#P-completeness of Counting Update Digraphs, Cacti, and Series-Parallel Decomposition Method

Kévin Perrot(✉), Sylvain Sené, and Lucas Venturini

Université publique, Marseille, France
kevin.perrot@lis-lab.fr, sylvain.sene@lis-lab.fr,
lucas.venturini@ens-lyon.fr

Abstract. Automata networks are a very general model of interacting entities, with applications to biological phenomena such as gene regulation. In many contexts, the order in which entities update their state is unknown, and the dynamics may be very sensitive to changes in this schedule of updates. Since the works of Aracena et al., it is known that update digraphs are pertinent objects to study non-equivalent block-sequential update schedules. We prove that counting the number of equivalence classes, that is a tight upper bound on the synchronism sensitivity of a given network, is #P-complete. The problem is nevertheless computable in quasi-quadratic time for oriented cacti, and for oriented series-parallel graphs thanks to a decomposition method.

1 Introduction

Since their introduction by McCulloch and Pitts in the 1940s through the well known formal neural networks [20], automata networks (ANs) are a general model of interacting entities in finite state spaces. The field has important contributions to computer science, with Kleene's finite state automata [17], linear shift registers [14] and linear networks [12]. At the end of the 1960s, Kauffman and Thomas (independently) developed the use of ANs for the modeling of biological phenomena such as gene regulation [16,28], providing a fruitful theoretical framework [26].

ANs can be considered as a collection of local functions (one per component), and influences among components may be represented as a so called *interaction digraph*. In many applications the order of components update is a priori unknown, and different schedules may greatly impact the dynamical properties of the system. It is known since the works of Aracena *et al.* in [4] that *update*

Mainly funded by our salaries as French State agents (affiliated to Laboratoire Cogitamus (CN), Aix-Marseille Univ., Univ. de Toulon, CNRS, LIS, Marseille, France (KP, SS and LV), Univ. Côte d'Azur, CNRS, I3S, Sophia Antipolis, France (KP), and École normale supérieure de Lyon, Computer Science department, Lyon, France (LV)), and secondarily by the projects ANR-18-CE40-0002 FANs, ECOS-CONICYT C16E01, STIC AmSud 19-STIC-03 (Campus France 43478PD).

The original version of this chapter was revised: The fictitious author was removed. The correction is available at https://doi.org/10.1007/978-3-030-51466-2_34

M. Anselmo et al. (Eds.): CiE 2020, LNCS 12098, pp. 326–338, 2020.
https://doi.org/10.1007/978-3-030-51466-2_30

$$f_1(x) = 0 \qquad f_3(x) = x_1 \vee \neg x_4$$
$$f_2(x) = \neg x_4 \qquad f_4(x) = x_3$$

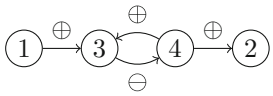

Fig. 1. Example of an AN f on the Boolean alphabet $[q] = \{1, 2\}$ (conventionally renamed $\{0, 1\}$), its interaction digraph, and a $\{\oplus, \ominus\}$-labeling $\mathrm{lab}_B = \mathrm{lab}_{B'}$ corresponding to the two equivalent update schedules $B = (\{1, 2, 3\}, \{4\})$ and $B' = (\{1, 3\}, \{2, 4\})$.

digraphs (consisting of labeling the arcs of the interaction digraphs with $\oplus$ and $\ominus$) capture the correct notion to consider a biologically meaningful family of update schedules called *block-sequential* in the literature. Since another work of Aracena *et al.* [3] a precise characterization of the valid labelings has been known, but their combinatorics remains puzzling. After formal definitions and known results in Sects. 2 and 3, we propose in Sect. 4 an explanation for this difficulty, through the lens of computational complexity theory: we prove that counting the number of update digraphs (valid $\{\oplus, \ominus\}$-labelings) is #P-complete. In Sect. 5 we consider the problem restricted to the family of oriented cactus graphs and give a $\mathcal{O}(n^2 \log n \log\log n)$ time algorithm, and finally in Sect. 6 we present a decomposition method leading to a $\mathcal{O}(n^2 \log^2 n \log\log n)$ algorithm for oriented series-parallel graphs.

2 Definitions

Given a finite alphabet $[q] = \{1, \ldots, q\}$, an *automata network* (AN) of size n is a function $f : [q]^n \to [q]^n$. We denote x_i the *component* $i \in [n]$ of some *configuration* $x \in [q]^n$. ANs are more conveniently seen as n *local functions* $f_i : [q]^n \to [q]$ describing the update of each component, *i.e.* with $f_i(x) = f(x)_i$. The *interaction digraph* captures the effective dependencies among components, and is defined as the digraph $G_f = ([n], A_f)$ with

$$(i, j) \in A_f \iff f_j(x) \neq f_j(y) \text{ for some } x, y \in [q]^n \text{ with } x_{i'} = y_{i'} \text{ for all } i' \neq i.$$

It is well known that the schedule of components update may have a great impact on the dynamics [5,13,21,23]. A *block-sequential update schedule* $B = (B_1, \ldots, B_t)$ is an ordered partition of $[n]$, defining the following dynamics

$$f^{(B)} = f^{(B_t)} \circ \cdots \circ f^{(B_2)} \circ f^{(B_1)} \quad \text{with} \quad f^{(B_i)}(x)_j = \begin{cases} f_j(x) & \text{if } j \in B_i \\ x_j & \text{if } j \notin B_i \end{cases}$$

i.e., parts are updated sequentially one after the other, and components within a part are updated in parallel. For the parallel update schedule $B^{\mathtt{par}} = ([n])$, we have $f^{(B^{\mathtt{par}})} = f$. Block-sequential update schedules are a classical family of update schedules considered in the literature, because they are perfectly fair: every local function is applied exactly once during each step. Equipped with an

update schedule, $f^{(B)}$ is a discrete dynamical system on $[q]^n$. In the following we will shortly say *update schedule* to mean *block-sequential update schedule*.

It turns out quite intuitively that some update schedules will lead to the same dynamics, when the ordered partitions are very close and the difference relies on components far apart in the interaction digraph (see an example on Fig. 1). Aracena *et al.* introduced in [4] the notion of *update digraph* to capture this fact. To an update schedule one can associate its *update digraph*, which is a $\{\oplus, \ominus\}$-labeling of the arcs of the interaction digraph of the AN, such that (i, j) is negative ($\ominus$) when i is updated strictly before j, and positive ($\oplus$) otherwise. Formally, given an update schedule $B = (B_1, \ldots, B_t)$,

$$\forall (i,j) \in A_f : \mathrm{lab}_B((i,j)) = \begin{cases} \oplus & \text{if } i \in B_{t_i} \text{ and } j \in B_{t_j} \text{ with } t_i \geq t_j, \\ \ominus & \text{if } i \in B_{t_i} \text{ and } j \in B_{t_j} \text{ with } t_i < t_j. \end{cases}$$

Remark 1. Loops are always labeled $\oplus$, hence we consider our digraphs *loopless*.

The following result has been established: given two update schedules, if the relative order of updates among all adjacent components are identical, then the dynamics are identical. It leads naturally to an *equivalence relation* on update schedules, at the heart of the present work.

Theorem 1 ([4]). *Given an AN f and two update schedules B, B', if $\mathrm{lab}_B = \mathrm{lab}_{B'}$ then $f^{(B)} = f^{(B')}$. Hence we denote $B \equiv B'$ if and only if $\mathrm{lab}_B = \mathrm{lab}_{B'}$.*

It is very important to note that, though every update schedule corresponds to a $\{\oplus, \ominus\}$-labeling of G_f, the reciprocal of this fact is not true. For example, a cycle with all arcs labeled $\ominus$ would lead to a contradiction where components are updated strictly before themselves. Aracena *et al.* gave a precise characterization of *valid* update digraphs (*i.e.* the ones corresponding to at least one update schedule).

Theorem 2 ([3]). *A labeling function $\mathrm{lab} : A \to \{\oplus, \ominus\}$ is valid if and only if there is no cycle $(i_0, i_1, \ldots, i_k)$, with $i_0 = i_k$, of length $k > 0$ such that both:*

- $\forall\ 0 \leq j < k : \mathrm{lab}((i_j, i_{j+1})) = \oplus \vee \mathrm{lab}((i_{j+1}, i_j)) = \ominus$,
- $\exists\ 0 \leq j < k : \mathrm{lab}((i_{j+1}, i_j)) = \ominus$.

In words, the multidigraph where the orientation of negative arcs is reversed, does not contain a cycle with at least one negative arc (forbidden cycle).

As a corollary, one can decide in polynomial time whether a labeling is valid (**Valid-UD Problem** is in P). We are interested in the following.

Update Digraphs Counting (#UD)
Input: a digraph $G = (V, A)$.
Output: $\#\mathtt{UD}(G) = |\{\mathrm{lab} : A \to \{\ominus, \oplus\} \mid \mathrm{lab} \text{ is valid}\}|$.

The following definition is motivated by Theorem 2.

Definition 1. *Given a digraph $G = (V, A)$, let $\bar{G} = (V, \bar{A})$ denote the undirected multigraph underlying G,* i.e. *with an edge $\{i, j\} \in \bar{A}$ for each $(i, j) \in A$.*

Remark 2. We can restrict our study to connected digraphs (that is, such that $\bar{G}$ is connected), because according to Theorem 2 the only invalid labelings contain (forbidden) cycles. Given some G with $V_1, \ldots, V_k$ its connected components, and $G[V_i]$ the subdigraph induced by V_i, we straightforwardly have $\#\mathtt{UD}(G) = \prod_{i \in [k]} \#\mathtt{UD}(G[V_i])$, and this decomposition can be computed in linear time from folklore algorithms.

Theorem 3. **#UD** *is in* #P.

Proof. The following non-deterministic algorithm runs in polynomial time (**Valid-UD Problem** is in P), and its number of accepting branches equals $\#\mathtt{UD}(G)$:

1. guess a labeling $\mathrm{lab} : A \to \{\oplus, \ominus\}$ (polynomial space),
2. accept if lab is valid, otherwise reject.

□

3 Further Known Results

The consideration of update digraphs has been initiated by Aracena *et al.* in 2009 [4], with their characterization (Theorem 2) in [3]. In Sect. 4 we will present a problem closely related to **#UD** that has been proven to be NP-complete in [3], **UD Problem**, and bounds that we can deduce on **#UD** (Corollary 1, from [2]). In [1] the authors present an algorithm to enumerate update digraphs, and prove its correctness. They also consider a surprisingly complex question: given an AN f, knowing whether there exist two block-sequential update schedules B, B' such that $f^{(B)} \neq f^{(B')}$, is NP-complete. The value of $\#\mathtt{UD}(G)$ is known to be $3^n - 2^{n+1} + 2$ for bidirected cycles on n vertices [23], and to equal $n!$ if and only if the digraph is a tournament on n vertices [2].

4 Counting Update Digraphs Is #P-complete

The authors of [3] have exhibited an insightful relation between valid labelings and feedback arc sets of a digraph. We recall that a *feedback arc set* (FAS) of $G = (V, A)$ is a subset of arcs $F \subseteq A$ such that the digraph $(V, A \setminus F)$ is acyclic, and its size is $|F|$. This relation is developed inside the proof of NP-completeness of the following decision problem. We reproduce it as a Lemma.

Update Digraph Problem (UD Problem)
Input: a digraph $G = (V, A)$ and an integer k.
Question: does there exist a valid labeling of size at most k?

The *size* of a labeling is its number of $\oplus$ labels. It is clear that minimizing the number of $\oplus$ labels (or equivalently maximizing the number of $\ominus$ labels) is the difficult direction, the contrary being easy because $\mathrm{lab}(a) = \oplus$ for all $a \in A$ is always valid (and corresponds to the parallel update schedule $B^{\mathtt{par}}$).

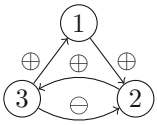

Fig. 2. $F = \{(1,2),(2,3),(3,1)\}$ is a FAS, but the corresponding labeling is not valid: component 3 is updated prior to 2, 1 not prior to 2, and 3 not prior to 1, which is impossible.

Lemma 1 (appears in [3, Theorem 16]**).** *There exists a bijection between minimal valid labelings and minimal feedback arc sets of a digraph $G = (V, A)$.*

Proof (sketch). To get the bijection, we simply identify a labeling lab with its set of arcs labeled $\oplus$, denoted $F_{\mathrm{lab}} = \{a \in A \mid \mathrm{lab}(a) = \oplus\}$. □

Any valid labeling corresponds to a FAS, and every minimal FAS corresponds to a valid labeling, hence the following bounds hold. The strict inequality for the lower bound comes from the fact that labeling all arcs $\oplus$ does not give a minimal FAS, as noted in [2] where the authors also consider the relation between update digraphs (valid labelings) and feedback arc sets, but from another perspective.

Corollary 1 ([3]). *For any digraph G, let $\#\mathtt{FAS}(G)$ and $\#\mathtt{MFAS}(G)$ be the number of FAS and minimal FAS of G, then $\#\mathtt{MFAS}(G) < \#\mathtt{UD}(G) \leq \#\mathtt{FAS}(G)$.*

From Lemma 1 and results on the complexity of FAS counting problems presented in [22], we have the following corollary (minimum FAS are minimal, hence the identity is a parsimonious reduction from the same problems on FAS).

Corollary 2. *Counting the number of valid labelings of minimal size is $\#\mathsf{P}$-complete, and of minimum size is $\#\cdot\mathsf{OptP}[\log \mathsf{n}]$-complete.*

However the correspondence given in Lemma 1 does not hold in general: there may exist some FAS F such that lab with $F_{\mathrm{lab}} = F$ is not a valid labeling (see Fig. 2 for an example). As a consequence we do not directly get a counting reduction to **#UD**. It nevertheless holds that **#UD** is $\#\mathsf{P}$-hard, with the following reduction.

Theorem 4. #UD *is $\#\mathsf{P}$-hard.*

Proof. We present a (polynomial time) parsimonious reduction from the problem of counting the number of acyclic orientations of an undirected graph, proven to be $\#\mathsf{P}$-hard in [18].

Given an undirected graph $G = (V, E)$, let $\prec$ denote an arbitrary total order on V. Construct the digraph $G' = (V, A)$ with A the orientation of E according to $\prec$, *i.e.* $(u, v) \in A \iff \{u, v\} \in E$ and $u \prec v$. An example is given on Fig. 3 (left). A key property is that G' is acyclic, because A is constructed from an order $\prec$ on V (a cycle would have at least one arc (u, v) with $v \prec u$).

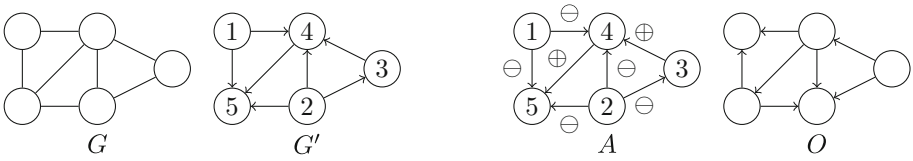

Fig. 3. Left: an undirected graph G (instance of acyclic orientation counting), and the obtained digraph G' (instance of update digraph counting). Right: a valid labeling A of G', and the corresponding orientation O of G.

We claim that there is a bijection between the valid labelings of G' and the acyclic orientations of G: to a valid labeling $\text{lab} : A \to \{\oplus, \ominus\}$ of G' we associate the orientation

$$\begin{aligned} O = \quad & \{(u,v) \mid (u,v) \in A \text{ and } \text{lab}((u,v)) = \oplus\} \\ \cup\ & \{(v,u) \mid (u,v) \in A \text{ and } \text{lab}((u,v)) = \ominus\}. \end{aligned}$$

First remark that O is indeed an orientation of E: each edge of E is transformed into an arc of A, and each arc of A is transformed into an arc of O. An example is given on Fig. 3 (right). Now observe that O is exactly obtained from G' by reversing the orientation of arcs labeled $\ominus$ by lab. Furthermore, a cycle in O must contain at least one arc labeled $\ominus$ by lab, because G' is acyclic and $\oplus$ labels copy the orientation of G'. The claim therefore follows directly from the characterization of Theorem 2. □

5 Quasi-quadratic Time Algorithm for Oriented Cacti

The difficulty of counting the number of update digraphs comes from the interplay between various possible cycles, as is assessed by the parsimonious reduction from acyclic orientations counting problem to #**UD**. Answering the problem for an oriented tree with m arcs is for example very simple: all of the 2^m labelings are valid. Cactus undirected graphs are defined in terms of very restricted entanglement of cycles, which we can exploit to compute the number of update digraphs for any orientation of its edges.

Definition 2. *A* cactus *is a connected undirected graph such that any vertex (or equivalently any edge) belongs to at most one simple cycle (cycle without vertex repetition). An* oriented cactus G *is a digraph such that* $\bar{G}$ *is a cactus.*

Cacti may intuitively be thought as trees with cycles. This is indeed the idea behind the *skeleton* of a cactus introduced in [9], via the following notions:

- a *c-vertex* is a vertex of degree two included in exactly one cycle,
- a *g-vertex* is a vertex not included in any cycle,
- remaining vertices are *h-vertices*,

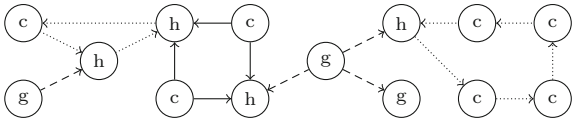

Fig. 4. An oriented cactus G, with $\{c, g, h\}$-vertex labels. Graft arcs are dashed, cycles forming directed cycles are dotted, and cycles not forming directed cycles are solid. Theorem 5 counts $\#\mathtt{UD}(G) = 2^1 2^3 (2^3 - 1)(2^5 - 1)(2^4 - 2) = 48\,608$.

and a *graft* is a maximal subtree of g- and h-vertices with no two h-vertices belonging to the same cycle. Then a cactus can be decomposed as grafts and cycles (two classes called *blocks*), connected at h-vertices according to a tree skeleton. These notions directly apply to oriented cacti (see an example on Fig. 4).

Theorem 5. #UD *is computable in time* $\mathcal{O}(n^2 \log n \log\log n)$ *for oriented cacti.*

Proof. The result is obtained from the skeleton of an oriented cactus G, since potential forbidden cycles are limited to within blocks of the skeleton. From this independence, any union of valid labelings on blocks is valid, and we have the product

$$\#\mathtt{UD}(G) = \prod_{H \in \mathcal{G}} 2^{|H|} \prod_{H \in \vec{\mathcal{C}}} (2^{|H|} - 1) \prod_{H \in \mathcal{C}} (2^{|H|} - 2)$$

where $\mathcal{G}$ is the set of grafts of G, $\vec{\mathcal{C}}$ is the set of cycles forming directed cycles, $\mathcal{C}$ is the set of cycles not forming directed cycles, and $|H|$ is the number of arcs in block H. Indeed, grafts cannot create forbidden cycles hence any $\{\oplus, \ominus\}$-labeling will be valid, cycles forming a directed cycle can create exactly one forbidden cycle (with $\ominus$ labels on all arcs), and cycles not forming a directed cycle can create exactly two forbidden cycles (one for each possible direction of the cycle). In a first step the skeleton of a cactus can be computed in linear time [9]. Then, since the size n of the input is equal (up to a constant) to the number of arcs, the size of the output contains $\mathcal{O}(n)$ bits (upper bounded by the number of $\{\oplus, \ominus\}$-labelings), thus naively we have $\mathcal{O}(n)$ terms, each of $\mathcal{O}(n)$ bits, and the $\mathcal{O}(n \log n \log\log n)$ Schönhage–Strassen integer multiplication algorithm gives the result. □

Remark 3. Assuming multiplications to be done in constant time in the above result would be misleading, because we are multiplying integers having a number of digits in the magnitude of the input size. Also, the result may be slightly strengthened by considering the $\mathcal{O}(n \log n \; 2^{2 \log^* n})$ algorithm by Fürer in 2007.

6 Series-Parallel Decomposition Method

In this section we present a divide and conquer method in order to solve **#UD**, *i.e.* in order to count the number of valid labelings (update digraphs) of a given

digraph. What will be essential in this decomposition method is not the orientation of arcs, but rather the topology of the underlying undirected (multi)graph $\bar{G}$. The (de)composition is based on defining two endpoints on our digraphs, and composing them at their endpoints. It turns out to be closely related to series-parallel graphs first formalized to model electric networks in 1892 [19]. In Subsect. 6.1 we present the operations of composition, and in Subsect. 6.2 we show how it applies to the family of oriented series-parallel graphs.

6.1 Sequential, Parallel, and Free Compositions

Let us first introduce some notations and terminology on the characterization of valid labelings provided by Theorem 2. Given $\mathrm{lab} : A \to \{\oplus, \ominus\}$, we denote $\tilde{G}_{\mathrm{lab}} = (V, \tilde{A})$ the multidigraph obtained by reversing the orientation of negative arcs:

$$(i,j) \in \tilde{A} \iff \begin{array}{c}(i,j) \in A \text{ and } \mathrm{lab}((i,j)) = \oplus, \\ \text{or } (j,i) \in A \text{ and } \mathrm{lab}((j,i)) = \ominus.\end{array}$$

For simplicity we abuse the notation and still denote lab the labeling of the arcs of $\tilde{G}_{\mathrm{lab}}$ (arcs keep their label from G to $\tilde{G}_{\mathrm{lab}}$). From Theorem 2, lab is a valid labeling if and only if $\tilde{G}_{\mathrm{lab}}$ does not contain any cycle with at least one arc labeled $\ominus$, called *forbidden cycle* (it may contain cycles with all arcs labeled $\oplus$). A path from i to j in $\tilde{G}_{\mathrm{lab}}$ is called *negative* if it contains at least one arc labeled $\ominus$, and *positive* otherwise.

Definition 3. *A* source-sink labeled graph (ss-graph) (G, α, β) *is a multigraph G with two distinguished vertices $\alpha \neq \beta$. A triple (G, α, β) with G a digraph such that $(\bar{G}, \alpha, \beta)$ is a ss-graph, is called an* oriented ss-graph (oss-graph).

We can decompose the set of update digraphs (denoted $\mathtt{UD}(G) = \{\mathrm{lab} : A \to \{\oplus, \ominus\} \mid \mathrm{lab} \text{ is valid}\}$) into an oss-graph (G, α, β), based on the follow sets.

$$\mathtt{UD}(G)^{+}_{\alpha\to\beta} = \{\mathrm{lab} \in \mathtt{UD}(G) \mid \text{there } \textbf{exists} \text{ a path from } \alpha \text{ to } \beta \text{ in } \tilde{G}_{\mathrm{lab}}, \text{ and } \textbf{all} \text{ paths from } \alpha \text{ to } \beta \text{ in } \tilde{G}_{\mathrm{lab}} \text{ are } \textbf{positive}\}$$

$$\mathtt{UD}(G)^{-}_{\alpha\to\beta} = \{\mathrm{lab} \in \mathtt{UD}(G) \mid \text{there } \textbf{exists} \text{ a } \textbf{negative} \text{ path from } \alpha \text{ to } \beta \text{ in } \tilde{G}_{\mathrm{lab}}\}$$

$$\mathtt{UD}(G)^{\varnothing}_{\alpha\to\beta} = \{\mathrm{lab} \in \mathtt{UD}(G) \mid \text{there exist } \textbf{nopath} \text{ from } \alpha \text{ to } \beta \text{ in } \tilde{G}_{\mathrm{lab}}\}$$

We define analogously $\mathtt{UD}(G)^{+}_{\beta\to\alpha}$, $\mathtt{UD}(G)^{-}_{\beta\to\alpha}$, $\mathtt{UD}(G)^{\varnothing}_{\beta\to\alpha}$, and partition $\mathtt{UD}(G)$ as:

1. $\mathtt{UD}(G)^{+,+}_{\alpha,\beta} = \mathtt{UD}(G)^{+}_{\alpha\to\beta} \cap \mathtt{UD}(G)^{+}_{\beta\to\alpha}$
2. $\mathtt{UD}(G)^{+,\varnothing}_{\alpha,\beta} = \mathtt{UD}(G)^{+}_{\alpha\to\beta} \cap \mathtt{UD}(G)^{\varnothing}_{\beta\to\alpha}$
3. $\mathtt{UD}(G)^{-,\varnothing}_{\alpha,\beta} = \mathtt{UD}(G)^{-}_{\alpha\to\beta} \cap \mathtt{UD}(G)^{\varnothing}_{\beta\to\alpha}$
4. $\mathtt{UD}(G)^{\varnothing,+}_{\alpha,\beta} = \mathtt{UD}(G)^{\varnothing}_{\alpha\to\beta} \cap \mathtt{UD}(G)^{+}_{\beta\to\alpha}$
5. $\mathtt{UD}(G)^{\varnothing,-}_{\alpha,\beta} = \mathtt{UD}(G)^{\varnothing}_{\alpha\to\beta} \cap \mathtt{UD}(G)^{-}_{\beta\to\alpha}$
6. $\mathtt{UD}(G)^{\varnothing,\varnothing}_{\alpha,\beta} = \mathtt{UD}(G)^{\varnothing}_{\alpha\to\beta} \cap \mathtt{UD}(G)^{\varnothing}_{\beta\to\alpha}$

Notice that the three missing combinations, $\mathtt{UD}(G)^{+,-}_{\alpha,\beta}$, $\mathtt{UD}(G)^{-,+}_{\alpha,\beta}$ and $\mathtt{UD}(G)^{-,-}_{\alpha,\beta}$, would always be empty because such labelings contain a forbidden cycle. For

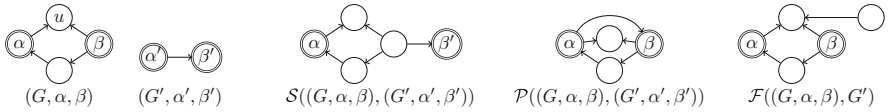

Fig. 5. Example of series and parallel compositions, and a free composition at u, β'.

convenience let us denote $S = \{(+,+),(+,\varnothing),(-,\varnothing),(\varnothing,+),(\varnothing,-),(\varnothing,\varnothing)\}$. Given any oss-graph (G,α,β) we have

$$\#\mathtt{UD}(G) = \sum_{(s,t)\in S} \#\mathtt{UD}(G)^{s,t}_{\alpha,\beta} \tag{1}$$

where $\#\mathtt{UD}(G)^{s,t}_{\alpha,\beta} = |\mathtt{UD}(G)^{s,t}_{\alpha,\beta}|$. Oss-graphs may be thought as black boxes, we will compose them using the values of $\#\mathtt{UD}(G)^{s,t}_{\alpha,\beta}$, regardless of their inner topologies.

Definition 4. *We define three types of compositions (see Fig. 5).*

- *The* series composition *of two oss-graphs* (G,α,β) *and* (G',α',β') *with* $V \cap V' = \emptyset$*, is the oss-graph* $\mathcal{S}((G,\alpha,\beta),(G',\alpha',\beta')) = (D,\alpha,\beta')$ *with* D *the one-point join of* G *and* G' *identifying components* β,α' *as one single component.*
- *The* parallel composition *of two oss-graphs* (G,α,β) *and* (G',α',β') *with* $V \cap V' = \emptyset$*, is the oss-graph* $\mathcal{P}((G,\alpha,\beta),(G',\alpha',\beta')) = (D,\alpha,\beta)$ *with* D *the two-points join of* G *and* G' *identifying components* α,α' *and* β,β' *as two single components.*
- *The* free composition at v,v' *of an oss-graph* $(G = (V,A),\alpha,\beta)$ *and a digraph* $G' = (V',A')$ *with* $V \cap V' = \emptyset$*,* $v \in V$*,* $v' \in V'$*, is the oss-graph* $\mathcal{F}((G,\alpha,\beta),G') = (D,\alpha,\beta)$ *with* D *the one-point join of* G *and* G' *identifying* v,v' *as one single component.*

Remark that the three types of compositions from Definition 4 also apply to (undirected) ss-graph $(\bar{G},\alpha,\beta)$. Series and free compositions differ on the endpoints of the obtained oss-graph, which has important consequences on counting the number of update digraphs, as stated in the following results. We will see in Theorem 6 from Subsect. 6.2 that both series and free compositions are needed in order to decompose the family of (general) *oriented series-parallel graphs* (to be defined).

Lemma 2. *For* $(D,\alpha,\beta') = \mathcal{S}((G,\alpha,\beta),(G',\alpha',\beta'))$*, the values of* $\#\mathtt{UD}(D)^{s,t}_{\alpha,\beta'}$ *for all* $(s,t) \in S$ *can be computed in time* $\mathcal{O}(n \log n \log\log n)$ *(with* n *the binary length of the values) from the values of* $\#\mathtt{UD}(G)^{s,t}_{\alpha,\beta}$ *and* $\#\mathtt{UD}(G')^{s,t}_{\alpha',\beta'}$ *for all* $(s,t) \in S$*.*

Proof (sketch). The result is obtained by considering the 36 couples of some $\mathtt{UD}(G)^{s,t}_{\alpha,\beta}$ and some $\mathtt{UD}(G')^{s',t'}_{\alpha',\beta'}$, each combination giving an element of $\mathtt{UD}(D)^{s'',t''}_{\alpha,\beta'}$ for some $(s'',t'') \in S$. □

Lemma 3. *For* $(D, \alpha, \beta) = \mathcal{P}((G, \alpha, \beta), (G', \alpha', \beta'))$*, the values of* $\#\texttt{UD}(D)^{s,t}_{\alpha,\beta}$ *for all* $(s,t) \in S$ *can be computed in time* $\mathcal{O}(n \log n \log\log n)$ *(with* n *the binary length of the values) from the values of* $\#\texttt{UD}(G)^{s,t}_{\alpha,\beta}$ *and* $\#\texttt{UD}(G')^{s,t}_{\alpha',\beta'}$ *for all* $(s,t) \in S$*.*

Proof (sketch). The proof is analogous to Lemma 2, except that some couples may create invalid labelings. □

Note that Remark 3 also applies to Lemmas 2 and 3. For the free composition the count is easier.

Lemma 4. *For* $(D, \alpha, \beta) = \mathcal{F}((G, \alpha, \beta), G')$*, we have* $\#\texttt{UD}(D)^{s,t}_{\alpha,\beta} = \#\texttt{UD}(G)^{s,t}_{\alpha,\beta} \#\texttt{UD}(G')$ *for all* $(s,t) \in S$*.*

Proof. The endpoints of the oss-graph (D, α, β) are the endpoints of the oss-graph (G, α, β), and it is not possible to create a forbidden cycle in the union of a valid labeling on G and a valid labeling of G', therefore the union is always a valid labeling of D, each one belonging to the part (s,t) of (D, α, β) corresponding to the part $(s,t) \in S$ of (G, α, β). □

6.2 Application to Oriented Series-Parallel Graphs

The series and parallel compositions of Definition 4 correspond exactly to the class of two-terminal series-parallel graphs from [25,29].

Definition 5. *A ss-graph* (G, α, β) *is* two-terminal series-parallel (a ttsp-graph) *if and only if one the following holds.*

- (G, α, β) *is a* base ss-graph *with two vertices* α, β *and one edge* $\{\alpha, \beta\}$*.*
- (G, α, β) *is obtained by a series or parallel composition*[1] *of two ttsp-graphs.*

In this case G *alone is called a* blind *ttsp-graph.*

Adding the free composition allows to go from two-terminal series-parallel graphs to (general) series-parallel graphs [11,29]. More precisely, it allows exactly to add tree structures to ttsp-graphs, as we argue now (ttsp-graphs do not contain arbitrary trees, its only acyclic graphs being simple paths; *e.g.* one cannot build a *claw* from Definition 5).

Definition 6. *A multigraph* G *is* series-parallel (sp-graph) *if and only if all its 2-connected components are blind ttsp-graphs. A digraph* G *such that* $\bar{G}$ *is an sp-graph, is called an* oriented sp-graph (osp-graph)*.*

The family of sp-graphs corresponds to the multigraphs obtained by series, parallel and free compositions from base ss-graphs.

Theorem 6. *G is an sp-graph if and only if (G, α, β) is obtained by series, parallel and free compositions from base ss-graphs, for some α, β.*

[1] With Definition 4 applied to (undirected) ss-graphs.

Proof (sketch). Free compositions allow to build all sp-graphs, because it offers the possibility to create the missing tree structures of ttsp-graphs: arbitrary 1-connected components linking 2-connected ttsp-graphs. Moreover free compositions do not go beyond sp-graphs, since the obtained multigraphs still have treewidth 2. □

Theorem 7. **#UD** *is solvable in time* $\mathcal{O}(n^2 \log^2 n \log\log n)$ *on osp-graphs (without promise).*

Proof (sketch). This is a direct consequence of Lemmas 2, 3 and 4, because all values are in $\mathcal{O}(2^n)$ (the number of $\{\oplus, \ominus\}$-labelings) hence on $\mathcal{O}(n)$ bits, the values of $\#\mathtt{UD}(G)^{s,t}_{\alpha,\beta}$ are trivial for oriented base ss-graphs, and we perform $\mathcal{O}(n \log n)$ compositions (to reach Formula 1). The absence of promise comes from a linear time recognition algorithm in [29] for ttsp-graphs, which also provides the decomposition structure. □

Again, Remark 3 applies to Theorem 7.

7 Conclusion

Our main result is the #P-completeness of **#UD**, *i.e.* of counting the number of non-equivalent block-sequential update schedules of a given AN f. We proved that this count can nevertheless be done in $\mathcal{O}(n^2 \log n \log\log n)$ time for oriented cacti, and in $\mathcal{O}(n^2 \log^2 n \log\log n)$ time for oriented series-parallel graphs. This last result has been obtained via a decomposition method providing a divide-and-conquer algorithm.

Remark that cliques or tournaments are intuitively difficult instances of **#UD**, because of the intertwined structure of potential forbidden cycles. It turns out that K_4 is the smallest clique that cannot be build with series, parallel and free decompositions, and that series-parallel graphs (Definition 6) correspond exactly to the family of K_4-minor-free graphs [11] (it is indeed closed by minor [27]). In further works we would like to extend this characterization and the decomposition method to (di)graphs with multiple endpoints.

The complexity analysis of the algorithms presented in Theorems 5 and 7 may be improved, and adapted to the parallel setting using the algorithms presented in [7,15]. One may also ask for which other classes of digraphs is $\#\mathtt{UD}(G)$ computable efficiently (in polynomial time)? Since we found such an algorithm for graphs of treewidth 2, could it be that the problem is fixed parameter tractable on bounded treewidth digraphs? Rephrased more directly, could a general tree decomposition (which, according to the proof of Theorem 6, is closely related to the series-parallel decomposition for treewidth 2) be exploited to compute the solution to **#UD**? Alternatively, what other types of decompositions one can consider in order to ease the computation of $\#\mathtt{UD}(G)$?

Finally, from the multiplication obtained for one-point join of two graphs (Lemma 4 on free composition), we may ask whether $\#\mathtt{UD}(G)$ is an evaluation of the Tutte polynomial? From its universality [8], it remains to know whether there is a deletion-contradiction reduction. However defining a Tutte polynomial for directed graphs is still an active area of research [6,10,24,30].

References

1. Aracena, J., Demongeot, J., Fanchon, É., Montalva, M.: On the number of different dynamics in Boolean networks with deterministic update schedules. Math. Biosci. **242**, 188–194 (2013)
2. Aracena, J., Demongeot, J., Fanchon, É., Montalva, M.: On the number of update digraphs and its relation with the feedback arc sets and tournaments. Discrete Appl. Math. **161**, 1345–1355 (2013)
3. Aracena, J., Fanchon, É., Montalva, M., Noual, M.: Combinatorics on update digraphs in Boolean networks. Discrete Appl. Math. **159**, 401–409 (2011)
4. Aracena, J., Goles, E., Moreira, A., Salinas, L.: On the robustness of update schedules in Boolean networks. Biosystems **97**, 1–8 (2009)
5. Aracena, J., Gómez, L., Salinas, L.: Limit cycles and update digraphs in Boolean networks. Discrete Appl. Math. **161**, 1–12 (2013)
6. Awan, J., Bernardi, O.: Tutte polynomials for directed graphs. J. Comb. Theory Ser. B **140**, 192–247 (2020)
7. Bodlaender, H.L., de Fluiter, B.: Parallel algorithms for series parallel graphs. In: Diaz, J., Serna, M. (eds.) ESA 1996. LNCS, vol. 1136, pp. 277–289. Springer, Heidelberg (1996)
8. Brylawski, T.H.: A decomposition for combinatorial geometries. Trans. Am. Math. Soc. **171**, 235–282 (1972)
9. Burkard, R.E., Krarup, J.: A linear algorithm for the pos/neg-weighted 1-median problem on a cactus. Computing **60**, 193–215 (1998)
10. Chan, S.H.: Abelian sandpile model and Biggs-Merino polynomial for directed graphs. J. Comb. Theory Ser. A **154**, 145–171 (2018)
11. Duffin, R.J.: Topology of series-parallel networks. J. Math. Anal. Appl. **10**, 303–318 (1965)
12. Elspas, B.: The theory of autonomous linear sequential networks. IRE Trans. Circuit Theory **6**, 45–60 (1959)
13. Fatès, N.: A guided tour of asynchronous cellular automata. J. Cell. Automata **9**, 387–416 (2014)
14. Huffman, D.A.: Canonical forms for information-lossless finite-state logical machines. IRE Trans. Inf. Theory **5**, 41–59 (1959)
15. Já Já, J.: An Introduction to Parallel Algorithms. Addison-Wesley, Boston (1992)
16. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. **22**, 437–467 (1969)
17. Kleene, S.C.: Representation of events in nerve nets and finite automata. Project RAND RM-704, US Air Force (1951)
18. Linial, N.: Hard enumeration problems in geometry and combinatorics. SIAM J. Algebraic Discrete Methods **7**, 331–335 (1986)
19. MacMahon, P.A.: The combinations of resistances. The Electrician **28**, 601–602 (1892)
20. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. J. Math. Biophys. **5**, 115–133 (1943)
21. Noual, M., Sené, S.: Synchronism versus asynchronism in monotonic Boolean automata networks. Nat. Comput. **17**, 393–402 (2017)
22. Perrot, K.: On the complexity of counting feedback arc sets. arXiv:1909.03339 (2019)
23. Perrot, K., Montalva-Medel, M., de Oliveira, P.P.B., Ruivo, E.L.P.: Maximum sensitivity to update schedule of elementary cellular automata over periodic configurations. Nat. Comput. **19**, 51–90 (2020)

24. Perrot, K., Pham, V.T.: Chip-firing game and partial Tutte polynomial for Eulerian digraphs. Electron. J. Comb. **23**, P1.57 (2016)
25. Riordan, J., Shannon, C.E.: The number of two-terminal series-parallel networks. J. Math. Phys. **21**, 83–93 (1942)
26. Robert, F.: Blocs-H-matrices et convergence des méthodes itératives classiques par blocs. Linear Algebra Appl. **2**, 223–265 (1969)
27. Robertson, N., Seymour, P.D.: Graph minors. XX. Wagner's conjecture. J. Comb.Theory Ser. B **92**, 325–357 (2004)
28. Thomas, R.: Boolean formalization of genetic control circuits. J. Theor. Biol. **42**, 563–585 (1973)
29. Valdes, J., Tarjan, R.E., Lawler, E.L.: The recognition of series parallel digraphs. In: Proceedings of STOC 1979, pp. 1–12 (1979)
30. Yow, K.S.: Tutte-Whitney polynomials for directed graphs and maps. Ph.D. thesis, Monash University (2019)