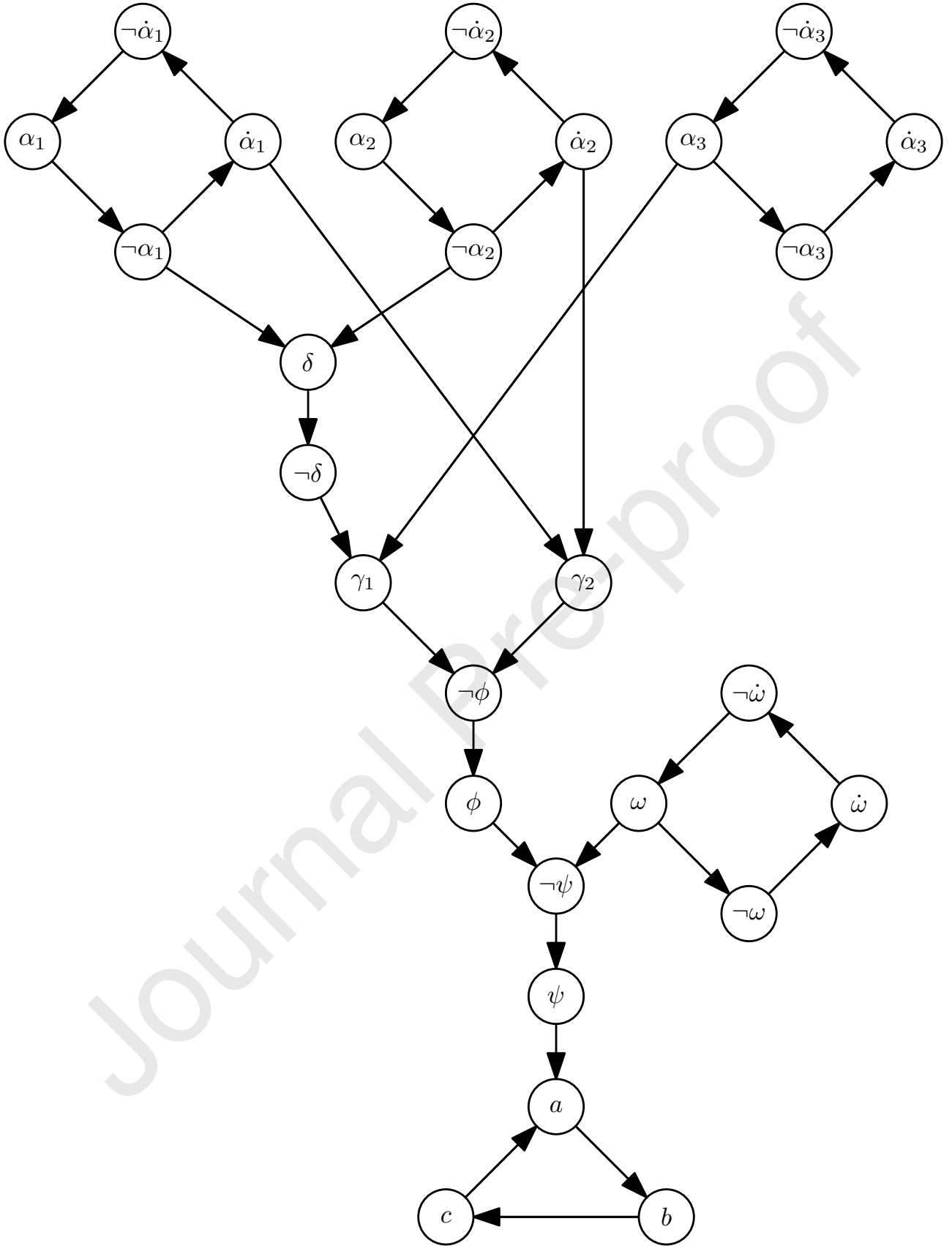


Figure 15: Illustration of the reduction from 3-TAUT to FIXABLE where each literal appears at most twice.

Proof of Proposition 7.6. If $x = 0$ then the proposition trivially holds as $G[\mathbf{1}(x)]$ is the empty digraph and $y = x$, so the geodesic is the empty word; we now consider $x \neq 0$.

We first prove that if $x \mapsto_1 y$, then $y_C \neq 0$ for all $C \in \mathcal{C}(x)$. Suppose $y = l^w(x)$ satisfies $y_C = 0$, with $w = w_{1:l}$, and that $y_C^{l-1} \neq 0$. Then $w_l \in C$ and there exists a vertex u such that $y_u^{l-1} = x_u = 1$ and $u \rightarrow w_l$; but then $u \in C$ and hence $y_C \neq 0$, which is the desired contradiction.

We now prove that if $y \leq x$ and $y_C \neq 0$ for all $C \in \mathcal{C}(x)$, then there is a geodesic from x to y . We use Lemma 2.1. The geodesic is a concatenation of words w^B , one for each strong component B of $G[\mathbf{1}(x)]$, in reverse topological order. The word w^B is described as follows.

- Case 1: $y_B = 0$.

Then B is not an initial component, hence there exists $u \in B$ with an in-neighbour in a parent strong component and T a spanning out-tree of B rooted at u . Then w^B is the word obtained by traversing the tree from leaves to root.

- Case 2: $y_B \neq 0$.

Let $S = \mathbf{1}(y) \cap B$ and T be the spanning out-forest of B rooted at S . Then w^B is the word obtained by traversing the out-forest but missing out the roots. (In particular, if $S = B$, then w^B is empty.)

It is easy to verify that w is indeed a geodesic from x to y . □

We now determine the words that fix the independent network. We generalise the concept of a vertex cover to digraphs as follows: a **directed vertex cover** of a digraph is a set of vertices S such that for every symmetric edge uv , $\{u, v\} \cap S \neq \emptyset$ and for every oriented edge (u, v) , $v \in S$.

Proposition 7.7 (Words fixing the independent network). *Let G be a digraph. Then w fixes the independent set network $\mathbf{l}(G)$ if and only if $[w]$ is a directed vertex cover of G .*

Proof. Suppose w fixes \mathbf{l} . If uv is a symmetric edge with $\{u, v\} \cap [w] = \emptyset$, then for any configuration x with $x_{uv} = 11$, we have $y_{uv} = 11$. If (u, v) is an oriented edge with $v \notin [w]$, then if $x_{N^{\text{in}}(u), u, v} = (0, 1, 1)$, we have $y_{uv} = 11$.

Suppose $[w]$ is a directed vertex cover. Suppose $y_{uv} = 11$. If uv is symmetric, without loss let v be updated last. Let y^a be the configuration before the last update of v , then $y_u^a = y_u = 1$, which implies $y_v = 1$, which is a contradiction. If (u, v) is oriented, then let y^a be the configuration before the last update of v , then $y_u^a = y_u = 1$, which implies $y_v = 1$, which is a contradiction. □

↓ FIXING WORD

Input: A digraph $G = (V, E)$ and a word w .

Question: Does w fix $\mathbf{l}(G)$?

Corollary 7.8. ↓ FIXING WORD is in **P**.

7.3 The dominating network

The **dominating network** on G , denoted by $D(G)$, is defined by

$$D(x)_v = x_v \vee \bigwedge_{u \rightarrow v} \neg x_u,$$

with $D(x)_v = 1$ if $N^{\text{in}}(v) = \emptyset$. We then have $\text{Fix}(D(G)) = D(G)$. Moreover, once again, any permutation of V fixes the dominating network.

We first settle the reachability problem for the dominating network. We use the simple fact that if $x \mapsto_D y$, then $y \geq x$. We characterise the configurations y that are reachable from a given configuration x . For any configuration x , let $A(x) = \mathbf{0}(x) \cap N^{\text{out}}(\mathbf{1}(x))$ and $B(x) = \mathbf{0}(x) \setminus N^{\text{out}}(\mathbf{1}(x))$.

Proposition 7.9 (Reachability for the dominating network). *Let $x, y \in \{0, 1\}^V$ be two configurations. The following are equivalent:*

1. $x \mapsto_D y$;

2. $x \xrightarrow{\text{geo}}_{\mathbf{D}} y$;

3. $y \geq x$, $y_{A(x)} = 0$, and $G[\mathbf{1}(y_{B(x)})]$ is acyclic.

Proof. We first prove that if $x \mapsto_{\mathbf{D}} y$, then $y_{A(x)} = 0$. We prove that $y_{A(x)}^a = 0$ by induction on a . This is clear for $a = 0$, so assume it holds for a . If $w_{a+1} \notin A(x)$, then we're done. Otherwise, since $y^a \geq x$, we have $y_{\mathbf{1}(x)}^a = 1$, therefore $\mathbf{D}(y^a)_{w_{a+1}} = 0$ and hence $y_{A(x)}^{a+1} = 0$.

We now prove that if $x \mapsto_{\mathbf{D}} y$, then $G[\mathbf{1}(y_{B(x)})]$ is acyclic. Suppose that $G[\mathbf{1}(y_{B(x)})]$ has a cycle v_1, \dots, v_k and without loss, suppose that v_k is the last updated. Let z be the configuration before that last update, then $z_{v_{k-1}} = 1$ hence $y_{v_k} = \mathbf{D}(z)_{v_k} = 0$, which is the desired contradiction.

We finally prove that if $y \geq x$, $y_{A(x)} = 0$, and $H = G[\mathbf{1}(y_{B(x)})]$ is acyclic, then there is a geodesic from x to y . Let w be a traversal of H in reverse topological order; it is easily shown by induction on a that $N^{\text{in}}(w_{a+1}; G) \subseteq \mathbf{0}(y^a)$ and hence $\mathbf{D}(y^a)_{w_{a+1}} = 1$, as required. \square

We now determine the words that fix the dominating network.

We denote the set of closed twins of v by $\langle v \rangle$ and we note that $\langle v \rangle \subseteq N^{\text{in}}(v)$. Note that V can be efficiently partitioned into these equivalence classes.

Proposition 7.10 (Words fixing the dominating network). *Let G be a digraph. Then w fixes the dominating set network $\mathbf{D}(G)$ if and only if $[w] \cap \langle m \rangle \neq \emptyset$ for all $m \in \mathbf{B}(G)$.*

Proof. Suppose that there exists $m \in \mathbf{B}(G)$ such that $[w] \cap \langle m \rangle = \emptyset$, then we exhibit a configuration x such that $y_{N^{\text{in}}[m]} = 0$. Let $A = N^{\text{in}}[m] \setminus \langle m \rangle$ and $B = N^{\text{in}}(A) \setminus N^{\text{in}}[m]$. For any $a \in A$, there exists $b \in N^{\text{in}}(a) \setminus N^{\text{in}}[m] \subseteq B$, since otherwise we would have $N^{\text{in}}[a] \subseteq N^{\text{in}}[m]$ and hence $a \in \langle m \rangle$. Let $x_{\langle m \rangle, A, B} = (0, 0, 1)$, then $y_{\langle m \rangle} = x_{\langle m \rangle} = 0$, $y_B \geq x_B = 1$ and hence $y_A = 0$, thus $y_{N^{\text{in}}[m]} = 0$.

Suppose that for all $m \in \mathbf{B}(G)$, we have $[w] \cap \langle m \rangle \neq \emptyset$, and suppose that $y_{N^{\text{in}}[v]} = 0$ for some v . For all u with $N^{\text{in}}[u] \subseteq N^{\text{in}}[v]$, we also have $y_{N^{\text{in}}[u]} = 0$, therefore, there exists $m \in \mathbf{B}(G)$, $N^{\text{in}}[m] \subseteq N^{\text{in}}[v]$ such that $y_{N^{\text{in}}[m]} = 0$. Since $y \geq x$, we also have $x_{N^{\text{in}}[m]} = 0$. Let $m' \in [w] \cap \langle m \rangle \subseteq N^{\text{in}}[m]$ and let w_{a+1} be the first update of m' , then $y_{N^{\text{in}}[m']}^a = y_{N^{\text{in}}[m]}^a = 0$, thus $\mathbf{D}(y^a)_{w_{a+1}} = 1 = y_{m'}$ and $y_{N^{\text{in}}[m]} \neq 0$, which is the desired contradiction. \square

D FIXING WORD

Input: A digraph $G = (V, E)$ and a word w .

Question: Does w fix $\mathbf{D}(G)$?

Corollary 7.11. **D FIXING WORD** is in **P**.

8 Conclusion

Summary of results In this paper, we have considered the generalisation of the MIS algorithm to allow for any initial configuration and to use update words that are not necessarily permutations. We have defined many decision problems with respect to this generalisation, such as:

- Given G and a configuration x , can x reach all MIS?
- Given G and a word w , does w fix $\mathbf{M}(G)$?
- Given G and a set of vertices S , can we fix $\mathbf{M}(G)$ by only updating S ?
- Given G , is there a word fixing $\mathbf{M}(G)$ that skips a vertex?
- Given G and a permutation w , does w fix $\mathbf{M}(G)$ (i.e. is w a permis of G)?
- Given G , does G have a permis?

Even though every graph has a fixing word that guarantees terminating at a MIS regardless of the initial configuration, all the decision problems about the MIS algorithm in this paper are computationally hard. Additionally, we exhibit broad classes of graphs with and without permises, and relate these to existing graph classes. We introduce the class of near-comparability (a strict superclass of comparability graphs, which themselves encompass interval graphs and bipartite graphs, among others) and show that all near-comparability graphs are permissible.

We further extended the MIS algorithm to digraphs; in this case, deciding whether the kernel network has a fixing word is computationally hard once again. Lastly, we consider the independent network and the dominating network, which are both related to the kernel network, and show that the analogous problems for these networks are tractable.

Future work This paper can be extended in several ways. We give three potential avenues below.

1. Graph classes.

Since our problems are **NP**- or **coNP**-hard for the class of all graphs, it is natural to examine the complexity of those problems when we restrict ourselves to particular graph classes. The main tool for reductions is the CONSTITUENCY decision problem. However, the reductions used in this paper did not preserve certain graph classes. For instance, CONSTITUENCY remains **NP**-complete even for bipartite graphs, while PERMISSIBLE is trivial for comparability graphs.

2. Minimum length of a fixing word.

We have investigated the existence of fixing words, but not their lengths. From our results on prefixing and suffixing words, we get an upper bound on the minimum length of a fixing word: $a + b - 1$, where a is the minimum size of a non-district and b is the minimum size of a vertex cover. We conjecture that the problem of determining the minimum length of a fixing word is computationally hard. The analogous problem when we can only start at the all-zero configuration is obviously **NP**-hard, as this amounts to determining the minimum size of a maximal independent set.

3. Permises with bounded diameter.

Let w be a permutation of V , and for any vertex v , let $\delta(v)$ denote the maximum length of a path terminating at v in G^w . Let $C_i = \{v \in V : \delta(v) = i\}$, then $V = C_0 \cup \dots \cup C_d$, where d is the diameter of w . Instead of updating the vertices sequentially according to w , one can update all the vertices in C_i at once, thus only requiring $d + 1$ time steps. As such, the diameter of a permis w measures the time it takes to fix the MIS network when we allow for some amount of synchronicity. If Δ is the maximum degree of G , then G always has a permutation of diameter Δ : partition V into colour classes C_0, \dots, C_Δ (since the chromatic number is at most $\Delta + 1$), then $C_0 \succ C_1 \succ \dots \succ C_\Delta$. This is best possible if G is complete, for instance. We therefore ask: if G has a permis, then does it have a permis of diameter bounded by a function of Δ ?

Acknowledgments

We would like to thank William K. Moses Jr. and Peter Davies-Peck for their insights on self-stabilization and distributed computing.

References

- [1] Yehuda Afek, Noga Alon, Ziv Bar-Joseph, Alejandro Cornejo, Bernhard Haeupler, and Fabian Kuhn. Beeping a maximal independent set. *Distributed computing*, 26(4):195–208, 2013.
- [2] Yehuda Afek, Noga Alon, Omer Barad, Eran Hornstein, Naama Barkai, and Ziv Bar-Joseph. A biological solution to a fundamental distributed computing problem. *Science*, 331(6014):183–185, 2011.

- [3] J. Aracena, A. Richard, and L. Salinas. Maximum number of fixed points in and-or-not networks. *Journal of Computer and System Sciences*, 80(7):1175 – 1190, 2014.
- [4] J. Aracena, A. Richard, and L. Salinas. Synchronizing boolean networks asynchronously. *Journal of Computer and System Sciences*, 136:249–279, 2023.
- [5] Julio Aracena, Maximilien Gadouleau, Adrien Richard, and Lilian Salinas. Fixing monotone boolean networks asynchronously. *Information and Computation*, 274(104540), October 2020.
- [6] Julio Aracena, Eric Goles, A. Moreira, and Lilian Salinas. On the robustness of update schedules in boolean networks. *BioSystems*, 97:1–8, 2009.
- [7] Jorgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2009.
- [8] Joffroy Beauquier, Janna Burman, Fabien Dufoulon, and Shay Kutten. Fast Beeping Protocols for Deterministic MIS and $(\Delta + 1)$ -Coloring in Sparse Graphs. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1754–1762, 2018.
- [9] Guy E. Blelloch, Jeremy T. Fineman, and Julian Shun. Greedy sequential maximal independent set and matching are parallel on average. In *SPAA '12: Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*, pages 308–317, June 2012.
- [10] Béla Bollobás and Imre Leader. Connectivity and dynamics for random subgraphs of the directed cube. *Israel Journal of Mathematics*, 83:321–328, 1993.
- [11] S. Bornholdt. Boolean network models of cellular regulation: prospects and limitations. *Journal of The Royal Society Interface*, 5(Suppl 1):S85–S94, 2008.
- [12] A. Casteigts, Y. Métivier, J.M. Robson, and A. Zemhari. Design patterns in beeping algorithms: Examples, emulation, and analysis. *Information and Computation*, 264:32–51, 2019.
- [13] Artur Czumaj and Peter Davies. Communicating with beeps. *J. Parallel Distrib. Comput.*, 130(C):98–109, aug 2019.
- [14] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, nov 1974.
- [15] Maximilien Gadouleau and Adrien Richard. On fixable families of boolean networks. In *Proc. Workshop on Asynchronous Cellular Automata*, pages 396–405, September 2018.
- [16] Mohsen Ghaffari. Distributed maximal independent set using small messages. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 805–820, 2019.
- [17] Mohsen Ghaffari. Local computation of maximal independent set. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 438–449, 2022.
- [18] E. Goles and M. Noul. Disjunctive networks and update schedules. *Advances in Applied Mathematics*, 48(5):646–662, 2012.
- [19] Eric Goles. Dynamics of positive automata networks. *Theoretical Computer Science*, 41:19–32, 1985.
- [20] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.
- [21] S. A. Kauffman. Metabolic stability and epigenesis in randomly connected nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

- [22] Christoph Lenzen, Jukka Suomela, and Roger Wattenhofer. Local Algorithms: Self-Stabilization on Speed. In Sándor Fekete, Stefan Fischer, Martin Riedmiller, and Suri Subhash, editors, *Algorithmic Methods for Distributed Cooperative Systems*, volume 9371 of *Dagstuhl Seminar Proceedings (DagSemProc)*, pages 1–18, Dagstuhl, Germany, 2010. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [23] Ross M. McConnell and Jeremy Spinrad. Linear-time transitive orientation. In *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 19–25, 1997.
- [24] Mathilde Noual and Sylvain Sené. Synchronism versus asynchronism in monotonic boolean automata networks. *Natural Computing*, 17:393–402, June 2018.
- [25] Landon Rabern, Brian Rabern, and Matthew Macauley. Dangerous reference graphs and semantic paradoxes. *Journal of Philosophical Logic*, 42(5):727–765, 2013.
- [26] A. Richard and P. Ruet. From kernels in directed graphs to fixed points and negative cycles in boolean networks. *Discrete Applied Mathematics*, 161(7):1106–1117, 2013.
- [27] F. Robert. Iterations sur des ensembles finis et automates cellulaires contractants. *Linear Algebra and its Applications*, 29:393–412, 1980.
- [28] Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2), mar 2013.
- [29] R. Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563 – 585, 1973.
- [30] Steven Yablo. Paradox without self-reference. *Analysis*, 53(4):251–252, 1993.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: