Advances in Applied Mathematics 64 (2015) 111–123

Contents lists available at ScienceDirect

Advances in Applied Mathematics

www.elsevier.com/locate/yaama

The complexity of the majority rule on planar graphs

Eric Goles [a,1], Pedro Montealegre [b,*,2]

[a] *Facultad de Ciencias y Tecnología, Universidad Adolfo Ibáñez, Santiago, Chile*
[b] *Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, FR-45067 Orléans, France*

ARTICLE INFO

Article history:
Received 15 November 2012
Accepted 19 November 2014
Available online 6 January 2015

MSC:
68Q17

Keywords:
Automata networks
Computational complexity
Majority
P-Completeness
NC
Planar graph

ABSTRACT

We study the complexity of the majority rule on planar automata networks. We reduce a special case of the Monotone Circuit Value Problem to the prediction problem of determining if a vertex of a planar graph will change its state when the network is updated with the majority rule.

1. Introduction

Let $G = (V, E)$ be a simple undirected finite graph, where V is the set of vertices and E the set of edges. An *Automata Network* is a triple $\mathcal{A} = (G, Q, (f_i : i \in V))$, where Q

* Corresponding author.
E-mail addresses: eric.chacc@uai.cl (E. Goles), pedro.montealegre@etu.univ-orleans.fr (P. Montealegre).

[1] This work was partially supported by FONDECYT 1140090 and Basal project PFB-03, Centro de Modelamiento Matemático, UMI 2807 CNRS, Universidad de Chile.
[2] This work was partially supported by Conicyt-Becas Chile-72130083.

http://dx.doi.org/10.1016/j.aam.2014.11.005
0196-8858/

is the finite set of states and $f_i : Q^{|V|} \to Q$ is the transition function associated to the vertex i. The set $Q^{|V|}$ is called the set of configurations, and automaton's global transition function $F : Q^{|V|} \to Q^{|V|}$, is constructed from the local functions $(G, Q, (f_i : i \in V))$ such that $(F(x))_i = f_i(x)$.

In the special case where $Q = \{0, 1\}$ we say that the state 1 means that the vertex is *active*, while state 0 represents *passive* vertices. Let $N(v)$ be the neighborhood of v, i.e. the set of vertices $\{u \mid uv \in E\}$, consider the following transition function:

$$f_i(x) = \begin{cases} 0 & \text{if } \sum_{j \in N(i)} x_j < \frac{|N(v)|}{2} \\ x_i & \text{if } \sum_{j \in N(i)} x_j = \frac{|N(v)|}{2} \\ 1 & \text{if } \sum_{j \in N(i)} x_j > \frac{|N(v)|}{2} \end{cases}$$

In other words, a vertex takes the state of the majority of its neighbors, and in case of a tie, keeps its state. The global transition function given by $(f_i : i \in V)$ is called the *the majority rule*.

In this paper we will study the computational complexity of predicting the state of a vertex in a given graph from an initial configuration. Let G be a graph, and x a configuration of G. A trajectory obtained from x is the set $\{x(t) : t \geq 0\}$ where $x(0) = x$, F is the majority rule, and $x(t+1) = F(x(t))$. We define **PER** as the decision problem which consists in predicting if a single vertex in a graph will change its state, in the trajectory given by some initial configuration. Formally:

PER. Let $G = (V, E)$ be an undirected graph, $x \in \{0, 1\}^{|V|}$ an initial configuration of G, $v \in V$ a vertex initially passive ($x_v = 0$), and F some global transition function. Determine if there exists $T > 0$ such that $X(T)_v = 1$.

The computational complexity of a decision problem can be defined as the amount of resources, like time or space, needed to predict it. In this case, we consider two fundamental classes: **P**, the class of problems solvable in polynomial time on a serial computer; and **NC**, the class of problems solvable in poly-logarithmic time with a polynomial amount of processors in a PRAM machine. It is easy to prove that $\mathbf{NC} \subset \mathbf{P}$. Informally **NC** is known as the class of problems which have a fast parallel algorithm [5]. It is a well known conjecture that $\mathbf{NC} \neq \mathbf{P}$, and if so, there exist "inherently sequential" problems, this is, that belong to **P** and do not belong to **NC**. The most likely to be inherently sequential are **P**-Complete problems, to which any other problem in **P** can be reduced by an **NC**-reduction or a logarithmic space reduction. If any of these problems have a fast parallel algorithm, then $\mathbf{P} = \mathbf{NC}$ [5,6].

One such problem is the *Circuit Value Problem* (**CVP**), which consists in predicting the truth value of an output of a Boolean circuit, given a truth value of its inputs. This problem is **P**-Complete since any deterministic Turing machine computation of length k can be converted into a Boolean circuit of depth k; thus polynomial time computations

are equivalent to polynomial size and depth circuits; a complete analysis of this reduction can be found in [5].

We define the *layer* of a gate v, denoted layer(v), to be zero for input gates and for the other gates, is the length of the longest path from an input to v. A circuit is *synchronous* if all inputs to a gate v come from vertices at layer $[\text{layer}(v) - 1]$. The **CVP** remains **P**-Complete when the circuit is restricted to be synchronous, monotone (that is, with AND and OR gates but without negation) and all vertices have in degree (fan in) and out degree (fan out) exactly two, with the obvious exceptions of the input with in degree zero, and the outputs with out degree zero [5]. We call **S2MCVP** this restriction of the **CVP**.

A different case is when we restrict the circuit to be planar, because *Planar monotone circuit value problem* or **PMCVP**, the restriction of **CVP** to a planar and monotone circuit, is in **NC** [2].

C. Moore in [6] studied the complexity of the majority rule in the d-dimensional lattice, with $d \geq 3$. He proved that in that kind of graphs **PER** is **P**-Complete, but leaves as an open question what happens in the 2-dimensional lattice.

In a previous work [3] we studied the problem **PER** with the bootstrap percolation rule, this is, passive vertices become active if the majority of its neighbors are active, but once a vertex is active, it never changes again. We found that the complexity of this problem depends on the characteristics of the graph, moreover, the problem is **P**-Complete in the general case, but in **NC** if we restrict the vertices to have degree less or equals than 4. In that paper we left as an open question what happens in the case when the graph is restricted to a planar one.

We approach this open problem proving that, for the majority rule and the family of planar graphs, **PER** is **P**-Complete. Then the main theorem of this article is:

Theorem 1. *Over the family of planar graphs, the problem* **PER** *associated to the majority rule is* **P***-Complete.*

In the following sections we will give a proof of this theorem, which follows from three lemmas. First, in Section 2, we will show the membership in **P** for **PER** with the majority rule. Then, we will give a proof of the **P**-Hardness of **PER** with the majority rule. Finally, we will show a **P**-Hardness proof for the planar case. At the end we give some conclusions.

2. The majority rule is in P

To prove the membership in **P** of the majority rule, we notice that from any configuration x of G, the next state of any vertex can be calculated in $\mathcal{O}(n)$ time, and then the whole next configuration $F(x)$ can be calculated in at most $\mathcal{O}(n^2)$ time. Let $\{x(t) : t \geq 0\}$ a trajectory obtained from a configuration x. We say that the dynamic of

x enters into a cycle if $\exists t_1, t_2 \geq 0$, $t_1 < t_2$ such that $x(t_1) = x(t_2)$, and the length of the cycle is $t_2 - t_1$.

To decide **PER**, we should prove that for any configuration the dynamic of x enters into a cycle of polynomial length in at most a polynomial number of steps.

Lemma 2. *For the majority rule,* **PER** *is in* **P**.

Proof. We will use the results obtained in [1,4]. Let $\eta : \mathbb{R} \to \{0,1\}$ the threshold function

$$\eta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

we can characterize the majority rule using this function:

$$x_v(t+1) = \begin{cases} \eta\left(\sum_{u \in N(v)} x_u(t) + x_v(t) - \frac{|N(v)|+1}{2}\right) & \text{if } |N(v)| \text{ is even} \\ \eta\left(\sum_{u \in N(v)} x_u(t) - \frac{|N(v)|}{2}\right) & \text{if } |N(v)| \text{ is odd} \end{cases}$$

If $\bar{\eta} : \mathbb{R}^{|V|} \to \{0,1\}^{|V|}$ is the multidimensional threshold function: $\bar{\eta}((x_i)_{i \in V}) = (\eta(x_i))_{i \in V}$, then $x(t+1) = \bar{\eta}(Ax(t) - b)$ where $A = (a_{uv})$ is a square matrix of order $|V|$ and b is a vector of size $|V|$ with

$$a_{uv} = \begin{cases} 1 & \text{if } uv \in E \\ 1 & \text{if } i = j \wedge |N(v)| \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$

and

$$b_v = \begin{cases} \frac{|N(v)|+1}{2} & \text{if } |N(i)| \text{ is even} \\ \frac{|N(v)|}{2} & \text{if } |N(i)| \text{ is odd} \end{cases}$$

Using this characterization, let $\{x(t) : t \geq 0\}$ be a trajectory of the majority rule with initial condition $x(0)$, and define the following functional for $t \geq 1$:

$$\begin{aligned} E\big(x(t)\big) = \sum_{v \in V} \left(2b_v - \sum_{u \in V} a_{vu}\right) \big(2x_v(t) + 2x_v(t-1) - 2\big) \\ - \big(2x_v(t) - 1\big) \sum_{u \in V} a_{vu} \big(2x_u(t-1) - 1\big) \end{aligned}$$

We have that this functional satisfies

$$\begin{aligned} \Delta_t E &= E\big(x(t)\big) - E\big(x(t-1)\big) \\ &= -4 \sum_{v \in V} \big(x_v(t) - x_v(t-2)\big) \left(\sum_{u \in V} a_{vu} x_u(t-1) - b_v \right) \end{aligned}$$

Notice that if $(x_v(t) - x_v(t-2)) > 0$ iff $x_v(t) = 1$ and then $a_{vu}x_u(t-1) > b_v$. It follows that $\Delta_t E \leq 0$, and then E is a strictly decreasing function in t. Hence, if $\{x(t) : t \in [t_1, t_2]\}$ is a cycle, then $E(x(t))$ is necessarily constant for any $t \in [t_1, t_2]$. Actually, since for any $x \in \{0,1\}^{|V|}$ and $v \in V$ we have $\sum_{u \in V} a_{vu}x_u \neq b_v$, then the cycles are only fixed points and/or cycles of length two, i.e. $t_2 = t_1$ or $t_2 = t_1 + 2$.

Then we have that for any initial configuration, the dynamics necessarily enters in a cycle of length at most 2. We must prove now that the entrance occurs in a step that is polynomial on the size of the graph. Let again $\{x(t) : t \geq 0\}$ be a trajectory of some rule with initial condition $x(0)$. We can define its transient length [1,4] by:

$$\tau(x) = \max\{t \geq 0 : x(t) \text{ enters a cycle for the first time}\}$$

and the transient length for the Automata Network is the greatest of these values:

$$\tau(A, b) = \max\{\tau(x(0)) : x(0) \in \{0,1\}^{|V|}\}.$$

Also by a result in [1,4] we have that $\tau(A, b) \leq |V|^3$ for the majority rule. This tells us that in a number of steps that is polynomial in the size of the input, the trajectory enters into a cycle of length 2. Since any iteration can be simulated in polynomial time, **PER** is in P. □

3. The majority rule is P-Hard

We will give now a proof of **P**-Hardness of the majority rule, reducing it to a restricted case of **CVP**. Clearly proving the **P**-Completeness in the family of planar graphs is stronger than proving it over any graph. Moreover, the **P**-Hardness of **PER** with the majority rule follows from the result of C. Moore in [6] who proved that for the majority rule **PER** is **P**-Complete in the d-dimensional lattice, for d grater than 3. We will provide our own proof of the **P**-Hardness, since it will help us to explain better the planar case.

Lemma 3. *For the majority rule* **PER** *is* **P***-Hard.*

Proof. To prove that for the majority rule **PER** is **P**-Hard, we will reduce **S2MCVP** to it. Since **S2MCVP** is **P**-Complete, if the reduction uses only a logarithmic space, then **PER** would be **P**-Complete.

To reduce **S2MCVP** to **PER**, we will build, given a monotone circuit, a graph that simulates its gates, where active vertices represent truth, and passive vertices represent false. Since the circuit is monotone, we only have to simulate the AND and OR gates.

The AND gate (Fig. 1 (a)) is simulated by an initially passive middle vertex with degree 3, two of them will be the inputs and the other the output. By the majority rule, this vertex will become active only if two more neighbors become active. In a similar way, the OR gate (Fig. 1 (b)) has an initially passive middle vertex, with degree 5 and

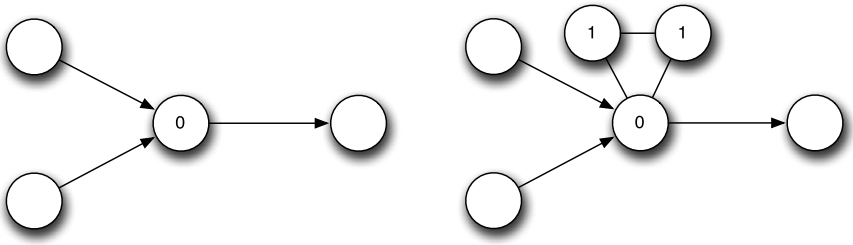

Fig. 1. a) The AND gate. b) The OR gate.

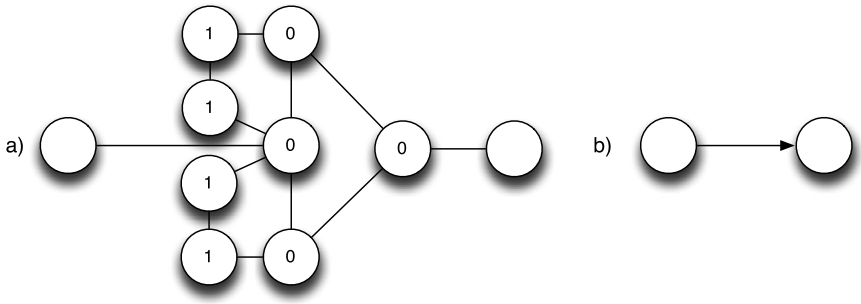

Fig. 2. a) The diode. b) Symbol of a diode.

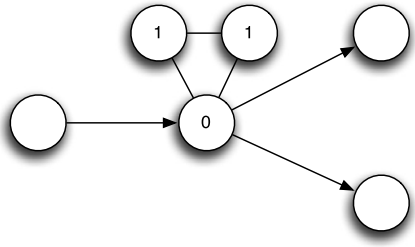

Fig. 3. OR gate 'backwards' to multiply the information of an output.

two neighbors initially active, then, this vertex will become active if any passive neighbor becomes active. Connections between active vertices are used to keep them active.

In order to avoid problems with the flow of information, Fig. 2 (a) shows the construction of a 'diode' which only allows the flow of information in 'one way': If the left vertex is active, then the right vertex will become active. But if the right vertex is active, the left vertex remains in its state. We simplify the drawing with an arrow Fig. 2 (b). In both AND and OR gates (Fig. 1), the diode is included.

Remember that in our restricted version of the circuit value problem **S2MCVP**, every gate of the original circuit must have fan-out exactly 2, so we use OR gates 'backwards' in order to multiply the information of the output, as in Fig. 3. Finally, we connect the gates with 'wires' (Fig. 4) that corresponds to initially passive vertices with degree 3 and one neighbor initially active, so they stay passive unless another neighbor becomes

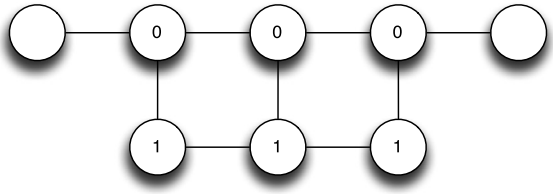

Fig. 4. A wire of length 3.

active. Then, with these constructions, given ϕ a monotone Boolean circuit and I an input, we can build a graph G that simulates ϕ in I. Letting active the input nodes that are true, passive the rest, and considering v the vertex that simulates the state of the output o of the circuit, (G, x, v) belongs to **PER** if and only if (ϕ, I, o) belongs to **S2MCVP**.

This reduction can be done by a Turing machine in logarithmic space: reading the input (a monotone circuit) of size n, the machine only has to determine which construction corresponds to each gate ($\mathcal{O}(1)$ space), and then determine the connectivity of every gate ($\mathcal{O}(\log n)$ space). Then, the whole construction requires $\mathcal{O}(\log n)$ space.

Then for the majority rule **PER** is **P**-Hard. □

4. P-Hardness of the majority rule for planar graphs

Despite of the constructions of Lemma 3 that are all planar, the input graph may be not be a planar one. Moreover, as we have said in the introduction, the monotone circuit value problem restricted to the family of planar graphs **PMCVP** is not likely to be a **P**-Complete problem, since there is a NC-Algorithm that solves it [2]. Then we should build a planar graph that simulates a not necessarily planar circuit.

Let (ϕ, x) an instance of our special case of **S2MCVP**, this is, ϕ is a monotone, in degree two, out degree two, layered synchronous circuit, where x is an input of ϕ. We will need that our circuit is lexicographically ordered, which means that the gates of the circuit are numbered from 1 to n, where n is the number of gates, and the numbering respects the order of the layouts, i.e., if n_1, n_2 are the numberings of the gates in layers l_1 and l_2 respectively, then $l_1 < l_2 \Rightarrow n_1 < n_2$. **CVP** with this restrictions still is **P**-Complete [5].

We have that the gates in the input layer (layer 0) are numbered from 1 to n_0, the gates in the layer 1 are numbered from $n_0 + 1$ to n_1 and so on. We define $N_l = n_l - n_{l-1}$, the number of gates in the layer l.

Lemma 4. *Over the family of planar graphs, the problem* **PER** *associated to the majority rule is* **P***-Hard.*

Proof. In this case the **P**-Hardness also is obtained reducing a circuit, but this time we must guarantee that the resulting graph is planar. To ensure this, in our reduction

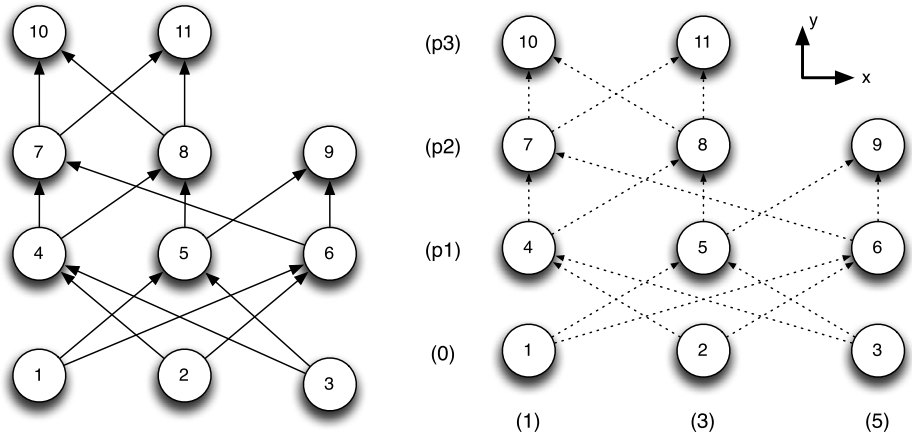

Fig. 5. An example of a circuit and the position of the gates in the planar embedding. Notice that the y coordinate is unknown yet for the vertices that do not represent gates in the input layer, but is the same for every vertex which represent gates in the same layer. If we just draw the edges we clearly won't obtain a planar embedding.

algorithm we also give the positions of every vertex in the XY plane in order to obtain a planar embedding.

We will first assign positions in the XY plane to the vertices that will simulate the gates of the circuit, which we know, by the proof of Lemma 3, that are: a passive vertex joined with two initially active and stable vertices if the gate is of type OR, and just a passive vertex if the gate is of type AND (Fig. 1). In both cases the initially passive vertices are assigned to coordinates $(2u-1, p_l)$, where u is the number of the simulated gate, relatively to its layer l (then $u \in \{1, \ldots, N_l\}$) and p_l is some number bounded by $\mathcal{O}(N_l^2)$ which is the same for every gate in layer l (Fig. 5). In the case that the gate is of type OR, we give to the initially active vertices positions very close to $(2u-1, p_l)$ in the same layout of Fig. 1, for example $(2u-\epsilon, p_l-\epsilon)$, $(2u-\epsilon, p_l+\epsilon)$, for some small ϵ.

This can be done in $\mathcal{O}(\log(n))$ space, since we just need to keep the numbering of the gate that we are simulating ($\mathcal{O}(\log(n))$ space) and its type ($\mathcal{O}(1)$ space). To obtain the values of p_l, we will execute the following induction: We define $p_0 = 0$, and then that we have defined p_l we obtain p_{l+1} by the following steps.

Step 1. First we use the gadget for multiplying the information (Fig. 3), in order to be able to "separate" the two outputs of every gate in the layer l. Remember that our case of CVP ensures us that every gate has exactly two inputs and two outputs. For a vertex that simulates a gate of the layer l, say numbered u, we draw the gadget of Fig. 3 in order that the positions of the outputs are in position $(2u-1, p_l+1)$ and $(2u, p_l+1)$ (Fig. 6). We will call these new vertices u_a and u_b.

Step 2. Let g_1 the first gate in the layer $l+1$, and let i and j, its inputs, with $i < j$. Then, g_1 will have as inputs i_a and j_a in the simulated circuit. In general, if i is an input of a gate g, in our simulated circuit the vertex that simulates g will have as input i_a if g is the first gate in the layer $l+1$ which has i as input, and i_b otherwise.

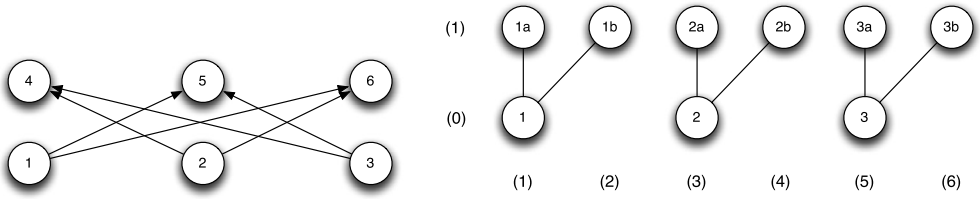

Fig. 6. Left: The two first layers of the circuit of the example in Fig. 5. Right: Scheme of the positions of the vertices that simulates the input layer and the positions of their two outputs.

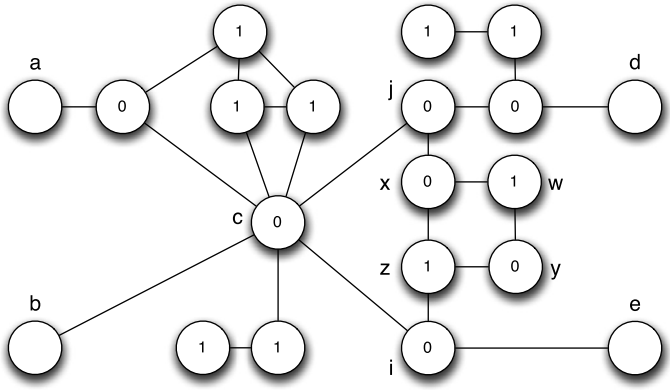

Fig. 7. Gadget for 'Crossing cables' using 'Traffic Lights'.

Too keep planarity, we would like that the inputs of the gate k-th gate of the layer $l+1$ have positions $(2k-1, p_{l+1}-1)$ and $(2k, p_{l+1}-1)$. Since normally this is not the case, we will introduce a new gadget that will help us to shift the positions.

We are going to change the places where there are 'crossing cables', building a gadget that uses 'traffic lights', letting the information pass in one way or another depending in the parity of the step. Fig. 7 shows this gadget. Vertices **a** and **b** are the 'inputs', **d** and **e** are the 'outputs' of the gadget. Before **a** and **b** and after **d** and **e** there are diodes, that are omitted for simplicity.

The key here is that the initially active vertices that simulate inputs will blink between active and passive states. Since they are connected only to the first vertex of a diode of the gadget for multiplying info (Fig. 3), in the first step all become passive, in the second the initially active become active again, and so on. Moreover, with exceptions of the initially active nodes that are used to build the gadgets, every vertex that simulates a gate and becomes active at some step, begins to blink between active and passive states.

Suppose then that at least one of the inputs of the crossing cable gadget become active in an even step (and then is passive in odd steps, we are supposing that the inputs of the gadgets are synchronized). If they become active in an even step, we change $x_x(0) = x_y(0) = 0$ and $x_z(0) = x_w(0) = 1$ and the rest is analogous.

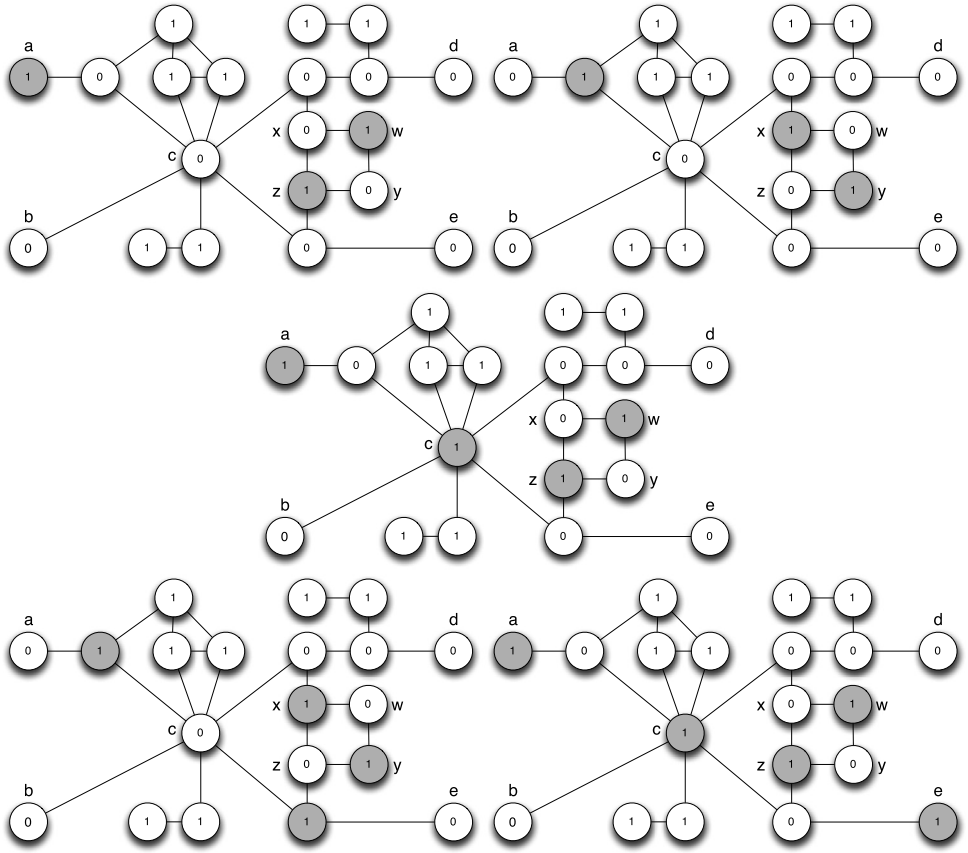

Fig. 8. Steps for the crossing information gadget from **a** to **e**. Notice that before **a** and **b** and after **d** and **e** there are diodes, which can be crossed in 4 steps. Then the whole gadget is crossed in 13 steps.

We have that $x_x(0) = x_y(0) = 1$ and $x_z(0) = x_w(0) = 0$ (Fig. 7), then for every $k \in \mathbb{N}$

$$x_x(2k) = x_y(2k) = 1, \qquad x_z(2k) = x_w(2k) = 0,$$
$$x_x(2k-1) = x_y(2k-1) = 0 \quad \text{and} \quad x_z(2k-1) = x_w(2k-1) = 1.$$

Let **i** the vertex connected to **c**, **z** and **e**, in Fig. 7. Notice that it will become active only if both are active. Since **z** is active only in odd steps, the only way that **i** becomes active is that **a** is active at some even step. But in even steps, the vertex **x** will be passive, and then **j** will remain passive. Then, we can take the information from **a** to **e** without any "contamination" to **d** (Fig. 8). We can cross information from **b** to **d** in a similar fashion.

Back in Step 2, we are going to use our gadget to change positions of the vertices that carry the information from layer l to layer $l+1$. We start with the inputs of the first gate of the layer $l+1$, which we called g_1. Let i_a the smallest input of g_1, if the position of i_a is not $(1, p_l + 1)$, we use our crossing cables gadget (Fig. 7) with $\mathbf{a} = (i-1)_b$, $\mathbf{b} = i_a$,

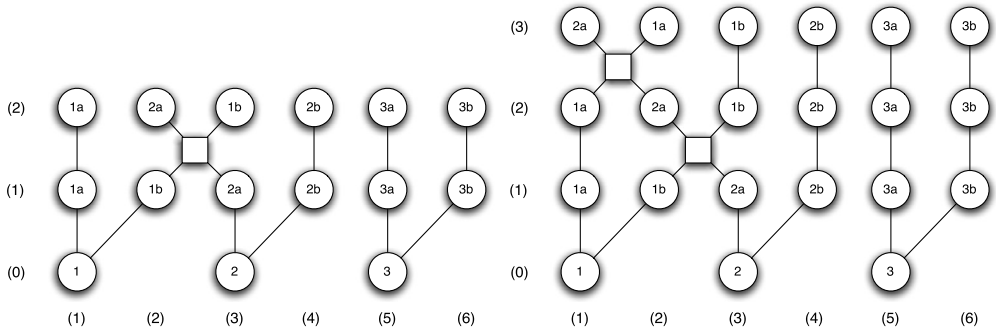

Fig. 9. Scheme of Step 2: Squared node represents the crossing cables gadget. Edges between vertices of y-coordinate (1) and (2), and y coordinate (2) and (3) represent wires of length 13.

$\mathbf{d} = (2(i-1), p_l + 2)$ and $\mathbf{e} = (2i-1, p_l + 2)$ and the rest of the vertices of the gadget very close to $(2i - 3/2, p_l + 3/2)$. We connect with a wire of length 13 (Fig. 4) between every vertex with position $(x, p_l + 1)$ to $(x, p_l + 2)$, where $x \in \{1, \ldots, N_l\} \backslash \{i_a, (i-1)_b\}$ (Fig. 9).

Summing-up, we just add a new row of vertices such that i_a changes x coordinate with $(i-1)_b$ using the crossing cable gadget, and everyone else keeps their position (Fig. 9). We repeat this until i_a reaches position 1, and then repeat the same for the other input j until it reaches position 2. Once we have both inputs of the first layer in their correct positions, apply the same to the second vertex in the layer $l+1$, and so on (Fig. 9).

This can be done in $\mathcal{O}(\log(n))$ space. For a vertex v in the layer $l+1$ we need to:

- Identify its inputs (if they are i_a or i_b), which requires $\mathcal{O}(\log(n))$ space, since we only need to save its inputs i, j ($\mathcal{O}(\log(n))$) and a flag $\mathcal{O}(1)$ to determine if v is the first vertex in the layer which has this inputs.
- Calculate the initial position of an input i_x of v, which is $I[i_x] = 2i + Q[x] + P[i]$, where $Q[x] = -1$ if $x = a$ and 0 if $x = b$, and $P[i] = N_a[i] + N_b[i]$, with

$$N_a[i] = \left|\{j \in \text{layer } l \mid j_a \text{ is input of } u < v, \text{ and } j > i\}\right|$$
$$N_b[i] = \left|\{j \in \text{layer } l \mid j_b \text{ is input of } u < v, \text{ and } j > i\}\right|$$

  $P[i]$ can be calculated in $\mathcal{O}(\log(n))$ since we just need to check from 1 to v if any gate has none, one or both inputs smaller than i, and $Q[x]$ can be calculated with the identification done before.
- Build the gadgets from the initial position into the final one, which is $2v - 1$ or $2v$ depending if we are moving the first or the second input. This requires $\mathcal{O}(\log(1))$ since we just need to keep a counter for the number of times that we must build our gadgets.

Then Step 2 requires only $\mathcal{O}(\log(n))$ space.

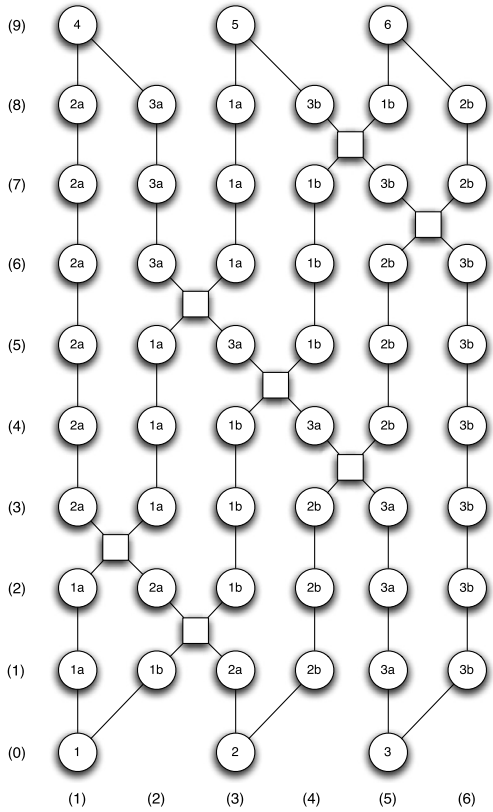

Fig. 10. Scheme of the planar embedding that simulates the two first layers of a circuit in Fig. 6. Squared nodes represent the crossing cables gadget. Edges between vertices with different y-coordinate grater that 1 represent wires of length 13.

Step 3. Finally, we have that for every vertex v that simulates a gate in layer $l+1$ has its inputs in positions $2v-1$ and $2v$, we just draw the corresponding diodes (Fig. 10). This requires $\mathcal{O}(1)$ space since we just need to take a counter for parity.

We repeat this for every layer of the circuit. Then, with these constructions and gadgets, given ϕ a monotone Boolean circuit and I an input, we can build a planar graph G that simulates ϕ in I. Letting active the input nodes that are true, passive the rest, and considering v the vertex that simulates the state of the output o of the circuit, (G, x, v) belongs to **PER** if and only if (ϕ, I, o) belongs to **S2MCVP**.

Since this reduction can be done in $\mathcal{O}(\log n)$ space, for the majority rule **PER** is **P**-Hard on the family of planar graphs. □

5. Conclusions

We have discussed the majority rule in planar graphs, proving that the problem is **P**-Complete. The membership in **P** was proven defining a decreasing energy function, which ensure us that the dynamics converge into cycles of length at most two in a

polynomial number of steps. Then, using some gadgets, where the most important one is the crossing cables gadget with *traffic lights*, we were able to simulate a monotone circuit with the majority rule in a planar graph.

On the other hand, C. Moore studied this rule in the d-dimensional lattice [6]. He proved that for $d \geq 3$ the problem is **P**-Complete, but leaves open the case of $d = 2$, conjecturing membership in **NC**. Clearly, two dimensional lattices are planar graphs, and in this context our result shows that the planarity alone is not enough to classify the problem into **NC**. If the conjecture is true we may consider more specific cases, like restricting the degree of the graph.

In a previous work [3] we have studied the complexity of bootstrap percolation, which is the majority rule but when *active* vertices remains always *active*. In that case we proved that for the family of graphs with degree strictly lower than 5 the rule is in **NC**, so for the two dimensional lattice the particular case majority (Bootstrap Percolation) is in **NC**.

These approaches enclose the conjecture of C. Moore, showing that both the topological properties of the lattice (for example the restrictions on degree and regularity) and the symmetry of the rule for both states (zeros and ones may change) are relevant to study this problem.

References

[1] E. Chacc, Comportement oscillatoire d'une famille d'automates cellulaires non uniformes, Thèse IMAG, Grenoble, France, http://books.google.fr/books?id=V5lstwAACAAJ, 1980.
[2] A.L. Delcher, S.R. Kosaraju, An $\mathcal{NC}$ algorithm for evaluating monotone planar circuits, SIAM J. Comput. 24 (1995) 369–375, http://dx.doi.org/10.1137/S0097539792226278.
[3] E. Goles, P. Montealegre-Barba, I. Todinca, The complexity of the bootstraping percolation and other problems, Theoret. Comput. Sci. 504 (2013) 73–82, http://dx.doi.org/10.1016/j.tcs.2012.08.001.
[4] E. Goles-Chacc, F. Fogelman-Soulie, D. Pellegrin, Decreasing energy functions as a tool for studying threshold networks, Discrete Appl. Math. 12 (1985) 261–277.
[5] R. Greenlaw, H.J. Hoover, W.L. Ruzzo, Limits to Parallel Computation: P-Completeness Theory, Oxford University Press, Inc., New York, NY, USA, 1995.
[6] C. Moore, Majority-vote cellular automata, Ising dynamics, and P-completeness, J. Stat. Phys. 88 (1997) 795–805.