International Journal of Foundations of Computer Science
Vol. 20, No. 5 (2009) 869–886

World Scientific
www.worldscientific.com

SWITCHING GRAPHS

JAN FRISO GROOTE
J.F.Groote@tue.nl

BAS PLOEGER*
s.c.w.ploeger@tue.nl

Eindhoven University of Technology,
Department of Mathematics and Computer Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Received 25 November 2008
Accepted 15 May 2009
Communicated by Vesa Halava and Igor Potapov

Switching graphs are graphs that contain switches. A switch is a pair of edges that start in the same vertex and of which precisely one edge is enabled at any time. By using a Boolean function called a switch setting, the switches in a switching graph 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 μ-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.

Keywords: Switching graphs; parity games; Boolean equation systems; model checking.

2000 Mathematics Subject Classification: 05C38, 68Q17, 68Q60, 68R10

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) [16], model checking formulas in the modal μ-calculus [14] and finding a winning strategy in parity games [3, 18]. Especially the intriguing algorithms that use hill-climbing techniques with smartly chosen progress measures drew our attention [11, 20]. While studying the

*This author is partially supported by the Netherlands Organisation for Scientific Research (NWO) under VoLTS grant number 612.065.410.

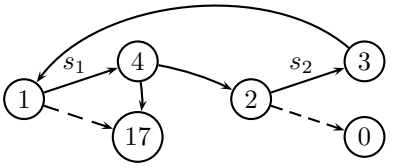


Fig. 1. A directed labelled switching graph.

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 Figure 1; in Figure 2(a) 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 Figure 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. Meinel [17] uses a slightly different notion of switching graphs, essentially omitting switches and unifying non-determinism with switches. Girard [5] proposes proof nets which closely resemble switching graphs, except that here switches move in pairs. Other notions in the literature (like switch graphs [10], bidirected graphs [2] and skew-symmetric graphs [6]) 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 μ-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 [8].

Moreover, we prove that it is impossible to synchronise switches in the context of the parity loop problem, which is the v-parity loop problem in which *all* loops are considered, not just the ones that are reachable from a given vertex v. 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 [9], but we now show that this algorithm is incorrect.

We first define switching graphs in Section 2. Next, we investigate several problems on switching graphs and provide their characterisation in terms of complexity (Section 3). In Section 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 Section 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. We also thank the anonymous referee who showed us that the loop problem can be solved in linear time.

2. Preliminaries

2.1. *Switching graphs*

Definition 1. *A directed labelled switching graph (DLSG) is a four-tuple* (V, E, S, ℓ) *where*

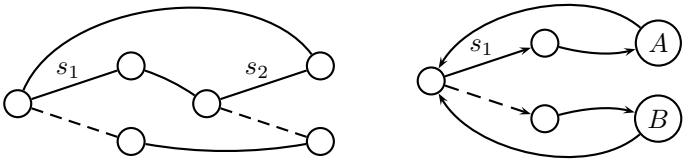
- V *is a set of vertices;*
- $E \subseteq V \times V$ *is a set of directed edges;*
- $S \subseteq V \times V \times V$ *is a set of directed switches;*
- $\ell : V \to \mathbb{N}$ *is a labelling function.*

For edges v and w, we write $v \to w$ if $(v, w) \in E$. If $(v_1, v_2, v_3) \in S$, we write $v_1 \to_1 v_2$ and $v_1 \to_0 v_3$. 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 $f : S \to \{0, 1\}$. 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 $v_i \to v_{i+1}$, 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:

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



(a) (b)

Fig. 2. An undirected (a) and a directed (b) switching graph.

A switching graph with no labelling function is simply called a *directed switching graph*. A switching graph of which the edges are *unordered* pairs, is called an *undirected* (*labelled*) *switching graph*.

An example of an undirected switching graph is shown in Figure 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.).

In Figure 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 Section 3.

Definition 2. *A* 3CNF formula *is a logical formula of the form*

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

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.*

The famous Cook–Levin theorem states that the SAT problem, *i.e.* satisfiability of any Boolean formula, is NP-complete [1, 15]. By a reduction from SAT to 3SAT, Karp showed that the same holds for the 3SAT problem [12]. 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* $\mathcal{X}$ 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:

- ϵ 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 ϕ is a *proposition formula*.

A fixed-point symbol is either μ for least fixed-point, or ν for greatest fixed-point. Proposition formulae ϕ are defined by the following grammar:

$$\phi ::= \bot \mid \top \mid X \mid \phi \wedge \phi \mid \phi \vee \phi$$

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

The BES problem is the problem of finding the *solution* of a BES. This solution is an assignment of either 0 or 1 to each variable X such that every equation is satisfied, and the assignment to X is as strong as possible if its fixed-point symbol is μ, and as weak as possible if it is ν. Moreover, the order of the equations is important: equations that come first take precedence over equations that follow. We illustrate this by an example.

Example 3. *Consider the following BES:*

$$(\mu X = Y)\ (\nu Y = X)\,.$$

Ignoring the fixed-point symbols for now, there are two candidate solutions for this BES: $(X, Y) := (0, 0)$ *and* $(X, Y) := (1, 1)$. *Because the fixed-point symbol for* X *is* μ, *we have to take the strongest possible assignment to* X, *which is* 0. *Given this assignment, we have to find the weakest possible assignment to* Y *such that* (X, Y) *is a solution, which can only be* 0. *Hence, the solution to this BES is* $(X, Y) := (0, 0)$.

Now consider the BES obtained by swapping the equations in the previous BES:

$$(\nu Y = X)\ (\mu X = Y)\,.$$

When determining the solution to this BES, we first assign to Y *and then to* X, *instead of the other way around. Because the fixed-point symbol of* Y *is* ν, *the assignment to* Y *is* 1. *Given this assignment, the strongest possible assignment to* X *such that all equations are satisfied, is* 1. *Hence, the solution to this BES is* $(X, Y) := (1, 1)$.

The solution of a BES is defined formally as follows. A *proposition environment* is a function $\theta : \mathcal{X} \to \{0,1\}$ that assigns a Boolean value to every proposition variable in $\mathcal{X}$. For any environment θ, variable X and Boolean value b, we denote by $\theta[b/X]$ the environment θ in which X is mapped to b, *i.e.* $\theta[b/X](X) = b$ and $\theta[b/X](Y) = \theta(Y)$ for all $Y \neq X$.

Definition 4. *Let θ be a proposition environment. The* solution of a BES *in the context of θ is a proposition environment defined inductively as follows, for any BES $\mathcal{E}$:*

$$\begin{aligned} [\![\epsilon]\!]\theta &= \theta \\ [\![(\sigma X = \phi)\ \mathcal{E}]\!]\theta &= [\![\mathcal{E}]\!]\theta[\phi([\![\mathcal{E}]\!]\theta[b_\sigma/X])/X] \end{aligned}$$

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

It is known that the BES problem is equivalent to the μ-calculus model-checking problem [16]. This problem was shown to be in NP ∩ co-NP by Emerson, Jutla and Sistla by showing equivalence to the non-emptiness problem of parity tree automata [3]. Later, Stirling reduced the problem to that of deciding the winner in parity games, obtaining the same complexity result [18].

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

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

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

- in *standard form* if every ϕ_i is either X_j or $X_j \vee X_k$ or $X_j \wedge X_k$;
- in *2-conjunctive normal form* (2CNF) if every ϕ_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 $X_j \vee X_k$;
- *conjunctive* if no ϕ_i contains a $\vee$;
- *disjunctive* if no ϕ_i contains a $\wedge$.

It is known that every BES can be linearly transformed to standard form such that its solution is maintained, modulo renaming of variables [16]. If $\mathcal{E}$ is conjunctive or disjunctive, the *dependency graph* of $\mathcal{E}$ is a directed labelled graph (V, E, ℓ) where $V = \{1, \ldots, n\}$, $E = \{(i,j) \mid X_j \text{ occurs in } \phi_i\}$ and $\ell(i) = 2i + \mathit{sign}(\sigma_i)$ with $\mathit{sign}(\mu) = 1$ and $\mathit{sign}(\nu) = 0$. The following result was obtained in [8] as Lemma 3.[a]

Lemma 5. *(Lemma 3 of [8].) Let θ be an environment, $\mathcal{E}$ be a conjunctive BES and G be its dependency graph. Then for any variable X_i of $\mathcal{E}$ we have that $([\![\mathcal{E}]\!]\theta)(X_i) = 0$ iff G contains a loop that is reachable from i on which the lowest label is odd.*

[a] In [8] the labelling function is a function ℓ' 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 μ. 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.

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 [8].

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 6. *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. □

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 7. *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. □

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 8. *The loop problem is solvable in linear time.*

Proof. Construct an ordinary graph G' by replacing every switch (u, v, w) in G by two edges (u, v) and (u, w), and compute the strongly connected components of G'. This can be done in linear time [19]. The answer to the loop problem on G is *yes* if and only if G' contains a non-trivial strongly connected component. □

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. Observe that we are allowed to modify the switching graph G in the following way, without affecting the answer to the problem: for every vertex $v \in V$, if v has no outgoing edges (*i.e.* there is no w such that $v \rightarrow w$, and there are no u, w such that $(v, u, w) \in S$) then we add an edge (v, v) to G and we set $\ell(v) := 2$. In this way we obtain a switching graph that is *total*, in the sense that every vertex has an outgoing edge or is the root of a switch. Hence, for the remainder of this section we can assume without loss of generality that G is total, which is convenient for the correspondence of the v-parity loop problem with solving parity games (as described below) and with solving BESs (as described in Section 4).

A parity game is played on a game graph, which is a directed, vertex-labelled graph of which every vertex has an outgoing edge and is labelled by a natural number, called a *priority*. The parity game has two players, called *Even* and *Odd*, and the set of vertices of the graph is partitioned into two blocks: V_{Even} and V_{Odd}. The game proceeds as follows. A token is placed on vertex x and is repeatedly moved along an edge of the graph from its current vertex to an adjacent vertex by one of the players. If the token is currently on a vertex in V_{Even} then it is *Even*'s turn to move the token; if it is on a vertex in V_{Odd} then it is *Odd*'s turn. By repeatedly moving the token, the players construct an infinite path in the graph. Player *Even* wins the game if the lowest priority that occurs infinitely often on the path is even, and *Odd* wins if that priority is odd.

It is not hard to see that the v-parity loop problem on a total switching graph is equivalent to the problem of finding a winning strategy for player *Even* in a parity game, starting from a given vertex x of the game graph. Hence, the v-parity loop problem is in NP $\cap$ co-NP (and UP $\cap$ co-UP) 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 Section 4.

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 and no polynomial algorithm is known.

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 parity loop problem is *yes*, whereas if the same switches are fixed to equal positions, the answer to the parity loop problem on the remaining graph is *no*. In the cancellation and connection problems (Sections 3.8 and 3.9), it is exactly this capability of synchronising switches that allows us to prove NP-completeness.

In order to formulate the non-synchronisation property we write $L(f) = 1$ for

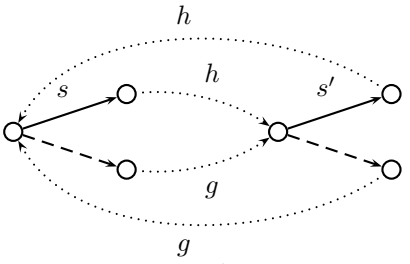


Fig. 3. The situation in the proof of Lemma 9. A dotted arrow marked by a switch setting f indicates an f-path in the switching graph.

a switch setting f if all f-loops in G have an even number as lowest label, and $L(f) = 0$ otherwise. For a given f, $L(f)$ can be computed in polynomial time by determining the strongly connected components of the f-graph [19] and checking whether the lowest label in every component is even.

Lemma 9. *Consider two switches s and s'. Suppose that for all switch settings f, $f(s) \neq f(s')$ implies $L(f) = 1$. 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 $L(g) = L(h) = 0$. Hence, there is a g-loop on which the lowest label m is odd. Consider one of these loops with minimal m. If this loop does not pass through s, then the $g[s{:=}1]$-graph also has a loop with m as lowest number, hence $L(g[s{:=}1]) = 0$ which contradicts the assumption of the lemma. Similarly, if the loop does not pass through s', then $L(g[s'{:=}1]) = 0$ which is a contradiction. Hence, the loop passes through both s and s'. Using the same line of reasoning, there is an h-loop through both s and s' with an odd number n as lowest label, and we take one of these loops with minimal n. See Figure 3.

Assume without loss of generality that $m \leq n$. There is a vertex labelled by m on the g-path from s_0 to the root of s' or on the g-path from s'_0 to the root of s. In both cases we can construct a switch setting f based on g and h such that $f(s) \neq f(s')$ and there is an f-loop that contains the label m. As $m \leq n$, there is no even label on this loop that is smaller than m. Hence, $L(f) = 0$ which contradicts the assumption of the lemma. All cases have led to a contradiction, which proves the lemma. □

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 Figure 4 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

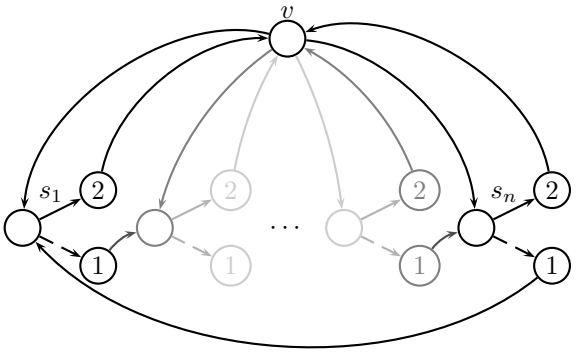


Fig. 4. Synchronising switches in the parity loop-through-v problem.

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 [9], we claimed a polynomial time algorithm for this problem. It was a weight improvement algorithm inspired by [11, 20]. However, it turns out that our algorithm and its correctness proof were incorrect. We give an example to illustrate this.

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.* 1-loop) can be f-reached from v, then the weight is †. 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 † 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 5 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-

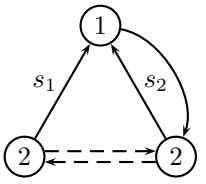


Fig. 5. Counterexample for correctness of our previously presented algorithm for the 1-2-loop problem.

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.

Nevertheless, it is not hard to see that the 1-2-loop problem is decidable in polynomial time indeed. The problem is equivalent to the problem of deciding a winning strategy for a parity game, where d – being the number of different priorities in the parity-game graph – is equal to 2. As parity games are known to be solvable in polynomial time for fixed d (see *e.g.* [11, 20]), this implies that the 1-2-loop problem is decidable in polynomial time as well.

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 10. *The cancellation loop problem is NP-complete.*

Proof. It is straightforward to see that this problem is in NP. For a given switch setting f, checking that every f-loop with even label n does not have a label $n-1$ can be done in polynomial time by computing the strongly connected components of the f-graph. 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 $\bigwedge_{i\in I}(l_{i,1}\vee l_{i,2}\vee l_{i,3})$ 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 $l_{i,1} \vee l_{i,2} \vee l_{i,3}$ 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 Figure 6 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

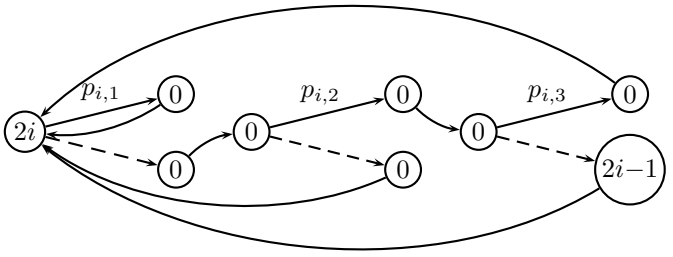


Fig. 6. Translation of a clause $p_{i,1} \vee \neg p_{i,2} \vee p_{i,3}$ for the cancellation loop problem.

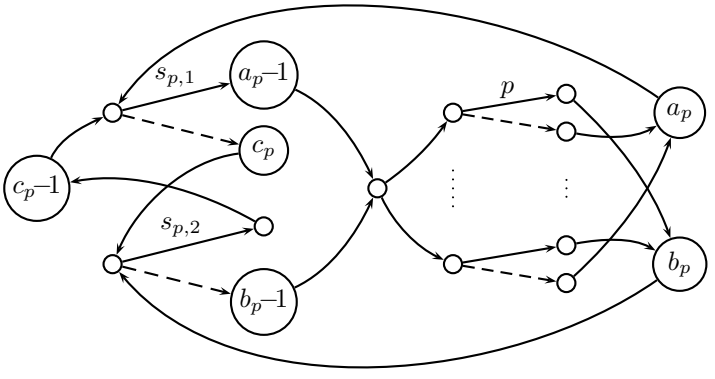


Fig. 7. Construction for keeping all switches for one proposition letter p synchronised in the translation of the cancellation loop problem.

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 Figure 7. 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 11. *The connection problem is NP-complete.*

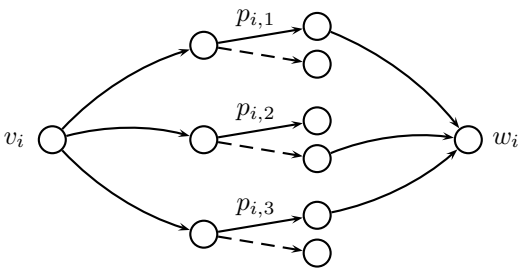


Fig. 8. Translation of a clause i of the shape $p_1 \vee \neg p_2 \vee p_3$ for the connection problem.

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:

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

for an index set I. We introduce one switch for every proposition letter occurring in the formula. We translate each clause $l_{i,1} \vee l_{i,2} \vee l_{i,3}$ 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 Figure 8 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. □

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 12. *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:

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

for an index set I. We introduce a switch for every *occurrence* of a proposition letter in a clause. For each clause $l_{i,1} \vee l_{i,2} \vee l_{i,3}$ 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

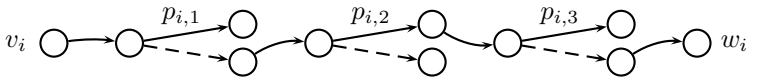


Fig. 9. Translation of a clause $p_{i,1} \vee \neg p_{i,2} \vee p_{i,3}$ for the disconnection problem.

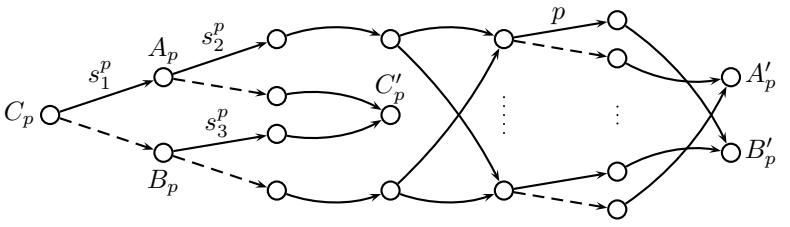


Fig. 10. Construction for keeping all switches for one proposition letter p synchronised in the translation of the disconnection problem.

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 Figure 9 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.

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 Figure 10. 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 Figure 10 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 setting f.

Definition 13. *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 $j \in I_i$: $f(i, c_{i,j}) = X_k$ if $c_{i,j} = X_k$, and either $f(i, c_{i,j}) = X_k$ or $f(i, c_{i,j}) = X_l$ if $c_{i,j} = X_k \vee X_l$. 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 [16].

Lemma 14. *Given a BES $\mathcal{E}$ and environment θ. 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 of [16]. □

Proposition 15. *(Proposition 3.36 of [16].) For every BES $\mathcal{E}$ and environment θ 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 [7], 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 16. *The* DLSG corresponding to $\mathcal{E}$ *is a DLSG (V, E, S, ℓ) where $V = \{1, \ldots, n\}$ and:*

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

Theorem 17. *Let G be the DLSG corresponding to $\mathcal{E}$, θ be an arbitrary environment and $1 \leq i \leq n$. Then the BES-problem on $\mathcal{E}$, θ 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 $([\![\mathcal{E}]\!]\theta)(X_i) = 1$ iff there is a switch setting of G such that on every loop reachable from vertex i the lowest label is even.

($\Rightarrow$) 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 $([\![\mathcal{E}]\!]\theta)(X_i) = 0$. By Proposition 15 there is a selection function f such that $[\![\mathcal{E}_f]\!]\theta = [\![\mathcal{E}]\!]\theta$. Take this f and construct a switch setting $\hat{f}$ of G as follows,

for any $(i,j,k) \in S$: $\hat{f}(i,j,k) = 1$ if $f(i, X_j \vee X_k) = X_j$, and $\hat{f}(i,j,k) = 0$ if $f(i, X_j \vee X_k) = X_k$. (Note that $\phi_i = X_j \vee X_k$ by construction of G.) Observe that the $\hat{f}$-graph $G_{\hat{f}}$ is the dependency graph of the f-BES $\mathcal{E}_f$. Then by Lemma 5 we have $([\![\mathcal{E}_f]\!]\theta)(X_i) = 0$, hence $([\![\mathcal{E}]\!]\theta)(X_i) = 0$.

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

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

(Note that ϕ_i contains the required clauses – X_j or $X_j \vee X_k$ – by construction of G.) Observe that the $\hat{f}$-graph $G_{\hat{f}}$ is the dependency graph of the f-BES $\mathcal{E}_f$ and that the lowest label on every loop in $G_{\hat{f}}$ reachable from i is even. Then by Lemma 5 we have $([\![\mathcal{E}_f]\!]\theta)(X_i) = 1$. Hence, by Lemma 14, $([\![\mathcal{E}]\!]\theta)(X_i) = 1$. □

4.2. *Reducing the v-parity loop problem to the BES problem*

We fix a DLSG $G = (V, E, S, \ell)$. Without loss of generality, we assume that $V = \{1, \ldots, n\}$ for some $n \geq 1$, $\ell(i) \leq \ell(i+1)$ for all $1 \leq i < n$, and G is total as described in Section 3.4.

Definition 18. *The* BES corresponding to G *is the following BES:*

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

where, for all $1 \leq i \leq n$*:*

- $\sigma_i = \mu$ *if* $\ell(i)$ *is odd and* $\sigma_i = \nu$ *otherwise;*
- $\phi_i = \bigwedge\{X_j \mid (i,j) \in E\} \ \wedge \ \bigwedge\{(X_j \vee X_k) \mid (i,j,k) \in S\}$.

Theorem 19. *Let* $1 \leq i \leq n$*,* $\mathcal{E}$ *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* $\mathcal{E}$*,* θ *and* X_i*.*

Proof. Observe that $\mathcal{E}$ can be constructed in polynomial time and is in 2CNF. We have to show that $([\![\mathcal{E}]\!]\theta)(X_i) = 1$ 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 17 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 NP $\cap$ 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 μ-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) 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-edge-connected (*i.e.* at least k edges have to be removed in order for the f-graph to become disconnected).

Recently, problem (1) was shown to be polynomially decidable [13]. In the same paper, it is shown that it is almost linear to find a path from a vertex v to a vertex w in an undirected switching graph, and it is observed that the problem of finding a switch setting such that the resulting graph is bipartite, is NP-hard.

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) 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 (*i.e.* vertex-crossing free) loop in an ordinary directed graph, and this problem is NP-complete [4].

References

[1] Cook, S.A., *The Complexity of Theorem Proving Procedures*, in: *Proc. 3rd ACM Symposium on Theory of Computing*, ACM (1971), pp. 151–158.

[2] Edmonds, J. and E.L. Johnson, *Matching: A well-solved class of integer linear programs*, in: *Combinatorial Optimization*, LNCS 2570 (2003), pp. 27–30.

[3] Emerson, E.A., C.S. Jutla and A.P. Sistla, *On model-checking for fragments of μ-calculus*, in: *Proc. CAV 1993*, LNCS 697 (1993), pp. 385–396.

[4] Fortune, S., J. Hopcroft and J. Wyllie, *The directed subgraph homeomorphism problem*, Theoretical Computer Science **10** (1980), pp. 111–121.

[5] Girard, J.-Y., *Linear logic*, Theoretical Computer Science **50** (1987), pp. 1–102.

[6] Goldberg, A. V. and A.V. Karzanov, *Path problems in skew-symmetric graphs*, Combinatorica **16** (1996), pp. 353–382.

[7] Groote, J.F. and M.K. Keinänen, *A sub-quadratic algorithm for conjunctive and disjunctive BESs*, CS-Report 04-13, Technische Universiteit Eindhoven (2004).

[8] Groote, J.F. and M.K. Keinänen, *Solving disjunctive/conjunctive Boolean equation systems with alternating fixed points*, in: *Proc. TACAS 2004*, LNCS 2988 (2004), pp. 436–450.

[9] Groote, J.F. and B. Ploeger, *Switching Graphs*, in: *Proc. RP 2008*, ENTCS (2008), pp. 113–129.

[10] Hochstein, J.M. and K. Weihe, *Edge-disjoint routing in plane switch graphs in linear time*, Journal of the ACM **51** (2004), pp. 636–670.

[11] Jurdziński, M., *Small progress measures for solving parity games*, in: *Proc. STACS 2000*, LNCS 1770 (2000), pp. 290–301.

[12] Karp, R.M., *Reducibility among combinatorial problems*, in: R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, Plenum Press, 1972 pp. 85–103.

[13] Katz, B., I. Rutter and G. Woeginger, *An algorithmic study of switch graphs.* Accepted for publication in: *Proc. WG 2009*, LNCS (2009).

[14] Kozen, D., *Results on the propositional μ-calculus*, Theoretical Computer Science **27** (1983), pp. 333–354.

[15] Levin, L., *Universal search problems*, Problemy Peredachi Informatsii **9**:3 (1973), pp. 265–266. In Russian.

[16] Mader, A., "Verification of Modal Properties Using Boolean Equation Systems," Ph.D. thesis, Technische Universität München (1997).

[17] Meinel, C., *Switching graphs and their complexity*, in: *Proc. MFCS 1989*, LNCS 379 (1989), pp. 350–359.

[18] Stirling, C., *Model checking and other games* (1996), notes for Mathfit Workshop on Finite Model Theory.

[19] Tarjan, R., *Depth-first search and linear graph algorithms*, SIAM Journal on Computing **1** (1972), pp. 146–160.

[20] Vöge, J. and M. Jurdziński, *A discrete strategy improvement algorithm for solving parity games (extended abstract)*, in: *Proc. CAV 2000*, LNCS 1855 (2000), pp. 202–215.