Switching graphs

Citation for published version (APA):

Document status and date:
Published: 01/01/2008

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Switching Graphs
Jan Friso Groote J.F.Groote@tue.nl
Bas Ploeger∗ S.C.W.Ploeger@tue.nl
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands

Abstract
Switching graphs are graphs containing switches. By using boolean functions called switch settings, these switches can be put in a fixed direction to obtain an ordinary graph. For many problems, switching graphs are a remarkable straightforward and natural model, but they have hardly been studied. We study the complexity of several natural questions in switching graphs of which some are polynomial, and others are NP-complete. We started investigating switching graphs because they turned out to be a natural framework for studying the problem of solving Boolean equation systems, which is equivalent to model checking of the modal $\mu$-calculus and deciding the winner in parity games. We give direct, polynomial encodings of Boolean equation systems in switching graphs and vice versa, and prove correctness of the encodings.

1 Introduction

We are inspired by the long open problem of finding a polynomial algorithm for the equivalent problems of solving Boolean equation systems (BES) [14], model checking formulas in the modal $\mu$-calculus [13] and finding a winning strategy in parity games [2] [10]. Especially the intriguing algorithms that use hill-climbing techniques with smartly chosen progress measures drew our attention [10] [18]. While studying the question how ‘progress’ could be captured better in the formalism, we started to use what we call switching graphs.

A switching graph is an ordinary graph with switches. A typical instance of a directed labelled switching graph is given in Fig. 1 in Fig. 2(a)}
Figure 1: A directed labelled switching graph

An undirected unlabelled switching graph is depicted. A switch is a triple of vertices \((v_1, v_2, v_3)\) where it is possible to move either from \(v_1\) to \(v_2\) or from \(v_1\) to \(v_3\) depending on a switch setting, which is a function from switches to booleans. In Fig. 1, there are two switches \(s_1\) and \(s_2\). If the switch setting for \(s_1\) is true, it is possible to move from the vertex labelled 1 to the vertex labelled 4, indicated by a straight solid arrow. If the switch setting for \(s_1\) is false, the dashed arrow must be followed. The curved arrows are ordinary directed edges in the graph. Note that only if both switches are set to 1, there is a loop on which the lowest number is odd.

As switching graphs are a natural extension to ordinary graphs and a natural abstract domain for representing concrete problems, we expected switching graphs to be widely studied, but this appears not to be the case. In [15], a slightly different notion of switching graphs is used essentially omitting switches, unifying nondeterminism with switches. In [4], proof nets are proposed which closely resemble switching graphs, except that here switches move in pairs. Other notions in the literature (like switch graphs [9], bidirected graphs [1] and skew-symmetric graphs [5]) also deal with choices excluding other options, but appear to be of a different nature than switching graphs.

We show that solving BESs can naturally be formulated as a problem on directed labelled switching graphs, namely as the \(v\)-parity loop problem: does a switch setting exist such that the lowest label on every loop reachable from \(v\) is even. The \(v\)-parity loop problem is essentially the same problem as that of deciding the winner in parity games. This problem is already known to be in NP \(\cap\) co-NP (or even UP \(\cap\) co-UP) and is also known to be equivalent to the BES problem via translations to/from the \(\mu\)-calculus model checking problem. The fact that the \(v\)-parity loop problem is equivalent to solving BESs is therefore not surprising, neither is the immediate complexity result. However, a direct encoding of BESs in switching graphs (and vice versa) is, to our opinion, more intuitive and a natural generalisation of the encoding of conjunctive/disjunctive BESs in ordinary graphs [7].

Moreover, we prove that it is impossible to synchronise switches in the context of the \(v\)-parity loop problem. Although we believe that this property is key to finding a polynomial algorithm we were not able to discover such an algorithm.
We also investigate the complexities of several variations of the parity loop problem, in order to gain more insight into its nature. If we relax the total ordering of labels we obtain the NP-complete cancellation loop problem. If we relax the requirement of loops we get the NP-complete connection- and disconnection problems. If we relax the requirements on labels we obtain the polynomial loop problem and the 1-2-loop problem. For the latter problem, we presented a polynomial-time algorithm in a previous paper [8], but we now show that this algorithm is incorrect.

In this paper we first define switching graphs (Sect. 2). Next, we investigate several problems on switching graphs and provide their characterisation in terms of complexity (Sect. 3). In Sect. 4 we connect the $v$-parity loop problem to that of solving BESs. As we found switching graphs to be interesting in themselves as a natural abstract model for many problems, we list a number of problems of which the complexity is unknown (at least to us) in Sect. 5. It should be obvious that these are only a fraction of the interesting questions that exist about switching graphs.

Acknowledgements. We thank Mark de Berg, Herman Geuvers, Jaco van de Pol and Gerhard Woeginger for various insights and references to relevant literature.

2 Preliminaries

2.1 Switching graphs

Definition 2.1. A directed labelled switching graph (DLSG) is a four-tuple $G = (V, E, S, \ell)$ where

- $V$ is a set of vertices;
- $E \subseteq V \times V$ is a set of directed edges;
- $S \subseteq V \times V \times V$ is a set of directed switches;
- $\ell : V \rightarrow \mathbb{N}$ is a labelling function.

For edges $v$ and $w$, we write $v \rightarrow w$ if $(v, w) \in E$. If $(v_1, v_2, v_3) \in S$, we write $v_1 \rightarrow_1 v_2$ and $v_1 \rightarrow_0 v_3$. For a switch $s = (v_1, v_2, v_3)$ we write $s^1$ for $v_2$ and $s^0$ for $v_3$. We call $v_1$ the root of $s$, $v_2$ the 1-vertex of $s$ and $v_3$ the 0-vertex of $s$.

A switch setting is a function $f : S \rightarrow \{0, 1\}$. An $f$-path from vertex $v$ to vertex $w$ is a sequence $v_1, v_2, \ldots, v_n$ with $v = v_1$, $w = v_n$ and for all $1 \leq i < n$ it either holds that $v_i \rightarrow v_{i+1}$, or there is a switch $s \in S$ with root $v_i$ and $v_{i+1} = s^{f(s)}$. An $f$-path from $v$ to $v$ is called an $f$-loop through $v$. For $b \in \{0, 1\}$ we write $f[s:=b]$ for the switch setting $f$ where switch $s$ is set
to $b$. Similarly, $f[S:=b]$ denotes the switch setting $f$ in which every switch in the set $S$ is set to $b$.

For any DLSG $G = (V, E, S, \ell)$ and switch setting $f$ over $S$, the directed labelled graph $G_f = (V, E', \ell)$ is called the $f$-graph of $G$ where:

$$E' = E \cup \{(v, w) \mid \exists u \in V . ((v, w, u) \in S \land f(v, w, u) = 1) \lor ((v, u, w) \in S \land f(v, u, w) = 0)\}$$

A switching graph with no labelling function is simply called a directed switching graph. A switching graph of which the set of edges is symmetric, is called an undirected (labelled) switching graph.

**Example 2.2.** An example of an undirected switching graph is found in Fig. 2(a). This graph is only not connected if both switches are set to 1.

A typical application comes from investigating possible partitions of a set. The elements of the set are the vertices. An edge between two vertices indicates that they are related. Two elements are equivalent if they are connected. Using switches alternative relations between individual elements can be represented. Typical questions can be whether a switch setting exists such that the equivalence partition has exactly $k$ elements (or more than $k$, or exactly an even number, etc.).

**Example 2.3.** In Fig. 2(b) a simple example of a directed switching graph is depicted. It illustrates that ordinary graphs and switching graphs are not quite the same. Compare the switching graph with an ordinary graph where both branches of switch $s_1$ are ordinary edges. The question whether there is a (switch setting with a) loop through both $A$ and $B$ is answered with yes in the graph, and with no in the switching graph. If the switching graph is compared to the graph that contains only one branch of the switch, the question whether there is a (switch setting with a) loop through $A$ and a (switch setting with a) loop through $B$ is answered with no for the graph, and with yes for the switching graph.
2.2 The 3SAT problem

We use the well-known 3SAT problem for proving NP-completeness of several problems in Sect. 3.

Definition 2.4. A *3CNF formula* is a logical formula of the form

$$\bigwedge_{i \in I} (l_{i,1} \lor l_{i,2} \lor l_{i,3})$$

where $I$ is a finite index set and $l_{i,j}$ are literals, i.e. formulas of the shape $p$ or $\neg p$ for a proposition letter $p$ taken from a set $P$. The *3SAT problem* is the question whether a given 3CNF formula is satisfiable.

A famous result by Karp is that the 3SAT problem is NP-complete [11]. In this paper, we assume some ordering $<$ on the proposition letters $P$ and assume that the literals in each clause of a 3CNF formula are ordered according to $<$, such that the leftmost proposition letter in that clause is the smallest, and the rightmost is the largest.

2.3 Boolean equation systems

We assume the existence of a set of proposition variables $X$ with typical elements $X, X_1, X_2, \ldots$ and a type $\mathbb{B}$ representing Booleans with elements $\bot$ and $\top$ representing 0 (false) and 1 (true) respectively.

A *Boolean equation system* (BES) is a sequence of fixed-point equations over the Boolean lattice. It can be defined inductively as follows:

- $\epsilon$ denotes the empty BES;

- for every BES $\mathcal{E}$, $(\sigma X = \phi) \mathcal{E}$ is also a BES, where $\sigma \in \{\mu, \nu\}$ is a fixed-point symbol, $X$ is a proposition variable and $\phi$ is a proposition formula.

A fixed-point symbol is either $\mu$ for least fixed-point, or $\nu$ for greatest fixed point. Proposition formulae $\phi$ are defined by the following grammar:

$$\phi ::= \bot \mid \top \mid X \mid \phi \land \phi \mid \phi \lor \phi$$

where $X \in \mathcal{X}$ is a proposition variable of type $\mathbb{B}$.

The BES problem is the problem of finding an assignment of either 0 or 1 to each variable $X_i$ such that every equation is satisfied and for every $1 \leq i \leq n$, the assignment to $X_i$ is as strong as possible if $\sigma_i = \mu$, and as weak as possible if $\sigma_i = \nu$. We call this assignment the solution of the BES, which we now define formally. A *proposition environment* is a function $\theta : \mathcal{X} \rightarrow \{0,1\}$ that assigns a Boolean value to every proposition variable in $\mathcal{X}$.
Definition 2.5. The solution of a BES in the context of a proposition environment $\theta$ is defined inductively as follows, for any BES $E$:

$$
\begin{align*}
& [\epsilon] \theta = \theta \\
& [\sigma X = \phi] E \theta = [E] \theta[b_{\sigma} / X] / X
\end{align*}
$$

where $b_\mu = 0$ and $b_\nu = 1$. The BES problem is the problem of determining $([E] \theta)(X)$ for a given BES $E$, environment $\theta$ and variable $X$.

It is known that the BES problem is equivalent to the $\mu$-calculus model-checking problem [14]. This problem was shown to be in NP $\cap$ co-NP by Emerson, Jutla and Sistla by showing equivalence to the non-emptiness problem of parity tree automata [2]. Later, Stirling reduced the problem to that of deciding the winner in parity games, obtaining the same complexity result [16].

In this paper, we only consider BESs of the following form:

$$
E = (\sigma_1 X_1 = \phi_1) \ldots (\sigma_n X_n = \phi_n)
$$

for some $n \in \mathbb{N}$. Moreover, $E$ is:

- in standard form if every $\phi_i$ is either $X_j$ or $X_j \lor X_k$ or $X_j \land X_k$;
- in 2-conjunctive normal form (2CNF) if every $\phi_i$ is of the form $\bigwedge_{p \in I_i} c_{i,p}$ where $I_i$ is an index set and $c_{i,p}$ is either $X_j$ or $X_j \lor X_k$;
- conjunctive if no $\phi_i$ contains a $\lor$;
- disjunctive if no $\phi_i$ contains a $\land$.

It is known that every BES can be linearly transformed to standard form such that its solution is maintained, modulo renaming of variables [14]. If $E$ is conjunctive or disjunctive, the dependency graph of $E$ is a directed labelled graph $(V, E, \ell)$ where $V = \{1, \ldots, n\}$, $E = \{(i, j) \mid X_j \text{ occurs in } \phi_i\}$ and $\ell(i) = 2i + \text{sign}(\sigma_i)$ with $\text{sign}(\mu) = 1$ and $\text{sign}(\nu) = 0$. The following result was obtained in [7] as Lemma 3.

\begin{lemma}
(Lemma 3 in [7].) Let $\theta$ be an environment, $E$ be a conjunctive BES and $G$ be its dependency graph. Then for any variable $X_i$ of $E$ we have that $(E \theta)(X_i) = 0$ iff $G$ contains a loop that is reachable from $i$ on which the lowest label is odd.
\end{lemma}

For disjunctive BESs the dual result holds (i.e. replacing 0 by 1 and “odd” by “even”). The loop problem on the dependency graph can be solved in polynomial time, which is also shown in [7].

\footnote{In [7] the labelling function is a function $\ell'$ that maps vertices to $\{\mu, \nu\}$. In the original lemma, $G$ should contain a loop on which the vertex with the lowest index has label $\mu$. It is obvious that $\ell'(i) = \mu$ iff $\ell(i)$ is odd and $\ell'(i) = \nu$ iff $\ell(i)$ is even. Also $i < j$ iff $\ell(i) < \ell(j)$. Hence, the result indeed carries over to our version here.}
3 Switching graph problems and their complexities

In this section we investigate the complexity of several problems on a fixed DLSG \( G = (V, E, S, \ell) \). For many problems there are straightforward and efficient algorithms. Therefore, we often only state the complexity and sketch the algorithm without further explanation.

3.1 The \( v\)-\( w\)-connection problem

The \( v\)-\( w\)-connection problem is the question whether there is a switch setting \( f \) such that there is an \( f \)-path in \( G \) from a given vertex \( v \) to a given vertex \( w \).

**Theorem 3.1.** The \( v\)-\( w\)-connection problem is solvable in linear time.

**Proof.** A straightforward depth-first search suffices where both branches of each switch are taken as ordinary edges. When a path from \( v \) to \( w \) is found, no vertex, and hence no switch, occurs on this path more than once. The switch setting \( f \) is set accordingly, where those switches not on the path can be set arbitrarily. \( \square \)

3.2 The \( v\)-\( w\)-disconnection problem

The \( v\)-\( w\)-disconnection problem is the question whether there is a switch setting \( f \) such that there is no \( f \)-path in \( G \) from a given vertex \( v \) to a given vertex \( w \).

**Theorem 3.2.** The \( v\)-\( w\)-disconnection problem is solvable in linear time.

**Proof.** Mark vertices backwards from \( w \) in a breadth-first fashion. A vertex \( u \) is marked if an outgoing edge leads to a marked vertex, or if both branches of a switch with root \( u \) lead to marked vertices. This marking algorithm will terminate in linear time. Answer yes if \( v \) was not marked and no otherwise. The switch setting is obtained by setting the switches such that they point to an unmarked vertex. Switches that point to vertices that are both marked or unmarked can be set arbitrarily. \( \square \)

3.3 The loop problem

The loop problem is the question whether there is a switch setting \( f \) such that \( G \) contains an \( f \)-loop.

**Theorem 3.3.** The loop problem is solvable in quadratic time.

**Proof.** This problem is equivalent to the \( v\)-\( v\)-connection problem for every vertex \( v \). If for some vertex \( v \) a switch setting \( f \) is found such that \( v \) is connected to itself, then the answer is yes. Otherwise, the answer is no. \( \square \)
3.4 The \( v\)-parity loop problem

The \( v\)-parity loop problem is the question whether there is a switch setting \( f \) such that every \( f \)-loop in \( G \) that is reachable from a given vertex \( v \), has an even number as lowest label.

It is not hard to see that this problem is equivalent to that of deciding the winner of a parity game, starting from a vertex \( v \) of the game graph. Hence, this problem is in \( \text{NP} \cap \text{co-NP} \) and it is an open question whether a polynomial algorithm exists. We provide direct translations from this problem to that of solving BESs and vice versa in Sect. 4.

We now show that it is not possible to keep two switches synchronised in the following sense. Consider a switching graph containing at least two switches. It is not possible that if two switches are fixed to opposite positions, the answer to the \( v\)-parity loop problem is \text{yes}, whereas if the same switches are fixed to equal positions, the answer to the \( v\)-parity loop problem on the remaining graph is \text{no}. A similar non-synchronisation property holds for the loop problem. In the cancellation and connection problems (Sects. 3.8 and 3.9), it is exactly this capability of synchronising switches that allows us to prove \( \text{NP}\)-completeness.

In order to formulate the non-synchronisation property we write \( L(f) = 1 \) for a switch setting \( f \) if all \( f \)-loops reachable from \( v \) in \( G \) have an even number as lowest label, and \( L(f) = 0 \) otherwise. For concrete values of \( f \), \( L(f) \) can be calculated in polynomial time.

**Lemma 3.4.** Consider two switches \( s \) and \( s' \). If for all switch settings \( f \) with \( f(s) \neq f(s') \) with \( L(f) = 1 \), then there are no switch settings \( g \) and \( h \) such that \( g(s) = g(s') = 0 \), \( h(s) = h(s') = 1 \) and \( L(g) = L(h) = 0 \).

**Proof.** Assume the contrary, i.e. there are switch settings \( g \) and \( h \) such that \( g(s) = g(s') = 0 \), \( h(s) = h(s') = 1 \) and \( L(g) = L(h) = 0 \). Consider a \( g \)-loop reachable from \( v \) that has an odd value \( m \) as lowest label. This loop must pass through both \( s \) and \( s' \), as if it does not go through \( s \), then \( g[s:=1] \) also has a loop with \( m \) as lowest number, contradicting the assumption of the lemma. Similarly, if the loop does not pass through \( s' \), then \( g[s' :=1] \) has a loop with \( m \) as lowest label. Using the same line of reasoning, there is an \( h \)-loop reachable from \( v \) with lowest odd label \( n \) going through both \( s \) and \( s' \).

Assume without loss of generality that \( m \leq n \). The vertex labelled with \( m \) can be positioned between the 0-edge of \( s \) and the 0-edge of \( s' \) or vice versa between the 0-edge of \( s' \) and the 0-edge of \( s \). In the first case \( g[s' :=1] \) also has a loop reachable from \( v \) with \( m \) as lowest odd label, contradicting the assumption. In the second case \( g[s :=1] \) has a loop reachable from \( v \) with \( m \) as lowest odd label also contradicting the assumption. Note that as \( m \leq n \), there are no even labels on these loops smaller than \( m \). Hence, all cases lead to a contradiction, proving the lemma.
3.5 The parity loop problem

The parity loop problem is the question whether a switch setting $f$ exists such that all $f$-loops in $G$ have an even number as lowest label. This problem is very similar to the $v$-parity loop problem. In particular no polynomial algorithm is known, and it is also not possible to synchronise switches in this setting.

3.6 The parity loop-through-$v$ problem

The parity loop-through-$v$ problem is the question whether a switch setting $f$ exists such that all $f$-loops through $v$ in $G$ have an even number as lowest label. This problem is very similar to the parity loop problem, but it allows a single set of switches to be synchronised. In Fig. 3 the switches $s_1, s_2, \ldots, s_n$ are synchronised. If all switches are in the 1 position, the lowest label of any loop through $v$ is 2. If all switches are in the 0 position, there is no loop through $v$. But if there are switches in both the 1 and in the 0 position, then there is a loop through $v$ on which the lowest label is 1. All non-labelled nodes are assumed to have a high, and therefore irrelevant label.

As it is possible to synchronise a single set of switches, we expect that it will even be harder to find a polynomial algorithm for this problem than for the $v$-parity loop problem.

3.7 The 1-2-loop problem

A simplification of the parity loop problem is the 1-2-loop problem where the labels are restricted to the values 1 and 2. In our previous work [8], we claimed a polynomial time algorithm for this problem. It was a weight improvement algorithm inspired by [10, 18]. However, it turns out that our algorithm and its correctness proof were incorrect. We give an example to
Figure 4: Counterexample for correctness of our previously presented algorithm for the 1-2-loop problem.

Assume \( \ell(v) \in \{1, 2\} \) for all \( v \in V \). The 1-2-loop problem is the question whether a switch setting \( f \) exists such that every \( f \)-loop in \( G \) has 2 as lowest label. In our original algorithm for this problem we first defined the weight of every vertex \( v \) for a switch setting \( f \) as follows. If no loop containing a label 1 (i.e. a 1-loop) can be \( f \)-reached from \( v \), then the weight is \( \dagger \). Otherwise, the weight is \((k, d, p)\) where \( k \) is the maximal number of 1’s on a path from \( v \) to a 1-loop, \( d \) is the minimal number of switches on a path with \( k \) 1-labelled vertices from \( v \) to a 1-loop, and \( p \) is the number of 1’s on this loop. The weight \( \dagger \) is the smallest of all. Furthermore, \( (k, d, p) < (k', d', p') \) iff \( k < k' \) or, \( k = k' \) and \( d > d' \) or, \( k = k' \), \( d = d' \) and \( p < p' \).

Our algorithm then proceeded as follows. Take an arbitrary switch setting \( f \). Repeatedly try changing every switch and accept the change if the weight of the root of this switch decreases. If no switch can be changed decreasing its root’s weight, the algorithm stops. If in the final switch setting there are no 1-loops, report yes. Otherwise report no.

We now show that this algorithm is incorrect. Consider the example in Figure 4 and take the switch setting in which both switches are set to 1. Observe that there is a 1-loop. The root of switch \( s_1 \) has weight \((0, 1, 1)\) and the root of \( s_2 \) has weight \((0, 0, 1)\). None of these switches can be changed such that the weight of its root decreases. Hence, our algorithm terminates and reports no as its answer to the 1-2-loop problem. However, by setting both switches to 0 we find a switch setting in which no 1-loop exists, so our algorithm gave the wrong answer.

3.8 The cancellation loop problem

The cancellation loop problem is the question whether there is a switch setting \( f \) such that each \( f \)-loop in \( G \) that contains an even label \( n \) does not contain the label \( n - 1 \).

Theorem 3.5. The cancellation loop problem is NP-complete.
Switching graphs

Figure 5: Translation of a clause $p_{i,1} \lor \neg p_{i,2} \lor p_{i,3}$ for the cancellation loop problem.

Proof. It is straightforward to see that this problem is in NP. If a switch setting is given, checking that every loop with even label $n$ does not have a label $n-1$ can be done in polynomial time. In order to show that the problem is NP-hard, we reduce the 3SAT problem to the cancellation loop problem. We start with a 3CNF formula $\bigwedge_{i \in I} (l_{i,1} \lor l_{i,2} \lor l_{i,3})$ where $I$ is an index set that does not contain 0. We generate a switch for every occurrence of a proposition letter in a clause. For each clause $l_{i,1} \lor l_{i,2} \lor l_{i,3}$ with proposition letters $p_{i,1}$, $p_{i,2}$ and $p_{i,3}$ three switches are generated. The root of the first switch is labelled with an even number $2i$. All other vertices have label 0. If $l_{i,1}$ is a positive literal then the 0-vertex of the switch for $p_{i,1}$ is connected to the next switch and the 1-vertex is connected to the root of the switch for $p_{i,1}$. Otherwise, these connections are interchanged. In the same way the second switch is connected to the third switch, and the third switch to the first switch. If $l_{i,3}$ is positive, the 0-vertex of the third switch is labelled with $2i - 1$. Otherwise, its 1-vertex is labelled with $2i - 1$. If two or more of the proposition letters in a clause are the same, then either the clause is omitted in case the literals have opposite signs, or only one or two switches are necessary. See Fig. 5 for an example.

Now assume for each proposition letter $p$ that all different switches for $p$ are synchronised, in the sense that they are all set to 0, or they are all set to 1. Then it is straightforward to see that every loop with even label $n$ does not contain label $n-1$ iff the formula is satisfiable.

We now show that all the switches for a proposition letter $p$ can be synchronised. For this, we introduce for each proposition letter $p$ three unique even numbers greater than 0: $a_p$, $b_p$ and $c_p$. Uniqueness means that these numbers are neither equal to any $a_q$, $b_q$ or $c_q$ for any other proposition letter $q$ nor are they equal to $2i$ for any $i \in I$. For any $p$ we add two switches $s_{p,1}$ and $s_{p,2}$ and we connect the switches as indicated in Fig. 6. The column of switches under the letter $p$ are all the different switches introduced for every occurrence of $p$ in some clause. We also label certain vertices by $a_p$, $b_p$, $c_p$, $a_p - 1$, $b_p - 1$ and $c_p - 1$. Unlabelled vertices are considered to have the value 0.
Observe that all switches (including $s_{p,1}$ and $s_{p,2}$) either must all be set to 0, or they must all be set to 1: otherwise there is a loop containing labels $n$ and $n - 1$. As the required case distinction is straightforward, we omit it here.

3.9 The connection problem

Given a set of pairs of vertices \( \{(v_i, w_i) \mid i \in I\} \) for some index set \( I \). The connection problem is the question whether there is a switch setting \( f \) such that, for every \( i \in I \), there is an \( f \)-path in \( G \) from \( v_i \) to \( w_i \).

**Theorem 3.6.** The connection problem is NP-complete.

**Proof.** It is obvious that this problem is in NP and therefore we only show that it is NP-hard. We reduce the 3SAT problem to the connection problem in a straightforward way. Consider the following 3CNF formula:

\[
\bigwedge_{i \in I} (l_{i,1} \lor l_{i,2} \lor l_{i,3})
\]

for an index set \( I \). We introduce one switch for every proposition letter occurring in the formula. We translate each clause \( l_{i,1} \lor l_{i,2} \lor l_{i,3} \) in which proposition letters \( p_{i,1}, p_{i,2} \) and \( p_{i,3} \) occur, as follows. We introduce a start vertex \( v_i \), and an end vertex \( w_i \). The vertex \( v_i \) is connected to the roots of the switches \( p_{i,1}, p_{i,2} \) and \( p_{i,3} \). Moreover, if \( l_{i,k} \) is a positive literal then the 1-vertex of switch \( p_{i,k} \) is connected to \( w_i \), otherwise its 0-vertex is connected to \( w_i \). See Fig. 4 for an example.

We can now directly observe that the 3CNF formula is satisfiable iff there is a switch setting \( f \) such that for all \( i \in I \) there is an \( f \)-path from \( v_i \) to \( w_i \).

\[\square\]
3.10 The disconnection problem

Given a set of pairs of vertices \( \{(v_i, w_i) \mid i \in I \} \) for some index set \( I \). The disconnection problem is the question whether there is a switch setting \( f \) such that, for every \( i \in I \), there is no \( f \)-path in \( G \) from \( v_i \) to \( w_i \).

**Theorem 3.7.** The disconnection problem is NP-complete.

**Proof.** It is again easy to show that this problem is in NP. For showing NP-hardness, we again reduce the 3SAT problem to this problem but the translation is considerably more involved than for the connection problem. We start again with a 3CNF formula:

\[
\bigwedge_{i \in I} (l_{i,1} \lor l_{i,2} \lor l_{i,3})
\]

for an index set \( I \). We introduce a switch for every occurrence of a proposition letter in a clause. For each clause \( l_{i,1} \lor l_{i,2} \lor l_{i,3} \) with proposition letters \( p_{i,1}, p_{i,2} \) and \( p_{i,3} \) a start vertex \( v_i \), an end vertex \( w_i \) and three switches are generated. The vertex \( v_i \) is connected to the root of the first switch. If \( l_{i,1} \) is a positive literal then the 0-vertex is connected to the next switch, otherwise the 1-vertex is connected to the next switch. In the same way the second switch is connected to the third switch, which is in turn connected in the same way to the vertex \( w_i \). If two or more of the proposition letters in a clause are the same, then either the clause is omitted in case the literals have opposite signs, or only one or two switches are necessary. See Fig. 8 for an example.

Now assume for each proposition letter \( p \) that all different switches for \( p \) are synchronised, i.e. they are all set to 0, or they are all set to 1. Then it is straightforward to see that for all \( i \in I \) there is no path from \( v_i \) to \( w_i \) iff the 3CNF formula is satisfiable.

![Switching graph diagram](image-url)
Figure 8: Translation of a clause $p_{i,1} \lor \neg p_{i,2} \lor p_{i,3}$ for the disconnection problem.

Figure 9: Construction for keeping all switches for one proposition letter $p$ synchronised in the translation of the disconnection problem.

We now show that synchronisation of all switches for a proposition letter $p$ can be enforced. In order to do so, we add three pairs of start and end vertices for $p$, namely $(A_p, A'_p)$, $(B_p, B'_p)$ and $(C_p, C'_p)$, and three switches $s_{p,1}$, $s_{p,2}$ and $s_{p,3}$. We connect the switches and vertices as indicated in Fig. 9. The column of switches under the letter $p$ are all the switches introduced for every occurrence of $p$ in some clause.

We claim that only if all switches under $p$ are set to 0, or all switches are set to 1, there are no paths between any of the pairs $(A_p, A'_p)$, $(B_p, B'_p)$ and $(C_p, C'_p)$. In order to see that all switches for $p$ must be synchronised, it suffices to note that all switches (including $s_{p,1}$, $s_{p,2}$ and $s_{p,3}$) in Fig. 9 either must all be set to 0, or they must all be set to 1. Otherwise there is an unwanted path. As the required case distinction is straightforward, we omit it here.

4 Equivalence of the $v$-parity loop and BES problems

In this section we provide reductions from the BES problem to the $v$-parity loop problem and vice versa. For the proofs we need to be able to remove every disjunction from a BES in 2CNF by selecting one of the disjuncts and eliminating the other one, resulting in a conjunctive BES. The operation resembles that of obtaining the $f$-graph $G_f$ from a DLSG $G$ and a switch
Definition 4.1. Given the following BES in 2CNF:

\[ \mathcal{E} = (\sigma_1 X_1 = \bigwedge_{j \in I_1} c_{1,j}) \ldots (\sigma_n X_n = \bigwedge_{j \in I_n} c_{n,j}) \]

for clauses \( c_{i,j} \) and index sets \( I_i \). Let \( f \) be a function such that for every \( 1 \leq i \leq n \) and \( j \in I_i \): \( f(i, c_{i,j}) = \phi_k \) if \( c_{i,j} = \phi_k \), and either \( f(i, c_{i,j}) = \phi_k \) or \( f(i, c_{i,j}) = \phi_l \) if \( c_{i,j} = \phi_k \lor \phi_l \). We call \( f \) a selection function. Then the \( f \)-BES \( \mathcal{E}_f \) is the following conjunctive BES:

\[ \mathcal{E}_f = (\sigma_1 X_1 = \bigwedge_{j \in I_1} f(1, c_{1,j})) \ldots (\sigma_n X_n = \bigwedge_{j \in I_n} f(n, c_{n,j})). \]

The following results follow directly from results by Mader [14].

Lemma 4.2. Given a BES \( \mathcal{E} \) and environment \( \theta \). For all selection functions \( f \) and variables \( X \) of \( \mathcal{E} \) we have that \( (\mathcal{E}_f)^\theta(X) \) implies \( (\mathcal{E})^\theta(X) \).

Proof. By Lemma 3.16 in [14]. \( \square \)

Proposition 4.3. (Proposition 3.36 in [14].) For every BES \( \mathcal{E} \) and environment \( \theta \) there is a selection function \( f \) such that \( [\mathcal{E}_f]^{\theta} = [\mathcal{E}]^{\theta} \).

4.1 Reducing the BES problem to the \( v \)-parity loop problem

Our translation is similar to the one for conjunctive and disjunctive BESs given in [6], which produces a directed labelled graph without switches. We fix a BES \( \mathcal{E} = (\sigma_1 X_1 = \phi_1) \ldots (\sigma_n X_n = \phi_n) \) for some \( n \in \mathbb{N} \). Without loss of generality, we assume that \( \mathcal{E} \) is in standard form.

Definition 4.4. The DLSG corresponding to \( \mathcal{E} \) is a DLSG \((V, E, S, \ell)\) where \( V = \{1, \ldots, n\} \) and:

- \( E = \{(i, j) \mid \phi_i \text{ contains } X_j \text{ but not } \lor\} \);
- \( S = \{(i, j, k) \mid \phi_i = X_j \lor X_k\} \);
- \( \ell(i) = 2i + \text{sign}(\sigma_i) \) for all \( i \in V \).

Theorem 4.5. Let \( G \) be the DLSG corresponding to \( \mathcal{E} \), \( \theta \) be an arbitrary environment and \( 1 \leq i \leq n \). Then the BES-problem on \( \mathcal{E} \), \( \theta \) and \( X_i \) is polynomially equivalent to the \( i \)-parity loop problem on \( G \).
Proof. Observe that $G$ can be constructed in polynomial time. We show that $([E]θ)(X_i) = 1$ iff there is a switch setting of $G$ such that on every loop reachable from vertex $i$ the lowest label is even.

$(⇒)$ We prove this part by contraposition. Assume that for all switch settings of $G$ there is a loop reachable from $i$ on which the lowest label is odd. We have to show that $([E]θ)(X_i) = 0$. By Proposition 4.3 there is a selection function $f$ such that $[E]θ = [f]θ$. Take this $f$ and construct a switch setting $f$ of $G$ as follows, for any $(i, j, k) ∈ S$: $f(i, j, k) = 1$ if $f(i, X_j ∨ X_k) = X_j$, and $f(i, j, k) = 0$ if $f(i, X_j ∨ X_k) = X_k$. (Note that $φ_i = X_j ∨ X_k$ by construction of $G$.) Observe that the $f$-graph $G_f$ is the dependency graph of the $f$-BES $E_f$. Then by Lemma 2.6 we have $([E_f]θ)(X_i) = 0$, hence $([E]θ)(X_i) = 0$.

$(⇐)$ Assume the existence of a switch setting $f$ such that on every $f$-loop reachable from $i$ the lowest label is even. Construct a selection function $f$ as follows. For any $(i, j) ∈ E$ define $f(i, X_j) = X_j$ and for any $(i, j, k) ∈ S$ define:

$$f(i, X_j ∨ X_k) = \begin{cases} X_j & \text{if } f(i, j, k) = 1 \\ X_k & \text{if } f(i, j, k) = 0. \end{cases}$$

(Note that $φ_i$ contains the required clauses -- $X_j$ or $X_j ∨ X_k$ -- by construction of $G$.) Observe that the $f$-graph $G_f$ is the dependency graph of the $f$-BES $E_f$ and that the lowest label on every loop in $G_f$ reachable from $i$ is even. Then by Lemma 2.6 we have $([E_f]θ)(X_i) = 1$. Hence, by Lemma 4.2 $([E]θ)(X_i) = 1$. □

4.2 Reducing the $ν$-parity loop problem to the BES problem

We fix a DLSG $G = (V, E, S, ℓ)$. Without loss of generality, we assume that $V = \{1, \ldots, n\}$ for some $n ≥ 1$, and $ℓ(i) ≤ ℓ(i + 1)$ for all $1 ≤ i < n$.

Definition 4.6. The BES corresponding to $G$ is the following BES:

$$E = (σ_1X_1 = φ_1) \ldots (σ_nX_n = φ_n)$$

where, for all $1 ≤ i ≤ n$:

- $σ_i = µ$ if $ℓ(i)$ is odd and $σ_i = ν$ otherwise;
- $φ_i = \bigwedge\{X_j \mid (i, j) ∈ E\} \land \bigwedge\{(X_j ∨ X_k) \mid (i, j, k) ∈ S\}$.

Theorem 4.7. Let $1 ≤ i ≤ n$, $E$ be the BES corresponding to $G$ and $θ$ be an arbitrary environment. Then the $i$-parity loop problem on $G$ is polynomially equivalent to the BES problem on $E$, $θ$ and $X_i$.

Proof. Observe that $E$ can be constructed in polynomial time and is in 2CNF. We have to show that $([E]θ)(X_i) = 1$ iff there is a switch setting of $G$ such that on every loop reachable from vertex $i$ the lowest label is even. The proof is very similar to that of Theorem 4.5 and is therefore omitted. □
5 Conclusions

In this paper, we introduced switching graphs as a natural extension to ordinary graphs and determined the complexity of several problems on such graphs. The \( v-w \)-connection, \( v-w \)-disconnection and loop problems are solvable in polynomial time. The cancellation loop, connection and disconnection problems are NP-complete by reductions from the 3SAT problem. Finally, the \( v \)-parity loop problem is in \( \text{NP} \cap \text{co-NP} \), both by the already known result on parity games and by the direct translations to/from the BES problem presented here. The complexity of the strongly related parity loop problem is left as an open question.

Of course, by its equivalence to the BES and \( \mu \)-calculus model-checking problems, the \( v \)-parity loop problem is particularly interesting. Studying this problem and related problems in the context of switching graphs may help in answering the long open question whether all of these problems have a polynomial algorithm.

We conclude this paper by listing a number of switching graph problems of which we do not know the complexity. Resolving these may help in finding the answer to the open question mentioned above. Typically, those problems that have a polynomial solution for ordinary graphs can become hard in the setting of switching graphs. Open questions regarding undirected switching graphs are:

1. Is there a switch setting \( f \) such that the \( f \)-graph is connected?\(^2\)

2. Is there a switch setting \( f \) such that the \( f \)-graph contains an Euler tour (i.e. a path through the \( f \)-graph that traverses each edge exactly once)?

3. Is there a switch setting \( f \) such that the \( f \)-graph is planar?

4. Find a switch setting \( f \) such that the \( f \)-graph has a minimal number of connected vertices (to some vertex \( v \)), or is \( k \)-connected (i.e. after removing \( k \) edges the \( f \)-graph is still connected).

Another recent observation is that it is NP-complete to find a switch setting such that the resulting graph is bipartite [19].

Open questions regarding directed switching graphs are:

1. Is there a switch setting \( f \) such that the \( f \)-graph is a strongly connected component?\(^2\)

\(^2\)Recently, this problem was shown to be polynomially decidable [12]. In [12] it is also shown that it is almost linear to find a path from a vertex \( v \) to a vertex \( w \) in an undirected switching graph.
2. Is there a switch setting $f$ such that the $f$-graph can be topologically sorted? This is equivalent to finding a switch setting $f$ such that the $f$-graph does not contain non-trivial connected components.

3. Is there a switch setting $f$ such that the $f$-graph is a tree, has a network flow of sufficient capacity or has a path between two vertices shorter than a given $k$?

4. Given two vertices $v$ and $w$. Is there a switch setting $f$ such that an $f$-loop through $v$ and $w$ exists? We expect that this problem is NP-complete, because finding a path back from $w$ to $v$ may require to adapt the path from $v$ to $w$, leading to backtracking. A very similar problem is to find a simple (=vertex-crossing free) loop in an ordinary directed graph, and this problem is NP-complete [3].

References


