On the complexity of enumerating possible dynamics of sparsely connected Boolean network automata with simple update rules

Predrag T. Tošić

Automata 2010 — 16th Intl. Workshop on CA and DCS DMTCS proc. **AL**, 2010, 125–144

On the complexity of enumerating possible dynamics of sparsely connected Boolean network automata with simple update rules

Predrag T. Tošić

Department of Computer Science, University of Houston, Houston, USA. Email: ptosic@uh.edu

We study how hard is to determine some fundamental properties of dynamics of certain types of network automata. We address the computational complexity of determining how many different possible dynamic evolutions can arise from some structurally very simple, deterministic and sparsely connected network automata. In this as well as our prior, related work, we try to push the limits on the underlying simplicity of two structural aspects of such network automata: (i) the uniform sparseness of their topologies, and (ii) severely restricted local behaviors of the individual agents (that is, the local update rules of the network nodes).

In this endeavor, we prove that counting the Fixed Point (FP) configurations and the predecessor and ancestor configurations in two classes of network automata, called Sequential and Synchronous Dynamical Systems (SDSs and SyDSs, respectively), are computationally intractable problems. Moreover, this intractability is shown to hold when each node in such a network is required to update according to (i) a monotone Boolean function, (ii) a symmetric Boolean function, or even (iii) a simple threshold function that is both monotone and symmetric. Furthermore, the hardness of the exact enumeration of FPs and other types of configurations of interest remains to hold even in some severely restricted cases with respect to both the network topology and the diversity (or lack thereof) of individual node's local update rules. Namely, we show that the counting problems of interest remain hard even when the nodes of an SDS or SyDS use at most two different update rules from a given restricted class, and, additionally, when the network topologies are constrained so that each node has only $c = O(1)$ neighbors for small values of constant c.

Our results also have considerable implications for other discrete dynamical system models studied in applied mathematics, physics, biology and computer science, such as Hopfield networks and spin glasses. In particular, one corollary of our results is that determining the memory capacity of sparse discrete Hopfield networks (viewed as associative memories) remains computationally intractable even when the interconnection and dependence structure among the nodes of a Hopfield network is severely restricted.

Keywords: network and graph automata, cellular automata, Hopfield networks, discrete dynamical systems, computational complexity, enumeration problems, **#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, biological and socio-technical distributed infrastructures. We are interested in the computational complexity of determining several fundamental *configuration space properties* of such network automata. The complexity of answering questions about, for instance, the existence [9], the number [67, 69] or the reachability [8] of *fixed points* (that is, the stable configurations) of an appropriate class of network automata can be argued to provide important insights into the *collective dynamics* of multi-agent systems found in distributed artificial intelligence [69], as well as other complex physical, biological, and socio-technical networks that are abstracted via those formal network automata.

In this paper, as well as in related prior work (see, e.g., [6, 7, 10, 11, 9, 66, 61, 62, 69]), the general approach has been to investigate mathematical and computational configuration space properties of such network 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?

Our own focus in the context of dynamic behaviors of complex network and graph automata has been on determining *how many possible dynamics*, and in particular how many of certain types of configurations, can such discrete dynamical systems have – and *how hard* are the computational problems of determining the exact or approximate number of those various types of configurations [60, 61, 62, 63, 64, 69, 67]. We have been particularly interested in addressing the problem of counting how many *fixed point* (FP) configurations such network automata have, and how hard is the computational problem of counting those FP configurations. In this paper, we show 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 rather severely restricted. 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 follows from those results that, for the networked dynamical systems that can be abstracted via a class of formal network automata, a complex and generally unpredictable global dynamics can be obtained even via uniformly sparse couplings of simple, monotonic local interactions. The implication for Hopfield networks is that determining their memory capacity (when viewed as a model of associative memory) is computationally intractable, even when the structure of the underlying weight matrices of discrete, binary-valued Hopfield network is of a very particular and restricted kind.

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 [10, 11, 12] as an abstract model for computer simulations. These models have been successfully applied in the development of large-scale socio-technical simulations such as the *TRANSIMS* project at the Los Alamos National Laboratory [13]. A more detailed discussion of the motivation behind these models, as well as their application to large-scale simulations, can be found in [9, 61, 62] 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. *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 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 neighbors of node v_i in G, plus 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 the vertex set $V = \{v_1, v_2, \ldots, v_n\}$, 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 node state s_i, where the node states are updated according to the order Π.*

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 the new value. In the sequel, we shall often slightly abuse the notation, and not explicitly distinguish between an SDS's node itself, v_i, and its state, s_i. The intended meaning will be clear from the context.

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 update their state values.*

Thus, SyDSs are similar to the finite classical parallel *cellular automata* (CA) [22, 23, 25, 28, 76, 77], 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 (such as, e.g., a line, a rectangular grid, or a hypercube). Another difference is that, while in the classical CA all nodes update according to the same rule, in an SyDS different nodes, in general, may use different update rules [9, 61].

Given the importance of the number of stable configurations of a Hopfield network viewed as an *associative memory* [29, 24], we next define discrete Hopfield networks. We will briefly summarize what has been known about the problem of counting their stable configurations in the subsequent sections.

Definition 2.3 *A* discrete Hopfield network *(DHN) [29] 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 a* linear threshold function *of the form*

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

where, in order to ensure that $x_i \in \{-1, +1\}$, we define $sgn(0) = +1$.

In the sequel, we will often not bother to explicitly distinguish between an S(y)DS's or DHN's node, v_i, and this node's state, denoted s_i for S(y)DSs and x_i for Hopfield networks; the 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 above. However, *asynchronous Hopfield networks*, where the nodes update sequentially, one at a time, have also been studied [20, 29]. In these sequential DHNs, however, it is not required that the nodes update according to a *fixed permutation* like in our SDS model. We emphasize that these differences are inconsequential insofar as the fixed points are concerned.

In most 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 $w_{ii} = 0$ along the main diagonal of W for all $i = 1, ..., n$ [20].

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 (e.g., [6, 7, 10, 11, 67]). 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 the states of nodes v_j in $N(i)$, 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 as defined by S. Wolfram [74, 75, 76]. The computational complexity of counting various configurations in SDSs and SyDSs with symmetric Boolean update rules is addressed in [61, 67].

We consider in this paper the SDSs, SyDSs and Hopfield networks with the local update rules that are restricted to *monotone* Boolean / binary-valued functions. Our preliminary hardness results about the counting problems in monotone Boolean SDSs and SyDSs can be found in [60, 62]. The SDSs with the local transition rules that are both monotone and symmetric are, in essence, *sequential threshold cellular automata* [65, 66, 68] that are defined over arbitrary finite graphs, as opposed to the usual regular *Cayley graphs* of the classical cellular automata [22]. We will consider the monotone update rules that are not necessarily symmetric; however, these monotone Boolean functions will be required to be of a linear threshold kind, so that our subsequent results would imply analogous results for discrete Hopfield networks [29, 30], whose update rules are, by default, always required to be linear (but not necessarily monotone) threshold functions.

We next define the notion of *monotone* Boolean functions. This definition of monotonicity readily extends to other partially ordered domains such as $\{-1, +1\}$ that has been commonly used in Hopfield networks literature.

Definition 2.4 *Given two Boolean vectors, $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$, 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 2.4 allows us to compare only Boolean vectors of the same length.

Definition 2.5 *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 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 $F_{\mathcal{S}} : \{0, 1\}^n \to \{0, 1\}^n$ 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.

Definition 2.6 *The* configuration space *(also called* phase space*)* $\mathcal{P}_{\mathcal{S}}$ *of an SDS or SyDS* $\mathcal{S}$ *is a directed graph whose nodes are configurations and whose directed edges capture* transitions *from a configuration to its successor configuration. More formally, 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* $F_{\mathcal{S}}(\mathcal{C}') = \mathcal{C}$.

Note that, since an SDS or SyDS is deterministic, each vertex in its phase space has the out-degree of 1. Since the domain $\mathcal{D}$ of state values is assumed finite, and the number of nodes in the S(y)DS is finite, the number of configurations in the phase space is also finite. If the size of the domain (that is, the number of possible states of each node) is $|\mathcal{D}|$, then the number of global configurations in $\mathcal{P}_{\mathcal{S}}$ is $|\mathcal{D}|^n$.

We next define some prominent types of configurations that are of particular interest insofar as capturing the important qualitative (and quantitative) properties of a discrete dynamical system's global behavior (that is, its dynamics).

Definition 2.7 *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 given configuration is a special case of an ancestor.

Definition 2.8 *A configuration* $\mathcal{C}$ *of an S(y)DS* $\mathcal{S}$ *is a* GARDEN OF EDEN (GE) *configuration if* $\mathcal{C}$ *does not have a predecessor.*

Definition 2.9 *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. Also note, that the fixed point configurations are also often referred to as *stable configurations* (esp. in the Hopfield networks literature); we will use the two terms interchangeably throughout the paper.

2.1 The Basics of Computational Complexity of Counting

We next define the computational complexity classes pertaining to the counting problems that we shall work with in the sequel. We also define the notion of *(weakly) parsimonious reductions* that are used to reduce one counting problem to another.

Definition 2.10 *A counting problem* Ψ *belongs to the class* **#P** *if there exists a polynomial time bounded* nondeterministic Turing machine (NTM) *such that, for each instance* I *of* Ψ*, the number of nondeterministic computational paths this NTM takes that lead to acceptance of this problem instance equals the number of solutions of* $\mathrm{I}(\Psi)$.

For an alternative but equivalent definition of the class **#P** in terms of *polynomially balanced relations*, we refer the reader to [51].

The hardest problems in the class **#P** are the **#P**-complete problems.

Definition 2.11 *A counting problem* Ψ *is* **#P***-complete if and only if (i) it belongs to the class* **#P***, and (ii) it is* hard *for that class, i.e.,* every *other problem in* **#P** *is efficiently reducible to* Ψ.

Thus, if we could solve any particular **#P**-complete problem in (deterministic) polynomial time, then all the problems in class **#P** would be solvable in (deterministic) polynomial time, and the entire class **#P** would collapse to **P**.[(i)]

As one would expect, the counting versions of the standard decision **NP**-complete problems, such as SATISFIABILITY or HAMILTON CIRCUIT, are **#P**-complete [51]. What is curious, however, is that the counting versions of some tractable decision problems, such as BIPARTITE MATCHING or MONOTONE 2CNF SATISFIABILITY, are also **#P**-complete [71, 72].

Definition 2.12 *Given two problems* Π *and* $\Pi^{'}$, *a* PARSIMONIOUS REDUCTION *from* Π *to* $\Pi^{'}$ *is a polynomial-time transformation* g *that preserves the number of solutions; that is, if an instance* I *of* Π *has* n_I *solutions, then the corresponding instance* g(I) *of* $\Pi^{'}$ *also has* $n_{g(I)} = n_I$ *solutions.*

In practice, one often resorts to reductions that are "almost parsimonious", in a sense that, while they do not exactly preserve the number of solutions, n_I in the previous definition can be efficiently recovered from $n_{g(I)}$.

Definition 2.13 *Given two problems* Π *and* $\Pi^{'}$, *a* WEAKLY PARSIMONIOUS REDUCTION *from* Π *to* $\Pi^{'}$ *is a polynomial-time transformation* g *such that, if an instance* I *of* Π *has* n_I *solutions, and the corresponding instance* g(I) *of* $\Pi^{'}$ *has* $n_{g(I)}$ *solutions, then* n_I *can be computed from* $n_{g(I)}$ *in polynomial time.*

We observe that any *parsimonious* reduction is also, trivially, *weakly parsimonious*.

All of our results on the computational complexity of counting various kinds of configurations in SDSs, SyDSs and discrete Hopfield networks will be obtained by reducing counting problems about certain types of Boolean formulae that are *known to be* **#P**-complete to the problems about S(y)DSs or Hopfield nets of a desired, appropriately restricted kind. That such reductions suffice follows from the well-known property of every problem that is *hard* for a given complexity class; for the record, we state that property in the context of the class **#P** in the Lemma below.

Lemma 2.1 [51] *Given two decision problems* Π *and* $\Pi^{'}$, *if the corresponding counting problem* $\#\Pi$ *is known to be* **#P***-hard, and if there exists a (weakly) parsimonious reduction from* Π *to* $\Pi^{'}$, *then the counting problem* $\#\Pi^{'}$ *is* **#P***-hard, as well.*

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., [22, 47, 68]), to complex dynamical systems [24, 25, 38], to theoretical biology [43, 44]. Beside the classical (parallel) cellular automata [22, 28] and

[(i)] Strictly speaking, since **#P** is a class of *function problems* (as opposed to the classes of *decision problems* such as **P**, **NP** and **PSPACE**), if an **#P**-complete problem turns out to be solvable in deterministic polynomial time, that would imply that $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}$.

their sequential or asynchronous variants [34, 66, 68], perhaps the most studied class of models of network or graph automata are the *Hopfield networks* [29, 30].

Computational aspects of the classical Cellular Automata have been studied in many contexts. Prior to the 1980s, most of the theoretical work dealt with infinite CA and the fundamental (un)decidability results about the global CA properties. Some examples of such properties of infinite CA are surjectivity, injectivity, and invertibility of a cellular automaton's *global map*; see, e.g., [2, 49, 52]. Systematic study of other computational aspects of CA, from topological to formal language theoretic to computational complexity theoretic, was prompted in the 1980s by the seminal work of S. Wolfram [74, 75, 76]. Among other issues, Wolfram addressed the fundamental characteristics of CA in terms of their computational expressiveness and universality. He also offered the first broadly accepted classification of all CA into four qualitatively distinct classes in terms of the structural complexity of the possible computations or, equivalently, dynamical evolutions. The state of the art pertaining to a broad variety of computational properties of CA in both theoretical and experimental domains by the end of the "golden decade" of cellular automata research (the 1980's) can be found in [28].

Since most interesting global properties of sufficiently general infinite CA have been shown to be formally undecidable, the computational complexity proper (that is, as contrasted with the computability theory) has been mainly concerned with the computational aspects of *finite* CA, or those pertaining to *finite subconfigurations* of infinite CA. Most work within that framework has focused on the fundamental *decision problems* about the possible CA computations. We include below a very short survey of some of the more important results in that context.

The first **NP**-complete problems for CA are shown by Green in [26]; these problems are of a general REACHABILITY flavor, i.e., they address the properties of the FORWARD DYNAMICS of CA. Kari studies the reverse dynamics, more specifically, the reversibility and surjectivity problems about CA [39]. Sutner also addresses the BACKWARD DYNAMICS problems, such as the problem of an arbitrary configuration's PREDECESSOR EXISTENCE, and their computational complexity in [57]. In the same paper, Sutner establishes the efficient solvability of the predecessor existence problem for an arbitrary CA with a *fixed neighborhood radius.* In [14], Durand solves the injectivity problem for arbitrary 2-D CA but restricted to the *finite subconfigurations* only; that paper contains one of the first results on **coNP**-completeness of a natural and important problem about CA. Furthermore, Durand addresses the REVERSIBILITY PROBLEM in the same, two-dimensional CA setting in [15]. Some good surveys on various directions of computational complexity-theoretic research on cellular automata can be found in [40, 47].

The SDS and SyDS models introduced in Section 2 are closely related to the *graph automata* (GA) models studied in [46, 50] and the *one-way cellular automata* studied in [54]. In fact, the general finite-domain SyDSs exactly correspond to the graph automata of Nichitiu and Remila in [50]. Barrett, Mortveit and Reidys [10, 11, 48] and Laubenbacher and Pareigis [45] 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. Those problems include the REACHABILITY, PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE problems [7, 8]. Problems related to the existence of the GARDEN OF EDEN and the FIXED POINT configurations are studied in [9]. In particular, **NP**-completeness for the problem of FIXED POINT EXISTENCE (FPE) in various restricted classes of Boolean S(y)DSs is proven in [9]. 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 [71, 72]. Counting

problems naturally arise in many contexts, from approximate reasoning and Bayesian belief networks in AI (e.g., [55]), to network reliability and fault-tolerance [70], to theoretical biology [3], to *phase transitions* in statistical physics, which is a large body of work in itself (some representative references include [4, 35, 36, 38]).

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 [27, 70]. 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 those problems' solutions [27].

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 [70]. Some of the relatively recent results in that context, such as those on the hardness of counting in *planar graphs* [32], and especially in *sparse graphs* [27, 70], have directly inspired our recent work (see [60, 61, 62, 67]), as well as the investigations summarized in this paper.

Counting problems naturally arise in the context of discrete dynamical systems, as well. Indeed, being able to efficiently solve certain counting problems is essential for the full understanding of the underlying dynamical system's qualitative behavior. The most obvious counting problem is to determine (or estimate) the number of *attractors* of the dynamical system [3]. As noted earlier, we refer to these attractors and other, not necessarily attracting *stable configurations* as to the *fixed points* (FPs); we do not address the issue of distinguishing among different types of those stable configurations (e.g., attractive vs. repulsive, etc.) in this paper.

For example, in the context of Hopfield networks, the interpretation of the FP count is that it tells us how many distinct *patterns* a given Hopfield network can *memorize* [3, 29, 30]. Computational complexity of counting FPs and other structures of interest in discrete Hopfield networks is addressed in [18, 19, 20]. We shall discuss in the next section how our results in this paper strengthen those in [18, 19] for the *symmetric* discrete Hopfield nets with integer weight matrices and *linear threshold* update rules.

4 Complexity of Counting Various Configurations of Monotone SDSs, SyDSs and Discrete Hopfield Networks

Our general approach to establishing the computational intractability of the counting problems of interest about SDSs, SyDSs and Discrete Hopfield Nets (DHNs) will be as follows. We first identify certain restricted variants of Boolean Satisfiability problem, whose counting versions (that is, determining how many satisfying truth assignment an arbitrary Boolean formula in a particular, restricted form has) are known to be intractable, that is, **#P**-complete. We then construct an S(y)DS or DHN and show that exactly enumerating a particular kind of such network automaton's configurations (e.g., its fixed points) is at least as hard as exactly enumerating the satisfying assignments of the Boolean formula. For showing intractability of counting problems, we use weakly parsimonious reductions that approximately preserve the number of solutions; for details on complexity of counting in general, and (weakly) parsimonious reductions from one counting problem to another in particular, we refer the reader to any standard reference on computational complexity, such as the book by C. Papadimitriou [51].

Monotone Boolean functions, formulae and circuits [73] 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., [9, 66]). The problem of counting FPs of *monotone* Boolean SDSs and SyDSs is originally addressed in [60, 62]. 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* [60, 62, 64]. In particular:

Lemma 4.1 [62] *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 of the star is given as a* MON 2CNF *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 [55], subsequently strengthened by S. Vadhan [70], the problem of *approximately* counting FPs in the setting as in Lemma 4.1 above is **NP**-hard [62].

To summarize, 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 $2^{n^{1-\epsilon}}$ 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 4.2 below, the problem of enumerating stable configurations of other types of discrete dynamical systems, such as the discrete Hopfield networks, is also in general computationally intractable. Moreover, that intractability holds for those discrete dynamical systems even when they are defined on very sparse underlying graphs or networks.

Given the importance of the number of stable configurations of a Hopfield network viewed as an *associative memory* (e.g., [24]), we next summarize what has been known about the problem of counting their stable configurations.

In [18], Floreen and Orponen establish the following two interesting results:[(ii)]

Theorem 4.1 *(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 $w_{ii} = 0$ 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 $w_{ii} = 0$ along the main diagonal, is **#P***-complete.*

For proving (i), Floreen and Orponen devise a Hopfield network that is quite dense, i.e., with many 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 [60, 62] for more details.

Prior to moving to our main results, for the sake of completeness, we state the following

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

[(ii)] We slightly rephrase the statement of these results from the linear algebra language originally used in [18] into the discrete language we are using throughout this paper, in order to make the comparison and contrast with our own results clear.

4.1 Counting Configurations of Uniformly Sparse SDSs and SyDSs with Monotone Update Rules

The first major result of this paper pertains to the computational complexity of counting the fixed point configurations of monotone Boolean SDSs and SyDSs that are defined over *uniformly sparse* underlying graphs.

One of the original motivations behind our interest in that problem was to improve upon the complexity of counting results in [18, 19]. We shall show below that the result (i) from [18] discussed above 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 the 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 update rules.

We remark that, since each node of an SDS or SyDS in the Theorem that follows is required to have only $O(1)$ neighbors, the issue of *encoding* of the local update rules, that is discussed in detail in [62], is essentially irrelevant in the present context. In particular, even a truth table with one row for each combination of the values of a given node's neighbors is permissible [61, 62].

In the sequel, BOOL-MON-S(Y)DS will stand for a *monotone Boolean* SDS or SyDS.

Theorem 4.2 *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*;*
- *the S(y)DS is with memory, and such that, along the main diagonal,* $w_{ii} = 1$ *for all indices* i*,* $1 \leq i \leq n$*;*
- *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 rules 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 [21] 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* [21].

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 together 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

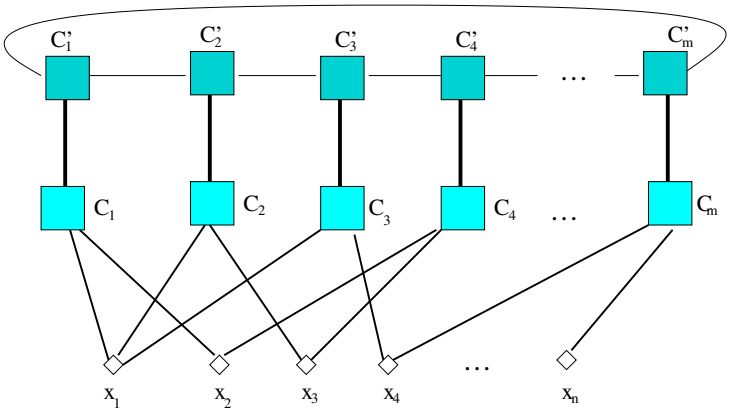

Fig. 1: Figure 4.1: *The underlying graph of a bounded-degree monotone linear threshold Boolean S(y)DS in the construction of Theorem 4.2. The original clause nodes are marked C_j, the cloned clause nodes are primed, as in C'_j, and the variable nodes are denoted by x_i.*

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 4.1.

In the sequel, we shall slightly abuse the notation and use x_i *both* to denote a variable in the Boolean formula, and the corresponding *variable node* in the S(y)DS or discrete Hopfield network we are constructing. Similarly, in this proof as well as throughout the rest of the paper, C_j will denote both clauses in the Boolean formulae and clause nodes in the S(y)DSs or Hopfield networks that are being constructed from those formulae. Again, the intended meaning will be clear from the context.

With these conventions in mind, we now define the update rules for the clause nodes, cloned clause nodes, and variable nodes of the constructed SyDS. 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*

on four inputs. 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 will turn to the related complexity of counting results in the context of Hopfield networks in the next subsection.

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 summarize the case analysis. If, at any time step t, one of the cloned clause nodes C'_j evaluates to 0, that will ensure that, within no more than $\frac{m}{2}$ 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 an n-tuple of Boolean values that corresponds to a satisfying truth assignment $(x_1, ..., x_n)$ to the underlying 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 [27], 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, when the update rules are written in the form as in expressions (2) and (3), it is immediate that 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 monotone linear threshold SyDSs are concerned.

Finally, by the invariance of FPs with respect to the node update ordering [48], it follows that exactly enumerating FPs of 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 $(C^m, C'^m) \neq (1^m, 1^m)$, 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 that 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***-hard. Moreover, this intractability result holds even when all restrictions from* Theorem 4.2 *are simultaneously imposed on the S(y)DS's structure.*

4.2 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 4.3 *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 4.2, 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.[(iii)] Insofar as the asynchronous DHNs whose nodes update in arbitrary sequential orders are concerned, while indeed those 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 4.3 about discrete Hopfield networks is nothing but rephrasing Theorem 4.2, 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

[(iii)] 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 $\{0, 1\}$.

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* 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 4.2, we will eliminate the cloned clause nodes C'_j and, instead, connect the ordinary clause 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 not essential, we will adopt this practice through the rest of the paper when it comes to the discrete Hopfield networks. With that in mind, 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 the stated assumptions, for every $i \in \{1, ..., |V|\}$, 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*) [18, 20].

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

Theorem 4.4 *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. In the Hopfield network context, we will identify the Boolean value FALSE of a variable in the MON-2CNF formula with the corresponding DHN variable node's state -1, whereas the Boolean value TRUE of a variable in the formula will be mapped to the state $+1$ of the corresponding DHN variable node.

The case analysis is similar to that in the proof of Theorem 4.2. In particular, every 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 will evolve to the other fixed point configuration, $(1^n, 1^m) = (+1)^{n+m}$. 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 immediately 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 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***-hard. Moreover, this intractability holds even when all the restrictions from* Theorem 4.4 *on the Hopfield network instances are simultaneously imposed.*

We remark that the #ANC problem for S(y)DSs and DHNs is indeed **#P**-complete whenever the *basin of attraction* of a fixed point – or, for that matter, of an arbitrary configuration that has ancestors – of the dynamical system in question is *shallow*. This shallowness, in particular, ensures that the problem of enumerating ancestors of all generations is in the class **#P**. However, for arbitrary basins of attraction that need not necessarily be shallow, the question arises, whether it can always be verified in polynomial time if one configuration is an ancestor of another configuration. In fact, the results in [8] imply that, given two arbitrary configurations $\mathcal{C}$ and $\mathcal{C}'$ of a monotone Boolean SDS or SyDS, determining whether $\mathcal{C}'$ is an ancestor of $\mathcal{C}$ (alternatively, whether $\mathcal{C}$ is *reachable* from $\mathcal{C}'$) is, in general, **PSPACE**-complete.

The implication of the results in [8] for the complexity of counting ancestors of an arbitrary configuration of a monotone Boolean S(y)DS or a discrete Hopfield network with a nonnegative and symmetric weight matrix is that the problem #ANC need not be in the class **#P**. In particular, it is an open problem whether #ANC $\in$ **#P** under the *sparseness* restrictions as in our three main results earlier in this paper. Therefore, all we can offer at this stage is a more conservative characterization of the complexity of #ANC in comparison to the complexity of #FP and #PRED – hence the **#P**-hardness, rather than **#P**-completeness, statements about the problem #ANC in Corollaries 4.1 and 4.2.

5 Summary

We have shown in [61, 62, 60, 64] that the problem of enumerating the fixed point configurations of two related classes of Boolean network automata, called Sequential and Synchronous Dynamical Systems is, in general, computationally intractable. We continue the general line of inquiry from our prior work in the present paper, as well. 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 only $O(1)$ 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 Hopfield networks as a model of 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. This computational intractability remains to hold 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, our hardness result still 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 result 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 (or, in the SDS terminology, fixed point) 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 work [60, 61, 62, 63, 67, 64, 69] all pertain to the worst-case complexity of counting the stable configurations and other structures of discrete dynamical systems. Of a considerable interest to statistical physics, connectionist AI and large-scale multi-agent systems research communities, however, is the problem of determining average complexity of the relevant decision, search and counting problems about the underlying system's dynamics.

Another important problem is that of the hardness of approximate counting of the fixed points and other types of configurations of interest; we have partially solved that problem for certain classes of underlying network topologies and node update rules [60, 62, 67, 64], but not for the particular restricted classes of topologies and update rules for which we have established the hardness of exact counting in this paper and the extended technical report [63]. Hence, the approximate counting in the settings discussed in the present paper, as far as we know, is still open.

To summarize our contribution in the present paper, the results in Theorems 4.2 - 4.4, and in particular the constructions in their corresponding proofs (see also [63] for details), clearly indicate that there are various restricted classes of uniformly sparse Boolean network automata and Hopfield networks for which exactly enumerating the stable configurations (FPs), as well as the predecessor and the arbitrary ancestor configurations, are all computationally intractable in the worst case. These hardness results have some interesting implications and interpretations – for example, in the context of pattern storage capacity of sparsely connected Hopfield networks viewed as associative memories. However, what is the average complexity of these important counting problems (and in particular, under what specific assumptions are those average or expected case problems tractable), are wide-open problems. We hope to address the average case complexity of those and other similar counting problems about discrete dynamical networks in our future work.

Acknowledgements

Many thanks to my colleague Ricardo Vilalta, as well as Department of Computer Science and Texas Learning & Computation Center (TLC2) at University of Houston.

References

[1] M. Anthony. "Threshold Functions, Decision Lists, and the Representation of Boolean Functions", NeuroCOLT Techn. Report Series (NC-TR-96-028), January 1996

[2] S. Amoroso, Y. Patt. "Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures", *Journal of Computer and System Sciences* (JCSS), vol. 6, pp. 448 – 464, 1972

[3] R. J. Bagley, L. Glass. "Counting and Classifying Attractors in High Dimensional Dynamical Systems", *Journal of Theoretical Biology*, vol. 183, pp. 269–284, 1996

[4] F. Barahona. "On the computational complexity of Ising spin glass models", *Journal of Physics A: Mathematical and General*, vol. 15, pp. 3241 – 3253, 1982

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

[6] 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, 2000

[7] 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, 2001

[8] 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 Computer Science*, vol. 295, issues 1–3, pp. 41–64, February 2003

[9] 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", *Discrete Mathematics and Theoretical Computer Science* (DMTCS), spec. ed. Proc. AA DM–CCG, pp. 95–110, 2001

[10] C. Barrett, H. Mortveit, and C. Reidys. "Elements of a theory of simulation II: sequential dynamical systems" *Applied Mathematics and Computation*, vol. 107 (2–3), pp. 121–136, 2000

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

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

[13] 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

[14] B. Durand. "Inversion of 2D cellular automata: some complexity results", *Theoretical Computer Science*, vol. 134 (2) , pp. 387 – 401, November 1994

[15] B. Durand. "A random **NP**-complete problem for inversion of 2D cellular automata", *Theoretical Computer Science*, vol. 148 (1) , pp. 19 – 32, August 1995

[16] B. Durand. "Global properties of 2D cellular automata", in E. Goles, S. Martinez (eds.), *"Cellular Automata and Complex Systems"*, Kluwer, Dordrecht, 1998

[17] C. Dyer. "One-way bounded cellular automata", *Information and Control*, vol. 44, pp. 54 – 69, 1980

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

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

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

[21] 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

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

[23] E. Goles, S. Martinez. *"Neural and Automata Networks: Dynamical Behavior and Applications"*, Mathematics and Its Applications series, vol. 58, Kluwer, 1990

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

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

[26] F. Green. "NP-Complete Problems in Cellular Automata", *Complex Systems*, vol. 1 (3), pp. 453–474, 1987

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

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

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

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

[31] B. Huberman, N. Glance. "Evolutionary games and computer simulations", *Proc. National Academy of Sciences* (USA), vol. 90, pp. 7716–7718, August 1993

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

[33] L.P. Hurd. "On Invertible cellular automata" *Complex Systems,* vol. 1(1), pp. 69-80, 1987

[34] 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

[35] S. Istrail. "Statistical Mechanics, Three-Dimensionality and NP-completeness: I. Universality of Intractability for the Partition Function of the Ising Model Across Non-Planar Lattices (Extended Abstract)", *Proceedings of the 32nd ACM Symposium on Theory of Computing* (STOC '00), Portland, Oregon, pp. 87 – 96, 2000

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

[37] M. Jerrum, A. Sinclair. "Approximating the permanent", *SIAM Journal of Computing*, vol. 18, pp. 1149–1178, 1989

[38] M. Jerrum, A. Sinclair. "Polynomial-time approximation algorithms for the Ising model", *SIAM Journal of Computing*, vol. 22, pp. 1087–1116, 1993

[39] J. Kari. "Reversibility and surjectivity problems of cellular automata", *Journal of Computer and System Sciences*, vol. 48, pp. 149 – 182, 1994

[40] J. Kari. "Theory of cellular automata: A survey", *Theoretical Computer Science*, vol. 334, pp. 3 – 33, 2005

[41] R. Karp, M. Luby. "Monte-Carlo algorithms for enumeration and reliability problems", *IEEE Symposium on Foundations of Computer Science*, No. 24, pp. 56 – 64, 1983

[42] R. Karp, M. Luby. "Monte Carlo algorithms for the planar multiterminal network reliability problem", *Journal of Complexity*, vol. 1, pp. 45 – 64, 1985

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

[44] S. A. Kauffman. "Emergent properties in random complex automata", *Physica D: Nonlinear Phenomena*, Volume 10, Issues 1–2, pp. 145–156, January 1984

[45] R. Laubenbacher and B. Pareigis. "Finite Dynamical Systems", Technical report, Department of Mathematical Sciences, New Mexico State University, Las Cruces, New Mexico, 2000

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

[47] 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

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

[49] J. Myhill. "The converse of Moore's Garden-of-Eden theorem", *Proc. Amer. Math. Soc.* vol. 14, pp. 685-686, 1963

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

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

[52] D. Richardson. "Tessellations with local transformations", *J. of Computer and System Sciences (JCSS)*, 6, pp. 373-388, 1972

[53] C. Robinson. *"Dynamical systems: stability, symbolic dynamics and chaos"*, CRC Press, New York, 1999

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

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

[56] K. Sutner. "De Bruijn graphs and linear cellular automata", *Complex Systems*, vol. 5 (1), pp. 19–30, 1990

[57] K. Sutner. "On the computational complexity of finite cellular automata", *Journal of Computer and System Sciences* (JCSS), vol. 50 (1), pp. 87–97, February 1995

[58] K. Sutner. "Computation theory of cellular automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, August 1998

[59] C. Schittenkopf, G. Deco and W. Brauer. "Finite automata-models for the investigation of dynamical systems" *Information Processing Letters*, vol. 63 (3), pp. 137–141, August 1997

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

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

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

[63] P. Tosic. "Computational Complexity of Some Enumeration Problems About Uniformly Sparse Boolean Network Automata", *Electronic Colloquium on Computational Complexity*, Report ECCC–TR06–159, 2006

[64] P. Tosic. "On the Complexity of Counting Fixed Points and Gardens of Eden in Sequential and Synchronous Dynamical Systems", *International Journal on Foundations of Computer Science* (IJFCS), vol. 17 (5), pp. 1179–1203, October 2006

[65] P. Tosic, G. Agha. "Concurrency vs. Sequential Interleavings in 1-D Threshold Cellular Automata", APDCM Workshop, in *Proc. of the 18th IEEE Int'l Parallel & Distributed Processing Symposium*, Santa Fe, New Mexico, USA, April 2004

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

[67] P. Tosic, G. Agha. "On Computational Complexity of Counting Fixed Points in Symmetric Boolean Graph Automata", in *Proc. of the 4th Int'l Conference on Unconventional Computation* (UC'05), Sevilla, Spain, October 2005; Springer's *Lecture Notes in Computer Science* (LNCS) series, vol. 3699, pp. 191–205

[68] 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

[69] P. Tosic, G. Agha. "On Computational Complexity of Predicting Dynamical Evolution of Large Agent Ensembles", *Proc. of the 3rd European Workshop on Multiagent Systems* (EUMAS'05), pp. 415–426, Flemish Academy of Sciences, Brussels, Belgium, December 2005

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

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

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

[73] I. Wegener. *"The Complexity of Boolean Functions"*, Teubner Series in Computer Science, Wiley, 1987

[74] S. Wolfram. "Computation theory of cellular automata", *Communications in Mathematical Physics*, vol. 96, 1984

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

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

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