Electronic Colloquium on Computational Complexity, Report No. 159 (2006)

Computational Complexity of Some Enumeration Problems About Uniformly Sparse Boolean Network Automata

PREDRAG T. TOŠIĆ
Department of Computer Science, University of Illinois at Urbana-Champaign
201 N. Goodwin Avenue, Urbana, IL 61801, U.S.A.
p-tosic@cs.uiuc.edu

Abstract

We study the computational complexity of *counting* the *fixed point* configurations (FPs), the *predecessor* configurations and the *ancestor* configurations in certain classes of *graph* or *network automata* viewed as discrete dynamical systems. Some early results of this investigation are presented in two recent ECCC reports [39, 40]. In particular, it is proven in [40] that both exact and approximate counting of FPs in the two closely related classes of Boolean network automata, called *Sequential* and *Synchronous Dynamical Systems* (SDSs and SyDSs, respectively), are computationally intractable problems when each node is required to update according to a *monotone* Boolean function. In the present paper, we further strengthen those results by showing that the intractability of exact enumeration of FPs of a monotone Boolean SDS or SyDS still holds even when (i) the monotone update rules are restricted to *linear threshold functions*, and (ii) the underlying graph is *uniformly sparse.* By *uniform sparseness* we mean that every node in the graph has its degree bounded by $c = O(1)$ for a small value of constant c. In particular, we prove that exactly enumerating FPs in such SDSs and SyDSs remains **#P**-complete even when no node degree exceeds $c = 3$. Among other consequences, we show that this result also implies intractability of determining the exact memory capacity of discrete Hopfield networks with *uniformly sparse* and *nonnegative* integer weight matrices.

Keywords: Cellular and network automata, sequential and synchronous dynamical systems, discrete Hopfield networks, fixed point configurations, computational complexity, **#P**-completeness

1 Introduction

We study certain classes of *network automata* that can be used as an abstraction of the large-scale multi-agent systems made of simple reactive agents, of *ad hoc* communication networks, and, more generally, of dynamical systems whose complex dynamics stems from coupling of and interaction among their relatively simple individual components. These network or graph automata can also be viewed as a theoretical model for the computer simulation of a broad variety of computational, physical, social, and socio-technical distributed infrastructures [1, 9]. In the present work, as well as in several prior, loosely related papers (see, e.g., [2, 3, 4, 5, 6, 7, 33, 38, 39, 40, 41, 42]), the general approach has been to investigate mathematical and computational *configuration space properties* of such graph automata, as a formal way of addressing the fundamental question: what are the possible *global behaviors* of the entire system, given the simple local behaviors of its components, and the interaction pattern among those components?

ISSN 1433-8092

Our recent [38, 39, 40, 42] and ongoing research focus has been on determining *how many* fixed point configurations such network automata have, and *how hard* is the computational problem of *counting* those configurations.

In the present paper, we establish computational intractability of determining the exact number of the fixed point configurations of *sparse Sequential* and *Synchronous Dynamical Systems*, as well as *discrete Hopfield networks*, whose node update rules are restricted to *monotone linear threshold functions*. Moreover, we show that intractability of the exact enumeration of fixed points holds even when the maximum node degree in the underlying graph is bounded by a small constant. We also show similar intractability results for the problems of exact enumeration of all predecessors and all ancestors of a given SDS, SyDS or Hopfield network configuration. It will then follow that, for the networked dynamical systems that can be abstracted via a class of formal network automata, a generally unpredictable global dynamics can be obtained even via *uniformly sparse* couplings of simple, monotonic local interactions.

2 Preliminaries

In this section, we define the discrete dynamical system models studied in this paper, as well as their configuration space properties. *Sequential Dynamical Systems* (SDSs) are proposed in [6, 7, 8] as an abstract model for computer simulations. This model has been successfully applied in the development of large-scale socio-technical simulations such as the *TRANSIMS* project at the Los Alamos National Laboratory [9]. A more detailed discussion of the motivation behind these models, as well as their application to large-scale simulations, can be found in [1, 5, 38, 39], and references therein.

Definition 2.1 *A* Sequential Dynamical System *(SDS)* $\mathcal{S}$ *is a triple* (G, F, Π) *whose components are as follows:*

1. $G(V, E)$ *is a connected undirected graph without multi-edges or self-loops.* $G = G_{\mathcal{S}}$ *is referred to as the* underlying graph *of* $\mathcal{S}$*. We often use* n *to denote* $|V|$ *and* m *to denote* $|E|$*.*

2. *Each node is characterized by its* state*. The state of a node* v_i*, denoted by* s_i*, takes on a value from some finite domain,* $\mathcal{D}$*. In this paper, we shall focus on* $\mathcal{D} = \{0, 1\}$*. We use* d_i *to denote the degree of the node* v_i*. Further, we denote by* $N(i)$ *the set of neighbors of node* v_i *in* G*, including the node* v_i *itself. Each node* v_i *has an associated* node update rule $f_i : \mathcal{D}^{d_i+1} \to \mathcal{D}$*, for* $1 \leq i \leq n$*. We also refer to* f_i *as the* local transition function*. The inputs to* f_i *are* s_i *and the current states of the neighbors of* v_i*. We use* $F = F_{\mathcal{S}}$ *to denote the* global map *of* $\mathcal{S}$*, obtained by appropriately composing together all the local update rules* f_i*,* $i = 1, ..., n$*.*

3. *Finally,* Π *is a permutation of* $V = \{v_1, v_2, \ldots, v_n\}$ *specifying the order in which the nodes update their states using their local transition functions. Alternatively,* Π *can be envisioned as a total ordering on the set of nodes V. In particular, we can view the global map as a sequential composition of the local actions of each* f_i *on the respective state* s_i*, where the node states are updated according to the order* Π*; that is,* $F_{\mathcal{S}} = (f_{\Pi^{-1}(s_1)}, f_{\Pi^{-1}(s_2)}, \ldots, f_{\Pi^{-1}(s_n)})$*.*

The nodes are processed in the *sequential* order specified by the permutation Π. The processing associated with a node consists of computing the new value of its state according to the node's update function, and changing its state to this new value.

If the nodes are required to update simultaneously, all at once, we arrive at the definition of *Synchronous Dynamical Systems* (SyDSs).

Definition 2.2 *A* Synchronous Dynamical System *(SyDS)* $\mathcal{S}' = (G, F)$ *is an SDS* without *the node permutation. In an SyDS, at each discrete time step, all the nodes perfectly synchronously in parallel compute and update their state values.*

Thus, SyDSs are similar to the finite classical cellular automata (CA) [14, 19, 48, 50], except that in an SyDS the nodes may be interconnected in an arbitrary fashion, whereas in a classical cellular automaton the nodes are interconnected in a regular fashion. Another difference is that, while in the classical CA all nodes update according to the same rule, in an SyDS different nodes are allowed to use different local update rules [39].

Much of the early work on sequential and synchronous dynamical systems has primarily focused on the SDSs and SyDSs with *symmetric Boolean functions* as the node update rules [1, 2, 3, 5, 6, 7]. By *symmetric* is meant that the future state of a node does not depend on the order in which the input values of this node's neighbors are specified. Instead, the future state of v_i depends only on $\Sigma_{j \in N(i)}\ s_j$, i.e., on how many of v_i's neighbors are currently in the state $s_j = 1$. In particular, symmetric Boolean SyDSs correspond to *totalistic (Boolean) cellular automata* of Wolfram [50, 51]. The computational complexity of counting various configurations in SDSs and SyDSs with symmetric Boolean update rules is addressed in [39, 42].

We are presently interested in SDSs with the local update rules that are restricted to *monotone* Boolean functions. Our preliminary hardness results about the counting problems in monotone Boolean SDSs and SyDSs can be found in [38, 40]. The SDSs with the local transition rules that are both monotone and symmetric are, in essence, sequential threshold cellular automata [41, 43] that are defined over *arbitrary* finite graphs, as opposed to the usual *regular Cayley graphs* of the classical cellular automata [14].

In this paper, we focus on the monotone update rules that are not necessarily symmetric; however, these monotone Boolean functions will be required to be of a *linear threshold* variety, so that our subsequent results would imply analogous results for *discrete Hopfield networks* [20], whose update rules are, by default, always required to be linear (not necessarily monotone) threshold functions.

We next define the notion of *monotone Boolean functions.*

Definition 2.3 *Given two Boolean vectors,* $X = \langle x_1, x_2, \ldots, x_n \rangle$ *and* $Y = \langle y_1, y_2, \ldots, y_n \rangle$*, define a binary relation "*$\preceq$*" as follows:* $X \preceq Y$ *if* $x_i \leq y_i$ *for all* i*,* $1 \leq i \leq n$*. An* n*-input Boolean function* f *is* monotone *if* $X \preceq Y$ *implies that* $f(X) \leq f(Y)$.

Notice that the notion of monotonicity given in Definition 2.3 allows us to compare only Boolean vectors of the same length.

A configuration of a Boolean SDS $\mathcal{S} = (G, F, \Pi)$ or an SyDS $\mathcal{S}' = (G, F)$ is a vector $(b_1, b_2, \ldots, b_n) \in \{0,1\}^n$. A configuration $\mathcal{C}$ can also be thought of as a function $\mathcal{C} : V \to \{0,1\}^n$.

The (global) map computed by an S(y)DS $\mathcal{S}$, denoted $F = F_{\mathcal{S}}$, specifies for each configuration $\mathcal{C}$ the next configuration $\mathcal{C}'$ reached by $\mathcal{S}$ after carrying out the updates of all the node states, whether in parallel or in the order given by Π. Thus, the map $F_{\mathcal{S}} : \{0,1\}^n \to \{0,1\}^n$ is a total function on the set of global configurations. This function therefore defines the dynamics of $\mathcal{S}$. We say that $\mathcal{S}$ moves from a configuration $\mathcal{C}$ to a configuration $F_{\mathcal{S}}(\mathcal{C})$ in a single transition step. Alternatively, we say that S(y)DS $\mathcal{S}$ moves from a configuration $\mathcal{C}$ at time t to a configuration $F_{\mathcal{S}}(\mathcal{C})$ at time $t+1$. Assuming that

each node update function f_i is computable in time polynomial in the size of the description of $\mathcal{S}$, each transition step will also take polynomial time in the size of the S(y)DS's description.

The *configuration space*[1] $\mathcal{P}_{\mathcal{S}}$ of an SDS or SyDS $\mathcal{S}$ is a directed graph defined as follows. There is a vertex in $\mathcal{P}_{\mathcal{S}}$ for each global configuration of $\mathcal{S}$. There is a directed edge from a vertex representing configuration $\mathcal{C}$ to that representing configuration $\mathcal{C}'$ if $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}'$. Since an SDS or SyDS is deterministic, each vertex in its configuration space has the out-degree of 1.

Definition 2.4 *Given two configurations $\mathcal{C}'$ and $\mathcal{C}$ of an SDS or SyDS $\mathcal{S}$, configuration $\mathcal{C}'$ is a* predecessor *of $\mathcal{C}$ if $F_{\mathcal{S}}(\mathcal{C}') = \mathcal{C}$, that is, if $\mathcal{S}$ moves from $\mathcal{C}'$ to $\mathcal{C}$ in one global transition step. Similarly, $\mathcal{C}'$ is an* ancestor *of $\mathcal{C}$ if there is a positive integer $t \geq 1$ such that $F_{\mathcal{S}}{}^{t}(\mathcal{C}') = \mathcal{C}$, that is, if $\mathcal{S}$ evolves from $\mathcal{C}'$ to $\mathcal{C}$ in one or more transitions.*

In particular, a predecessor of a configuration is a special case of an ancestor of that configuration.

Definition 2.5 *A configuration $\mathcal{C}$ of an S(y)DS $\mathcal{S}$ is a* fixed point *(FP) configuration if $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}$, that is, if the transition out of $\mathcal{C}$ is back to $\mathcal{C}$ itself.*

Note that a fixed point is a configuration that is its own predecessor.

We shall focus in this paper on the computational complexity of the problems of *counting* how many *fixed point* configurations (FPs), predecessor configurations, and/or ancestor configurations a given sequential or synchronous dynamical system has. We will show that these problems in general remain hard, even when both the underlying graph structure of an SDS or SyDS, and the local update rules, are severely restricted. Moreover, we will also establish similar results for discrete Hopfield networks.

3 Related Work

Various models of *cellular* and *network automata* have been studied in a variety of contexts, from unconventional models for parallel and distributed computing (e.g., [14, 32, 43]), to complex dynamical systems (e.g., [15, 16, 27]), to theoretical biology (e.g., [28, 29]). Beside the classical (parallel) cellular automata [14, 19] and their sequential or asynchronous variants [24, 43], perhaps the most studied class of models of network or graph automata are *Hopfield networks* [20, 21]. While our original interests in the area of cellular and network automata have primarily focused on studying the structural and behavioral properties of Sequential and Synchronous Dynamical Systems [5, 38, 39, 40, 42], given the prominence of Hopfield networks in the literature, we shall also briefly address the pertinent counting problems in the context of (discrete-time) Hopfield nets towards the end of this paper.

The SDS and SyDS models introduced in Section 2 are closely related to the *graph automata* (GA) models studied in [31, 34] and the one-way cellular automata studied in [36]. In fact, the general finite-domain SyDSs exactly correspond to the graph automata of Nichitiu and Remila in [34]. Barrett, Mortveit and Reidys [6, 7, 33] and Laubenbacher and Pareigis [30] investigate the mathematical properties of sequential dynamical systems. Barrett *et al.* study the computational complexity of several problems about the configuration space structure of SDSs and SyDSs. These problems include the REACHABILITY, PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE problems [3, 4]. Problems related to the existence of GARDEN OF EDEN and FIXED POINT configurations are studied in [5]. In particular, **NP**-completeness for the problem of FIXED POINT EXISTENCE (FPE) in various restricted classes of Boolean

[1] Also sometimes called *phase space* in the literature; see, e.g., [4, 5].

S(y)DSs is proven in [5]. However, the FPE problem becomes easy when all the nodes of a Boolean S(y)DS are required to update according to *monotone functions.*

The subarea of computational complexity that addresses counting or enumeration of various combinatorial structures dates back to the seminal work of L. Valiant in the late 1970s [45, 46]. Counting problems naturally arise in many contexts, from approximate reasoning and Bayesian belief networks in AI (e.g., [37]), to network reliability and fault-tolerance (e.g., [44]), to statistical physics (e.g., [25, 27]).

It has been observed, however, that the progress in understanding the complexity of counting problems has been much slower than the progress related to our understanding of decision and search problems [18, 44]. Since the reductions used in proving counting problems hard have to preserve the number of solutions, rather than just whether a solution exists or not, they are in general more difficult to devise than the reductions used to establish, say, **NP**-completeness of the corresponding decision problems. For example, most standard reductions used to establish computational hardness of certain decision or search problems on graphs tend to "blow up" the underlying graph, thereby destroying the local structures that impact the number of that problem's solutions [18]. One area where this understanding of the complexity of counting has been particularly poor, is whether the general counting problems that are provably hard remain hard when various restrictions are placed on the problem instances [44]. Some of the relatively recent results in this area, such as those on the hardness of counting in *planar graphs* [23], and especially in *sparse graphs* [18, 44], have directly inspired our recent work (see [38, 39, 40, 42]), as well as the investigations summarized in the present paper.

Insofar as the complexity of counting in the context of discrete dynamical systems, there have been surprisingly few theoretical results. Some results along the lines of our work in [39, 40] and the present paper, but in the context of *discrete Hopfield networks*, can be found in [10, 11, 12]. We will discuss at the end of the next Section how our results on Boolean SDSs and SyDSs with monotone linear threshold rules strengthen those in [10] for the *symmetric* discrete Hopfield nets with *sparse* and *nonnegative* integer weight matrices.

4 Counting Various Configurations of Monotone Boolean SDSs and SyDSs

Monotone Boolean functions, formulae and circuits [47] have been extensively studied in many areas of computer science, from machine learning to connectionist models in AI to VLSI circuit design. Cellular and other types of network automata with the local update rules restricted to monotone Boolean functions have also been of a considerable interest (e.g., [5, 41]). The problem of counting FPs in *monotone* Boolean SDSs and SyDSs is originally addressed in [38, 40]. It is shown there that, in general, counting FPs of such S(y)DSs either exactly or approximately is computationally intractable. This intractability holds even for the graphs that are simultaneously bipartite, planar, and very sparse *on average* [38, 40]. In particular:

Lemma 4.1 [40] *Counting exactly the fixed points of a monotone Boolean SDS or SyDS defined over a star graph, and such that the update rule of the central node is given as a* MONOTONE 2CNF *Boolean formula of size $O(n)$, where n is the number of nodes in the star graph, is* **#P***-complete.*

Moreover, by the results of D. Roth in [37], subsequently strengthened by S. Vadhan in [44], the problem of *approximately* counting FPs in the setting as in Lemma 4.1 above can be readily shown to be **NP**-hard [40].

In summary, enumerating the fixed points of *monotone* Boolean SDSs and SyDSs defined on bipartite, planar and sparse on average underlying graphs *exactly* is **#P**-complete, and for any $\epsilon > 0$, *approximating*

the number of FPs in such monotone S(y)DSs to within $2^{|V|^{1-\epsilon}}$ is **NP**-hard. Our next goal is to show that the hardness of the exact enumeration of FPs for monotone S(y)DSs holds even when the underlying graphs are required to be *uniformly sparse.* We will also argue that, as a consequence of our construction in the proof of Theorem 4.1 below, the problem of enumerating the stable configurations of *discrete Hopfield networks* with very sparse *weight matrices* is also in general computationally intractable.

Given the importance of the number of stable configurations of a Hopfield network viewed as an *associative memory* (e.g., [15]), we now informally introduce *discrete Hopfield networks*, and briefly summarize what has been known about the problem of counting their stable configurations.

A *discrete Hopfield network* (DHN) [20] is made of n binary-valued nodes; the set of node states is, by convention, $\{-1, +1\}$. Associated to each pair of nodes (v_i, v_j) is (in general, real-valued) *weight*, $w_{ij} \in \mathbf{R}$. The *weight matrix* of a DHN is defined as $W = [w_{ij}]_{i,j=1}^{n}$. Each node also has a fixed *threshold*, $h_i \in \mathbf{R}$. A node v_i updates its state x_i from time step t to step $t + 1$ according to

$$x_i^{t+1} \leftarrow sgn(\sum_{j=1}^{n} w_{ij} \cdot x_j^t - h_i) \tag{1}$$

In the sequel, we will not bother to explicitly distinguish between an S(y)DS's or DHN's node, v_i, and this node's state, s_i or x_i; the intended meaning will be clear from the context.

In the standard DHN model, the nodes update synchronously in parallel, similarly to the classical cellular automata and the SyDSs as defined in Section 2. However, *asynchronous Hopfield networks*, where the nodes update sequentially, one at a time, have also been studied [12, 20]. In these sequential DHNs, unlike SDSs, it is not required that the nodes update according to a *fixed permutation*. However, these differences are inconsequential insofar as the fixed points are concerned [33]. For simplicity and space constraints reasons, we will focus on synchronously updating, *parallel* DHNs in the rest of the paper.

In much of the Hopfield networks literature, the weight matrix W is assumed symmetric, i.e., for all pairs of indices $\{i, j\}$, $w_{ij} = w_{ji}$ holds. A DHN is called *simple* if $w_{ii} = 0$ for all $i = 1, ..., n$ [12]. Simple DHNs are thus a generalization of *memoryless* finite cellular automata with linear threshold update rules [14, 50].

In [10], Floreen and Orponen establish the following two interesting results:[2]

(i) the problem of determining the number of fixed point configurations of a simple discrete Hopfield network, with a symmetric weight matrix $W = [w_{ij}]$ such that all the weights w_{ij} are integers, and with $w_{ii} = 0$ along the main diagonal, is **#P**-complete; and

(ii) the problem of determining the number of predecessor configurations of a given configuration of a simple discrete Hopfield network, with a symmetric weight matrix $W = [w_{ij}]$ such that all the weights w_{ij} are from the set $\{-1, 0, +1\}$, and with $w_{ii} = 0$ along the main diagonal, is **#P**-complete.

For proving (i), Floreen and Orponen devise a Hopfield network that is relatively dense, i.e., with quite a few non-zero weights w_{ij}. This would correspond to an SDS or SyDS where there are, informally speaking, several nodes each of which having many neighbors. In contrast, our result in Lemma 4.1 allows only for a *single node* that has a large neighborhood; see [38, 40] for more details.

Prior to proving our first main result, for the sake of completeness, we state the following

[2] We (slightly) rephrase these results from the language originally used in [10] into the discrete dynamical systems language we have used in [38, 39, 40] and the present paper, in order to make the comparison and contrast with our results more transparent.

Lemma 4.2 *Counting FPs of an arbitrary SDS or SyDS all of whose nodes use Boolean-valued linear threshold rules is* **#P***-complete.*

We shall show next that the result (i) from [10] can be considerably strengthened along several dimensions. That is, the hardness of counting FPs will be proven to still hold even when the following restrictions on problem instances are *simultaneously* imposed:

– the underlying graphs will be required to be *uniformly sparse*, with no node degree exceeding 3;

– all linear threshold update rules will be restricted to *monotone* functions by disallowing negative weights;

– only two (positive) integer values for the weights will be allowed; and

– each S(y)DS node will choose one from only two allowed monotone linear threshold functions.

Since each node of an SDS or SyDS in the Theorem below is required to have only $O(1)$ neighbors, the issue of *encoding* of the local update rules, that is discussed in detail in [40], is essentially irrelevant here. In particular, even a truth table with one row for each combination of the values of a given node's neighbors is permissible [39, 40]. In the sequel, BOOL-MON-S(Y)DS will stand for a *monotone Boolean* SDS or SyDS.

Theorem 4.1 *Counting the fixed points of* BOOL-MON-S(Y)DS*s* exactly *is* **#P***-complete, even when all of the following restrictions on the structure of such an S(y)DS* simultaneously *hold:*

- *the monotone update rules are* linear threshold functions *– in particular, monotonicity of the linear threshold update rules implies that all weights satisfy* $w_{ij} \geq 0$*;*
- *the S(y)DS is with memory, and such that, along the main diagonal,* $w_{ii} = 1$ *for all indices* i*,* $1 \leq i \leq |V|$ *(where* $|V|$ *denotes the number of the S(y)DS's nodes);*
- *at most two different positive integer weights are used by each local update rule;*
- *each node has at most three neighbors in the underlying graph of this S(y)DS;*
- *only two different monotone linear threshold functions are used by the S(y)DS's nodes.*

Proof: We first describe the construction of a BOOL-MON-SYDS from an instance of a *Boolean monotone 2CNF* (MON-2CNF) formula [13] such that no variable appears in more than three different clauses. We then outline why is this reduction from the problem of counting satisfying assignments of such a formula to the problem of counting FPs in the resulting SyDS *weakly parsimonious* [13].

Let's assume that a MON-2CNF Boolean formula is given, such that there are n variables, m clauses, each variable appears in at least one clause, and no variable appears in more than three clauses. In particular, these requirements imply that $m = O(n)$, but we shall keep m and n as two distinct parameters for clarity.

The corresponding SyDS $\mathcal{S}$ is constructed as follows. To each variable in the formula corresponds a variable node, and to each clause, a clause node. In addition, a *cloned clause node* is introduced for each of the original m clause nodes. Thus, the underlying graph of $\mathcal{S}$ has exactly $n + 2m$ nodes. A variable node is adjacent to a clause node if and only if, in the Boolean formula, the corresponding variable appears in the corresponding clause. Each clause node is adjacent to its clone. Finally, the cloned clause nodes form a ring among themselves. Therefore, the underlying graph of this SyDS looks as in *Figure 1*.

In the sequel, we shall slightly abuse the notation and use x_i to denote *both* a variable in the Boolean formula, and the corresponding *variable clause* in the S(y)DS or discrete Hopfield network we are constructing; similarly, C_j will denote both a clause in a Boolean formula, and a clause node in the S(y)DSs

or Hopfield network that is being constructed from that formula. The intended meaning will be clear from the context.

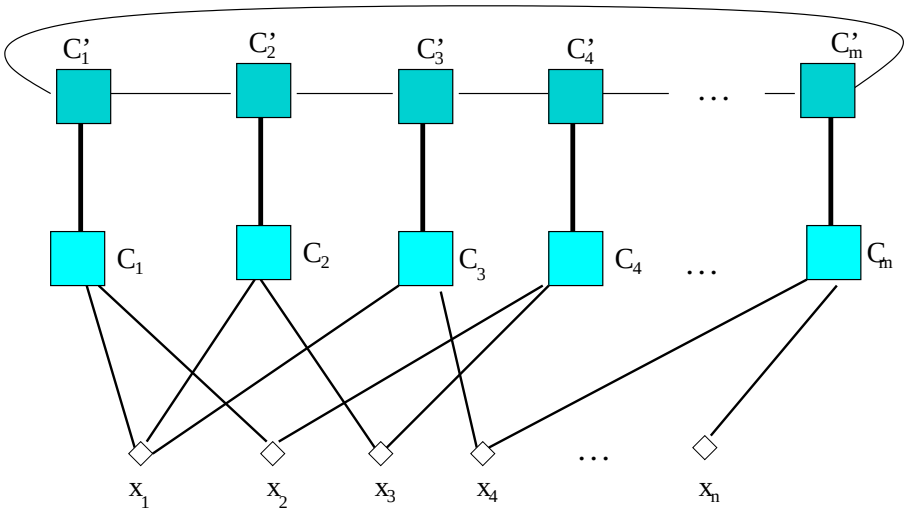

Figure 1: *The underlying graph of a bounded-degree monotone linear threshold Boolean S(y)DS in the construction of Theorem 4.1. The original clause nodes are marked C_j, the cloned clause nodes are primed, as in C'_j, and the variable nodes are denoted by x_i.*

With this convention in mind, we now define the update rules for the clause nodes, the cloned clause nodes, and the variable nodes of the SyDS that we are constructing from a MON-2CNF Boolean formula. The cloned clause nodes C'_j and the variable nodes x_i will update according to the Boolean *AND* rule. The original clause nodes, C_j, will update according to the following monotone linear threshold update rule:

$$C_j \leftarrow \begin{cases} 1, & \text{if } 2C'_j + C_j + x_{j_1} + x_{j_2} \geq 4 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where x_{j_1}, x_{j_2} is a shorthand for the two variable nodes that are adjacent to the clause node C_j.

The given construction can be slightly rephrased, in order to emphasize that the resulting SyDS also satisfies the *symmetry requirement* as it is usually defined in the Hopfield networks literature, namely, so that the underlying matrix of weights is a symmetric matrix. To that end, the Boolean *AND* rule used by the cloned clause nodes can be written in an equivalent, but more "linear-threshold-like", form:

$$C'_j \leftarrow \begin{cases} 1, & \text{if } 2C_j + C'_j + C'_{j-1} + C'_{j+1} \geq 5 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Notice that the function defined in equation (3) evaluates to 1 if and only if all of its inputs are 1, and thus, indeed, the given formula is nothing but a linear-threshold-like way of writing the ordinary Boolean *AND* of four variables. If this latter convention on how we write the update rules at the cloned clause nodes and the variable nodes is adopted, then the resulting $\mathcal{S}$ can be also viewed as a discrete Hopfield network with parallel node updates.

We now show that the reduction from the counting problem #MON-2CNF-SAT to the counting problem #FP for the constructed SyDS is, indeed, weakly parsimonious. To that end, we will just summarize the case analysis. If, at any time step t, any of the cloned clause nodes C'_j evaluates to 0, that will ensure that, within no more than $\frac{m}{2}+1$ parallel steps, all the cloned clause nodes will become 0, and stay in state 0 thereafter. This will also cause all the original clause nodes' states C_k, and, consequently, also all the variable nodes' states x_i, to become 0, as well. Thus, if at any point a single cloned clause node's state becomes 0, the entire SyDS will eventually collapse to the "sink" fixed point 0^{n+2m}. Clearly, this sink FP does not correspond to a satisfying assignment to the original Boolean formula.

Now, the only way that no cloned clause node ever evaluates to 0 is that the following two conditions simultaneously hold:

– each C'_k and C_k is initially in the state 1, for $1 \leq k \leq m$; and

– the initial states x_i of the variable nodes are such that they correspond to a satisfying truth assignment to the variables in the original Boolean formula.

If these conditions hold, then each such global configuration $(x^n, C^m, C'^m) = (x^n_{sat}, 1^m, 1^m)$ is a fixed point of $\mathcal{S}$, where $x^n_{sat} \in \{0,1\}^n$ is a short-hand for any n-tuple of Boolean values that corresponds to a satisfying truth assignment $(x_1, ..., x_n)$ to the original monotone 2CNF formula. Moreover, the satisfying truth assignments of the original Boolean formula are in a one-to-one correspondence with these non-sink FPs of $\mathcal{S}$.

Since no variable in the MON-2CNF formula from which we are constructing the SyDS appears in more than three clauses, each variable node x_i in the SyDS has at most three neighbors. Since we use 2CNF, each clause node C_j has two variable node neighbors, plus one cloned clause neighbor, C'_j, for the total of three neighbors. Finally, each cloned clause node C'_j clearly has exactly three neighbors. In particular, by the result of C. Greenhill in [18], we can make the underlying graph of SyDS $\mathcal{S}$ be 3-regular, and the **#P**-completeness of the counting problem #FP will still hold.

We also observe that *only two* different monotone linear threshold functions are used in the construction above; furthermore, at most two different integer weights are used in each of those two linear threshold functions. Hence, the claim of the Theorem follows insofar as the monotone linear threshold SyDSs are concerned. Finally, by the invariance of FPs with respect to the node update ordering [33], it follows that exactly enumerating the fixed point configurations of the monotone linear threshold SDSs defined on uniformly sparse graphs is **#P**-complete, as well. ■

In the construction above, the SyDS dynamics from *every* starting global configurations that is not of the form $(x^n_{sat}, 1^m, 1^m)$ will eventually converge to the sink state 0^{n+2m}. In particular, the *basin of attraction* of $\mathcal{C} = 0^{n+2m}$ includes all configurations of the form $(x^n_{unsat}, 1^m, 1^m)$, where x^n_{unsat} is a shorthand for an ordered n-tuple of Boolean values that corresponds to an *unsatisfying* (i.e., falsifying) truth assignment to the corresponding variables $x_1, ..., x_n$ in the original MON-2CNF formula. The rest of the configurations in the sink's basin of attraction are such that $(C^m, C'^m) \neq (1^m, 1^m)$ (and where $x^n \in \{0,1\}^n$ is arbitrary).

Hence, in order to determine *exactly* the size of the basin of attraction for the sink state $\mathcal{C} = 0^{n+2m}$, that is, the number of this configuration's ancestors, we must be able to exactly determine the number of falsifying truth assignments to the original MON-2CNF Boolean formula. It is easy to see that one can find an ordering Π under which the same claim holds for the corresponding BOOL-MON-SDS. As a consequence, we have

Corollary 4.1 *The problem of counting exactly all the* ancestors *of an arbitrary configuration of a* BOOL-MON-S(Y)DS*, denoted* #ANC*, is* **#P***-complete. Moreover, this intractability result holds even when all restrictions from* Theorem 4.1 *are simultaneously imposed on the S(y)DS's structure.*

4.1 Counting Configurations of Discrete Hopfield Networks

We now turn to the corresponding hardness of counting results for discrete Hopfield networks with appropriately restricted weight matrices. We start with the problem of fixed point enumeration in the context of Hopfield nets where each of the nodes has exactly one bit of memory – namely, its own (binary-valued) current state.

Theorem 4.2 *Determining the exact number of stable configurations of a parallel or asynchronous discrete Hopfield network is* **#P***-complete even when all of the following restrictions on the weight matrix* $W = [w_{ij}]$ *simultaneously hold:*

- *the matrix is symmetric:* $w_{ij} = w_{ji}$ *for all pairs of indices* $i, j \in \{1, ..., |V|\}$ *(where* $|V|$ *denotes the number of nodes in the underlying graph of this DHN);*
- $w_{ii} = 1$ *along the main diagonal for all* $i \in \{1, ..., |V|\}$*;*
- $w_{ij} \in \{0, 1, 2\}$ *for all pairs of indices* $i, j \in \{1, ..., |V|\}$*;*
- *each row and each column of* W *has* at most three *(alternatively,* exactly three*) nonzero entries off the main diagonal.*

Proof sketch: In case of the DHNs whose nodes update synchronously in parallel, the claim holds by virtue of Theorem 4.1, since an SyDS that is constructed as in the proof of that theorem can also be viewed as a parallel discrete Hopfield network whose weight matrix satisfies all the above listed conditions.[3] Insofar as the asynchronous DHNs whose nodes update in arbitrary sequential orders are concerned, while indeed these sequences of node updates need not be repetitions of a fixed permutation as in the corresponding SDSs, this difference can be easily shown to be immaterial insofar as the fixed point configurations are concerned. Therefore, Theorem 4.2 about discrete Hopfield networks is nothing but rephrasing Theorem 4.1, with parallel DHNs in place of SyDSs with monotone linear threshold update rules, and asynchronous/sequential DHNs replacing SDSs with the same kind of update rules. ■

Next, we consider the problems of enumerating predecessors as well as all ancestors of a given Hopfield network configuration. We shall establish the computational complexity of those two related counting problems in the context of *simple* DHNs, whose weight matrices satisfy $w_{ii} = 0$ for $\forall i \in \{1, ..., |V|\}$.

Before we proceed with a formal reduction from the problem #MON-2CNF-SAT to the problem #PRED of enumerating all predecessor configurations of a given DHN configuration, we establish the following additional conventions. First, the reduction will be from the MON-2CNF Boolean formulae with each variable appearing in at least one, and in at most (alternatively, exactly) four clauses. Second, we will abandon the usual convention in the Hopfield networks literature that the underlying graph is fully connected (i.e., a clique), and instead consider those pairs of vertices $\{v_i, v_j\}$ such that $w_{ij} = w_{ji} = 0$ not to be connected by an edge at all. We will require that the underlying DHN weight matrix W is *symmetric*

[3] For simplicity of the argument, in this proof sketch we are ignoring the syntactic difference that the state space of a node in a Hopfield network is $\{-1, +1\}$, not $\{0, 1\}$.

in the usual, Hopfield network sense; as a consequence, the underlying graph of such a discrete Hopfield network will be undirected, which is also in accordance with our convention about S(y)DSs. Third, in the construction used in proving Theorem 4.1, we will eliminate the cloned clause nodes C'_j and, instead, connect the ordinary close nodes into a ring.

We recall that, in a DHN, the set of possible states of a node is traditionally $\{-1, +1\}$ (instead of $\{0, 1\}$); while it's not essential, we will adopt this convention through the rest of the paper insofar as the Hopfield networks are concerned.

We define the update rule of a clause node C_j to be

$$C_j \leftarrow \begin{cases} +1, & \text{if } 2C_{j-1} + 2C_{j+1} + x_{j_1} + x_{j_2} > 3 \\ -1, & \text{otherwise} \end{cases} \tag{4}$$

For each variable x_i in the MON-2CNF formula from which we are constructing our DHN, let a_i denote *the number of clauses in which* x_i *appears*; thus, under our assumptions, for any $i \in \{1, ..., |V|\}$, we have $a_i \in \{1, 2, 3, 4\}$. We now define the variable node update rules as

$$x_i \leftarrow \begin{cases} +1, & \text{if } \sum_{\{j: x_i \in C_j\}} C_j > a_i - 1 \\ -1, & \text{otherwise} \end{cases} \tag{5}$$

Thus a variable node x_i updates to $+1$ if and only if *all* of the clause nodes $C_{j(i)}$ corresponding to those clauses in the formula in which variable x_i appears are currently in the state $+1$.

Finally, we observe that the resulting weight matrix W, while symmetric and with all entries $w_{ij} \in \{0, 1, 2\}$, also has $w_{ii} = 0$ along the main diagonal; therefore, the constructed Hopfield network is *simple* (i.e., *memoryless*) [10, 12].

We are now ready to establish the second main result of this paper:

Theorem 4.3 *The problem* #PRED *of determining the exact number of predecessors of a given configuration of a* simple *discrete Hopfield network is* **#P***-complete. Moreover, this claim holds even when all of the following restrictions on the Hopfield net's weight matrix* $W = [w_{ij}]$ *are simultaneously imposed:*

- *the matrix is symmetric:* $w_{ij} = w_{ji}$ *for all pairs of indices* $i, j \in \{1, ..., |V|\}$*;*
- $w_{ii} = 0$ *along the main diagonal for all* $i \in \{1, ..., |V|\}$*;*
- $w_{ij} \in \{0, 1, 2\}$ *for all pairs of indices* $i, j \in \{1, ..., |V|\}$*;*
- *each row and each column has* at most / exactly four *nonzero entries.*

Proof sketch: The claim of the Theorem will follow from the fact that the satisfying truth assignments to the Boolean variables $x_1, ..., x_n$ in the original MON-2CNF Boolean formula are in a one-to-one correspondence with the set of all predecessors of the configuration $(+1)^{n+m}$ in the Hopfield net constructed from that formula. The case analysis is similar to that in the proof of Theorem 4.1. In particular, any configuration with at least one clause node C_j in the state (-1) will eventually converge to the sink fixed point $(x^n, C^m) = ((-1)^n, (-1)^m)$. Among the configurations of the form $(x^n, C^m) = (x^n, (+1)^m)$, those and only those such that the n-tuple x^n corresponds to a satisfying truth assignment to the original MON-2CNF Boolean formula[4] will evolve to the other fixed point configuration, $(1^n, 1^m) = (+1)^{n+m}$.

[4]Here, we identify Boolean value FALSE of a variable in the MON-2CNF formula with the corresponding DHN variable node's state -1, whereas Boolean value TRUE is identified with the state $+1$ of the corresponding DHN variable node.

Moreover, this convergence to $(+1)^{n+m}$ is easily seen to take a single parallel transition. That is, the predecessors of $(+1)^{n+m}$ are precisely the configurations of the form $(x^n_{sat}, (+1)^m)$.

■

It follows from the discussion in the proof sketch above that *all* ancestors of the configuration $\mathcal{C} = (+1)^{n+m}$ are also this configuration's predecessors; that is, the convergence to $\mathcal{C}$ from every configuration in the basin of attraction of $\mathcal{C}$ takes *exactly one* (global) parallel step.

Corollary 4.2 *The problem* #ANC *of determining the exact number of all ancestors of an arbitrary configuration of a simple discrete Hopfield network is, in the worst case,* **#P***-complete. Moreover, this intractability holds even when all the restrictions from* Theorem 4.3 *on the Hopfield network instances are simultaneously imposed.*

5 Summary and Future Work

We have shown in [39, 40] that enumerating the fixed point configurations of two related classes of network automata, called Sequential and Synchronous Dynamical Systems (SDSs and SyDSs, respectively) is, in general, computationally intractable. We continue the general line of inquiry from [39, 40] in the present paper, as well. However, we now focus on those SDSs and SyDSs each of whose nodes is required to update its state according to a *monotone Boolean function*, and whose underlying network topologies are *uniformly sparse*, so that, in particular, each node has $O(1)$ neighbors.

Our main result in this paper is that exactly counting the fixed points of monotone, uniformly sparse Boolean SDSs and SyDSs such that no node has more than three neighbors is **#P**-complete. This result immediately implies similar intractability results for the sparse discrete Hopfield networks. Viewing a Hopfield net as an associative memory, our results imply that determining exactly how many different patterns can be stored in such an associative memory is, in general, computationally intractable – even when no inhibitive connections (i.e., no edges with negative weights) are allowed, and, simultaneously, no row or column of the weight matrix has more than four nonzero entries. Moreover, this intractability holds even for those DHNs with integer weight matrices all of whose entries are from the set $\{0, 1, 2\}$.

Similarly, determining the exact size of the basin of attraction of a given stable configuration of a discrete Hopfield network with a symmetric weight matrix is equally intractable; moreover, this intractability holds even when the Hopfield network is required to be *simple*, with a uniformly sparse weight matrix, and the same restrictions on the allowed values of weights w_{ij} as in our corresponding result about the enumeration of the stable configurations.

Insofar as the future work is concerned, it needs to be pointed out that our results in this paper, as well as similar in spirit results in our prior, related work [38, 39, 40, 42], all pertain to the *worst-case complexity* of counting the stable configurations and other structures of discrete dynamical systems. What is of a considerable interest to statistical physics, connectionist AI and large-scale multi-agent systems research communities, however, is the *average complexity* of the relevant decision, search and counting problems about the underlying system's dynamics. While our constructions in the proofs of Theorem 4.1 and Theorem 4.3 suggest that there may be many uniformly sparse monotone SDSs, SyDSs and Hopfield networks for which enumerating the stable configurations (FPs), as well as the predecessor and the ancestor configurations, are all computationally intractable, whether the average cases of these important counting problems are computationally tractable or not is still open. We hope to address the average case complexity of those and other similar problems in our future work.

Acknowledgements: The author expresses his gratitude to Gul Agha, Harry Hunt, Michael Loui, Madhav Marathe, Paul Schupp and Mahesh Viswanathan for useful discussions and feedback on various matters related to the research summarized in this paper. The early stages of this work were supported by the ONR MURI Grant, Contract N00014-02-1-0715.

References

[1] C. Barrett, B. Bush, S. Kopp, H. Mortveit and C. Reidys. "Sequential Dynamical Systems and Applications to Simulations", Technical Report, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1999

[2] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns. "Dichotomy Results for Sequential Dynamical Systems", Los Alamos National Laboratory Report, LA–UR–00–5984, Los Alamos, New Mexico, 2000

[3] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns. "Predecessor and Permutation Existence Problems for Sequential Dynamical Systems", Los Alamos National Laboratory Report, LA–UR–01–668, Los Alamos, New Mexico, 2001

[4] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns. "Reachability problems for sequential dynamical systems with threshold functions", *Theoretical Comp. Sci.,* vol. 295, issues 1–3, pp. 41 – 64, February 2003

[5] C. L. Barrett, H. B. Hunt, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, P. T. Tosic. "Gardens of Eden and Fixed Points in Sequential Dynamical Systems", Proc. AA DM–CCG, *Discrete Math. & Theoretical Comp. Sci.*, pp. 95 – 110, 2001

[6] C. Barrett, H. Mortveit, and C. Reidys. "Elements of a theory of simulation II: sequential dynamical systems" *Applied Math. & Comput.*, vol 107/2-3, pp. 121 – 136, 2000

[7] C. Barrett, H. Mortveit and C. Reidys. "Elements of a theory of computer simulation III: equivalence of SDS", *Applied Math. & Comput.,* vol. 122, pp. 325 – 340, 2001

[8] C. Barrett and C. Reidys. "Elements of a theory of computer simulation I: sequential CA over random graphs" *Applied Math. & Comput.*, vol. 98, pp. 241 – 259, 1999

[9] R. Beckman et. al. "TRANSIMS – Release 1.0 – The Dallas-Forth Worth case study", Tech. Report LA UR 97–4502, Los Alamos National Laboratory, Los Alamos, New Mexico, 1999

[10] P. Floreen, P. Orponen. "On the Computational Complexity of Analyzing Hopfield Nets", *Complex Systems*, vol. 3, pp. 577 – 587, 1989

[11] P. Floreen, P. Orponen. "Attraction radii in binary Hopfield nets are hard to compute", *Neural Computations* vol. 5, pp. 812 – 821, 1993

[12] P. Floreen, P. Orponen. "Complexity Issues in Discrete Hopfield Networks", *NeuroCOLT Technical Report Series,* NC–TR–94–009, October 1994

[13] M. R. Garey and D. S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-completeness",* W. H. Freeman and Co., San Francisco, California, 1979

[14] M. Garzon. *"Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks"*, Springer, 1995

[15] E. Goles, S. Martinez (editors). *"Cellular Automata, Dynamical Systems and Neural Networks"*, Math. and Its Applications series (vol. 282), Kluwer, 1994

[16] E. Goles, S. Martinez (editors). *"Cellular Automata and Complex Systems"*, Nonlinear Phenomena and Complex Systems series, Kluwer, 1999

[17] F. Green. "NP-Complete Problems in Cellular Automata", *Complex Systems*, vol. 1, No. 3, pp. 453 – 474, 1987

[18] C. Greenhill. "The Complexity of Counting Colourings and Independent Sets in Sparse Graphs and Hypergraphs", *Comput. Complexity*, vol. 9, pp. 52 – 72, 2000

[19] H. Gutowitz (Editor). *"Cellular Automata: Theory and Experiment"*, North Holland, 1989

[20] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities", *Proc. Nat'l Academy of Sciences (USA)*, vol. 79, pp. 2554 – 2558, 1982

[21] J. J. Hopfield, D. W. Tank. "Neural computation of decisions in optimization problems", *Biological Cybernetics*, vol. 52, pp. 141 – 152, 1985

[22] B. Huberman, N. Glance. "Evolutionary games and computer simulations", *Proc. Nat'l Academy of Sciences (USA)*, 1999

[23] H. B. Hunt, M. V. Marathe, V. Radhakrishnan, R. E. Stearns. "The Complexity of Planar Counting Problems", *SIAM J. Computing*, vol. 27, pp. 1142 – 1167, 1998

[24] T. E. Ingerson and R. L. Buvel. "Structure in asynchronous cellular automata", *Physica D: Nonlinear Phenomena*, vol. 10 (1-2), pp. 59 – 68, January 1984

[25] M. Jerrum. "Two-dimensional monomer-dimer systems are computationally intractable", *J. Statist. Physics*, vol. 48, pp. 121 – 134, 1987. Erratum in vol. 59, pp. 1087 – 1088, 1990

[26] M. Jerrum, A. Sinclair. "Approximating the permanent", *SIAM J. Comp.*, vol. 18, pp. 1149 – 1178, 1989

[27] M. Jerrum, A. Sinclair. "Polynomial-time approximation algorithms for the Ising model", *SIAM J. Comp.*, vol. 22, pp. 1087 – 1116, 1993

[28] S. A. Kauffman. "Metabolic stability and epigenesis in randomly connected nets", *J. Theoretical Biology*, vol. 22, pp. 437 – 467, 1969

[29] S. A. Kauffman. "Emergent properties in random complex automata", *Physica D: Nonlinear Phenomena*, Volume 10, Issues 1–2, pp. 59 – 68, January 1984

[30] R. Laubenbacher and B. Pareigis. "Finite Dynamical Systems", Technical report, Dept. of Mathematical Sciences, N. Mexico State Univ., Las Cruces, New Mexico, 2000

[31] B. Martin. "A Geometrical Hierarchy of Graphs via Cellular Automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, August 1998

[32] M. Mitchell. "Computation in Cellular Automata: A Selected Review", in T. Gramms, S. Bornholdt, M. Gross, M. Mitchell, T. Pellizzari (editors), *"Nonstandard Computation"*, pp. 95 – 140, Weinheim: VCH Verlagsgesellschaft, 1998

[33] H. Mortveit, C. Reidys. "Discrete sequential dynamical systems", *Discrete Mathematics*, pp. 281 – 295, vol. 226 , Issue 1–3, 2001

[34] C. Nichitiu and E. Remila. "Simulations of Graph Automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, August 1998

[35] C. Papadimitriou. *"Computational Complexity"* , Addison-Wesley, Reading, Massachusetts, 1994

[36] Sz. Roka. "One-way cellular automata on Cayley graphs", *Theoretical Computer Science*, 132 (1–2), pp. 259 – 290, September 1994

[37] D. Roth. "On the Hardness of Approximate Reasoning", *Artificial Intelligence,* vol. 82, pp. 273 – 302, 1996

[38] P. Tosic. "On Counting Fixed Point Configurations in Star Networks", *Advances in Parallel & Distributed Computational Models* Workshop (APDCM'05), within *The 19th IEEE Int'l Parallel & Distributed Processing Symp.*, Denver, Colorado, April 2005, in *Proc. IEEE-IPDPS'05* (CD-Rom)

[39] P. Tosic. "On Complexity of Counting Fixed Point Configurations in Certain Classes of Graph Automata", *Electronic Colloquium on Computational Complexity*, ECCC–TR05–051, April 2005

[40] P. Tosic. "Counting Fixed Points and Gardens of Eden of Sequential Dynamical Systems on Planar Bipartite Graphs", *Electronic Colloquium on Computational Complexity* ECCC–TR05–091, August 2005

[41] P. Tosic, G. Agha. "Characterizing Configuration Spaces of Simple Threshold Cellular Automata", in *Proc. of the 6th Int'l Conference on Cellular Automata for Research and Industry* (ACRI'04), Amsterdam, The Netherlands, October 2004; Springer's LNCS series, vol. 3305, pp. 861 – 870

[42] P. Tosic, G. Agha. "On computational complexity of counting fixed points in certain classes of graph automata", *Proc. of the 4th Int'l Conf. on Unconventional Computation* (UC'05), Sevilla, Spain, October 2005; Springer's *Lecture Notes in Computer Science* (LNCS) series, vol. 3699, pp. 191 – 205

[43] P. Tosic, G. Agha. "Parallel vs. Sequential Threshold Cellular Automata: Comparison and Contrast", in *Proc. of the First European Conference on Complex Systems* (ECCS'05), paper # 251; European Complex Systems Society, Paris, France, November 2005 (CD-Rom)

[44] S. Vadhan. "The Complexity of Counting in Sparse, Regular and Planar Graphs", *SIAM J. Computing,* vol. 31 (2), pp. 398 – 427, 2001

[45] L. Valiant. "The Complexity of Computing the Permanent", *Theoretical Computer Science,* vol. 8, pp. 189 – 201, 1979

[46] L. Valiant. "The Complexity of Enumeration and Reliability Problems", *SIAM J. Computing*, vol. 8 (3), pp. 410 – 421, 1979

[47] I. Wegener. *"The Complexity of Boolean Functions"*, Teubner Series Comp. Sci., Wiley, 1987

[48] S. Wolfram. "Computation theory of cellular automata", *Commun. Math. Physics*, vol. 96, 1984

[49] S. Wolfram. "Twenty problems in the theory of cellular automata", *Physica Scripta*, T9, pp. 170 – 183, 1985

[50] S. Wolfram. *"Theory and applications of cellular automata"*, World Scientific, 1986

[51] S. Wolfram (ed.). *"Cellular Automata and Complexity (collected papers)"*, Addison-Wesley, 1994

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/