Evolution of Similar Configurations in Graph Dynamical Systems

Joshua D. Priest[1], Madhav V. Marathe[1], S. S. Ravi[1,2(✉)], Daniel J. Rosenkrantz[1,2], and Richard E. Stearns[1,2]

[1] University of Virginia, Charlottesville, USA
{jdp8jb,marathe}@virginia.edu
[2] University at Albany – SUNY, Albany, USA
ssravi0@gmail.com, drosenkrantz@gmail.com, thestearns2@gmail.com

Abstract. We investigate questions related to the time evolution of discrete graph dynamical systems where each node has a state from {0,1}. The configuration of a system at any time instant is a Boolean vector that specifies the state of each node at that instant. We say that two configurations are similar if the Hamming distance between them is small. Also, a predecessor of a configuration B is a configuration A such that B can be reached in one step from A. We study problems related to the similarity of predecessor configurations from which two similar configurations can be reached in one time step. We address these problems both analytically and experimentally. Our analytical results point out that the level of similarity between predecessors of two similar configurations depends on the local functions of the dynamical system. Our experimental results, which consider random graphs as well as small world networks, rely on the fact that the problem of finding predecessors can be reduced to the Boolean Satisfiability problem (SAT).

1 Introduction

Discrete graph dynamical systems are generalizations of cellular automata (CA) [10,26]. They serve as a useful formal model in many contexts, including multi-agent systems, propagation of contagions in social networks and interaction phenomena in biological systems (see e.g., [1,17,25,27]). Here, we focus on one such class of graph dynamical systems, namely *synchronous* discrete dynamical systems (SyDSs). Informally, a SyDS[1] consists of an undirected graph[2] whose vertices represent entities and edges represent local interactions among entities. Each node v has a Boolean state and a local function f_v whose inputs are the current state of v and those of its neighbors; the output of f_v is the next state of v. The vector consisting of the state values of all the nodes at each time instant is referred to as the **configuration** of the system at that instant. In each time step, all nodes of a SyDS compute and update their states *synchronously*. Starting from a (given) initial configuration, the time evolution of a SyDS consists of a sequence of successive configurations, which is also called a **trajectory**.

[1] Formal definitions associated with SyDSs are presented in Sect. 2.

[2] Synchronous dynamical systems, where the underlying graph is directed, are called **Synchronous Boolean Networks** (see e.g., [12,13,19]).

R. M. Benito et al. (Eds.): COMPLEX NETWORKS 2020, SCI 943, pp. 544–555, 2021.
https://doi.org/10.1007/978-3-030-65347-7_45

In this paper, we examine questions related to the evolution of configurations that are *similar*. We measure the similarity between two configurations by their Hamming distance (i.e., the number of bit positions where the two configurations differ). Thus, two configurations are similar if the Hamming distance between them is small. It is known that certain dynamical systems may exhibit unpredictable behavior when the initial conditions are perturbed slightly [26]. A primary goal of our study is to obtain an understanding of when and how two similar configurations may arise from configurations that may be dissimilar. Such a study can be useful in understanding the sensitivity of a given dynamical system. As a concrete and simplified version of the general research question, we consider the following problem: given two similar configurations, how similar are their *predecessors* (i.e., configurations that just preceded the given configurations in the time evolution of the system)? A summary of our results is given below.

(a) Analytical Results. In Sect. 3, we show that SyDSs may exhibit extreme behaviors with respect to the evolution of configurations. For example, one of our results (Proposition 1) shows that there are SyDSs in which for any two configurations $\mathbb{C}_1$ and $\mathbb{C}_2$ which differ in h bits, there are respective predecessors $\mathbb{C}_1^{'}$ and $\mathbb{C}_2^{'}$ which also differ in exactly h bits. Further, we show (Proposition 2) that there are SyDSs where two very similar configurations (which differ in just one bit) have highly dissimilar predecessors (i.e., they differ in all the bits). In addition, we present examples of SyDSs (Corollary 1) in which highly dissimilar configurations have predecessors that differ in just one bit. We also show that computing similarity measures of the predecessors of two given configurations is, in general, computationally intractable. Further, we point out that the problem of computing a predecessor of a given configuration can be reduced to the Boolean Satisfiability problem (SAT).

(b) Experimental Results. Our experimental results (presented in Sect. 4) rely on the result that the problem of computing a predecessor of a given configuration can be reduced to SAT. While many public domain SAT solvers are available [22], we used Clasp [7] for our experiments. The reasons for this choice are explained in Sect. 4. Our experiments consider several classes of graphs (grids, Watts-Strogatz small world networks and Erdős-Rényi graphs). Since our analytical results indicate that non-monotone Boolean functions (e.g., exclusive OR) can cause extreme behaviors with respect to Hamming distance, we used threshold[3] functions (which are monotone) in our experiments. For small networks, our results show the exact maximum, minimum and average Hamming distance values for several threshold values. For larger networks, since it is computationally expensive to find all the predecessors and compute the exact Hamming distance values, we generated up to 10^4 predecessors and computed the Hamming distance values using those predecessors. In general, the results discussed in Sect. 4 indicate that for small threshold values, as the Hamming distance between a pair of configurations is increased, the average Hamming distance between their predecessor sets increases linearly; for larger threshold values, the average Hamming distance between predecessor sets remains more or less stable. We also present results showing the number of clauses generated by the transformation of the predecessor problem into SAT

[3] The class of threshold functions is defined in Sect. 2.

and the time used by two SAT solvers (namely, Clasp [7] and Glucose [8]) to solve the corresponding SAT instances.

Related Work. Computational problems associated with discrete dynamical systems have been addressed by many researchers. For example, Barrett et al. [3] and Rosenkrantz et al. [21] studied the reachability problem (i.e., given a SyDS $\mathbb{S}$ and two configurations $\mathbb{C}_1$ and $\mathbb{C}_2$, does $\mathbb{S}$ starting from $\mathbb{C}_1$ reach $\mathbb{C}_2$?) for undirected graphs. The same problem for directed graphs has been studied in [5,19]. Tosic [23,24] presented results for counting the number of fixed points[4] for systems with special forms of local functions. Kosub and Homan [15] presented dichotomy results that delineate computationally intractable and efficiently solvable versions of counting fixed points, based on the class of allowable local functions. The complexity of the predecessor existence problem for various classes of underlying graphs and local functions is investigated in [4]. A more general version of the predecessor existence problem, where the goal is to find t-step predecessors for values of $t \geq 2$, has been studied in [14,16]. Problems similar to predecessor existence have also been considered in the context of cellular automata [6,9]. Readers interested in the applications of graph dynamical systems are referred to [1,17].

Note: For space reasons, proofs are not included; they can be found in [20].

2 Preliminaries

Synchronous Dynamical Systems and Local Functions. We follow the presentation in [4] for the basic definitions associated with discrete dynamical systems. Let $\mathbb{B}$ denote the Boolean domain $\{0,1\}$. A **Synchronous Dynamical System** (SyDS) $\mathbb{S}$ over $\mathbb{B}$ is specified as a pair $\mathbb{S} = (G, \mathbb{F})$, where (a) $G(V,E)$, an undirected graph with $|V| = n$, represents the underlying graph of the SyDS and (b) $\mathbb{F} = \{f_1, f_2, \ldots, f_n\}$ is a collection of functions in the system, with f_i denoting the **local function** associated with node v_i, $1 \leq i \leq n$. Each node of G has a state value from $\mathbb{B}$. For any node v, we use $N[v]$ to denote the **closed neighborhood** of v, that is, the set consisting of v and all its neighbors. Each function f_i specifies the local interaction between node v_i and its neighbors in G. The inputs to function f_i are the state of the nodes in $N[v_i]$; function f_i maps each combination of inputs to a value in $\mathbb{B}$. This value becomes the next state of node v_i. It is assumed that each local function can be computed efficiently.

At any time τ, the **configuration** $\mathbb{C}$ of a SyDS is the n-vector $(s_1^\tau, s_2^\tau, \ldots, s_n^\tau)$, where $s_i^\tau \in \mathbb{B}$ is the state of node v_i at time τ $(1 \leq i \leq n)$. Given a configuration $\mathbb{C}$, the state of a node v in $\mathbb{C}$ is denoted by $\mathbb{C}(v)$. In a SyDS, all nodes compute and update their next state *synchronously*. Other update disciplines (e.g., sequential updates) have also been considered in the literature (e.g., [4,17]). Suppose a given SyDS transitions in one step from a configuration $\mathbb{C}'$ to a configuration $\mathbb{C}$. Then we say that $\mathbb{C}$ is the **successor** of $\mathbb{C}'$, and $\mathbb{C}'$ is a **predecessor** of $\mathbb{C}$. Since the SyDSs considered in this paper are deterministic, each configuration has a *unique* successor. However, a configuration may have zero or more predecessors. In the graph dynamical systems literature, configurations with no

[4] A fixed point of a SyDS is a configuration which is its own successor.

predecessors are called **Garden of Eden** (GE) configurations [17]. Given a configuration $\mathbb{C}$, we use the notation $\sigma(\mathbb{C})$ to denote the successor of $\mathbb{C}$, and $\Pi(\mathbb{C})$ to denote the set of all predecessors of $\mathbb{C}$.

SyDSs have been considered in the literature under many classes of local functions (see e.g., [4, 14]). We now present an example of a SyDS where the local function at each node is a **threshold function**. For each integer $k \geq 0$, the k-**threshold function** has the value 1 iff at at least k of its inputs are 1.

Example: The underlying graph of a SyDS shown in Fig. 1. The threshold value for each node is shown within parentheses. (Thus, the local function at b is the 2-threshold function while that at d is the 3-threshold function.) Suppose the initial configuration of the system is $(1,1,0,0,0)$; that is, a and b are in state 1 while c, d and e are in state 0. The reader can verify that starting from time 0, the system goes through the following sequence of configurations: $(1,1,0,0,0) \longrightarrow (1,1,1,0,0) \longrightarrow (1,1,1,1,0) \longrightarrow (1,1,1,1,1)$. Once the system reaches the configuration $(1,1,1,1,1)$ at time step 3, no further state changes occur in the subsequent time steps; that is, the configuration $(1,1,1,1,1)$ is a **fixed point**.

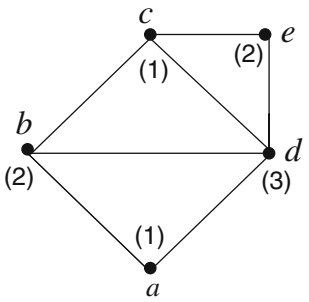


Fig. 1. An Example of a SyDS where each node has a threshold function. The threshold values are shown in parentheses.

The **phase space** $\mathbb{P}_{\mathbb{S}}$ of a SyDS $\mathbb{S}$ is a directed graph defined as follows. There is a node in $\mathbb{P}_{\mathbb{S}}$ for each configuration of $\mathbb{S}$. There is a directed edge from a node representing configuration $\mathbb{C}_1$ to that representing configuration $\mathbb{C}_2$ if there is a one step transition of $\mathbb{S}$ from $\mathbb{C}_1$ to $\mathbb{C}_2$. For a SyDS with n nodes, the number of nodes in the phase space is 2^n; thus, the size of phase space is *exponential* in the size of a SyDS. Each node in the phase space has an outdegree of 1 (since our SyDS model is deterministic). Also, in the phase space, each fixed point of a SyDS is a self-loop and each GE configuration is a node of indegree zero.

Hamming Distance and Similarity of Configurations. Given two configurations $\mathbb{C}_1$ and $\mathbb{C}_2$ of a SyDS over the domain $\{0,1\}$, the **Hamming Distance** between $\mathbb{C}_1$ and $\mathbb{C}_2$, denoted by $\mathbb{H}(\mathbb{C}_1, \mathbb{C}_2)$, is the number of positions in which they differ. For example, if $\mathbb{C}_1 = (1,0,0,1)$ and $\mathbb{C}_2 = (0,1,0,0)$, then $\mathbb{H}(\mathbb{C}_1, \mathbb{C}_2) = 3$. We say that two configurations $\mathbb{C}_1$ and $\mathbb{C}_2$ of a SyDS are h-**close** if $\mathbb{H}(\mathbb{C}_1, \mathbb{C}_2) = h$. Two configurations that are h-close for a small value of h can be thought of as 'similar' configurations. We note that in a SyDS with n nodes, the maximum Hamming distance between any pair of configurations $\mathbb{C}_1$ and $\mathbb{C}_2$ is n; this occurs when $\mathbb{C}_1$ is the bitwise complement of $\mathbb{C}_2$.

Similarity Measures for Sets of Configurations. Our focus is on studying the degree of similarity between predecessors of similar configurations. To do this, we define the following distance measures between two sets of nonempty configurations S_1 and S_2.

(a) **Minimum Separation** (MINSEP): This measure is defined as follows:

$$\text{MINSEP}(S_1, S_2) \;=\; \min\{\mathbb{H}(\mathbb{C}, \mathbb{C}') \;:\; \mathbb{C} \in S_1,\; \mathbb{C}' \in S_2\}.$$

(b) **Maximum Separation** (MAXSEP): This measure, which is analogous to minimum separation, is defined as follows.

$$\text{MAXSEP}(S_1, S_2) \;=\; \max\{\mathbb{H}(\mathbb{C}, \mathbb{C}') \;:\; \mathbb{C} \in S_1,\; \mathbb{C}' \in S_2\}.$$

(c) **Average Separation** (AVGSEP): This measure is defined as follows.

$$\text{AVGSEP}(S_1, S_2) \;=\; \frac{\sum_{\mathbb{C} \in S_1,\; \mathbb{C}' \in S_2} \mathbb{H}(\mathbb{C}, \mathbb{C}')}{|S_1| \times |S_2|}.$$

Among the above measures, a small value of MAXSEP provides the strongest guarantee of similarity. This is because if $\text{MAXSEP}(S_1, S_2) = \alpha$, and α is small, then the Hamming distance between any pair configurations $\mathbb{C}$ and $\mathbb{C}'$, where $\mathbb{C} \in S_1$ and $\mathbb{C}' \in S_2$, is at most α; in other words, each such configuration pair is α-close. For convenience, when at least one of the sets S_1 and S_2 is empty, we define the values of $\text{MINSEP}(S_1, S_2)$, $\text{MAXSEP}(S_1, S_2)$ and $\text{AVGSEP}(S_1, S_2)$ to be ∞.

The following lemma points out two simple properties of predecessors in SyDSs.

Lemma 1. *Let $\mathbb{S}$ be a SyDS. (i) Suppose $\mathbb{C}_1$ and $\mathbb{C}_2$ are two different configurations of $\mathbb{S}$. The sets $\Pi(\mathbb{C}_1)$ and $\Pi(\mathbb{C}_2)$ are disjoint. (ii) Suppose every configuration of $\mathbb{S}$ has a predecessor. Then each configuration of $\mathbb{S}$ has a unique predecessor.*

Proof: See [20].

Boolean Satisfiability Problem (SAT): Given an m-variable Boolean function F of in conjunctive normal form (CNF), the goal of the Satisfiability problem (SAT) problem is to determine whether there is an assignment of a Boolean values to each of the m variables so that the function F evaluates to true under the assignment. We will explain in Sect. 3 how the problem of finding predecessors of a given configuration can be reduced to an appropriate instance of SAT. Many public domain SAT solvers are currently available to obtain solutions to practical SAT instances [22]. Our experimental results in Sect. 4 were generated using SAT solvers.

3 Analytical Results

Overview. In this section, we first show that the problem of finding the predecessors of a given configuration of a SyDS can be expressed as an instance of SAT. This transformation forms the basis for the experimental results presented in Sect. 4. In addition, we present several analytical results regarding the similarities of predecessor sets of two configurations of a SyDS. Throughout this section, the reader should bear in mind that for any configuration $\mathbb{C}$, $\sigma(\mathbb{C})$ denotes the successor of $\mathbb{C}$ and $\Pi(\mathbb{C})$ denotes the set of all predecessors of $\mathbb{C}$.

Reducing Predecessor Finding to SAT. We assume that the nodes of the underlying graph of the given SyDS are numbered 1 through n and that the local function at node

i is denoted by f_i, $1 \leq i \leq n$. For each node i, let N_i denote the **closed neighborhood** of node i (defined in Sect. 2) in the underlying graph; thus, the states of the nodes in N_i are the inputs to the local function f_i, $1 \leq i \leq n$.

Let $\mathbb{C} = (c_1, c_2, \ldots, c_n)$ be the given configuration for which we need to find a predecessor (if one exists). Note that each c_i is a known 0 or 1 value, $1 \leq i \leq n$. We need to find a configuration $\mathbb{C}' = (x_1, x_2, \ldots, x_n)$ such that $\mathbb{C}'$ is a predecessor of $\mathbb{C}$ (if one exists). This condition can be transformed into an instance of SAT as follows.

Consider node i of the SyDS. As mentioned earlier, let $N_i = \{i_1, i_2, \ldots, i_r\}$ denote the closed neighborhood of node i, where $r = |N_i|$. Thus, the inputs to the local function f_i at node i are x_{i_1}, x_{i_2}, …, x_{i_r}. Since we want $\mathbb{C}'$ to be a predecessor of $\mathbb{C}$, the condition to be satisfied at node i is the following:

$$c_i \;\Leftrightarrow\; f_i(x_{i_1}, x_{i_2}, \ldots, x_{i_k}). \tag{1}$$

Since c_i is a known 0 or 1 value, the expression given in Eq. (1) can be simplified. If $c_i = 0$, the above expression simplifies to $\neg f_i(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$. Likewise, if $c_i = 1$, the above expression simplifies to $f_i(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$.

Using P_i to denote the subexpression given by Eq. (1) for node i, the condition to be satisfied for $\mathbb{C}'$ to be a predecessor of $\mathbb{C}$ is given by

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n. \tag{2}$$

As before, since each subexpression P_i can be expressed as an equivalent CNF, we can get a CNF formula with variables x_1, x_2, …, x_n from Eq. (2). Each solution to the resulting CNF formula (which can be obtained using a SAT solver) gives a predecessor of the given configuration $\mathbb{C}$. If there is no satisfying assignment to the CNF formula corresponding to the expression in Eq. (2), then $\mathbb{C}$ has no predecessor; that is, $\mathbb{C}$ is a Garden-of-Eden configuration. This SAT-based approach for finding predecessors will be incorporated into a software system called `net.science` that is being built in collaboration with several organizations [2].

Results on Similarities of Predecessor Sets. We now present our theoretical results regarding the similarity of predecessors of two configurations. Our first result points out that there are SyDSs where the Hamming distance between a pair of configurations is preserved when predecessors are considered.

Proposition 1. *Let G be an arbitrary graph. Then, there is a SyDS $\mathbb{S}$ with underlying graph G, such that $\mathbb{S}$ has the following properties: (i) every configuration has a predecessor, and (ii) for any pair of distinct configurations $\mathbb{C}_1$ and $\mathbb{C}_2$, $\mathbb{H}(\sigma(\mathbb{C}_1), \sigma(\mathbb{C}_2)) = \mathbb{H}(\mathbb{C}_1, \mathbb{C}_2)$ and $\textsc{MaxSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = \textsc{MinSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = \textsc{AvgSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = \mathbb{H}(\mathbb{C}_1, \mathbb{C}_2)$.*

Proof: See [20].

Our next result shows that there are SyDSs for which there are two distinct configurations that are 1-close, but their predecessors are highly dissimilar; that is, they have the maximum possible Hamming distance.

Proposition 2. *Let G be an arbitrary connected graph, and let n be the number of nodes in G. Then, there is a SyDS* $\mathbb{S}$ *with underlying graph G, such that* $\mathbb{S}$ *has the following properties: (i) every configuration has a predecessor and (ii) for every configuration* $\mathbb{C}_1$*, there is a configuration* $\mathbb{C}_2$ *such that* $\mathbb{H}(\mathbb{C}_1, \mathbb{C}_2) = 1$ *and* $\textsc{MaxSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = \textsc{MinSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = \textsc{AvgSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = n$.

Proof: See [20].

We now show the existence of SyDSs in which there are pairs of configurations which have the maximum level of dissimilarity but their predecessors are 1-close.

Proposition 3. *Let G be an arbitrary graph, and let* Δ *be the maximum node degree of G. Then, there is a SyDS* $\mathbb{S}$ *with underlying graph G, such that* $\mathbb{S}$ *has the following properties: (i) every configuration has a predecessor and (ii) for every configuration* $\mathbb{C}_1$*, there is a configuration* $\mathbb{C}_2$ *such that* $\mathbb{H}(\mathbb{C}_1, \mathbb{C}_2) = \Delta + 1$ *and* $\textsc{MaxSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = \textsc{MinSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = \textsc{AvgSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = 1$.

Proof: See [20].

The following corollary is a direct consequence of Proposition 3 by taking the underlying graph of the SyDS to be the star graph on n nodes.

Corollary 1. *For any integer* $n \geq 2$*, there is a SyDS* $\mathbb{S}$ *with n nodes satisfying the following properties: (i) there is a pair of configurations* $\mathbb{C}_1$ *and* $\mathbb{C}_2$ *with* $\mathbb{H}(\mathbb{C}_1, \mathbb{C}_2) = n$ *and* $\textsc{MaxSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) = 1$.

We now present a result that establishes the computational complexity of computing distance measures for predecessor configurations. The decision problem, which we call **Minimum Predecessor Separation** (MPS), is the following: given a SyDS $\mathbb{S}$, two configurations $\mathbb{C}_1$ and $\mathbb{C}_2$, and a positive integer q, is $\textsc{MinSep}(\Pi(\mathbb{C}_1), \Pi(\mathbb{C}_2)) \leq q$? Using the known result that the **Predecessor Existence** problem (i.e., given a SyDS $\mathbb{S}$ and a configuration $\mathbb{C}$, does $\mathbb{C}$ have a predecessor?) is **NP**-complete [4], it can be shown that MPS is also **NP**-complete. This result is stated below.

Proposition 4. *The MPS problem is* ***NP****-complete.*

Proof: See [20].

Our proof of Proposition 4 relies on the fact that it **NP**-hard to decide whether a configuration $\mathbb{C}$ has a predecessor. We now present a stronger **NP**-completeness result. We show that the MPS problem is **NP**-complete even when we are given predecessors of $\mathbb{C}_1$ and $\mathbb{C}_2$. We call the decision problem when this extra information is given **Minimum Predecessor Separation Given Predecessors** (MPSGP). Note that since the predecessors of $\mathbb{C}_1$ and $\mathbb{C}_2$ are specified in a given MPSGP problem instance, it is unnecessary to explicitly specify $\mathbb{C}_1$ and $\mathbb{C}_2$. Thus, we formalize the MPSGP problem as follows: given a SyDS $\mathbb{S}$, and two configurations $\mathbb{C}_1'$ and $\mathbb{C}_2'$, is $\textsc{MinSep}(\Pi(\sigma(\mathbb{C}_1')), \Pi(\sigma(\mathbb{C}_2'))) < \mathbb{H}(\mathbb{C}_1', \mathbb{C}_2')$? Our next result points out the **NP**-hardness of this problem.

Theorem 1. *The MPSGP problem is* ***NP****-complete.*

Proof: See [20].

4 Experimental Results

Overview. The analytical results presented in Sect. 3 show that in general, SyDSs may exhibit extreme behaviors with respect to evolution of configurations. So, in the experimental phase, our goal was to understand the behavior for restricted classes of graphs and local functions. We generated SyDSs whose underlying graphs are from special classes of graphs and whose local functions are from restricted classes of Boolean functions. We generated pairs of configurations that are h-close for small values of h and examined the range of Hamming distances for their sets of predecessors. We used the transformation from the predecessor problem to SAT discussed in Sect. 3.

SyDS Construction. We investigated several types of underlying graph structures including Erdős–Rényi models, lattice/grid graphs, and Watts-Strogatz small-world networks [18]. All graphs were created using the NetworkX library [11]. The Erdős–Rényi graphs were constructed such that the estimated mean degree of the graph was 16. Grid graphs were constructed such that each node connected to exactly four other nodes. Nodes in the Watts-Strogatz small world networks were initially wired to their eight nearest neighbors; then each edge had a 50% chance to be rewired to a random node in the graph.

To examine the similarity of configurations, we considered several local functions. All SyDSs constructed and tested were uniform SyDSs[5] with threshold functions ranging from threshold 1 (equivalent to Boolean OR) to threshold 4. We chose threshold functions as they are monotone Boolean functions. As shown in Sect. 3, SyDSs with similar configurations and non-monotone local functions (such as exclusive OR) can have predecessors with very high variability in their Hamming distances. With threshold functions, we expected the Hamming distances of the predecessors of similar configurations to show less extreme variance.

Procedure for Generating Configurations and Their Predecessors. We implemented the transformation from the predecessor problem to SAT in Python. We limited the number of predecessors generated for each configuration for the following reasons. In order to compute the minimum, average, and maximum Hamming distances between two sets S_1 and S_2 of predecessors, each predecessor in S_1 must be compared with each predecessor from S_2. For example, with just 10^4 predecessors for each configuration, the number of such comparisons is 10^8. In addition to time used for such a computation, attempting to exhaustively find and record every predecessor for larger graph sizes could generate several terabytes of data.

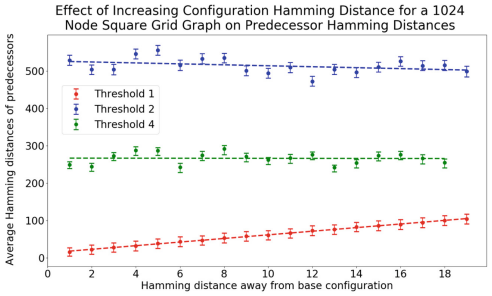
We defined our "base" configuration as the one with all node states set to 1. To generate a configuration with Hamming distance h from the base configuration, the states of h random nodes were changed from 1 to 0. In total, 20 configurations with different Hamming distances were generated. We generated up to 10^4 solutions for each predecessor problem. We computed the necessary Hamming distance values between the set of predecessors for the base configuration and the sets of predecessors of the 20 configurations derived from the base configuration. Our results provide an indication of the minimum and maximum Hamming distances. In the plots shown in this section, the

[5] A **uniform** SyDS is one in which all the nodes have the same local function.

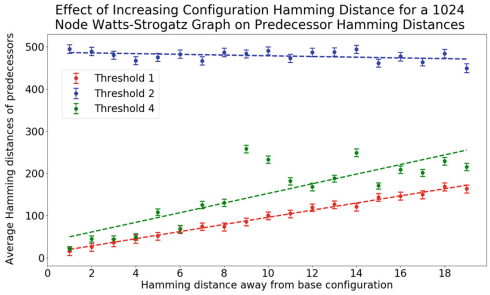
mean Hamming distance between the predecessors of the base configuration and those of the 20 derived configurations are shown, with error bars representing the minimum and maximum Hamming distances of the solution sets. For each threshold value, we fit a linear trend line to the results.

Table 1. Table showing minimum, maximum and average Hamming distance values for grids and Watts-Strogatz small world networks with 16 nodes

Threshold	Hamming distance from base Configuration	Square Grid			Watts-Strogatz Network		
		Predecessors' Hamming distance			Predecessors' Hamming distance		
		Minimum	Average	Maximum	Minimum	Average	Maximum
2	2	2	8.000	14	1	8.248	16
	4	2	8.376	16	1	8.304	16
	6	2	8.602	16	2	8.384	16
	14	5	9.605	16	3	8.490	16
3	2	2	6.905	11	1	8.250	16
	4	2	6.905	11	2	8.537	16
	6	2	7.502	13	1	8.473	16
	12	5	9.095	14	2	8.799	16
	14	5	9.540	15	4	8.883	16
4	2	1	3.789	5	1	7.872	15
	4	2	4.491	6	1	7.964	14
	10	4	6.421	8	3	8.486	15
	12	4	7.013	10	4	9.041	16
	14	4	8.191	12	5	9.163	15



(a) Graph showing average Hamming distance values for a 1024 node square grid network



(b) Graph showing average Hamming distance values for a 1024 node Watts-Strogatz network

Fig. 2. Average Hamming distance values for grid and Watts-Strogatz networks

Hamming Distance Results for Small Networks. Table 1 shows the minimum, average, and maximum Hamming distances for 16 node grids and Watts-Strogatz networks. For these small networks, we were able to generate all predecessors for each configuration. The table shows the results for the configurations for which both the grid and the Watts-Strogatz graph had predecessors. For both classes of graphs and all threshold values, the minimum and average predecessor Hamming distance show a roughly monotonic non-decreasing trend with increase in the Hamming distance of a configuration from the base configuration. The maximum Hamming distance also increased

monotonically for the grid graphs; however, for the Watts-Strogatz networks started at the highest value (16) and stayed very close to that value.

Hamming Distance Results for Large Networks. Our results for the 1024 node square grid network and the 1024 node Watts-Strogatz small world network are shown in Figs. 2a and 2b respectively. The average Hamming distances of predecessors for these two graphs show similar trends. For both networks, the Hamming distance values for threshold 1 were lower compared to the other threshold values; moreover, the average Hamming distance increased linearly with increase in the Hamming distance of a configuration from the base configuration. Threshold 2 showed the highest values of average Hamming distances for both networks; further, the average Hamming distance also showed a stable trend as configuration Hamming distance was increased. For the square grid graph, threshold 4 also showed a stable trend. In contrast, threshold 4 results for the Watts-Strogatz graph show a linearly increasing trend similar to Threshold 1. For both networks, the range of minimum and maximum Hamming distances was within 50 units of the average.

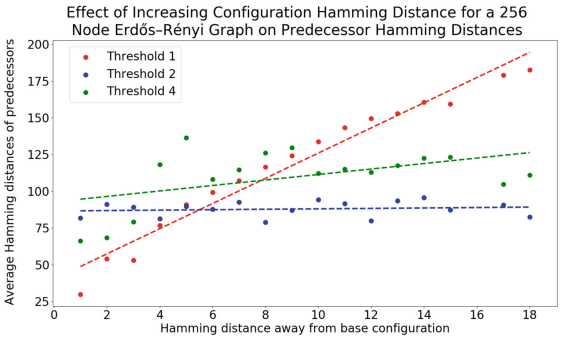


Fig. 3. Graph showing average Hamming distance values for a 256 node Erdős-Rényi network

Average Hamming distance values for an Erdős–Rényi graph with 256 nodes are shown in Fig. 3. There, the minimum and maximum Hamming distances in each set were within 30 units of the average and are not shown in Fig. 3 to avoid clutter. The average Hamming distance values for Threshold 1 once again show a linearly increasing trend with increase in the Hamming distance from the base configuration. Threshold 4 also shows a linearly increasing trend but with a slope smaller than that of threshold 1. The values for Threshold 2 show a more or less stable trend.

Number of Clauses Generated and SAT Solver Runtime. We conducted tests to compare the performance of the two most recently updated SAT solvers, namely Clasp [7] and Glucose [8]. For these experiments, graphs were generated in the same manner as previously mentioned except that Erdős–Rényi graphs for this experiment were constructed to have an average degree of 4. Assuming that each local function is the 1-threshold function, we computed the number of clauses generated for each predecessor problem with a uniform threshold of 1 on a sample of 20 predecessor problems and measured the average CPU time[6] it took each SAT solver to produce one solution. The results are shown in Table 2.

There was no significant difference between the Glucose and Clasp SAT solvers in terms of CPU time taken to obtain a single solution to a SAT problem. The only notable exception is that for the larger Watts-Strogatz graph, Clasp was faster than Glucose

[6] Experiments were run on a single core of a 2.80 GHz Intel Core i5-8400 CPU and with 16 GB of RAM.

Table 2. Table showing the number of clauses in the SAT instance generated from a predecessor problem and the CPU time to generate a solution for several networks

Network type	$2^{16} = 65,536$ Nodes			$2^{18} = 262,144$ Nodes		
	Number of clauses	Clasp time (seconds)	Glucose time (seconds)	Number of clauses	Clasp time (seconds)	Glucose time (seconds)
Square Grid	65806	0.059	0.054	262414	0.213	0.201
Watts-Strogatz	77614	0.552	0.772	299310	8.418	11.219
Erdős–Rényi	65686	0.096	0.129	262800	0.882	0.863

(8.418 s vs 11.219 s). The larger amount of time used for this graph could potentially be due to the larger average degree. Clasp was eventually chosen for our experiments because it can generate all the solutions for a given SAT instance.

5 Summary and Future Research Directions

We presented analytical and experimental results regarding the time evolution of similar configurations. We demonstrated the use of SAT solvers in studying these questions. There are several directions for future work. We considered one method of generating similar pairs of configurations starting from a base configuration. One may investigate other ways of generating similar configurations. Also, instead of considering one step predecessors, one may consider similarity issues for t-step predecessors for $t \geq 2$. Such generalized predecessor problems can also be reduced to SAT. Further, instead of Hamming distance, one may consider other measures of similarity between configurations; for example, two configurations may be considered similar if they have the same number of 1's.

Acknowledgments. We thank the referees for their comments. This work is partially supported by NSF Grants ACI-1443054 (DIBBS), IIS-1633028 (BIG DATA), CMMI-1745207 (EAGER), OAC-1916805 (CINES), CCF-1918656 (Expeditions) and IIS-1908530.

References

1. Adiga, A., Kuhlman, C.J., Marathe, M.V., Mortveit, H.S., Ravi, S.S., Vullikanti, A.: Graphical dynamical systems and their applications to bio-social systems. Springer Int. J. Adv. Eng. Sci. Appl. Math. **11**(2), 153–171 (2019)
2. Ahmed, N.K., Alo, R.A., Amelink, C.T., et al.: net.science: a cyberinfrastructure for sustained innovation in network science and engineering. In: Gateway (2020)
3. Barrett, C.L., Hunt III, H.B., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: Complexity of reachability problems for finite discrete dynamical systems. J. Comput. Syst. Sci. **72**(8), 1317–1345 (2006)
4. Barrett, C., Hunt III, H.B., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E., Thakur, M.: Predecessor existence problems for finite discrete dynamical systems. Theoret. Comput. Sci. **386**(1), 3–37 (2007)
5. Chistikov, D., Lisowski, G., Paterson, M., Turrini, P.: Convergence of opinion diffusion is PSPACE-complete. CoRR abs/1912.09864 (2019). http://arxiv.org/abs/1912.09864
6. Durand, B.: A random NP-complete problem for inversion of 2D cellular automata. Theoret. Comput. Sci. **148**(1), 19–32 (1995)

7. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Clasp: a conflict-driven answer set solver. In: Baral, C., Brewka, G., Schlipf, J. (eds.) Logic Programming and Nonmonotonic Reasoning, pp. 260–265. Springer, Heidelberg (2007)
8. The Glucose SAT solver (2016). https://www.labri.fr/perso/lsimon/glucose/
9. Green, F.: NP-complete problems in Cellular Automata. Complex Syst. **1**(3), 453–474 (1987)
10. Gutowitz, H.: Cellular Automata: Theory and Experiment. North Holland (1989)
11. Hagberg, A., Schult, D., Swart, P.: NetworkX reference (2020). https://networkx.github.io/documentation/latest/_downloads/networkx_reference.pdf
12. Kauffman, S., Peterson, C., Samuelsson, B., Troein, C.: Random Boolean network models and the yeast transcriptional network. Proc. Natl. Acad. Sci. (PNAS) **100**(25), 14796–14799 (2003)
13. Kauffman, S., Peterson, C., Samuelsson, B., Troein, C.: Genetic networks with canalyzing Boolean rules are always stable. Proc. Natl. Acad. Sci. (PNAS) **101**(49), 17102–17107 (2004)
14. Kawachi, A., Ogihara, M., Uchizawa, K.: Generalized predecessor existence problems for Boolean finite dynamical systems. In: 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017), pp. 8:1–8:13 (2017)
15. Kosub, S., Homan, C.M.: Dichotomy results for fixed point counting in Boolean dynamical systems. In: Proceedings of the 10th Italian Conference on Theoretical Computer Science, pp. 163–174 (2007)
16. Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: Computational aspects of fault location and resilience problems for interdependent infrastructure networks. In: International Conference on Complex Networks and their Applications, pp. 879–890. Springer, Heidelberg (2018)
17. Mortveit, H., Reidys, C.: An Introduction to Sequential Dynamical Systems. Springer, New York (2007)
18. Newman, M., Barabási, A.L., Watts, D.J.: The Structure and Dynamics of Networks. Princeton University Press, Princeton (2006)
19. Ogihara, M., Uchizawa, K.: Computational complexity studies of synchronous Boolean finite dynamical systems on directed graphs. Inf. Comput. **256**, 226–236 (2017)
20. Priest, J.D., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: Evolution of similar configurations in graph dynamical systems. Technical Report for 2020, Network Systems Science and Advanced Computing (NSSAC) Division, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA, USA. https://drive.google.com/file/d/1Bc2idtlFnk7uidLnDEi6U3iggET0O0dh/view?usp=sharing
21. Rosenkrantz, D.J., Marathe, M.V., Ravi, S.S., Stearns, R.E.: Testing phase space properties of synchronous dynamical systems with nested canalyzing local functions. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, 10–15 July 2018, pp. 1585–1594 (2018)
22. Information regarding SAT solvers (2018). http://www.satlive.org
23. Tosic, P.T.: On the complexity of enumerating possible dynamics of sparsely connected Boolean network automata with simple update rules. In: Automata 2010 - 16th International Workshop on CA and DCS, pp. 125–144 (2010)
24. Tosic, P.T.: Phase transitions in possible dynamics of cellular and graph automata models of sparsely interconnected multi-agent systems. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, 8-12 May 2017, pp. 474–483 (2017)
25. Valente, T.W.: Social network thresholds in the diffusion of innovations. Soc. Netw. **18**, 69–89 (1996)
26. Wolfram, S.: Theory and Applications of Cellular Automata. World Scientific (1987)
27. Wooldridge, M.: An Introduction to Multi-Agent Systems. Wiley, West Sussex (2002)