On the Computational Complexity of the Freezing Non-strict Majority Automata

Eric Goles, Diego Maldonado, Pedro Montealegre, Nicolas Ollinger

On the computational complexity of the freezing non-strict majority automata

Eric Goles[1,2,3], Diego Maldonado[2], Pedro Montealegre[2], and Nicolas Ollinger[2]

[1] Facultad de Ciencias y Tecnologia, Universidad Adolfo Ibañez, Santiago, Chile
[2] Univ. Orléans, LIFO EA 4022, FR-45067 Orléans, France
[3] LE STUDIUM, Loire Valley Institute for Advanced Studies, Orléans, France

Abstract. Consider a two dimensional lattice with the von Neumann neighborhood such that each site has a value belonging to $\{0, 1\}$ which changes state following a freezing non-strict majority rule, i.e., sites at state 1 remain unchanged and those at 0 change iff two or more of it neighbors are at state 1. We study the complexity of the decision problem consisting in to decide whether an arbitrary site initially in state 0 will change to state 1. We show that the problem in the class **NC** proving a characterization of the maximal sets of stable sites as the tri-connected components.

1 Introduction

Majority automata can be defined as the two-state cellular automata, where in each synchronous step each site takes most represented state in its neighborhood. This kind of automata models many kinds of physic and social phenomena [1,2,3,4,5]. Let us denote by 1 and 0 the two states of the majority automata, which may represent respectively states *active* or *inactive*, *alive* or *dead*, etc. This paper is about the problem of predicting, given an initial configuration of a states, if a given site will change its state.

The computational complexity of a prediction 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 in a PRAM machine, with a polynomial number of processors [6]. We say that **NC** is the class of problems which have a *fast parallel algorithm*. It is a well-known conjecture that $\mathbf{NC} \neq \mathbf{P}$, and so, if there exists "inherently sequential" problems, this is, problems 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 has a fast parallel algorithm, then **P**=**NC** [6,7].

In [7] Moore studied the computational complexity of the majority automata in a d-dimensional lattice. He showed that in three or more dimensions the prediction problem is as hard as evaluating monotone circuits, which implies that

the prediction problem is **P**-Complete. Roughly speaking, this means that in order to compute the state in a given site, the only option (unless $\mathbf{P} = \mathbf{NC}$) is to simulate the dynamics of the automaton until it reaches an attractor. However, Moore suggested that in two dimensions with von Neumann neighborhood it would be possible to predict exponentially faster, i.e., the problem is not **P**-Complete. In the same article, Moore also studied the *non-strict majority automata* (called also Half-or-More automata), which corresponds to the majority automata where the sites privilege state 1 over 0 in tie cases, not considering its own state in the neighborhood. Moore stated that the prediction problem for non-strict majority automaton is **P**-Complete in three or more dimensions, and also conjectured that in two dimensions the problem would not **P**-Complete.

In this article we study the prediction problem on the *freezing non-strict majority automata*. The freezing property means that a site in state 1 remains in that state in every future time step. Freezing automata model forest fires [8], infection spreading [9], bootstrap percolation [10] and voting systems [7]. Theoretical facts about those automata can be seen in [11].

The prediction problem for the *freezing majority automata* was studied by Goles et al. in [12], where the authors show that the prediction problem for the freezing majority automata is in **NC**, restricted to a two dimensional lattice with von Neumann neighborhood. This result is based on a characterization of *s*table sets of sites. A set of sites is called stable if a site in 0 remain in state 0 on any future time-step. The authors showed that for the freezing majority automata the stable sets can be characterized in terms of connected and biconnected components of the sets of sites initially in state 0. The prediction algorithm uses fast parallel algorithms computing connected and biconnected components due to Jájá [13].

In this paper, we show that the prediction problem for the non-strict majority automata is in **NC**. Our algorithm is based in a characterization of the stable sets for this rule. Unfortunately, the characterizations of stable sets for the freezing majority automaton is not valid for the non-strict one. In its place, we show that the stable sets in this case are roughly a set of sites initially in state 0 which form a tri-connected component, i.e., there are three disjoint paths between every pair of sites in the set. Then, we use a fast-parallel algorithm due to Jájá [14] to compute the tri-connected components of the sites initially in 0.

The article is organized as follows. In next section, we begin with the main formal definitions. In Section 3 we present a characterization of the stable sets for the freezing non-strict majority automata. In Section 4 we use the characterization of the previous section to obtain the main result. Finally, in Section 5 we conclude this article with a discussion and some open questions.

2 Preliminaries

Let us consider the freezing non-strict majority cellular automata (FNSMCA) as the cellular automata defined by the tuple $(\mathbb{Z}^2, \{0,1\}, N, f)$, where $N =$

$\{(0,0),(1,0),(-1,0),(0,1),(0,-1)\}$ is the von Neumann neighborhood and $f : \{0,1\}^5 \to \{0,1\}$ the local freezing non-strict majority function:

$$f(x) = \begin{cases} 1 \text{ if } x_1 = 1 \\ 1 \text{ if } (x_1 = 0) \wedge \left(\sum_{i \in [5]} x_i \geq 2 \right) \\ 0 \text{ otherwise} \end{cases}$$

For $c \in \{0,1\}^{\mathbb{Z}^2}$ and $i, j \in \mathbb{Z}$, call $c_{(i,j)+N}$ the vector $((i,j),(i+1,j),(i-1,j),(i,j+1),(i,j-1))$. The new state of each site of the lattice is computed synchronously, i.e., every site is updated at the same time, which is equivalent to the application of the global function $F : \{0,1\}^{\mathbb{Z}^2} \to \{0,1\}^{\mathbb{Z}^2}$ with $F(c)_v = f(c_{v+N})$.

The following definition characterizes the sites that are initially in 0 and never change, we call such sites *stable*.

Definition 1. *Given a configuration $c \in \{0,1\}^{\mathbb{Z}^2}$, we say that a site v is* stable *if and only if $c_v = 0$ and it remains at state 0 after any iterated application of the global rule, i.e., $F^t(c)_v = 0$ for all $t \geq 0$.*

One property of the strict majority automata that will be useful in our proofs is the *monotonicity*. For two configurations c and c', denote by $\leq$ the partial order relation over configurations, where $c \leq c'$ if and only if $c_u \leq c'_u$ for every $u \in \mathbb{Z}^2$. A function $G : \{0,1\}^{\mathbb{Z}^2} \to \{0,1\}^{\mathbb{Z}^2}$ is called *monotone* if $c \leq c'$ implies that $G(c) \leq G(c')$. Clearly, the strict majority automata (and its freezing version) is monotone.

A configuration $c \in \{0,1\}^{\mathbb{Z}^2}$ is called a *periodic configuration* if c consists in the periodic repetition of a finite configuration $x \in \{0,1\}^{[n]\times[m]}$, for $n, m > 0$. In such a case, we also call $c = c(x)$. Note that a FNSMCA in a periodic configuration $c(x)$, with $x \in \{0,1\}^{[n]\times[n]}$, reaches a fixed point in at most n^2 steps. Indeed, at each step before the dynamic reaches the fixed point, at least one site change from state 0 to state 1.

A natural problem consists in computing, given a periodic configuration c, the configuration obtained by the automata when it reaches the fixed point corresponding to c. We define the decision version of this problem, called (PREDICTION), as the problem consisting in decide, given a configuration, if a specific site initially in state 0 is not stable.

Prediction (PREDICTION)
Input: A finite configuration x of dimensions $n \times n$ and a site $u \in [n] \times [n]$ such that $x_u = 0$.
Question: Does there exists $T > 0$ such that $F^T(c(x))_u = 1$?

Clearly, if this problem is in **NC**, then we can compute the fixed point of the automaton running $\mathcal{O}(n^2)$ copies of the algorithm in each site initially in state

0. Indeed, the fixed point obtained from c can be computed choosing state 1 on all sites that are not stable for configuration c.

For a finite set of sites $S \subseteq \mathbb{Z}^2$, we call $G[S] = (S, E)$ the graph defined with vertex set S, where two vertices are adjacent if the corresponding sites are neighbors for the von Neumann neighborhood. For a graph $G = (V, E)$, a sequence of vertices $P = v_1, \ldots, v_k$ is called a v_1, v_k- *path* if $\{v_i, v_{i+1}\}$ is an edge of G, for each $i \in [k]$. Two u, v-paths P_1, P_2 are called *disjoint* if $P_1 \cap P_2 = \{u, v\}$.

Definition 2. *A graph G is called* tri-connected *if for every pair of vertices $u, v \in V(G)$, G contains three disjoint u, v-paths.*

A maximal set of vertices of a graph G that induces a tri-connected subgraph is called a *tri-connected component* of G. The following proposition states that is decidable in **NC** if a graph is tri-connected, moreover, gives a fast-parallel algorithm computing the tri-connected components of an input graph.

Proposition 1 ([14]). *There is an algorithm that computes the tri-connected components of a graph in time $\mathcal{O}(\log^2 n)$ with $\mathcal{O}(n^4)$ processors.*

Consider the relation R over vertices of a graph, which states that two vertices u and v are related by R if there exist three disjoint paths connecting u and v. For a pair of vertices u and v of a graph G, call $G_{u,v}$ the transitive closure of graph $G - \{u, v\}$. The transitive closure of a graph G is the graph in the same vertex set, where each connected component is a clique. In [14] it is shown that two vertices s, t are related by R if they are connected in $G_{u,v}$ for every pair $u, v \in V(G) \setminus \{s, t\}$. Moreover, the transitive closure can be computed in time $\mathcal{O}(\log^2 n)$ with $\mathcal{O}(n^2)$ processors [14]. The tri-connected components are then constructed roughly as follows: for every triple u, v, w of vertices of G that are mutually tri-connected (i.e. related by R), the set $\{l \in V : lRu \wedge lRv \wedge lRw\}$ is the tri-connected component that contains vertices u, v and w.

3 A characterization of stable sets

In this section, we characterize *stable sets* of a configuration c, i.e. sets of sites that are stable for c.

Lemma 1. *Let $x \in \{0,1\}^{n^2}$ be a finite configuration and $u \in [n] \times [n]$ a site. Then, u is stable for $c = c(x)$ if and only if there exist a set $S \subseteq [n] \times [n]$ such that:*

- *$u \in S$,*
- *$c_u = 0$ for every $u \in S$, and*
- *$G[S]$ is a graph of minimum degree 3.*

Proof. Suppose that u is stable and let S be the subset of $[n] \times [n]$ containing all the sites that are stable for c. We claim that S satisfy the desired properties. Indeed, since S contains all the sites stable for c, then u is contained in S. On

the other hand, since the automata is freezing, all the sites in S must be in state 0 on the configuration c. Finally, if $G[S]$ contains a vertex v of degree less than 3, it means that necessarily the corresponding site v has two non-stable neighbors that become 1 in the fixed point reached from c, contradicting the fact that v is stable.

On the other direction suppose that S contains a site that is not stable and let $t > 0$ be the minimum step such that a site v in S changes to state 1, i.e., $v \in S$ and t are such $F^{t-1}(c)_w = 0$ for every $w \in S$, and $F^t(c)_v = 1$. This implies that v has at least two neighbors in state 1 in the configuration $F^{t-1}(c)$. This contradicts the fact that v has three neighbors in S. We conclude that all the sites contained in S are stable, in particular u. □

For a finite configuration $x \in \{0,1\}^{[n]^2}$, let $D(x) \in \{0,1\}^{\{-n^2-n,\ldots,n^2+2n\}^2}$ be the finite configuration of dimensions $m \times m$, where $m = 2n^2 + 3n$, constructed with repetitions of configuration x in a rectangular shape, as is depicted in Figure 1, and sites in state 0 elsewhere. We also call $D(c)$ the periodic configuration $c(D(x))$. It is important to distinguish between $c(x)$ and $c(D(x))$: the first one is the periodic configuration defined as the repetition of the finite configuration x, while $c(D(x))$ corresponds to the periodic configuration obtained as repetitions of $D(x)$.

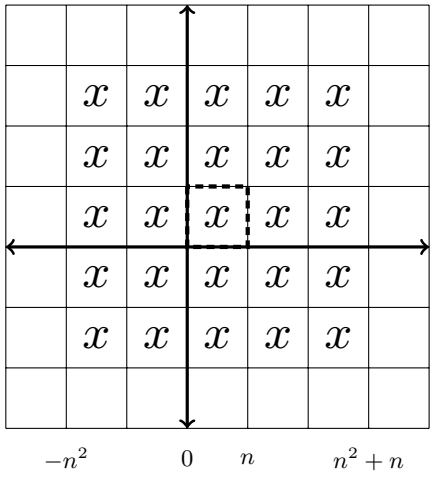


Fig. 1: Construction of the finite configuration $D(x)$ obtained from a finite configuration x of dimension $n \times n = 2 \times 2$. Note that $D(x)$ is of dimensions 14×14.

Lemma 2. *Let $x \in \{0,1\}^{[n]^2}$ be a finite configuration, and let u be a site in $[n] \times [n]$ such that $x_u = 0$. Then u is stable for $c = c(x)$ if and only if it is stable for $D(c)$.*

Proof. Suppose first that u is stable for c, i.e. in the fixed point c' reached from c, $c'_u = 0$. Call c'' the fixed point reached from $D(c)$. Note that $D(c) \leq c$ (where $\leq$ represent the inequalities coordinate by coordinate). Since the FNSMCA automata is monotonic, we have that $c'' \leq c'$, so $c''_u = 0$. Then u is stable for $D(c)$.

Conversely, suppose that $u \in [n] \times [n]$ is not stable for c, and let S be the set of all sites at distance at most n^2 from u. We know that in each step on the dynamics of c, at least one site in the periodic configuration changes its state, then in at most n^2 steps the site u will be in state 1. In other words, the state of u depends only on the states of the sites at distance at most n^2 from u. Note that for every $v \in S$, $c_v = D(c)_v$. Therefore, u is not stable in $D(c)$. □

The set of sites (i, j) of $D(x)$ satisfying $(i, j) \in [m] \times \big([-n^2 - n, -n^2 - 1) \cup (n^2 + n + 1, n^2 + 2n]\big)$ or $(i, j) \in \big([-n^2 - n, -n^2 - 1) \cup (n^2 + n + 1, n^2 + 2n]\big) \times [m]$, are called the *border* B of $D(x)$. Note that the border of $D(x)$ contains only sites in state 0. We call $D(x) - B$ the *interior* of $D(x)$. Note that B is tri-connected and forms a set of sites stable for $D(c)$ thanks to Lemma 1. We call Z the set of sites w in $[m] \times [m]$ such that $D(x)_w = 0$.

Lemma 3. *Let u be a site in $[n] \times [n]$ stable for $D(c)$. Then, there exist three disjoint paths on $G[Z]$ connecting u with sites of the border B. Moreover, the paths contain only sites that are stable for $B(c)$.*

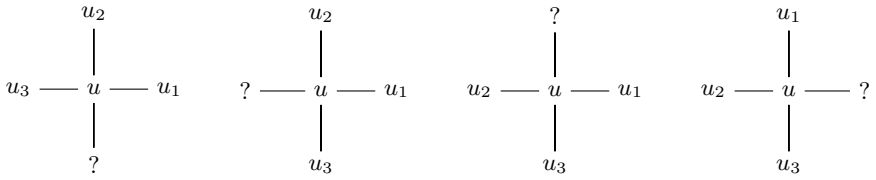
Proof. Suppose that u is stable. From Lemma 1 this implies that u has three stable neighbors. Let $0 \leq i, j \leq n$ be such that $u = (i, j)$. We divide the interior of $D(c)$ in four quadrants:

- The first quadrant contain all the sites in $D(x)$ with coordinates at the north-east of u, i.e., all the sites $v = (k, l)$ such that $k \geq i$ and $l \geq j$.
- The second quadrant contain all the sites in $D(x)$ with coordinates at the north-west of u, i.e., all the sites $v = (k, l)$ such that $k \leq i$ and $l \geq j$.
- The third quadrant contain all the sites in $D(x)$ with coordinates at the south-west of u, i.e., all the sites $v = (k, l)$ such that $k \leq i$ and $l \leq j$.
- The fourth quadrant contain all the sites in $D(x)$ with coordinates at the south-east of u, i.e., all the sites $v = (k, l)$ such that $k \geq i$ and $l \leq j$.

We will construct three disjoint paths in $G[Z]$ connecting u with the border, each one passing through a different quadrant. The idea is to first choose three quadrants, and then extend three paths starting from u iteratively picking different stable sites in the chosen quadrants, until the paths reach the border.

Suppose without loss of generality that we choose the first, second and third quadrants, and let u_1, u_2 and u_3 be three stable neighbors of u, named according to Figure 2.

Starting from u, u_1, we extend the path P_1 through the endpoint different than u, picking iteratively a stable site at the east, or at the north if the site in the north is not stable. Such sites will always exist since by construction the current endpoint of the path will be a stable site, and stable sites must have three stable neighbors (so either one neighbor at east or one neighbor at north).



(a) Case 1 (b) Case 2 (c) Case 3 (d) Caso 4

Fig. 2: Four possible cases for u_1, u_2 and u_3. Note that one of these four cases must exist, since u has at least three stable neighbors. From u_1 we will extend a path through the first quadrant, from u_2 a path through the second quadrant, and from u_3 a path through the third quadrant.

The iterative process finishes when P_1 reaches the border. Note that necessarily P_1 is contained in the first quadrant. Analogously, we define paths P_2 and P_3, starting from u_2 and u_3, respectively, and extending the corresponding paths picking neighbors at the north-west or south-west, respectively. We obtain that P_2 and P_3 belong to the second and third quadrants, and are disjoint from P_1 and from each other.

This argument is analogous for any choice of three quadrants. We conclude there exist three disjoint paths of stable sites from u to the border B. □

Lemma 4. *Let u, v be two sites in $[n] \times [n]$ stable for $D(c)$. Then, there exist three disjoint u, v-paths in $G[Z]$ consisting only of sites that are stable for $D(c)$.*

Proof. Let u, v be stable vertices. Without loss of generality, we can suppose that $u = (i, j)$, $v = (k, l)$ with $i \leq k$ and $j \leq l$ (otherwise we can rotate x to obtain this property). In this case u and v divide the interior of $D(x)$ into nine regions (see Figure 3). Let $P_{u,2}, P_{u,3}, P_{u,4}$ be three disjoint paths that connect u with the border through the second, third and fourth quadrants of u. These paths exist according to the proof of Lemma 3. Similarly, define $P_{v,1}, P_{v,2}, P_{v,3}$ three disjoint paths that connect v to the border through the first, second and third quadrants of v.

Observe fist that $P_{u,3}$ touches regions that are disjoint from the ones touched by $P_{v,1}, P_{v,2}$ and $P_{v,3}$. The same is true for $P_{v,1}$ with respect to $P_{u,2}, P_{u,3}, P_{u,4}$. The first observation implies that paths $P_{u,3}$ and $P_{v,1}$ reach the border without intersecting any other path. Let w_1 and w_2 be respectively the intersections of $P_{u,3}$ and $P_{v,1}$ with the border. Let now P_{w_1,w_2} be any path in G_B connecting w_1 and w_2. We call $P_{1,3}$ the path induced by $P_{u,3} \cup P_{w_1,w_2} \cup P_{v,1}$.

Observe now that $P_{u,2}$ and $P_{v,4}$ must be disjoint, as well as $P_{u,4}$ and $P_{v,2}$. This observation implies that $P_{u,2}$ either intersects $P_{v,2}$ or it do not intersect any other path, and the same is true for $P_{u,4}$ and $P_{v,4}$. If $P_{u,2}$ does not intersect $P_{v,2}$, then we define a path $P_{2,2}$ in a similar way than $P_{1,3}$, i.e., we connect the endpoints of $P_{u,2}$ and $P_{v,2}$ through a path in the border (we can choose this path disjoint from $P_{1,3}$ since the border is tri-connected). Suppose now that $P_{u,2}$ intersects $P_{v,2}$. Let w the first site where $P_{u,2}$ and $P_{v,2}$ intersect, let $P_{u,w}$ be the

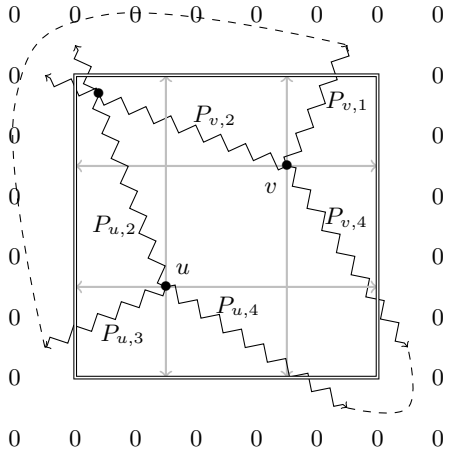


Fig. 3: Vertices u and v divides the interior of $D(x)$ into four regions each. Together they split the space into nine regions. According to Lemma 3), we can choose three disjoint paths connecting u and v, in such a way that each of the nine regions intersect at most one path. We use the border of $D(x)$ to connect the paths that do do not intersect in the interior of $D(x)$.

u, w-path contained in $P_{u,2}$, and let $P_{w,v}$ be the w, v-path contained in $P_{v,2}$. We call in this case $P_{2,2}$ the path $P_{u,w} \cup P_{w,v}$. Note that also in this case $P_{2,2}$ is disjoint from $P_{1,3}$. Finally, we define $P_{4,4}$ in a similar way using paths $P_{u,4}$ and $P_{v,4}$. We conclude that $P_{1,3}$, $P_{2,2,}$ and $P_{4,4}$ are three disjoint paths of stable sites connecting u and v in $G[Z]$. □

We are now ready to show our characterization of stable set of vertices.

Theorem 1. *Let $x \in \{0,1\}^{[n]^2}$ be a finite configuration, and let u be a site in $[n] \times [n]$. Then, u is stable for $c = c(x)$ if and only if u is contained in a tri-connected component of $G[Z]$.*

Proof. From Lemma 2, we know that u is stable for c if and only if it is stable $D(c)$. Let S be the set of sites stable for $D(c)$. We claim that S is a tri-connected component of $G[Z]$. From Lemma 4, we know that for every pair of sites in S there exist three disjoint paths in $G[S]$ connecting them, so the set S must be contained in some tri-connected component T of $G[Z]$. Since $G[T]$ is a graph of degree at least three, and the sites in T are contained in Z, then Lemma 1 implies that T must form a stable set of vertices, then T equals S.

On the other direction, Lemma 1 implies that any tri-connected component of $G[Z]$ must form a stable set of vertices for $D(c)$, so u is stable for c. □

4 The algorithm

We are now ready to show the main result of this paper.

Theorem 2. PREDICTION *is in* **NC**.

Proof. Let (x, u) be an input of PREDICTION, i.e. x is a finite configuration of dimensions $n \times n$, and u is a site in $[n] \times [n]$. Our algorithm for PREDICTION first computes from x the finite configuration $D(x)$. Then, the algorithm uses the algorithm of Proposition 1 to compute the tri-connected components of $G[Z]$, where Z is the set of sites w such that $D(x)_w = 0$. Finally, the algorithm answers *no* if u belongs to some tri-connected component of $G[Z]$, and answer *yes* otherwise.

Algorithm 1 PREDICTION

Input: x a finite configuration of dimensions $n \times n$ and $u \in [n] \times [n]$ such that $x_u = 0$.

1: Compute the finite configuration $D(x)$ of dimensions $m \times m$ with $m = 2n^2 + 3n$
2: Compute the set $Z = \{w \in [m] \times [m] \ : \ D(x)_w = 0\}$.
3: Compute the graph $G[Z]$.
4: Compute the set $\mathcal{T}$ of tri-connected components of $G[Z]$.
5: **for** each $T \in \mathcal{T}$ **do**
6: **if** $u \in T$ **then**
7: **return** *no*
8: **end if**
9: **end for**
10: **return** *yes*

The correctness of Algorithm 1 is given by Theorem 1. Indeed, the algorithm answers *yes* on input (x, u) only when u does not belong to a tri-connected component of $G[Z]$. From Theorem 1, it means that u is not stable, so there exists $t > 0$ such that $F^t(c(x))_u = 1$.

Step **1** can be done in $\mathcal{O}(\log n)$ time with $m^2 = \mathcal{O}(n^6)$ processors: one processor for each site of $D(x)$ computes from x the value of the corresponding site in $D(x)$. Step **2** can be done in time in $\mathcal{O}(\log m) = \mathcal{O}(\log n)$ with $\mathcal{O}(m^2)$ processors, representing Z as a vector in $\{0,1\}^{m^2}$, each coordinate is computed by a processor. Step **3** can be done in time $\mathcal{O}(\log n)$ and $\mathcal{O}(m^2)$ processors: we give one processor to each site in Z, which fill the corresponding four coordinates of the adjacency matrix of $G[Z]$. Step **4** can be done in time $\mathcal{O}(\log^2 n)$ with $\mathcal{O}(n)$ processors using the algorithm of Proposition 1. Finally, steps **5** to **10** can be done in time $\mathcal{O}(\log n)$ with $\mathcal{O}(n^2)$ processors: the algorithm checks in parallel if u is contained in each tri-connected components. There are $\mathcal{O}(n)$ tri-connected components, each of them containing $\mathcal{O}(n)$ elements. All together the algorithm runs in time $\mathcal{O}(\log^2 n)$ with $\mathcal{O}(n^6)$ processors.

5 Conclusion

We showed that the prediction problem for the two-dimensional freezing non-strict majority automaton is in **NC**. This question was posed in [12] and [7].

In [12] it is shown that the prediction problem for freezing non-strict majority automaton on an arbitrary graph of degree at most four is **P**-Complete, and in graphs of degree at most three is in **NC**. The authors conjectured that this problem is in **NC** on any planar topology. We remark that if we remove the hypothesis that the topology is a grid, then our characterization of stable sets (Theorem 1) is no longer true, even for planar regular graphs of degree four. Indeed a regular graph of degree four might not be tri-connected (see Figure 4).

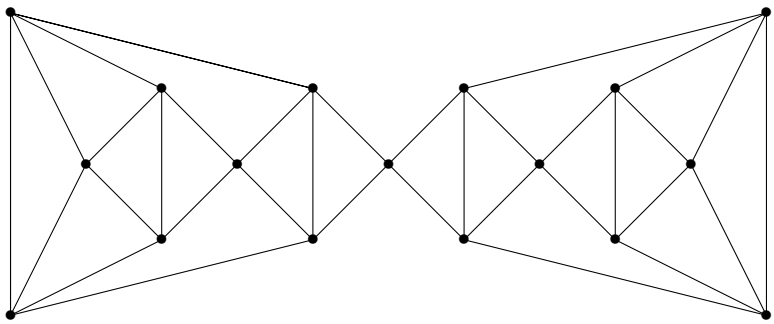


Fig. 4: Non-triconnected degree 4 graph. If all the vertices are initially in state 0, then the configuration is stable.

In our prediction problem, our goal was to compute the configuration obtained once the fixed point is reached. A variant of this problem could consider the complexity of the problem consisting in, given a configuration c and $T > 0$ and a site v, compute state of v in the configuration obtained after T steps, i.e., compute $F^T(c)_v$. Our algorithm is not valid for this problem, since a site might not be stable for the input configuration, but change its state in more than T steps. We believe that this version of the prediction problem is harder than the one treated in this paper. Indeed, we can create very simple gadgets that allow to simulate planar monotone circuitry only for a fixed number of steps, but that are destroyed when we do not bound the time of the simulation. To our knowledge, there are no example of an automata network capable to both simulate planar monotone circuitry, and which corresponding prediction problem is in **NC**.

References

1. Thomas C Schelling. *Micromotives and macrobehavior*. WW Norton & Company, 2006.
2. Pablo Medina, Eric Goles, Roberto Zarama, and Sergio Rica. Self-organized societies: On the sakoda model of social interactions. *Complexity*, 2017, 2017.
3. Nicolás Goles Domic, Eric Goles, and Sergio Rica. Dynamics and complexity of the schelling segregation model. *Physical Review E*, 83(5):056111, 2011.

4. Claudio Castellano, Santo Fortunato, and Vittorio Loreto. Statistical physics of social dynamics. *Reviews of modern physics*, 81(2):591, 2009.
5. Rainer Hegselmann. Modeling social dynamics by cellular automata. *Computer modeling of social processes*, pages 37–64, 1998.
6. Raymond Greenlaw, Howard Hoover, and Walter Ruzzo. *Limits to Parallel Computation: P-completeness Theory*. Oxford University Press, Inc., New York, NY, USA, 1995.
7. Cristopher Moore. Majority-vote cellular automata, ising dynamics, and p-completeness. Working papers, Santa Fe Institute, 1996.
8. Ioannis Karafyllidis and Adonios Thanailakis. A model for predicting forest fire spreading using cellular automata. *Ecological Modelling*, 99(1):87 – 97, 1997.
9. M.A. Fuentes and M.N. Kuperman. Cellular automata and epidemiological models with spatial dependence. *Physica A: Statistical Mechanics and its Applications*, 267(3):471–486, 1999.
10. J Chalupa, P L Leath, and G R Reich. Bootstrap percolation on a bethe lattice. *Journal of Physics C: Solid State Physics*, 12(1):L31, 1979.
11. Eric Goles, Nicolas Ollinger, and Guillaume Theyssier. Introducing Freezing Cellular Automata. In *Cellular Automata and Discrete Complex Systems, 21st International Workshop (AUTOMATA 2015)*, volume 24 of *TUCS Lecture Notes*, pages 65–73, Turku, Finland, June 2015.
12. Eric Goles, Pedro Montealegre-Barba, and Ioan Todinca. The complexity of the bootstraping percolation and other problems. *Theoretical Computer Science*, 504:73–82, 2013.
13. Joseph JáJá. *An Introduction to Parallel Algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1992.
14. Joseph JáJá and Janos Simon. Parallel algorithms in graph theory: Planarity testing. *SIAM J. Comput.*, 11(2):314–328, 1982.