An Efficient Algorithm for the Detection of Eden

David J. Warne[*]
School of Electrical Engineering and Computer Science,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia
Ross F. Hayward
School of Electrical Engineering and Computer Science,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia
Neil A. Kelson
High Performance Computing and Research Support,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia
Dann G. Mallet
School of Mathematical Sciences,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia

In this paper, a polynomial time algorithm is presented for solving the *Eden* problem for *graph cellular automata*. The algorithm is based on our *neighborhood elimination* operation which removes local neighborhood configurations which cannot be used in a pre-image of the given configuration. This paper presents a detailed derivation of our algorithm from first principles, and a detailed complexity and accuracy analysis is also given. In the case of time complexity, it is shown that the average case time complexity of the algorithm is $\Theta(n^2)$, and the best and worst cases are $\Omega(n)$ and $O(n^3)$ respectively. This represents a vast improvement in the upper bound over current methods, without compromising average case performance.

1. Introduction

Cellular automata and more generally discrete dynamical systems are powerful tools for modeling of complex phenomena [14]. This includes applications from physics, biology, and computer science [1]. Some have even speculated that the study of cellular automata may lead to a *Grand Unified Theory* of everything [13].

The study of the global dynamics of cellular automata (i.e., the

[*]E-mail address: david.warne@qut.edu.au

study of automata configuration transition graphs) can provide unique insight into complex systems [20]. Efficient construction of a configuration transition graph typically requires a method to determine if the given configuration is located on a leaf node of this graph [17].

This problem is known as the *Eden* problem, and has been shown to be computationally intractable for d-dimensional systems when $d > 1$. This is reflected in the worst case computational complexity of algorithms that solve the Eden problem for higher dimensions (e.g., Wuensche's general reverse algorithm [19]).

We present a new algorithm for approximately solving the Eden problem for *graph cellular automata* (i.e., cellular automata on graphs [8, 7]); the most general form of deterministic cellular automata. Although there exist rare instances in which the algorithm will fail to identify the non-existence of a pre-image, this is made up for by it's asymptotic complexity class which is $O(n^3)$ for the worst case and $\Theta(n^2)$ for the average case. This provides a method which is more computationally feasible in the worst case than approaches based on Wuensche and Lesser's reverse algorithm [20] and Wuensche's general reverse algorithm [19] for the study of the global dynamics of higher dimensional discrete dynamical systems with potentially a large number of cells.

2. Background

2.1 Discrete dynamical systems

A regular cellular automaton can be defined as a lattice of *finite state automata*, typically referred to as *cells* or *sites.* A state transition function defines how a cell updates its state based on it's current state and the state of it's neighbors. Cells update synchronously in discrete time intervals. The sequence of all cell states at a given time is referred to as the automaton's *configuration.*

Random boolean networks are binary cellular automata with one critical difference; there is no requirement that cells be located on a regular lattice [19]. Instead, neighborhoods are constructed via a random wiring. This random wiring makes random boolean networks useful for theoretical biological models of genetic regulatory networks [5, 18].

Graph cellular automata (also referred to as *Generalized automata networks* [11]) are a generalization of both cellular automata and random boolean networks. For a graph cellular automaton, cell connectivity is defined by a connected graph. The class of graph cellular automata contains regular cellular automata and random boolean networks as sub-classes. Cellular automata and random boolean networks can be considered as discrete dynamical systems. Despite their simple construction, discrete dynamical systems have been shown to be capable of very complex behavior [16, 15, 6]. Furthermore, computationally intractable, and formally undecidable problems relating to discrete dynamical systems have been shown to exist [3, 9].

2.2 The Eden problem

A particular problem of interest in the study of the global dynamics of discrete dynamical systems is the so-called *Eden* problem (also called the *Predecessor existence* problem [10, 2]). The Eden problem, attempts to determine, for a given automaton, if there exists a configuration (i.e., pre-image) that will evolve to the given configuration in the next time step. If the Eden problem is resolved to be $false$ then the configuration is called a *Garden-of-Eden* configuration (i.e., it has no pre-image). Wuensche and Lesser studied the Eden problem in depth and developed a reverse algorithm for one dimensional regular cellular automata [20]. Wuensche further generalized this approach to the case of random boolean networks, which may also be applied to graph cellular automata [19, 17]. While Wuensche and Lesser's method performs very well for small cellular automata, this methods upper bound is $O(2^n)$ (as we will show in Section 5.1) which prevents exploration of large discrete dynamical systems.

For one-dimensional finite cellular automata the Eden problem is in P, however for multi-dimensional finite cellular automata the Eden problem has been shown to be NP-Complete [10]. Even certain variants of the Eden problem in one-dimension (such as the *Constrained Eden* problem [9]) have been shown to be NP-Complete. Assuming that $P \neq NP$, then there does not exist a polynomial time algorithm to solve the Eden problem for graph cellular automata.

If we assume $P \neq NP$, then a complete solution to the Eden problem for graph cellular automata is computationally intractable. However, this does not exclude the possibility of a good solution (i.e., one that can identify most Garden-of-Eden configurations) being achievable in polynomial time. In this paper, we present an algorithm which provides a good solution to the Eden problem for graph cellular automata in cubic time. By solving the problem for graph cellular automata we, by extension, solve the problem for regular cellular automata and random boolean networks. Furthermore, we can show that our algorithm solves the Eden problem exactly when the topology of the graph cellular automaton is equivalent to a one dimensional finite cellular automaton with periodic boundary conditions.

2.3 Formal definition of graph cellular automata

In this section, we provide a formal definition of graph cellular automata. Our formalism is based heavily on the work of Fates [4], Marr et al. [8, 7], and Tomassini [11].

We consider a graph cellular automaton to be defined as a 4-tuple consisting of a connected graph, a set of states, a set of neighborhood mappings and a set of state transition functions. This is given formally in Definition 1.

Definition 1. Let $\mathbf{A} = (G, \Sigma, U, \Gamma)$ define a *graph cellular automaton*, where $G = (V, E)$ is a graph with vertices $V \subset \mathbb{Z}$ and edges $E \subseteq V \times V$, Σ is a finite set of symbols referred to as the alphabet, $U = \{h_i : i \in V\}$ is the set of neighborhoods $h_i = \{i\} \cup \{j : (i,j) \in E \vee (j,i) \in E\}$, and $\Gamma = \{g_i : i \in V\}$ is the set of all state transition functions $g_i : \Sigma^{|h_i|} \to \Sigma$.

In Definition 1, the vertices of the graph G represent the cells of the automaton $\mathbf{A}$. Note that the neighborhood, h_i, of each cell, i, is effectively the set of cells that are connected to cell i via the set of edges E including i itself[1].

At any time t each cell is associated with a state σ. For this we define the mapping in Definition 2. From this we can construct the global configuration of the automaton in Definition 3.

Definition 2. Let $C : V \to \Sigma$ be a mapping from a cell $i \in V$ to a state $\sigma \in \Sigma$ such that $C^t(i)$ represents the state of cell i at time t. Let $C^t(h_i) \in \Sigma^{|h_i|}$ be the *neighborhood configuration* of i.

Definition 3. Let $\phi^t = \{C^t(i) : i \in V\}$ be the configuration of the automaton $\mathbf{A}$ at time t. $\phi^t \in \Phi$ where Φ is the set of all possible configurations of $\mathbf{A}$.

Finally we define the evolution of a graph cellular automaton as the sequence of configurations generated by repeated synchronous application of the local state transition functions. This is given as a recurrence relation expressed in terms of the global configuration transition function. This is given in Definition 4.

Definition 4. Let the recurrence relation $\phi^{t+1} = f(\phi^t), t \geq 0$ be the evolution of $\mathbf{A}$, where $f : \Phi \to \Phi$ is the global configuration transition function $f(\phi^t) = \{(\phi^t, \phi^{t+1}) : \phi^t = \{C^t(i) : i \in V\} \wedge \phi^{t+1} = \{g_i(C^t(h_i)) : i \in V\}\}$.

We can now define formally an instance of the Eden problem.

Definition 5. *Problem*: The Eden problem (EDEN).
Instance: A graph cellular automaton $\mathbf{A}$ and a configuration $\phi \in \Sigma^{|V|}$.
Question: Does there exist an initial configuration ϕ^0 such that $\phi = f(\phi^0)$ under the evolution of $\mathbf{A}$?

[1]Note that the construction of h_i in Definition 1 assumes an undirected graph, the definition for a directed graph would be $h_i = \{i\} \cup \{j : (j,i) \in E\}$.

In Section 3, we will rely on the formalism given in this section to derive a polynomial time algorithm which provides the solution to EDEN($\mathbf{A}$,ϕ) in all but rare circumstances.

3. The algorithm

In this section, we present a detailed derivation of our *Eden detection* algorithm, denoted by EDEN-DET($\mathbf{A}, \phi$). There is a number of steps involved in this derivation. Firstly, some new mathematical constructions are defined. Then the fundamental operation of EDEN-DET($\mathbf{A}, \phi$), the *neighborhood elimination* operation, denoted by NH-ELIM($\mathbf{A}, H$), is derived. After presenting NH-ELIM($\mathbf{A}, H$) a *simple Eden detection* algorithm is provided, denoted by S-EDEN-DET($\mathbf{A}, \phi$). Using S-EDEN-DET($\mathbf{A}, \phi$) as a starting point we then derive a two phase construction of EDEN-DET($\mathbf{A}, \phi$).

3.1 Preliminaries

The graph cellular automata formalism given in Section 2.3 is not quite sufficient for us to express our Eden detection algorithm clearly. In this section, we present the definitions and notations that form the mathematical foundations of the algorithm. All definitions, notations, and theorems in this section assume the formalism in Section 2.3 to be given, and hence symbols used from Section 2.3 will not be re-defined.

We will assume, without loss of generality, that $\forall i \in V, |h_i| = k$. This is done purely for notational convenience. All of the concepts applied in the construction of our algorithm can be extended trivially to non-uniform $|h_i|$. Note that we do not assume a uniform update rule across all cells $\forall i, j \in V, g_i = g_j$.

To describe our algorithm, we need a method of consistently referring to a specific neighborhood configuration (see Section 3.2). The notation for this reference is given in the following definition.

Definition 6. Assume that some ordering scheme has been applied to the set of all neighborhood configurations Σ^k. Subject to this ordering, the nth neighborhood configuration is denoted by $\psi_n \in \Sigma^k$.

Note that the actual ordering scheme is arbitrary, all we require is an index into the possible neighborhood configuration space. For our implementation we simply map each configuration to its raw binary representation.

It is necessary for us to specify a set that contains all the cells that join adjacent neighborhoods. We refer to this set using the notation ${}^i\Xi^j$, and it is defined in Definition 7.

Definition 7. Let ${}^i\Xi^j = \{i\} \cup \{j\} \cup \{x : ((x,i) \in E \vee (i,x) \in E) \wedge ((x,j) \in E \vee (j,x) \in E)\}$ denote the *boundary* set of h_i and h_j. The

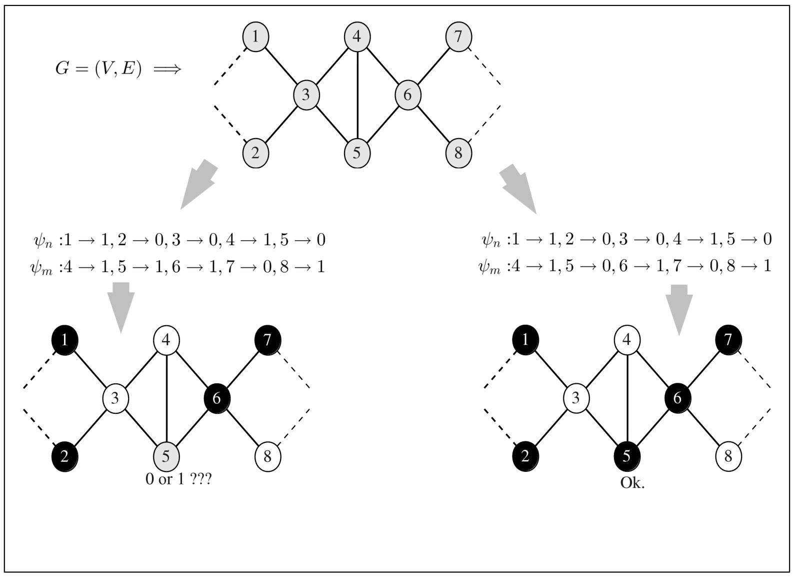
$G = (V, E) \implies$
1
2
3
4
5
6
7
8
$\psi_n : 1 \to 1, 2 \to 0, 3 \to 0, 4 \to 1, 5 \to 0$
$\psi_m : 4 \to 1, 5 \to 1, 6 \to 1, 7 \to 0, 8 \to 1$
0 or 1 ???
$\psi_n : 1 \to 1, 2 \to 0, 3 \to 0, 4 \to 1, 5 \to 0$
$\psi_m : 4 \to 1, 5 \to 0, 6 \to 1, 7 \to 0, 8 \to 1$
Ok.

Figure 1. Example of i, j-consistency where $i = 3$, $j = 6$, and ${}^i\Xi^j = \{3, 4, 5, 6\}$. LEFT: ψ_n is not i, j-consistent with respect to ψ_m since they cause an inconsistent state in the boundary set (i.e., cell 5). RIGHT: A modification to ψ_m allows consistency across the boundary set, hence ψ_n is now i, j-consistent to ψ_m.

nth boundary cell, $x \in V$, is denoted by $x = {}^i\Xi^j_n$.

The basis of our algorithm is the detection and removal of neighborhood configurations which cannot exist in any pre-image of ϕ^t due to an inconsistency across boundary sets.

Definition 8. If there exists an initial configuration ϕ^0 such that $C^0(h_i) = \psi_n$ and $C^0(h_j) = \psi_m$, then ψ_n is said to be i, j-*consistent* with respect to ψ_m.

The concept of i, j-consistency is readily visualized as shown in Figure 1. However, it would be preferable if a direct method of evaluating the i, j-consistency of two neighborhood configurations could be found. The function we require is given in Definition 9.

Definition 9. Let $\theta_i^{{}^i\Xi^j} : \Sigma^k \to \Sigma^{|{}^i\Xi^j|}$ be a function which maps neighborhood configurations of h_i to the configuration of the boundary set ${}^i\Xi^j$. The function is defined as $\theta_i^{{}^i\Xi^j} = \{(\psi_n, \psi'_n) : \psi'_{n,s} = \psi_{n,q} \wedge y = h_{i,q} \wedge y = {}^i\ \Xi^j_s \wedge s \in [0, {}^i\Xi^j)\}$.

The definition of θ given in Definition 9 may seem strange, but it leads us into Theorem 1 which is a vital component of our algorithm.

Theorem 1. If $\theta_i^{i\Xi^j}(\psi_n) = \theta_j^{i\Xi^j}(\psi_m)$ then ψ_n is i,j-consistent with respect to ψ_m.

A formal proof is given in Appendix A.

3.2 Neighborhood elimination

We can now formulate the core operation of our Eden detection algorithm. This core operation we call *neighborhood elimination* and denote it as NH-ELIM($\mathbf{A}$,H). As the name may suggest, its function is to eliminate neighborhood configurations which cannot be a component of any pre-image of the automaton configuration in question.

To explain how we perform this operation, we first consider the matrix $H \in \{0,1\}^{|\Sigma|^k \times |V|}$ where

$$H_{i,j} = \begin{cases} 0, & \text{impossible for } \psi_i = C^0(h_j) \\ 1, & \text{otherwise} \end{cases}. \tag{1}$$

It is important to note in Equation (1), that $H_{i,j} = 1$ should not be interpreted as $\psi_i = C^0(h_j)$ in at least one pre-image. Instead, $H_{i,j} = 1$ means we cannot yet determine if $\psi_i = C^0(h_j)$ or not. This is not the case for $H_{i,j} = 0$, which indicates that we have proven that there is no pre-image such that $\psi_i = C^0(h_j)$.

The algorithm can be described as follows: Consider the case in which we have already determined that $H_{i,j} = 0$ for specific i, j by techniques described in Section 3.3. If we start with an arbitrary cell neighborhood h_i then the column vector $H_{*,i}$ provides us with the neighborhood configurations still under consideration. If $H_{n,i} = 1$, but the neighborhood configuration ψ_n is not i,j-consistent with respect to *any* candidate configurations in one or more connected neighborhoods h_j, then ψ_n can be excluded from the realm of possible configurations for h_i as at least one boundary cell state cannot be satisfied consistently. By updating $H_{*,i}$ this will affect the validity of other configurations, so we repeat the process for every neighborhood.

Theorem 1 provides us with a comparison operation for testing the i,j-consistency of two neighborhood configurations. With the function $\theta_i^{i\Xi^j}$ as defined in Section 3.1, we arrive at NH-ELIM($\mathbf{A}, H$) (i.e., Algorithm 1).

One step of NH-ELIM($\mathbf{A}, H$) is shown in Figure 2 displaying contents of the data structures Θ_i,Θ_j, and ζ_i along with the effect on the state of H. It should be noted that although the example in Figure 2 is for a small 1-d cellular automaton with $k = 3$, NH-ELIM($\mathbf{A}, H$) is general enough to operate on graph cellular automata.

One particularly useful property of NH-ELIM($\mathbf{A}, H$) is that the number of zero elements in H can never decrease. Therefore, repeating NH-ELIM($\mathbf{A}, H$) on H in an iterative fashion will eventually result in

Algorithm 1 NH-ELIM($\mathbf{A}$,H): Neighborhood elimination.

```
for all $i \in V$ do
  for all $j \in h_i - \{i\}$ do
    $\Theta_i \leftarrow \left\{x : x \in \theta_i^{i\Xi j}(\psi_p) \wedge H_{p,i} = 1\right\}$
    $\Theta_j \leftarrow \left\{x : x \in \theta_i^{i\Xi j}(\psi_q) \wedge H_{q,j} = 1\right\}$
    $\zeta_i \leftarrow \{x : x \in \Theta_i \wedge x \notin \Theta_j\}$
    $\forall p, H_{p,i} \leftarrow 0$ if $\theta_i^{i\Xi j}(\psi_p) \in \zeta_i$
  end for
end for
```

$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

$$H = \begin{array}{c} \begin{array}{cccc} & k & k+1 & \end{array} \\ \left[\begin{array}{cccc} ... & 0 & 0 & ... \\ ... & 1 & 0 & ... \\ ... & 0 & 1 & ... \\ ... & 1 & 1 & ... \\ ... & 0 & 1 & ... \\ ... & 0 & 0 & ... \\ ... & 0 & 0 & ... \\ ... & 0 & 0 & ... \end{array}\right] \end{array}$$

$$\Theta_i \leftarrow \left\{x : x \in \theta_i^{i\Xi j}(\psi_p) \wedge H_{p,i} = 1\right\}$$

$$\Theta_i = \{01, 11\}$$

$$\Theta_j \leftarrow \left\{x : x \in \theta_i^{i\Xi j}(\psi_q) \wedge H_{q,j} = 1\right\}$$

$$\Theta_j = \{01, 10\}$$

$$\zeta_i \leftarrow \{x : x \in \Theta_i \wedge x \notin \Theta_j\}$$

$$\zeta_i = \{11\}$$

$$H_{p,i} \leftarrow 0 \; \forall p, (\theta_i^{i\Xi j} \in \zeta_i)$$

$$H_{4,k} = 0$$

$$H' = \left[\begin{array}{cccc} ... & 0 & 0 & ... \\ ... & 1 & 0 & ... \\ ... & 0 & 1 & ... \\ ... & 0 & 1 & ... \\ ... & 0 & 1 & ... \\ ... & 0 & 0 & ... \\ ... & 0 & 0 & ... \\ ... & 0 & 0 & ... \end{array}\right]$$

Figure 2. One Step of NH-ELIM($\mathbf{A}, H$). The upper left matrix is H at the start of the iteration, after the iteration is completed $H_{4,k}$ is set to 0, resulting in the lower left matrix H'.

an array H in which only configurations i, j-consistent with respect to all neighbors are candidates for pre-image construction. This property also enables us to put an upper bound on the number of iterations required, which aids us in our complexity analysis (see Section 4).

3.3 Garden of Eden detection

In this section, we will describe our Eden detection algorithm (EDEN-DET($\mathbf{A}$,ϕ)) in full. Throughout this description we rely heavily on the formalism in Section 2.3 and Section 3.1.

So far we have assumed that H is not all ones or all zeros, but we have not mentioned how H is initialized. If we are given an instance of EDEN($\mathbf{A}$,ϕ), we can prove the impossibility of some neighborhood configurations explicitly by using ϕ and the state transition functions $g_i \in \Gamma$, that is,

$$H_{i,j} = \begin{cases} 0, & g_j(\psi_i) \neq \phi_j \\ 1, & g_j(\psi_i) = \phi_j \end{cases}. \tag{2}$$

In Section 3.2, it was stated for NH-ELIM($\mathbf{A}, H$) (Algorithm 1) that the number of zeros in H can never decrease. Therefore, repeated invoking of NH-ELIM($\mathbf{A}, H$) is guaranteed to converge to a steady state.

Once H is initialized, we can repeatedly operate the neighborhood elimination algorithm on H. Clearly, if for any column $\forall i, H_{i,j} = 0$ during a iteration, then there is no possible ψ_i that can be selected for h_j in any pre-image. Furthermore, the steady state that H will converge to in this case will be $\forall i, j, H_{i,j} = 0$. Therefore we can conclude that ϕ is a Garden of Eden configuration.

We might also assume that all Garden of Eden configurations will cause the condition, $\forall i, H_{i,j} = 0$. Therefore, we could simply iterate until a steady state is reached and then look at the elements in H for any non-zero elements. This leads us to derive our initial algorithm for Eden detection, which we call *simple eden detection* and denote as S-EDEN-DET($\mathbf{A}, \phi$) (Algorithm 2).

Algorithm 2 S-EDEN-DET($\mathbf{A}$,ϕ): Simple Eden detection.

```
$\forall i, j, H_{i,j} \leftarrow 0$ if $i, j, (g_j(\psi_i) \neq \phi_j)$
$\forall i, j, H_{i,j} \leftarrow 1$ if $i, j, (g_j(\psi_i) = \phi_j)$
while $H \neq H'$ do
    $H' \leftarrow H$
    $H \leftarrow$ NH-ELIM$(\mathbf{A}, H')$
end while
if $\forall i, j, H_{i,j} = 0$ then
    $GoE \leftarrow true$
else
    $GoE \leftarrow false$
end if
return $GoE$
```

Unfortunately, S-EDEN-DET($\mathbf{A}, \phi$) is not quite complete[2]. It can be shown that $\forall i, H_{i,j} = 0$ is a *sufficient* but not *necessary* condition of Eden. It is possible for cells within a cycle of G to have i, j-consistent neighbors, but there does not exist a combination of possible neighborhood configurations that can form a consistent chain. Figure 3 gives an example of such a case, note that for a 1-d cellular automaton the topology graph G contains one cycle which includes all cells. Clearly, more processing is required once S-EDEN-DET($\mathbf{A}, \phi$) has converged and

S-EDEN-DET($\mathbf{A}_{30}$,$\{0,0,1,1\}$)

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

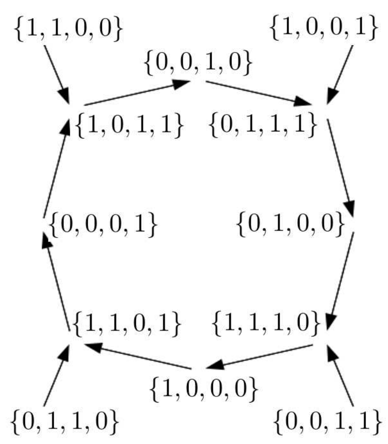

Figure 3. Counter example for S-EDEN-DET($\mathbf{A}, \phi$) (Algorithm 2). LEFT: The resulting non-zero steady state of H where $\mathbf{A}_{30}$ is the elementary cellular automaton *rule 30* (according to Wolfram's numbering scheme [16]) with periodic boundary conditions and $|V| = 4$. RIGHT: The main configuration transition graph for A_{30}, clearly $\phi = \{0, 0, 1, 1\}$ has no pre-image.

there does not exist a $j \in V$ such that $\forall i, H_{i,j} = 0$.

If for any possible neighborhood configuration (i.e., $H_{i,j} = 1$) we can construct at least one pre-image, then we can conclude that ϕ is not a Garden of Eden configuration. However, if it is found that a valid pre-image cannot be constructed with $C^0(h_j) = \psi_i$ then we can set $H_{i,j} = 0$ and repeat S-EDEN-DET($\mathbf{A}, \phi$) until a new steady state is reached. This leads us to a second and more complete approach EDEN-DET($\mathbf{A}, H$).

In practice, we locate each instance of $H_{i,j} = 1$, and temporarily set $\forall k \neq i, H_{k,j} = 0$. This has the effect of *assuming* that $C^0(h_j) = \psi_i$. We then apply one iteration of NH-ELIM($\mathbf{A}, H$) ensuring that cell $j \in V$ is visited last, then we examine the state of $H_{i,j}$. If $H_{i,j} = 1$, then we have no reason to reject our assumption. Otherwise, our assumption is disproved via contradiction, so we set $H_{i,j} = 0$ and repeat the loop in S-EDEN-DET($\mathbf{A}, \phi$). If none of the $H_{i,j} = 1$ can be disproved, then it is reasonable to conclude that ϕ has at least one pre-image (We show in Section 6 that there are rare cases when this is an invalid conclusion).

We now have a two phase procedure. Phase 1, denoted by PH1($\mathbf{A}, H$)

[2]Hence the name *simple Eden detection.*

(Algorithm 3), is effectively the loop from S-EDEN-DET$(\mathbf{A}, \phi)$. Phase 2, denoted by PH2$(\mathbf{A}, H)$ (Algorithm 4), is the assumption testing process described in the preceding paragraph. These two phases are then combined to form our full *Eden detection* algorithm EDEN-DET$(\mathbf{A},\phi)$ (Algorithm 5). A implementation of EDEN-DET$(\mathbf{A}, \phi)$ is provided as part of analysis software developed by Warne [12][3].

Algorithm 3 PH1$(\mathbf{A},H)$: Eden detection phase 1.

```
while $H \neq H'$ do
    $H' \leftarrow H$
    $H \leftarrow$ NH-ELIM$(\mathbf{A}, H')$
end while
```

Algorithm 4 PH2$(\mathbf{A},H)$: Eden detection phase 2.

```
for all $i \in V$ do
    for all $j \in \Sigma^{|k|}$ do
        if $H_{i,j} = 1$ then
            $H^{tmp} \leftarrow H$
            $\forall s, (s \neq j), H^{tmp}_{i,s} \leftarrow 0$
            $H^{tmp} \leftarrow$ NH-ELIM$(\mathbf{A}, H^{tmp})$
            if $H^{tmp}_{i,j} = 0$ then
                $H_{i,j} \leftarrow 0$
                return
            end if
        end if
    end for
end for
```

Leaving the details to Section 4, we simply claim that EDEN-DET$(\mathbf{A}, \phi)$ is guaranteed to complete in polynomial time. More specifically, it can be shown to have a cubic worst case time efficiency. Furthermore, when EDEN-DET$(\mathbf{A}, \phi)$ returns $GoE = false$, then H encodes the complete set of pre-images to ϕ^t (except for rare cases when $GoE = false$ is a false negative as shown in Section 6).

4. Time complexity analysis

In this section, we present the time complexity analysis for EDEN-DET$(\mathbf{A},\phi)$ (Algorithm 5). We show that the number of operations for the best case is a linear function of the number of cells, the worst case is shown to be cubic, and the average case is shown to be quadratic.

[3]This software, called `GCALab`, is an command line analysis tool designed for parallel computation of graph cellular automata properties.

Algorithm 5 EDEN-DET($\mathbf{A}$,ϕ): Eden detection.

```
$\forall i, j, H_{i,j} \leftarrow 0$ if $i, j, (g_j(\psi_i) \neq \phi_j)$
$\forall i, j, H_{i,j} \leftarrow 1$ if $i, j, (g_j(\psi_i) = \phi_j)$
repeat
  CALL PH1($\mathbf{A}$,$H$)
  if $\forall i, j, H_{i,j} = 0$ then
    $GoE \leftarrow true$
    return $GoE$
  else
    CALL PH2($\mathbf{A}$,$H$)
    $GoE \leftarrow false$
  end if
until $H' = H$
return $GoE$
```

Experimental results are also presented to reinforce theory with practice.

4.1 Time complexity of NH-ELIM($\mathbf{A}, H$)

The fundamental operation of EDEN-DET($\mathbf{A}$,ϕ) is cleary NH-ELIM($\mathbf{A}$,H). From the pseudo code for NH-ELIM($\mathbf{A}, H$) (Algorithm 1), it is also clear that the number of operations executed by NH-ELIM($\mathbf{A}$,H) is a function of the number of cells $n = |V|$. We will show that this operation is in $\Theta(n)$.

The four lines within the innermost loop of NH-ELIM($\mathbf{A}, H$) are only dependent on the number of neighborhood configurations. Without loss of generality, we assume $\forall i, |h_i| = k$, thus the construction of Θ_i and Θ_j require searching only a single column of H. That is, $C_\Theta = c_0|\Sigma^k|$ where $c_0 \approx k$ is the number of operations to evaluate $\theta_i^{|^i\Xi^j|}$. The construction of ζ_i is dependent only on the size of the Θ's, hence $C_\zeta \leq C_\Theta$. Furthermore, the number of elements in H is equal the number of elements in $\zeta_i \leq |\Sigma^k|$. Thus the total operation count within the inner loop is given by,

$$C_{inner} = 2C_\Theta + C_\zeta + |\zeta| \approx 3|\Sigma^k|. \tag{3}$$

Given Equation (3) we can derive the total operation count for NH-ELIM($\mathbf{A}, H$).

$$\begin{aligned} C_{NH}(n) &= \sum_{i=1}^{n}\sum_{j=1}^{k} C_{inner} \\ &= kC_{inner}n \\ &\approx 3k|\Sigma^k|n. \end{aligned} \tag{4}$$

Therefore $C_{NH}(n) \in \Theta(n)$.

4.2 Best case

We now consider the best case time complexity of EDEN-DET($\mathbf{A}$,ϕ). The best case occurs when there exist few possible i, j-consistent pairs for some sub-sequence in ϕ. This is very common in cellular automata in which Langton's λ [6] is small. An example of this is when $\mathbf{A}$ is the elementary cellular automaton rule 2, and ϕ has a contiguous sequence of 1's.

In this special case, only PH1($\mathbf{A}, H$) (Algorithm 3) will be required. Furthermore a column of zeros will developed very quickly as each iteration will eliminate at least one possible configuration from the unnatural area (due to few or no i, j-consistent neighborhood pairs), that is $I < |\Sigma^k|$ where I is the number of iterations of the while loop. Using the results from Equation (4) we have,

$$\begin{aligned} C_{best}(n) &= \sum_{i=1}^{|\Sigma^k|} C_{NH}(n) && (5)\\ &= |\Sigma^k| C_{NH}(n) \\ &\approx 3k|\Sigma^k|^2 n \end{aligned}$$

Therefore $C_{best} \in \Omega(n)$.

4.3 Worst case

For the worst case we must consider the full expression for the number of operations executed by EDEN-DET($\mathbf{A}$,ϕ). This is given by,

$$C_{ops}(n) = \sum_{t=1}^{J} \left(\underbrace{\sum_{i=1}^{I} C_{NH}(n)}_{\text{PH1}(\mathbf{A},H)} + \underbrace{\sum_{j=1}^{|V|} \sum_{i=1}^{|\Sigma^k|} C_{NH}(n)}_{\text{PH2}(\mathbf{A},H)} \right) \qquad (6)$$

where J and I simply denote the number of iterations taken by the conditional loops. We require an upper bound on these loops.

In Section 3.3 we noted that the number of 0's in H can never decrease. Now we also note that if the number of 0's in H does not increase after an execution of PH2($\mathbf{A}, H$) (Algorithm 4) then EDEN-DET($\mathbf{A}$,ϕ) terminates with $GoE = false$. Hence for the worst case we must assume that the number of 0's decreases by exactly one. Furthermore, every iteration of PH1($\mathbf{A}, H$) will either increase the number of 0's, terminate EDEN-DET($\mathbf{A}$,ϕ) with $GoE = true$, or continue to an iteration of PH2($\mathbf{A}, H$). Since $H \in \{0,1\}^{|\Sigma^k| \times n}$, it must hold that

$$J\,(I+1) \leq |\Sigma^k| n. \qquad (7)$$

We want to maximize the value of J as it has the greater effect on the total number calls to NH-ELIM($\mathbf{A}$,H). If we assume the upper bound is reachable, then as $I \to 1$ we have $J \to \frac{|\Sigma^k|}{2}n$. We can apply this result to Equation (6) to obtain an upper bound on $C_{ops}(n)$,

$$\begin{aligned} C_{ops}(n) &\leq \sum_{t=1}^{\frac{|\Sigma^k|}{2}n} \left(C_{NH}(n) + \sum_{j=1}^{|V|} \sum_{i=1}^{|\Sigma^k|} C_{NH}(n) \right) \\ &= \frac{|\Sigma^k|}{2} n \left(C_{NH}(n) + |V||\Sigma^k| C_{NH}(n) \right) \\ &= \frac{|\Sigma^k|}{2} \left(|\Sigma^k| n^2 + n \right) C_{NH}(n). \end{aligned}$$

Furthermore, we have already shown that $C_{NH} \in \Theta(n)$. Therefore $C_{worst} \in O(n^3)$.

4.4 Average case

Best and worst case bounds are important but of limited practical use without an indication of the likelihood of EDEN($\mathbf{A}$,ϕ) instances which cause these bounds to occur. In this section we will show, using empirical data, that the average case is quadratic in time.

Consider Equation (6), the values affecting the computational complexity are the number of iterations taken by the guard loops and whether PH2($\mathbf{A}, H$) needs to be executed. As in Section 4.3, we will denote the number of outer loops as J and the number of PH1($\mathbf{A}, H$) loops as I. Furthermore, we denote the number of iterations in which PH2($\mathbf{A}, H$) is executed as K.

We took random EDEN($\mathbf{A}$,ϕ) instances for $|V| = n = 2^i, 2 \leq i \leq 13$, where G is a single circuit. For each value of n over 1000 samples were taken. The values of I,J, and K were counted for each sample. The expected values computed from these samples are shown in Figure 4

From Figure 4 we can derive the overall expected values $E(I) = 2.88$, $E(J) = 1.06$, and $E(K) = 0.25$. So it is reasonable to approximate the average case as follows,

$$\begin{aligned} C_{average}(n) \approx& \left(\sum_{i=1}^{3} C_{NH}(n) \right) \times \Pr(\neg K) \\ &+ \left(\sum_{i=1}^{3} C_{NH}(n) + \sum_{j=1}^{|V|} \sum_{i=1}^{|\Sigma^k|} C_{NH}(n) \right) \times \Pr(K) \\ =& (3C_{NH}(n)) \times \frac{3}{4} + \left(3C_{NH}(n) + |\Sigma^k| n C_{NH}(n) \right) \times \frac{1}{4} \\ =& \frac{|\Sigma^k|}{4} n C_{NH}(n) + 3C_{NH}(n). \end{aligned}$$

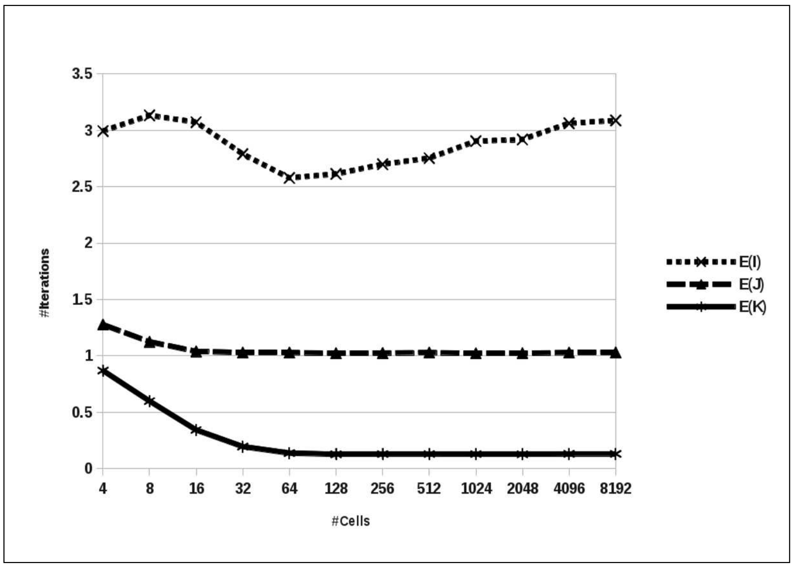

Figure 4. Expected Iteration Values. Based on 1000 random samples for each number of cells.

Since $C_{NH}(n) \in \Theta(n)$, the approximate overall expected time complexity is $C_{average}(n) \in \Theta(n^2)$. Section 4.5 provides further experimentation to validate this approximation.

4.5 Experimental results

For validation of the average case we took a new random sample of 1000 instances of EDEN($\mathbf{A}$,ϕ) for $|V| = n = 2^i, 2 \leq i \leq 13$. For each sample the average runtime of 5 separate runs was taken. Results were separated into two groups based on whether EDEN-DET($\mathbf{A}$,ϕ) returned with $GoE = true$ or $GoE = false$. The resulting average run times are shown in Figure 5.

Note that on average the runtime when $GoE = false$ is approximately 16 times the runtime when $GoE = true$. This is because only a Garden-of-Eden configuration ϕ_e can cause EDEN-DET($\mathbf{A}$,ϕ_e) to return before Phase 2 is executed, which will complete in $O(n)$ operations.

To confirm that the curves in Figure 5 are in fact quadratic, we can take the ratio $R = \frac{C(2^{i+1})}{C(2^i)}$ where $C(n)$ is the average runtime as a function of the number of cells n. We would expect $R \approx 4$ for a quadratic (i.e., doubling the input takes 4 times longer). Figure 6 shows the R for there samples taken for Figure 5.

From Figure 6 it is clear that $R \approx 4$ (Considering that $R \approx 2$ for linear and $R \approx 8$ for cubic). This provides support for our approximate average time complexity for EDEN-DET($\mathbf{A}$,ϕ) that we provided in

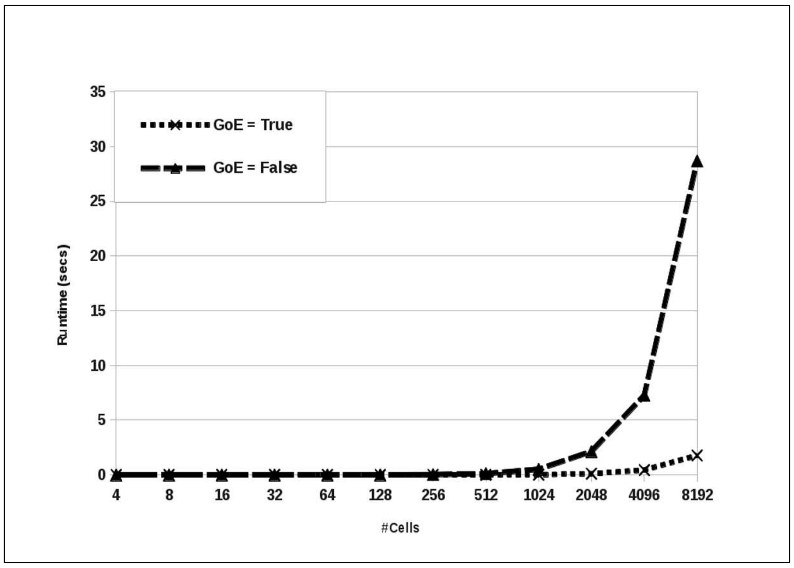

Figure 5. Run times for EDEN-DET (Algorithm 5).

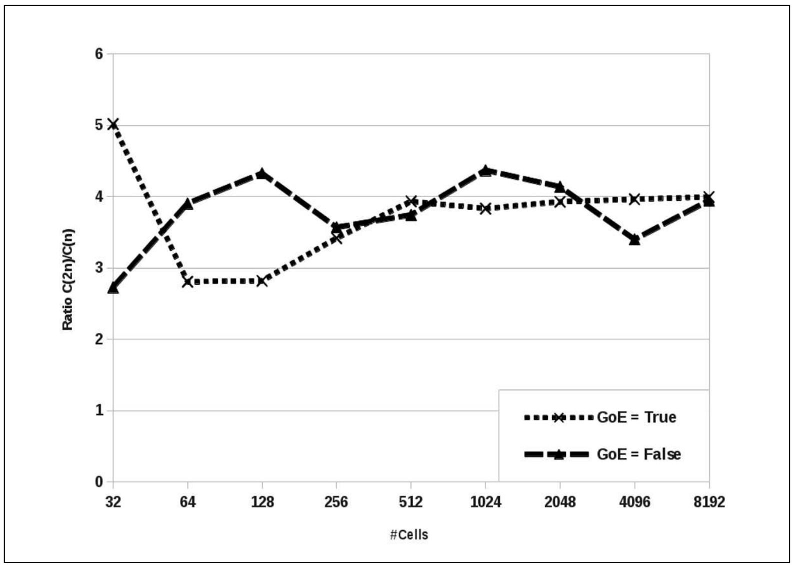

Figure 6. Ratio of run times $C(2n)/C(n)$).

Section 4.4.

5. Comparison with Wuensche and Lesser's reverse algorithm

In this section, we will compare the performance of EDEN-DET against the reverse algorithm developed by Wuensche and Lesser [20]. For the sake of simplicity, we will restrict the comparison to the simplest form of cellular automata; that of finite elementary cellular automata with periodic boundary condition. As a result we must emphasize that the following discussion and analysis relates specifically to Wuensche and Lesser's one dimensional reverse algorithm [20] and not Wuensche's more general reverse algorithm which applies to random boolean networks and graph cellular automata [19, 17]. The results of this analysis, however, can certainly be generalized to the graph cellular automata case.

For a finite elementary cellular automata with periodic boundary conditions $\mathbf{A}$, a configuration ϕ^t and a partial pre-image ϕ^{t-1} in which the first i cell states are known, Wuensche and Lesser's method is described as follows [20]:

1. If $g(\phi^{t-1}_{i-1}, \phi^{t-1}_i, 0) = g(\phi^t_{i-1} - 1, \phi^t_i, 1) \neq \phi^t_i$, then abandon the partial pre-image. Resume derivation of next partial pre-image (go to step 5).
2. If $g(\phi^{t-1}_{i-1}, \phi^{t-1}_i, 0) \neq g(\phi^t_{i-1} - 1, \phi^t_i, 1)$, then ϕ^{t-1}_{i+1} can be uniquely determined. Proceed with next cell (go to step 1).
3. If $g(\phi^{t-1}_{i-1}, \phi^{t-1}_i, 0) = g(\phi^t_{i-1} - 1, \phi^t_i, 1) = \phi^t_i$, then ϕ^{t-1}_{i+1} could be 0 or 1. Push the partial pre-image $(\phi^{t-1}_0, \phi^{t-1}_1, ..., \phi^{t-1}_i, 1)$ onto the pre-image queue to be processed later and continue with $\phi^{t-1}_{i+1} = 0$.
4. When $i = n - 1$ check that $g(\phi^{t-1}_{n-2}, \phi^{t-1}_{n-1}, \phi^{t-1}_0) \neq g(\phi^{t-1}_{n-1}, \phi^{t-1}_0, \phi^{t-1}_1)$ then abandon this pre-image, otherwise add to the valid pre-image list.
5. Take a new partial pre-image from the queue and continue processing (step 1).
6. When the partial pre-image queue is empty, all possible pre-images starting with the start values of $\phi^{t-1}_0, \phi^{t-1}_1$ are derived. Repeat for all possible $\phi^{t-1}_0, \phi^{t-1}_1$.

Note that the primary purpose of Wuensche and Lesser's method is the construction of all valid pre-images, but it can be utilised directly to compute the solution to the Eden problem. Clearly, we can assert $GoE = false$ as soon as a valid pre-image is found. We need not compute all of them. $GoE = true$ will be asserted when not pre-images are found.

5.1 Worst case complexity analysis

With a brief description of Wuensche and Lesser's one dimensional reverse algorithm, we can now show that the worst case computation time is not bounded by a polynomial in the number of cells n. Consider Algorithm 6 which depicts Wuensche and Lesser's method modified for solving the Eden problem without computing all pre-images.

Algorithm 6 REVERSE($\mathbf{A}$,ϕ): Wuensche and Lesser's Reverse Algorithm.

```
$GoE \leftarrow true$
for all $(p_1, p_2) \in \{(0,0),(0,1),(1,0),(1,1)\}$ do
  $\phi^{t-1} \leftarrow (p_1, p_2)$
  $Q \leftarrow \{\phi^{t-1}\}$
  while $Q \neq \{\}$ do
    $\phi^{t-1} \leftarrow pop(Q)$
    $x = |\phi^{t-1}| - 1$
    for all $i \in [x, n]$ do
      $T_0 \leftarrow g(\phi^{t-1}_{i-1}, \phi^{t-1}_i, 0)$
      $T_1 \leftarrow g(\phi^{t-1}_{i-1}, \phi^{t-1}_i, 1)$
      if $T_0 = T_1 \neq \phi^t_i$ then
        break for loop
      else
        if $T_0 \neq T_1$ then
          if $T_0 = \phi^{t-1}_i$ then
            $\phi^{t-1} \leftarrow \phi^{t-1} \cup \{0\}$
          else
            $\phi^{t-1} \leftarrow \phi^{t-1} \cup \{1\}$
          end if
        else
          $push(Q, \phi^{t-1} \cup \{1\})$
          $\phi^{t-1} \leftarrow \phi^{t-1} \cup \{0\}$
        end if
      end if
    end for
    $T_n \leftarrow g(\phi^{t-1}_{n-1}, \phi^{t-1}_n, \phi^{t-1}_1)$
    $T_1 \leftarrow g(\phi^{t-1}_n, \phi^{t-1}_1, \phi^{t-1}_2)$
    if $T_n = T_1$ then
      $GoE \leftarrow false$
      return $GoE$
    end if
  end while
end for
return $GoE$
```

Let C_{inner} denote the number of operations performed on a single iteration of the innermost loop in REVERSE($\mathbf{A}$,ϕ). Without loss of generality, we will assume C_{inner} is a constant[4].

Now, let $C_x(n)$ denote the number of operations required to complete ignoring partial pre-images already in Q before reaching the x-th cell in the current pre-image. We can express $C_x(n)$ as the following recurrence relation,

$$C_x(n) = \sum_{i=x}^{n} C_{inner} + e(x) \sum_{i=x+1}^{n} a(i) C_i(n)$$

where $e(x) = 0$ if $T_0 = T_1 \neq \phi_x^{t-i}$ otherwise $e(x) = 1$, and $a(x) = 1$ if $T_0 = T_1 = \phi_x^{t-1}$ otherwise $a(x) = 0$.

The worst case for $C_x(n)$ occurs when the number of partial pre-images being pushed onto the queue is every iteration. In this case, we have $\forall i > x, (a(i) = 1 \wedge e(i) = 1)$ and the recurrence relation becomes,

$$C_x(n) = \sum_{i=x}^{n} C_{inner} + \sum_{i=x+1}^{n} C_i(n).$$

We can now solve this recurrence relation. First consider expanding the $C_{x+1}(n)$ term in the summation,

$$\begin{aligned}
C_x(n) =& \sum_{i=x}^{n} C_{inner} + \sum_{i=x+1}^{n} C_i(n) \\
=& \sum_{i=x}^{n} C_{inner} + C_{x+1}(n) + \sum_{i=x+2}^{n} C_i(n) \\
=& C_{inner} + \sum_{i=x+1}^{n} C_{inner} + \left(\sum_{i=x+1}^{n} C_{inner} + \sum_{i=x+2}^{n} C_i(n) \right) \\
& + \sum_{i=x+2}^{n} C_i(n) \\
=& \sum_{i=x}^{n} C_{inner} + \left(\sum_{i=x+1}^{n} C_{inner} + \sum_{i=x+2}^{n} C_i(n) \right) + \sum_{i=x+2}^{n} C_i(n) \\
=& C_{inner} + 2 \sum_{i=x+1}^{n} C_{inner} + 2 \sum_{i=x+2}^{n} C_i(n).
\end{aligned}$$

[4]Clearly this is not true in reality, but instead $3 \leq C_{inner} \leq 6$

Now, expending the $C_{x+2}(n)$ term,

$$\begin{aligned}
C_x(n) =& C_{inner} + 2\sum_{i=x+1}^{n} C_{inner} + 2\sum_{i=x+2}^{n} C_i(n) \\
=& C_{inner} + 2\sum_{i=x+1}^{n} C_{inner} + 2C_{x+2}(n) + 2\sum_{i=x+3}^{n} C_i(n) \\
=& C_{inner} + 2C_{inner} + 2\sum_{i=x+2}^{n} C_{inner} \\
&+ 2\left(\sum_{i=x+2}^{n} C_{inner} + \sum_{x+3}^{m} C_i(n)\right) + 2\sum_{i=x+3}^{n} C_i(n) \\
=& C_{inner} + 2C_{inner} + 4\sum_{i=x+2}^{n} C_{inner} + 4\sum_{i=x+3}^{n} C_i(n).
\end{aligned}$$

Repeating this process yields,

$$\begin{aligned}
C_x(n) &= C_{inner} + 2C_{inner} + 4C_{inner} + ... + 2^{n-x}C_{inner} \\
&= C_{inner}\sum_{i=x}^{n} 2^{i-x}.
\end{aligned}$$

The worst case for REVERSE($\mathbf{A}$,ϕ) requires that $C_2(n)$ operations be executed four times,

$$\begin{aligned}
C_{worst} =& 4C_2(n) \\
=& 4C_{inner}\sum_{i=2}^{n} 2^{i-2} \\
=& C_{inner}\sum_{i=2}^{n} 2^{i}
\end{aligned}$$

hence REVERSE($\mathbf{A}$,ϕ) is in $O(2^n)$. It is worth noting that this worst case can only be achieved if the ϕ is a Garden-of-Eden configuration, and the cell which determines this is the n-th cell. For example, $\phi = (0, 0, ..., 0, 1, 1)$ for the elementary cellular automaton *rule 2*. However, according to Wuensche and Lesser [20] the average case is orders of magnitude better. We confirm this experimentally in Section 5.2.

5.2 Experimental comparison

We benchmarked EDEN-DET($\mathbf{A}$,ϕ) against REVERSE($\mathbf{A}$,ϕ). Each experiment consisted of solving the Eden problem for 1000 random configurations. Experiments were performed for both EDEN-DET($\mathbf{A}$,ϕ) and

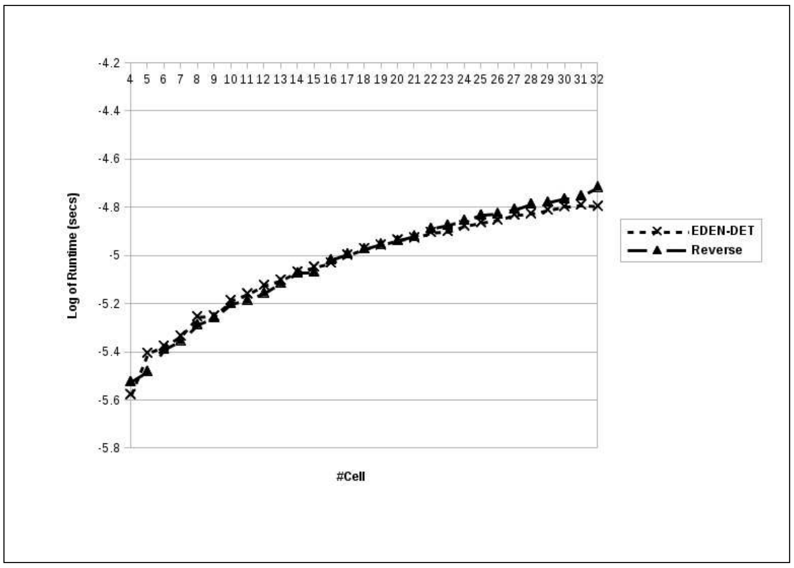

Figure 7. Comparison of EDEN-DET($\mathbf{A}$,ϕ) against REVERSE($\mathbf{A}$,ϕ) using a random sampling of configurations.

REVERSE($\mathbf{A}$,ϕ) using all the elementary cellular automata with cell counts ranging from 4 to 32. As shown in Figure 7, the benchmark average case is effectively the same order of magnitude for both methods.

The worst case for REVERSE($\mathbf{A}$,ϕ) is only approached for Garden-of-Eden configurations which is nearly identical to a non Garden-of-Eden configuration only differing in the last few cells. This is more likely to be possible with sparse configurations (i.e., very few 1 states compared with 0 states). If we restrict the random sample of test configurations to that of sparse configurations, then the probability of selecting a configuration which degrades the performance of REVERSE($\mathbf{A}$,ϕ).

Figure 8 indicates that the benchmark results are very different when we restrict the configuration sample this way. Such cases place a limitation on the usability of REVERSE($\mathbf{A}$,ϕ) for large cell counts[5]. The performance of EDEN-DET($\mathbf{A}$,ϕ), however, is hardly affected by such sparse configurations.

The main difference in our approach which provides such a large improvement in the worst case performance is the neighborhood elimination step. This operation performance is not affected by shifts (or rotations in higher dimension) in the same configuration, because it treats each cell neighborhood independently of each other. As a result, EDEN-DET($\mathbf{A}$,ϕ) provides a solution to the Eden problem which is scalable to very large cellular automata. EDEN-DET($\mathbf{A}$,ϕ) could be considered as a more stable alternative to Wuesnche and Lesser's RE-

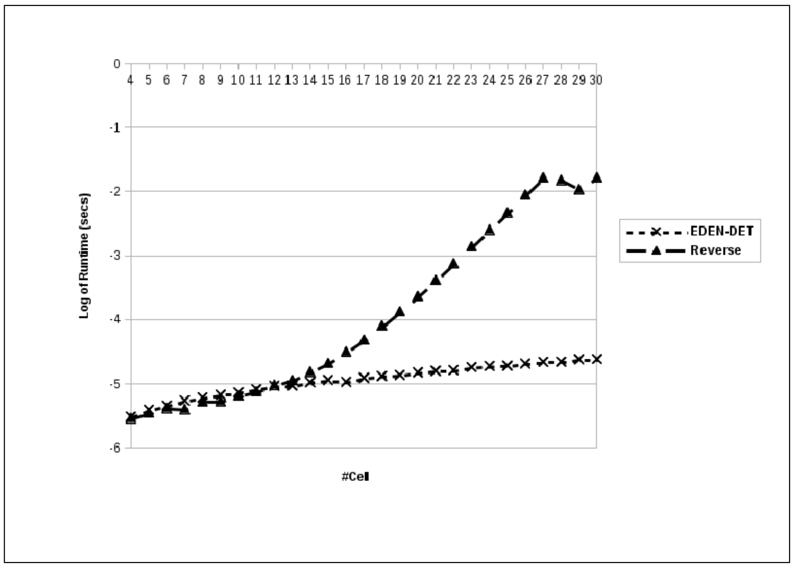

Figure 8. Comparison of EDEN-DET($\mathbf{A}$,ϕ) against REVERSE($\mathbf{A}$,ϕ) using a random sampling of sparse configurations.

VERSE($\mathbf{A}$,ϕ) as the worst case is vastly improved without degrading the average case.

6. Algorithm correctness

In this section, we discuss the correctness of the EDEN-DET($\mathbf{A}$,ϕ) in solving the Eden problem for graph cellular automata. We are able to show that EDEN-DET($\mathbf{A}$,ϕ) is completely correct for graphs with a single cycle. For graphs with more than one cycle it is possible for incorrect results to be returned[6] (i.e., false negatives), however we show that these cases are rare.

It is first worth discussing the correctness of the solution when EDEN-DET($\mathbf{A}$,ϕ) returns with $GoE = true$. This result will never occur if ϕ has a pre-image (i.e., false positives cannot occur). This is because elements in H are only ever set to 0 when there is no i, j-consistent pair in a neighbor cell. If $GoE = true$ is returned then at some point there must have existed an i such that $\forall j, H_{j,i} = 0$ (i.e., a cell has no possible i, j-consistent neighborhood configurations). For ϕ to have a pre-image each cell must have at least one i, j-consistent neighborhood

[5] As the cell count increases any configuration with a relatively small sparse sub-sequence could render the Eden problem computationally intractable for REVERSE($\mathbf{A}$,ϕ)

[6] If this were not so the title of this paper would be "$P = NP$"!

configuration. Therefore only a true Garden-of-Eden configuration can cause $GoE = true$ to be returned.

When EDEN-DET($\mathbf{A}$,ϕ) returns with $GoE = false$ there are possible false identifications. That is, it is possible for a Garden-of-Eden configuration to cause $GoE = false$ to be returned. However, this rare case is only a possibility when the graph G has more than one cycle.

We will now show that EDEN-DET($\mathbf{A}$,ϕ) returning $GoE = false$ is always correct if G contains only one cycle. Consider H which has reached a non-zero steady state after executing PH1($\mathbf{A}$,H). If we assume a neighborhood configuration $H_{j,i} = 1$ (see Section 3.3), and carry out an iteration of NH-ELIM($\mathbf{A}$,H) we are essentially propagating the assumption around the cycle of G. When this propagation returns to i then there are only two possibilities: 1) The assumed $H_{j,i}$ is not eliminated meaning a chain of i, j-consistent pairs can be constructed (i.e., a pre-image can exist under this assumption), and 2) The assumed $H_{j,i}$ is eliminated, hence ψ_j cannot contribute to any pre-image. Since EDEN-DET($\mathbf{A}$,ϕ) only returns $GoE = false$ when every element in H has passed assumption testing, we can conclude this can only occur if ϕ does in fact have a possible pre-image. Therefore EDEN-DET($\mathbf{A}$,ϕ) is completely correct for G with a single cycle.

These correctness results for the single cycle (i.e., 1-d) case have also been supported by experimental results. We executed EDEN-DET($\mathbf{A}$, ϕ) on the entire configuration space for all elementary cellular automata where $n = [4, 8, 16]$. Each return value was validated via a brute force search for a pre-image. This resulted in a 100% success rate.

Unfortunately, things are not so easy for G with multiple cycles. The assumption testing method we apply in PH2($\mathbf{A}$, H) is really only powerful enough to test consistency within a single cycle. It may be possible for every $H_{j,i}$ to pass the assumption test but any choice made from one cycle breaks consistency in another. Hence a complete solution would require looking at pairs of cycles, triples of cycles etc.[7]. This is likely the result of the NP-complete nature of the Eden problem in more than one dimension.

Again, we look to empirical data to show that in the majority of cases the single cycle accuracy is all we need. This time over $170,000$ random instances of EDEN($\mathbf{A}$, ϕ) (for a fixed choice of rules representing Wolfram Classes I,II, and III [15]) were taken as inputs[8]. Every result was compared to a brute force approach.

We found that for Class I cellular automata (i.e., point attractors) no false negatives ever seem to occur. Rules that fall under Class II

[7]Of course we do not have a rigorous proof of this. If we did, the title of this paper would be "$P \neq NP$"!

[8]The topology of the graph G was equivalent to a dodecahedron. Since $|V| = 20$, the complete configuration space is 2^{20}.

(i.e., simple structures, maybe periodic) had a low number of false negatives; around 0.01%. Class III cellular automata (i.e., chaotic) are a different story, with around 16% of cases in which EDEN-DET($\mathbf{A}$,ϕ) returned $GoE = false$ were incorrect[9]. Over all samples, the false negative rate was around 10%.

False negatives can be detected without resorting to a brute force sweep. As previously stated in Section 3.3, the final state of H completely encodes all possible pre-images. Neighborhoods in H can be stitched together using a method similar to Wuensche's general reverse algorithm [19], if no pre-image can be constructed then we have detected a false negative. In light of this, our algorithm could also be considered as a search reduction step to be used prior to invoking Wuensche's general method. Combined, this would provide a completely correct and more efficient method for constructing configuration transition graphs.

7. Conclusion

In this paper we have presented an efficient algorithm (i.e., average case in $\Theta(n^2)$), EDEN-DET($\mathbf{A}$,H), for solving the Eden problem for graph cellular automata. By changing the topology of the graph G, the Eden problem can be solved for all classes of deterministic discrete dynamical systems (e.g., regular cellular automata, and random boolean networks). This analysis provides a firm foundation for further study of the global dynamics of discrete dynamical systems.

Appendix

A. Proof of Theorem 1

Proof. First consider the equality,

$$\theta_i^{{}^i\Xi^j}(\psi_n) = \theta_j^{{}^i\Xi^j}(\psi_m).$$

Given Definition 9, we can expand the above expression. This yields,

$$\begin{aligned}&\equiv \forall s, (\psi'_{n,s} = \psi'_{m,s} \wedge \exists p, (\psi_{n,p} = \psi'_{n,s} \wedge y = h_{i,p} \wedge y = {}^i\Xi^j_s) \\ &\quad \wedge \exists q, (\psi_{m,q} = \psi'_m, s \wedge z = h_{j,q} \wedge z = {}^i\Xi^j_s)).\end{aligned}$$

[9]It is interesting to note that it is only Class III cellular automata that seem to cause PH2($\mathbf{A}$, H) to be executed in the 1-d case.

This can be reduced using predicate calculus,

$$\begin{aligned}
&\equiv \forall s, (\psi'_{n,s} = \psi'_{m,s} \wedge \exists p, (\psi_{n,p} = \psi'_{n,s} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s}) \\
&\quad \wedge \exists q, (\psi_{m,q} = \psi'_{m}, s \wedge h_{j,q} = {}^{i}\Xi^{j}_{s})) \\
&\equiv \forall s, (\psi'_{n,s} = \psi'_{m,s} \wedge \\
&\quad \exists p, q(\psi_{n,p} = \psi'_{n,s} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge \psi_{m,q} = \psi'_{m}, s \wedge h_{j,q} = {}^{i}\Xi^{j}_{s})) \\
&\equiv \forall s, (\exists p, q(\psi'_{n,s} = \psi'_{m,s} \wedge \psi_{n,p} = \psi'_{n,s} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \\
&\quad \wedge \psi_{m,q} = \psi'_{m}, s \wedge h_{j,q} = {}^{i}\Xi^{j}_{s})) \\
&\equiv \forall s, (\exists p, q(\psi_{n,p} = \psi_{m,q} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s}))
\end{aligned}$$

Now let $C^0(h_i) = \psi_n$ and $C^0(h_j) = \psi_m$, hence $C^0(h_{i,p}) = \psi_{n,p}$ and $C^0(h_{j,q}) = \psi_{m,q}$

$$\begin{aligned}
&\models \forall s, (\exists p, q(\psi_{n,p} = \psi_{m,q} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s}) \wedge C^0(h_{i,p}) = \psi_{n,p} \\
&\quad \wedge C^0(h_{j,q}) = \psi_{m,q}) \\
&\equiv \forall s, (\exists p, q(h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s}) \wedge C^0(h_{i,p}) = C^0(h_{j,q}))
\end{aligned}$$

This satisfies our definition of i, j-consistency (i.e., Definition 8). Therefore ψ_n and ψ_m are i, j-consistent. ■

References

[1] S. Bandini, G. Mauri, and R. Serra, "Cellular automata: From a theoretical parallel computational model to its application to complex systems," *Parallel Computing*, **27** (2001) 539–553.

[2] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Streans, and M. Thakur, "Predecessor existence problems for finite discrete dynamical systems," *Theoretical Computer Science*, **386** (2007) 3–37.

[3] M. Cook, "Universality in elementary cellular automata," *Complex Systems*, **15** (2004) 1–40.

[4] N. Fates, "Critical phenomena in cellular automata: Perturbing the update, the transitions, the topology,"*Acta Physica Polonica B, Proceedings Supplement*, **3** (2010) 315–325.

[5] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, **22** (1969) 439–467.

[6] C. G. Langton, "Computation at the edge of chaos: phase transitions and emergent computation," *Physica D*, **42** (1990) 12–37.

[7] C. Marr, and M. T. Hütt, "Outer-totalistic cellular automata on graphs," *Physics Letters A*, **373** (2009) 546–549.

[8] C. Marr, and M. T. Hütt, "Topology regulates pattern formation capacity of binary cellular automata on graphs," *Physica A*, **354** (2005) 641–662.

[9] S. Seif, "Constrained Eden," *Complex Systems*, **18** (2009) 379–385.

[10] K. Sutner, "On the computational complexity of finite cellular automata," *Journal of Computer and System Science*, **50** (1995) 87–97.

[11] M. Tomassini, "Generalized automata networks," *Lecture Notes in Computer Science*, **4173** (2006) 14–28.

[12] D. J. Warne, `GCALab` [C program for analysis of graph cellular automata] (available on GitHub, https://github.com/davidwarne/GCALab).

[13] S. Wolfram, *A new kind of science* (Wolfram Media Inc., Champaign, IL, 2002).

[14] S. Wolfram, "Complex Systems Theory," *Emerging Syntheses in Science: Proceedings of the Founding Workshops of the Santa Fe Institute* (Addison-Wesley, 1988)

[15] S. Wolfram, "Universality and complexity in cellular automata," *Physica D*, (1984) 1–35.

[16] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of Modern Physics*, **55** (1983) 601–644.

[17] A. Wuensche, *Exploring discrete dynamics* (Luniver Press, Frome, UK, 2011).

[18] A. Wuensche, "Genomic regulation modeled as a network with basins of attraction," *Pacific Symposium on Biocomputing'98*, Singapore (1998) 89–102.

[19] A. Wuensche, "The ghost in the machine: basins of attraction of random boolean networks," in *Artificial Life III*, edited C.G. Langton (Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, Reading, MA, 1994).

[20] A. Wuensche, and M. Lesser, *The global dynamics of cellular automata* (Addison-Wesley, Reading, MA, 1992).