2016 IEEE 55th Conference on Decision and Control (CDC)
ARIA Resort & Casino
December 12-14, 2016, Las Vegas, USA

On a Framework for Analysis and Design of Cascades on Boolean Networks

Griffin Kearney and Makan Fardad

Abstract— **We consider Boolean networks defined on directed graphs in which the state of every node belongs to the set $\{0, 1\}$. We think of a node in state 1 as having 'failed'. The state of every node at the next time instant is a function of the states of those nodes that link to it. Nodes fail according to a set of rules, and once a node fails it stays so forever. We develop a mathematical framework that allows us to find the smallest set of nodes whose failure at time zero causes the eventual failure of all nodes in a desired target set. Our methods are based on modeling network dynamics using Boolean polynomials and exploiting their properties. Rather than propagating the state forward using a nonlinear map, we characterize all possible steady-state configurations as the fixed points of the network's dynamics, and provide a simple algorithm for finding all such 'stable' configurations. We demonstrate the utility of our framework with the help of illustrative examples.**

Index Terms— **Boolean networks, Boolean polynomials, cascading failures**

I. Introduction

There is a growing need for the analysis of cascading behavior in large-scale networks [1]–[5]. The list of applications for such work includes prevention of epidemics, inciting social change, viral marketing, and the prevention of blackouts in power networks and the emerging smart grid. For instance, in viral marketing one wishes to induce a cascade and cause wide adoption of a product in the marketplace by selectively targeting initial customers. On the other hand, it may be desired to prevent cascading blackouts and increase the resilience of the power grid against hostile threats by identifying and protecting key infrastructure.

General classes of network cascades have proven to be difficult to analyze for a number of reasons as documented in [6]. In particular, cascades are a result of inherently nonlinear dynamics that do not lend themselves well to linearization schemes. Multiple approximation-based approaches for the analysis of cascading behavior have thus emerged in the literature. In one line of attack, a randomized version of the original problem is considered and the probabilistic setting is exploited to render a tractable problem [7], [8]. Another approximation method is to consider a lumped representation in which certain characteristics of the network are modeled in an aggregate sense, such as the compartmental models used in epidemiology [3], [9]. These approaches have been fruitful in providing some insight into networks and their behavior, but also intrinsically result in the loss of information about the actual states of individual nodes and/or the network topology. In a different line of attack, exact mathematical models are proposed to describe network dynamics in [10]–[13]. Our work fits into this category, in that we develop a general framework without approximations.

In this paper we consider finite Boolean networks which lack a recovery mechanism, where by the latter we mean networks in which once a node fails it remains failed indefinitely. We provide a method for the computation of the minimal initial failures (hereafter also referred to as perturbations) required to force the eventual failure of any target set of nodes. We do this by formalizing the concept of stable configurations, ones which will cause no further changes in the network's state, and we present an algorithm to find all such configurations. We then search the stable configurations for patterns which can be exploited to construct minimal perturbations to catalyze cascades. These patterns provide a description of the most important initial node failures in order to cause large-scale network failure.

While this paper was under review we became aware of [11]. In the present work we consider a smaller class of networks than those in [11], but exploit this restriction to deliver computational tools for the design of network cascades. The work of [11] uses the structure of generalized maps and their fixed points to describe large classes of network behavior. On the other hand, in this paper we use more restricted functions in the form of Boolean polynomials on a smaller class of networks. In this context, the roots of these polynomials are equivalent to the fixed points discussed in [11], but by limiting our focus we are able to explicitly calculate the minimal perturbations which lead to cascades.

II. Problem Statement

In this work we consider networks, described by graphs, in which the state of every node evolves in discrete time and takes values in $\{0, 1\}$. We refer to the 1-state as failure. Nodes have (possibly different) rules that dictate their state at the next time instant as a function of the state of those nodes they interact with. We assume that once a node has failed it will remain so thereafter. The main focus of the present work can be summarized as follows.

Main Problem: Given a set of target nodes, and the failure rule for every node, find a set of nodes with minimal cardinality whose failure at time $t = 0$ causes the failure of all target nodes in steady-state.

Clearly, as a special case one can seek the fewest number of initial failures that will lead to full network failure.

Financial support from the National Science Foundation under awards EAGER ECCS-1545270 and CNS-1329885 is gratefully acknowledged.

G. Kearney and M. Fardad are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244. E-mails: gmkearne@syr.edu, makan@syr.edu.

Given the nodal failure rules, it is generally an easy task to discover what failure configurations will cause the failure of a given node at the next immediate time step. However, as networks grow in size and complexity it becomes increasingly difficult to determine all of the initial failure configurations which will lead to the eventual failure of some node or set of nodes.

III. Mathematical Preliminaries

This section will serve to introduce the basic mathematical structures which we will use to develop our theory. Before proceeding, we note that in this work we use the terms vertex and node interchangeably; the same applies to the terms edge and link.

Definition: Let $X = (V, E)$ denote a network with vertex set V and edge set E. Take $\mathscr{C}_X = \mathbb{F}_2^n$ to be the *configuration space* of the network X, where $\mathbb{F}_2$ denotes the field consisting of $\{0, 1\}$ and n is the number of vertices.

The configuration space of a network is thus the collection of binary sequences of length equal to the cardinality of V, where every configuration is an ordered n-tuple $x = (x_1, x_2, \ldots, x_n) \in \mathscr{C}_X$ with $x_i \in \mathbb{F}_2$. We recall that unit entries in c imply the failure of the corresponding node. In this work we will only concern ourselves with finite networks.

To develop the mathematics needed for our analysis, it is helpful to make the following simple observations. It is obviously true that $0^2 = 0$ and $1^2 = 1$. Since every element of the n-tuple in the configuration space $\mathscr{C}_X$ must take one of these values then it is necessarily true that $x_i^2 = x_i$ for every x_i. Additionally, in $\mathbb{F}_2$ it is true that $1 + 1 = 0$ so that the addition of one to x_i is equivalent to toggling the value of x_i. Finally, if we consider $0 + 0 = 0$ and $1 + 1 = 0$ it is clear that $x_i + x_i = 0$.

Lemma 3.1: For each configuration $c \in \mathscr{C}_X$ there exists a unique polynomial $f_c : \mathscr{C}_X \to \mathbb{F}_2$ such that $f_c(x) = 1$ if and only if $x = c$.

Proof: Recall that c represents a binary n-tuple and that adding one to a variable in $\mathbb{F}_2$ toggles its value. Let

$$h_i(x_i) = \begin{cases} x_i & \text{if } c_i = 1, \\ 1 + x_i & \text{if } c_i = 0. \end{cases}$$

Observe that h_i is a Boolean polynomial for each i and take $f_c(x_1, x_2, \ldots, x_n) = \prod_{i=1}^n h_i(x_i)$, which is also a Boolean polynomial. By this construction we see that f_c is one only if $h_i = 1$ for all i. This is clearly the case for the configuration c. Additionally, for any configuration $c' \neq c$ at least one of the digits of c' and c would differ, say the i-th digit. In that case $h_i(c_i') = 0$ and therefore $f_c(c') = 0$. This completes the proof. ∎

Our aim is to build polynomials which take the value one on a specified *subset* of $\mathscr{C}_X$ rather than a single configuration. Note that since there are only two elements of $\mathbb{F}_2$ the ability to specify configurations on which a polynomial takes the value one implicitly specifies the configurations on which it takes the value zero. In this way we will think of the basic polynomials of Lemma 3.1 as the simple building blocks of more general polynomials and continue in the following manner.

Lemma 3.2: The binary operation $\vee : \mathbb{F}_2 \times \mathbb{F}_2 \to \mathbb{F}_2$ defined as $y \vee z = y + z + yz$ has the property that $y \vee z = 0$ if and only if $y = z = 0$.

Lemma 3.2 is standard [14] and the proof is omitted for brevity. We can now use the $\vee$ operation and two polynomials f_c and $f_{c'}$ defined as in the statement of Lemma 3.1 to generate a new polynomial which takes the value one on each of the two configurations c and c'. In what follows, we demonstrate that this procedure can be generalized to an arbitrary number of configurations.

Definition: Let $\vee_{i=1}^n x_i = x_1 \vee x_2 \vee x_3 \ldots \vee x_n$, which is well defined since $\vee$ is associative. It is clear that $\vee_{i=1}^n x_i = 0$ if and only if $x_i = 0$ for all i.

The following theorem allows us to define polynomials which are one on any arbitrary subset of the configuration space of the network and zero on its complement.

Theorem 3.3: Let $C \subset \mathscr{C}_X$. There exists a polynomial $F_C : \mathscr{C}_X \to \mathbb{F}_2$ such that $F_C(c) = 1$ for every configuration $c \in C$, and $F_C(x) = 0$ if $x \notin C$.

Proof: Since the network is finite, $\mathscr{C}_X$ is countable and thus C is countable. Therefore $F_C(x) = \vee_{c \in C} f_c(x)$ is well defined. It is clear that by this definition $F_C = 1$ on C since if $c' \in C$ then $f_{c'}(c') = 1$ and thus $\vee_{c \in C} f_c(c') = 1$. If $x \notin C$ then $f_c(x) = 0$ for all $c \in C$ and so $F_C(x) = 0$. ∎

As we move forward, for $C \subseteq \mathscr{C}_X$ we will take F_C to be understood as the polynomial which is one on C and zero on $\mathscr{C}_X \backslash C$.

IV. Stable Configurations

In this section we adapt the tools introduced in the previous section to examine node failures. Specifically, we wish to investigate which configurations will induce further failure propagation and which will not. We refer to the latter types of configurations as *stable* configurations.

For any finite network with a set of failure rules, we will assume that it is possible to generate a collection of sets $\{C_i\}_{i=1}^n$, where C_i is the subset of the configuration space which will lead to the failure of the i-th node at the next time step. It should be stressed for clarity that the i-th component of all configurations in C_i must be zero, since the i-th node can not fail at the next time instant if it has already failed. If the network's current state belongs to C_i then the state of the i-th node will be forced to one at the next time step. We refer to $\{C_i\}_{i=1}^n$ as the *trigger configurations*,

$$C_i : \text{ trigger configurations of node } i.$$

We view the specification of network failure dynamics as being equivalent to characterizing the collection of trigger configurations for each node in the network.

Definition: Given the set C_i of triggering configurations of node i, we call $F_{C_i}(x) : \mathscr{C}_X \to \mathbb{F}_2$, whose existence was proved in Theorem 3.3, the i-th *indicator function*. For simplicity of notation we hereafter use $F_{C_i}(x)$ and $F_i(x)$ interchangeably.

Notice that F_i takes the value one if and only if the i-th node's state is being forced to change. This property makes the indicator functions appealing building blocks for network analysis. Indeed, if $x(k)$ denotes the configuration of the network at time $k = 1, 2, \ldots$ then the evolution of the state of node i satisfies $x_i(k+1) = x_i(k) + F_i(x(k))$ or

$$x(k+1) = x(k) + F(x(k)), \tag{1}$$

where $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))$. Motivated by (1) we make the following definition.

Definition: Define the map $\psi : \mathscr{C}_X \to \mathscr{C}_X$ as one which satisfies

$$\psi(x) = x + F(x) \tag{2}$$

and refer to it as the *evolution map*. Define *stable* configurations as those which satisfy $F(x) = 0$, or equivalently $F_i(x) = 0$ for $i = 1, 2, \ldots, n$. Finally, let

$$G(x) = \vee_{i=1}^{n} F_i(x)$$

and refer to G as the *characteristic function* of the network.

Clearly, if $G(x) = 0$ then each of the indicator functions F_i must be zero at x. We now state the following theorem.

Theorem 4.1: The following are equivalent:
(i) $F(\overline{x}) = 0$, i.e., configuration $\overline{x}$ is stable.
(ii) $G(\overline{x}) = 0$.
(iii) $\psi(\overline{x}) = \overline{x}$.

Proof: Proofs that these statements are equivalent are routine and are thus omitted for brevity. ∎

Theorem 4.1 provides a natural way to formally partition the configuration space into stable and unstable states

$$\mathscr{C}_X = \mathscr{S} \cup \mathscr{D},$$

with

$$\mathscr{S} = \{x \in \mathscr{C}_X \mid G(x) = 0\}$$

denoting the *set of stable configurations* and

$$\mathscr{D} = \{x \in \mathscr{C}_X \mid G(x) = 1\}.$$

We define the following map for the purposes of our forthcoming discussion.

Definition: Let X be a network that lacks a recovery mechanism, in the sense that any failed node remains failed. Define $\phi : \mathscr{C}_X \to \mathscr{C}_X$ as the map that takes any initial configuration to its corresponding steady-state configuration,

$$\phi(x) = \lim_{k \to \infty} \psi^k(x), \tag{3}$$

where exponentiation on the right is to be understood as repeated composition. It should be noted that the well-definedness of ϕ relies on the lack-of-recovery assumption on all network nodes.

Proposition 4.2: $\phi(\mathscr{C}_X) = \mathscr{S}$.

Proof: If $x \in \mathscr{S}$ then it is a stable configuration and thus $\phi(x) = x$ so that $\phi|_{\mathscr{S}} = I$ where I is the identity map. Therefore $\mathscr{S} \subseteq \phi(\mathscr{C}_X)$.

For the reverse direction let $x \in \mathscr{C}_X$. Since we have assumed that the network lacks recovery then there must exist some positive integer k such that $\psi^{k+1}(x) = \psi^k(x)$. To justify the existence of such a k, consider that we necessarily have $\psi^{n+1}(x) = \psi^n(x)$; because nodes cannot return to zero state once failed, if a configuration is not stable then at least one node will change from state zero to one, and thus after n iterations either all nodes will have failed or the network must have stabilized prior. Therefore, given k, if we take $\psi^k(x) = x'$ it is clear that x' is a stable configuration and that $\phi(x) = x'$. Since x was an arbitrary element of the configuration space we conclude that $\phi(\mathscr{C}_X) \subseteq \mathscr{S}$. ∎

In the next section we present a simple algorithm for the computation of $\mathscr{S}$.

V. Computation of Stable Configurations Using Zero Sets

In the previous section we defined the set $\mathscr{S}$ of stable configurations as the steady-state result of initial failure configurations. In this section we describe how to find $\mathscr{S}$ using the notion of *zero sets*, which we define next.

Definition: For the Boolean polynomial p we define its *zero set* as $Z(p) = \{x \in \mathscr{C}_X \mid p(x) = 0\}$.

It follows from the definition of $\mathscr{S}$ that

$$Z(G) = \mathscr{S}.$$

Proposition 5.1: Let p, q be Boolean polynomials. Then
(i) $Z(p \vee q) = Z(p) \cap Z(q)$;
(ii) $Z(p\,q) = Z(p) \cup Z(q)$;
(iii) $Z(1 + p) = Z^{\mathrm{c}}(p)$.

Proof:

(i) $$\begin{aligned} Z(p \vee q) &= \{x \in \mathscr{C}_X \mid p(x) \vee q(x) = 0\} \\ &= \{x \in \mathscr{C}_X \mid p(x) = 0 \text{ and } q(x) = 0\} \\ &= Z(p) \cap Z(q). \end{aligned}$$

(ii) $$\begin{aligned} Z(p\,q) &= \{x \in \mathscr{C}_X \mid p(x)\,q(x) = 0\} \\ &= \{x \in \mathscr{C}_X \mid p(x) = 0 \text{ or } q(x) = 0\} \\ &= Z(p) \cup Z(q). \end{aligned}$$

(iii) $$\begin{aligned} Z(1 + p) &= \{x \in \mathscr{C}_X \mid 1 + p(x) = 0\} \\ &= \{x \in \mathscr{C}_X \mid p(x) = 1\} \\ &= \{x \in \mathscr{C}_X \mid p(x) = 0\}^{\mathrm{c}} \\ &= Z^{\mathrm{c}}(p). \end{aligned}$$ ∎

Recall that $G(x) = \vee_{i=1}^{n} F_i(x)$, and so we can apply Proposition 5.1 to conclude that

$$\mathscr{S} = Z(G) = \cap_{i=1}^{n} [Z(F_i)]. \tag{4}$$

We observe that calculating this intersection will be difficult, since we do not know the zeros of each of the triggering polynomials immediately. On the other hand, if we take the compliment of $Z(G)$ in $\mathscr{C}_X$ we obtain

$$Z^{\mathrm{c}}(G) = \cup_{i=1}^{n}[Z^{\mathrm{c}}(F_i)] = \cup_{i=1}^{n} C_i, \tag{5}$$

where in the last equality we have used the fact that $Z^{\mathrm{c}}(F_i) = C_i$. Once $Z^{\mathrm{c}}(G)$ is found then taking its complement a second time renders the desired set $\mathscr{S} = Z(G)$.

VI. Cascades on Boolean Networks: Main Results

We recall that the main objective of this work is to find initial configurations, also referred to as perturbations, with minimal cardinality that will fail a desired set of target nodes in steady-state. In this section we utilize the results developed in Sections III–V to calculate the minimum perturbations that will fail a specified target set.

Definition: Define the norm $\|\cdot\|$ on the space $\mathscr{C}_X$ such that if $x \in \mathscr{C}_X$ then $\|x\| = \sum_{i=1}^{n} x_i$. We will use the words norm and cardinality interchangeably. Moreover, for $x \in \mathscr{C}_X$ define the *compatibility of* x to be the set

$$\mathscr{S}_x = \{x' \in \mathscr{S} \mid \text{if } x_i = 1 \text{ then } x'_i = 1\}.$$

The next proposition gives insight into how combining initial failures restricts the states to which the network is able to stabilize, and is useful in solving the Main Problem.

Proposition 6.1: If $y, z \in \mathscr{C}_X$, and $x = y \vee z$ by using the definition of $\vee$ and extending it via component-wise operations on $\mathscr{C}_X$, then $\mathscr{S}_x = \mathscr{S}_y \cap \mathscr{S}_z$.

Proof: Since x has component values of 1 where either y or z have component values of 1 it is clear that $\mathscr{S}_x \subseteq \mathscr{S}_y \cap \mathscr{S}_z$. If $c \in \mathscr{S}_y \cap \mathscr{S}_z$ then it must be that $c_i = 1$ if $y_i = 1$ or $z_i = 1$. Thus $c_i = 1$ if $x_i = 1$ so that $c \in \mathscr{S}_x$, which implies $\mathscr{S}_x \supseteq \mathscr{S}_y \cap \mathscr{S}_z$. Therefore $\mathscr{S}_x = \mathscr{S}_y \cap \mathscr{S}_z$. ∎

Assumption 1 (Monotonicity of Failures): Let ϕ be defined as in (3) and let $x, y \in \mathscr{C}_X$. Then ϕ is such that $x \leq y$ implies $\phi(x) \leq \phi(y)$, with the inequality interpreted component-wise.

Assumption 1 is natural for large classes of physical systems, where the addition of more initial node failures increases the number of failures in steady-state.

Definition: For $x \in \mathscr{C}_X$ define $e_i(x) = x_i$, where x_i is the ith element of the n-tuple x. The lack of a recovery mechanism in particular implies that if $e_i(x) = 1$ then $e_i(\phi(x)) = 1$.

Proposition 6.2: If the network satisfies Assumption 1 and $x \in \mathscr{C}_X$ then $\mathscr{S}_x = \mathscr{S}_{\phi(x)}$.

Proof: Since we can write $\phi(x) = \phi(x) \vee x$ then by Proposition 6.1 we have that $\mathscr{S}_{\phi(x)} = \mathscr{S}_{\phi(x)\vee x} = \mathscr{S}_{\phi(x)} \cap \mathscr{S}_x$, which implies that $\mathscr{S}_{\phi(x)} \subseteq \mathscr{S}_x$.

For $c \in \mathscr{S}_x$ we have that $c \geq x$. This inequality together with Assumption 1 implies that $\phi(c) \geq \phi(x)$. But c is a stable configuration, $c = \phi(c)$, and therefore $c \geq \phi(x)$. We thus have that $c \in \mathscr{S}_{\phi(x)}$, which implies that $\mathscr{S}_x \subseteq \mathscr{S}_{\phi(x)}$. ∎

We are now in position to address the Main Problem stated in Section II. The following proposition is one of the main results of this work.

Proposition 6.3: Assume that the network satisfies Assumption 1. The solution to the Main Problem is given by the solution to the optimization problem

$$\begin{aligned} &\text{minimize} && \|x\| \\ &\text{subject to} && \mathscr{S}_x \subseteq \mathscr{S}_{\overline{x}},\ x \in \mathscr{C}_X, \end{aligned} \tag{6}$$

where $\overline{x}$ characterizes the desired target nodes for failure. The vector $\overline{x}$ has an i-th component of 1 if the i-th node is targeted for failure and an i-th component of 0 otherwise.

Proof: Proving this amounts to demonstrating the validity of the constraint $\mathscr{S}_x \subseteq \mathscr{S}_{\overline{x}}$. Suppose that x causes the failure of the target nodes, or equivalently that $\phi(x) \geq \overline{x}$. We may write $\phi(x) = \overline{x} \vee x'$ for some x'. If we apply Propositions 6.1, 6.2 then we conclude that $\mathscr{S}_x = \mathscr{S}_{\phi(x)} = \mathscr{S}_{\overline{x}} \cap \mathscr{S}_{x'} \subseteq \mathscr{S}_{\overline{x}}$. With this we see that any perturbation which fails the target nodes must satisfy the constraint.

On the other hand, suppose that there were an element $c \in \mathscr{S}_x$ such that $c \notin \mathscr{S}_{\overline{x}}$. From $\mathscr{S}_x = \mathscr{S}_{\phi(x)}$ it follows that c must also be compatible with $\phi(x)$. Since c is not compatible with $\overline{x}$ then there is some component of $\overline{x}$ which is one while the corresponding component of c is zero. Therefore the same component of $\phi(x)$ is also zero and thus x will have not failed all of the target nodes. ∎

A useful observation is that the constraint $\mathscr{S}_x \subseteq \mathscr{S}_{\overline{x}}$ is equivalent to $\mathscr{S}_x \cap (\mathscr{S} \backslash \mathscr{S}_{\overline{x}}) = \varnothing$. A method of solving the optimization (6) is given in Algorithm 1, and is based on replacing the first constraint in (6) by $\mathscr{S}_x \cap (\mathscr{S} \backslash \mathscr{S}_{\overline{x}}) = \varnothing$. A simple application of Proposition 6.3 and Algorithm 1 is demonstrated in the 'Square Network' example below. This procedure is also used for computing the solutions to the cascade problems presented in Section VII.

Algorithm 1 Solution of (6)

1: **given** trigger configurations $\{C_i\}_{i=1}^{n}$, and target $\overline{x}$.
2: Compute $\mathscr{S}$ using (4)-(5).
3: Arrange elements of $\mathscr{S}$ as rows of matrix S, omitting all rows in compatibility of target.
4: **for** $k = 1, 2, \ldots$ **do**
5: If $k = 1$, set $x = (0, 0, \ldots, 0)$.
6: Set s_i equal to sum of elements in i-th *column* of S.
7: Take i^* as index corresponding to minimal s_i, if minimal not unique take smallest such index.
8: Set i^*-th element of x equal to 1.
9: If $s_{i^*} = 0$ **quit**, else omit all rows of S which have 0 in their i^*-th element.
10: **end for**

Example (Square Network): Consider the square network in Fig. 1. We assume a failure rule that forces a node to fail if

Fig. 1: The Square Network

its two neighbors have failed. We explicitly list the triggering configurations of each node in the following table.

C_1	C_2	C_3	C_4
$(0,1,0,1)$	$(1,0,1,0)$	$(0,1,0,1)$	$(1,0,1,0)$
$(0,1,1,1)$	$(1,0,1,1)$	$(1,1,0,1)$	$(1,1,1,0)$

With the aid of this table we can calculate the stable configurations using Proposition 5.1 and (5) to obtain the sets $Z^{\mathrm{c}}(G)$ and $\mathscr{S} = Z(G)$.

$Z^{\mathrm{c}}(G)$
$(0,1,0,1)$
$(1,0,1,0)$
$(1,1,0,1)$
$(1,1,1,0)$
$(0,1,1,1)$
$(1,0,1,1)$

$\mathscr{S} = Z(G)$
$(0,0,0,0)$
$(1,0,0,0)$
$(0,1,0,0)$
$(0,0,1,0)$
$(0,0,0,1)$
$(1,1,0,0)$
$(0,1,1,0)$
$(0,0,1,1)$
$(1,0,0,1)$
$(1,1,1,1)$

$\mathscr{S}\backslash\mathscr{S}_{\overline{x}}$
$(0,0,0,0)$
$(1,0,0,0)$
$(0,1,0,0)$
$(0,0,1,0)$
$(0,0,0,1)$
$(1,1,0,0)$
$(0,1,1,0)$
$(0,0,1,1)$
$(1,0,0,1)$

If our target was to fail the entire network then $\overline{x} = (1,1,1,1)$ and $\mathscr{S}_{\overline{x}} = \{(1,1,1,1)\}$ so that $\mathscr{S}\backslash\mathscr{S}_{\overline{x}}$ is $\mathscr{S}$ excluding the final row. The idea now is to select unit perturbations and combine them using Proposition 6.1 to construct a perturbation whose compatibility sits entirely outside of $\mathscr{S}\backslash\mathscr{S}_{\overline{x}}$. Because of the symmetry of this problem any node which we choose for our first unit perturbation will work since each column has the same number of ones in it.

$\mathscr{S}_{x'} \cap (\mathscr{S}\backslash\mathscr{S}_{\overline{x}})$
$(1,0,0,0)$
$(1,1,0,0)$
$(1,0,0,1)$

$\mathscr{S}_{x' \vee x''} \cap (\mathscr{S}\backslash\mathscr{S}_{\overline{x}})$
empty

Suppose we chose $x' = (1,0,0,0)$ for our first failure. There is clearly a natural choice for the next unit perturbation since the third column of $\mathscr{S}_{x'} \cap (\mathscr{S}\backslash\mathscr{S}_{\overline{x}})$ contains no ones. Take $x'' = (0,0,1,0)$ as our second failure so that our total initial perturbation is $x = x' \vee x'' = (1,0,1,0)$, which has the property that $\mathscr{S}_x \cap (\mathscr{S}\backslash\mathscr{S}_{\overline{x}}) = \mathscr{S}_{x'} \cap \mathscr{S}_{x''} \cap (\mathscr{S}\backslash\mathscr{S}_{\overline{x}}) = \varnothing$. Therefore $x = (1,0,1,0)$ must fail the target. One can easily check that this is indeed the case.

Finally, the following theorem is of interest beyond the immediate scope of this work and its applications are a topic of current research.

Theorem 6.4: If the network satisfies Assumption 1 and $x \in \mathscr{C}_X$ then $\mathscr{S}_x$ has a unique minimal element with respect to the norm, and furthermore $\phi(x)$ is the minimal element of $\mathscr{S}_x$.

Proof: Since $x_i = 1$ implies $e_i(\phi(x)) = 1$, it is necessarily true that $\|\phi(x)\| \geq \|x\|$. First consider the case when $x \in \mathscr{S}$. In this case we have that x is a minimal element of $\mathscr{S}_x$ and that $\phi(x) = x$. Thus we only need to show that this element is unique. This is obvious, however, because if there is an $c \in \mathscr{S}_x$ such that $\|c\| = \|x\|$ then $c = x$ since $c_i = 1$ if $x_i = 1$.

Therefore all that remains to be shown is the case when $x \in \mathscr{D}$. In this case we utilize Proposition 6.2, which states that $\mathscr{S}_x = \mathscr{S}_{\phi(x)}$. Since $\phi(x)$ is a stable configuration and thus belongs to $\mathscr{S}$ then we know that $\mathscr{S}_{\phi(x)}$ has a unique minimal element $\phi(x)$. Hence the unique minimal element of $\mathscr{S}_x$ is $\phi(x)$. ∎

The power of Theorem 6.4 is that it provides an explicit way to obtain the stable state $\phi(x)$ which will be reached by a network with initial perturbation $x \in \mathscr{C}_X$. Given initial configuration x, we find the minimum norm stable configuration which contains x. Equivalently, $\phi(x)$ is equal to the unique solution of the optimization problem

$$\begin{aligned} &\text{minimize} && \|y\| \\ &\text{subject to} && y \in \mathscr{S}_x. \end{aligned}$$

VII. Illustrative Examples

This section will serve to further demonstrate the utility of the developed framework. In each example network that follows, we choose our target set to be the whole network; in other words we seek a minimal initial failure set which results in eventual full network failure. We color nodes to indicate their state at time $t = 0$; operational/healthy nodes are shown in green and failed nodes in red.

We begin with the simple example in Fig. 2, inspired by small-world networks. Here, each node is connected to four other nodes and all links are *undirected*. The failure rules are that a node will fail if at least two of its neighbors have failed. We now employ the tools developed in the previous sections. A minimal, though not necessarily unique, initial perturbation which satisfies our full failure requirement is depicted using red colored nodes in Fig. 2.

For our next example we use the network in Fig. 3. Each node is influenced by four other nodes and all links are *directed*. The failure rules are that a node will fail if at least two of the nodes which influence it have failed. A minimal initial perturbation which generates full network failure is depicted using red colored nodes in Fig. 3. This example is substantially less structured than the last, but our framework reveals that the entire network can be failed with just two initial node failures. We stress again that this solution may not be unique.

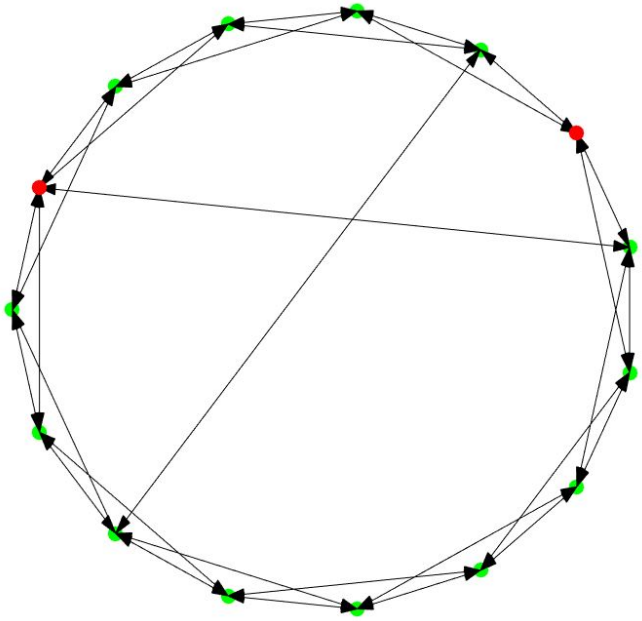

Fig. 2: A minimal perturbation that will cause full network failure of the first example.

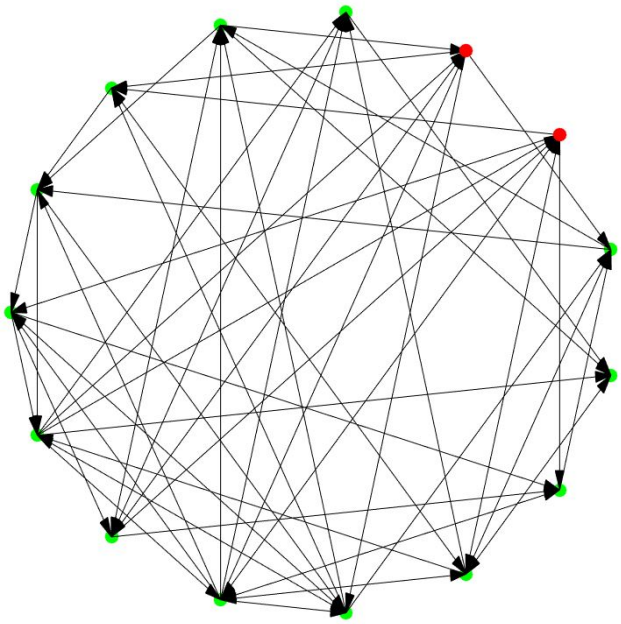

Fig. 3: A minimal perturbation that will cause full network failure of the second example.

VIII. Conclusions

In this work we consider finite Boolean networks that evolve in discrete time. A node, once in failed state, stays in that state indefinitely. We develop a framework to analyze such networks using Boolean polynomials. Specific contributions of importance are the construction of the network characteristic function, the computation of the stable configurations, and the generation of minimal initial perturbations/failures. The latter two of these constructions rely on the assumptions that the nodes take binary states and that the nodes lack a recovery mechanism. Further research is being pursued in order to eliminate or weaken the need for these assumptions. Specifically, the requirement that the nodes of the network take binary states is particularly restrictive, since there are numerous network applications with state spaces that are not binary; see for instance [12], [13]. Finally, an exact characterization of the computational complexity of our approach is the topic of current research.

IX. Acknowledgment

The first author thanks Prof. Graham Leuschke, Laura Ballard, and Alec Sullivan for their helpful comments.

References

[1] S. Morris, "Contagion," *Review of Economic Studies*, vol. 67, pp. 57–78, 2000.

[2] D. Centola and M. Macy, "Complex contagions and the weakness of long ties," *American Journal of Sociology*, vol. 113, pp. 702–734, 2007.

[3] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[4] L. Blume, D. Easley, J. Kleinberg, R. Kleinberg, and E. Tardos, "Which networks are least susceptible to cascading failures?" in *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science*, 2011.

[5] F. Chierichetti, J. Kleinberg, and A. Panconesi, "How to schedule a cascade in an arbitrary graph," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, 2012.

[6] A. E. Motter, "Networkcontrology," *Chaos*, vol. 25, p. 097621, 2015.

[7] D. J. Watts, "A simple model of global cascades on random networks," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 5766–5771, 2002.

[8] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of KDD*, 2003.

[9] M. E. J. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[10] E. M. Adam, M. A. Dahleh, and A. Ozdaglar, "Towards an algebra for cascade effects," in *Proceedings of the 53rd IEEE Conference on Decision and Control*, Dec. 2014, pp. 4461–4466.

[11] ——, "Towards an algebra for cascade effects," *arXiv 1506.06394*, June 2015.

[12] G. Como, K. Savla, D. Acemoglu, M. A. Dahleh, and E. Frazzoli, "Robust distributed routing in dynamical networks – Part I: Locally responsive policies and weak resilience," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 317–332, 2013.

[13] ——, "Robust distributed routing in dynamical networks – Part II: Strong resilience, equilibrium selection, and cascaded failures," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 333–348, 2013.

[14] C. C. Pinter, *A Book on Abstract Algebra*. Courier Corporation, 2010.