On Finding Minimum Deadly Sets for Directed Networks

NORBERT ZEH
School of Computer Science, Carleton University, Ottawa, Canada

NICOLA SANTORO
School of Computer Science, Carleton University, Ottawa, Canada

Abstract

Given a set S of elements in a *directed network* that are initially faulty, an element becomes (functionally) faulty if all its in-neighbors or all its out-neighbors are (functionally) faulty. A set S of initially faulty elements is called *deadly* if it causes the entire network to become faulty according to the above rule. We show that finding a minimum deadly set is NP-hard for arbitrary directed networks. For directed acyclic graphs (DAGs), we show that finding a weighted minimum deadly set is no harder than finding a minimum cut. We also study the case where a vertex becomes faulty if at least a certain percentage of its in-neighbors or out-neighbors is faulty. We call a set S of initially faulty elements ε-*deadly* if it causes the whole network to become faulty using this ε-majority rule. We show that finding a minimum ε-deadly set is NP-hard even for a restricted subclass of directed acyclic graphs.

Keywords

distributed computing, fault tolerance, majority rule, NP-hardness, minimum cut

1 Introduction

Background In most communication-based systems, the behavior of a single entity depends on the status of its neighboring entities in the system. This is true by definition in systems such as cellular automata, neural networks, or coupled map lattices; such dependencies can be found, sometimes unexpectedly, at different levels in VLSI systems, data communication networks, and distributed systems. This "locality" property can be used as a basic construct for distributed computations [24] and has been extensively employed in a variety of applications. Thus, it regulates the behavior of many communication protocols.

In all these environments, faulty elements can induce a faulty behavior in their neighbors. This is for example the case when the elements of a network maintain the consistency of crucial data by comparing their local copies with the ones

held by their neighbors, and resolving inconsistencies by performing majority voting [22]; thus, if the majority of its neighbors has corrupted data, a non-faulty element will exhibit a faulty behavior (e.g., its data will become corrupted) and will therefore be indistinguishable from a faulty one.

A large class of problems related to security and fault-tolerance have been modeled, analyzed and solved using this view of the system. In these studies, the system is viewed as a graph; vertices correspond to system entities; edges describe the neighborhood relation (e.g., direct communication links between entities). Every vertex is initially colored either "black" (faulty) or "white" (non-faulty). At each local time step, it recolors itself according to the colors held by the majority of its neighbors. Depending on the initial assignment of colors to the nodes and the definition of *majority*, different dynamics emerge.

These dynamics have been extensively studied in the context of synchronous systems (mostly cellular automata) with different majority functions (simple majority, strong majority, weighted threshold functions, convex functions) and different coloring sets (e.g., see [1, 11, 16, 25]).

In the context of distributed computing, the interest has been in simple and strong majority;[1] the focus has been on the patterns of initial faults which may cause the entire system to behave in a faulty manner. In terms of system dynamics, these are the patterns for which the system converges to a *monochromatic* fixed point. They have been called *dynamos* by Peleg who introduced their study in [23]. Most of the results are known for the static version of this process, that is, considering only one or two steps in the evolution [3, 4, 14, 22]. In the dynamic case, results on dynamos exist for a variety of topologies, including chordal rings, tori, and most interconnection networks [7, 8, 9, 10, 15, 23]. Recently nondeterministic rules and weighted majority functions have been investigated in [12, 13, 17].

All these studies have almost exclusively considered *undirected networks*. In this paper we are interested in the dynamics of majority rules in the more general case of *directed networks*. Surprisingly little is known for these systems. The only relevant results come from a separate line of investigation carried out in the area of VLSI arrays.

In these systems, fault-tolerance is commonly achieved by the combined use of component redundancy, bypass links, and reconfiguration techniques. However, a small number of strategically located faults can render the entire array unusable, regardless of the amount of redundancy and the cleverness of the reconfiguration technique; these sets of faults were called *deadly* and the patterns they formed were termed *catastrophic fault patterns* (CFP) by Nayak et al., who introduced their study in [20]. Clearly, a deadly set leads the system to a monochromatic fixed point, as does a dynamo. The fundamental difference between the dynamo and the CFP setting is that, in the latter, links are *unidirectional* and an

[1] That is, majority is defined as half of the neighbors. The difference is how ties are broken.

element becomes "dead" only if *all* its in-neighbors (out-neighbors) are "dead".[2] There have been several investigations on catastrophic fault patterns and their properties, the complexity of recognizing whether a set of faults is deadly, the efficient generation of such CFP, and how to route if the pattern is not catastrophic, etc. [27, 26, 18, 19, 20, 21].

Unidirectionality makes the results on CFP directly relevant for our investigation. However, those studies were limited to a restricted class of networks; indeed, due to the nature of the application system (VLSI array with regular bypass links), they apply only to directed *chordal rings* in one and two dimensions.

Our Results In this paper, we study the complexity of finding deadly and ε-deadly sets for arbitrary directed networks. In Section 2, we introduce the most important terminology and prove some basic properties of deadly and ε-deadly sets, which are the tools employed to prove the results of Sections 3, 4, and 5. In Section 3, we show that finding a minimum deadly set in the sense of [20] for arbitrary directed graphs is NP-hard, while Section 4 provides a linear time reduction of the problem of finding a minimum deadly set in directed acyclic graphs to that of finding a minimum cut in an *st*-graph constructed from the given DAG. In Section 5, we prove that finding a minimum ε-deadly set is NP-hard even for a very restricted class of directed acyclic graphs.

Definitions A *directed graph* (directed network) $G = (V,E)$ is an ordered pair of sets V and E, where the elements in E are ordered pairs (v,w) of elements $v,w \in V$. We call the elements of V the *vertices* or *nodes* of G; the elements of E are the *edges* of G. For an edge $(v,w) \in E$, we call vertices v and w the *endpoints* of (v,w); v and w are *adjacent*. The *in-neighborhood* of a vertex $v \in V$ is the set $\mathcal{N}^-(v) = \{u \in V : (u,v) \in E\}$. Analogously, the *out-neighborhood* of a vertex $v \in V$ is the set $\mathcal{N}^+(v) = \{w \in V : (v,w) \in E\}$. The *in-degree* and *out-degree* of a vertex v are defined as $\deg^-(v) = |\mathcal{N}^-(v)|$ and $\deg^+(v) = |\mathcal{N}^+(v)|$, respectively. A *path* in G is a sequence $P = (v_0, v_1, \ldots, v_k)$ of vertices in G such that $(v_{i-1}, v_i) \in E$, for all $1 \leq i \leq k$. P is a *cycle* if $v_0 = v_k$. A *directed acyclic graph* (DAG) is a directed graph that does not contain cycles. A vertex v in a DAG $G = (V,E)$ is a *source* if $\deg^-_G(v) = 0$; vertex v is a *sink* if $\deg^+_G(v) = 0$. An *st-graph* is a DAG $G = (V,E)$ with exactly one source s and exactly one sink t. In subsequent sections, we frequently perform the operation $G \cap S$ for a graph $G = (V,E)$ and a vertex set S. Viewing G as a collection of vertices and edges, this operation is naturally defined as $G \cap S = V \cap S$.

[2] The initially faulty elements are dead by definition; we are using the original terminology of [20] here.

2 Deadly Sets, Propagation Sequences, and Immortal Subgraphs

In this section, we study propagation sequences and immortal subgraphs, which are the basic tools employed to prove the results in Sections 3, 4, and 5. Given an assignment $\omega : V \to \mathbb{R}^+$ of weights to the vertices of a graph $G = (V,E)$, the weight $\omega(S)$ of a subsets $S \subseteq V$ is defined as $\omega(S) = \sum_{v \in S} \omega(v)$. A subset $S \subseteq V$ of vertices of G induces an ε-*propagation sequence* $S = S_0 \subseteq S_1 \subseteq S_2 \subset \ldots$ of subsets of V, for $0 < \varepsilon \leq 1$, where a vertex $v \in V$ is contained in set S_i, $i > 0$, if and only if

1. $v \in S_{i-1}$,
2. $\mathcal{N}^-(v) \neq \emptyset$ and $\omega(S_{i-1} \cap \mathcal{N}^-(v)) \geq \varepsilon \cdot \omega(\mathcal{N}^-(v))$, or
3. $\mathcal{N}^+(v) \neq \emptyset$ and $\omega(S_{i-1} \cap \mathcal{N}^+(v)) \geq \varepsilon \cdot \omega(\mathcal{N}^+(v))$.

Set S is ε-*deadly* if there exists some $k \geq 0$ such that $S_i = V$, for all $i \geq k$. We call a subgraph I of G ε-*immortal* if $I \cap S = \emptyset$ implies that $I \cap S_i = \emptyset$, for all $i \geq 0$. We call a set S *deadly* if it is 1-deadly. A subgraph I of G is *immortal* if it is 1-immortal. The *weighted* or *unweighted (ε-)deadly set problem* is the problem of finding a minimum (ε-)deadly set for a weighted or unweighted directed graph.[3]

Next we provide necessary and sufficient conditions for a subgraph of G to be ε-immortal, and for a subset of vertices of G to be ε-deadly.

Lemma 1 *A subgraph I of G is ε-immortal, $0 < \varepsilon \leq 1$, if and only if every vertex $v \in I$ satisfies the following two conditions:*

(i) Either $\mathcal{N}^-(v) = \emptyset$ or $\omega(\mathcal{N}^-(v) \cap I) > (1-\varepsilon)\omega(\mathcal{N}^-(v))$, and

(ii) Either $\mathcal{N}^+(v) = \emptyset$ or $\omega(\mathcal{N}^+(v) \cap I) > (1-\varepsilon)\omega(\mathcal{N}^+(v))$.

Proof. First assume that every vertex $v \in I$ satisfies the above conditions and that $S \cap I = \emptyset$. We have to show that $S_i \cap I = \emptyset$, for all $i \geq 0$, where $S = S_0, S_1, S_2, \ldots$ is the ε-propagation sequence induced by S. The proof is by induction.

The base case is trivial because $S = S_0$ and $S \cap I = \emptyset$. So assume that $S_i \cap I = \emptyset$, for $0 \leq i \leq k$. We have to show that $S_{k+1} \cap I = \emptyset$. Assume for the sake of contradiction that there is a vertex $v \in S_{k+1} \cap I$. Note that $v \notin S_k$. Thus, either $\mathcal{N}^-(v) \neq \emptyset$ and $\omega(\mathcal{N}^-(v) \cap S_k) \geq \varepsilon \cdot \omega(\mathcal{N}^-(v))$, or $\mathcal{N}^+(v) \neq \emptyset$ and $\omega(\mathcal{N}^+(v) \cap S_k) \geq \varepsilon \cdot \omega(\mathcal{N}^+(v))$. However, $\mathcal{N}^-(v) \neq \emptyset$ implies that $\omega(\mathcal{N}^-(v) \cap I) > (1-\varepsilon)\omega(\mathcal{N}^-(v))$, by the first condition of the lemma, and $\mathcal{N}^+(v) \neq \emptyset$ implies that $\omega(\mathcal{N}^+(v) \cap I) > (1-\varepsilon)\omega(\mathcal{N}^+(v))$, by the second condition of the lemma. This leads to a contradiction because $S_k \cap I = \emptyset$ and thus $\omega(\mathcal{N}^-(v) \cap (I \cup S_k)) > \omega(\mathcal{N}^-(v))$ and $\omega(\mathcal{N}^+(v) \cap (I \cup S_k)) > \omega(\mathcal{N}^+(v))$.

[3] In an unweighted graph, all vertices have equal weight.

Now assume that I is ε-immortal. We have to show that every vertex in I satisfies the two conditions of the lemma. So let $S = G \setminus I$ and $S = S_0, S_1, S_2, \ldots$ be the ε-propagation sequence induced by S. Let v be a vertex of I that does not satisfy the first condition of the lemma. That is, $\mathcal{N}^-(v) \neq \emptyset$ and $\omega(\mathcal{N}^-(v) \cap I) \leq (1-\varepsilon)\omega(\mathcal{N}^-(v))$. Then $\omega(\mathcal{N}^-(v) \cap S) \geq \varepsilon \cdot \omega(\mathcal{N}^-(v))$ because $S \cup I = V$, so that $v \in S_1$, contradicting the ε-immortality of I. A similar argument applies if v does not satisfy the second condition. □

Lemma 2 *A subset S of the vertices of G is ε-deadly, $0 < \varepsilon \leq 1$, if and only if $S \cap I \neq \emptyset$, for every ε-immortal subgraph I of G.*

Proof. First assume that there is an ε-immortal subgraph I of G such that $I \cap S = \emptyset$. Let $S = S_0, S_1, S_2, \ldots$ be the ε-propagation sequence induced by S. By the definition of ε-immortal subgraphs, $I \cap S_i = \emptyset$, for $i \geq 0$. Thus, S cannot be ε-deadly.

Now assume that S is not ε-deadly. We show that there has to be an ε-immortal subgraph I of G such that $I \cap S = \emptyset$. As S is not ε-deadly, there has to be an index k such that $S_i = S_k \neq V$, for $i \geq k$. Let $v_0 \in V \setminus S_{k+1}$.

We construct an ε-immortal subgraph I of G such that $I \cap S = \emptyset$. We do this by constructing a sequence of subgraphs $I_0 \subset I_1 \subset \cdots \subset I_r$ such that $V(I_0) = \{v_0\}$ and $I_r = I$. Given that we have constructed a sequence $I_0, \ldots, I_j$, we show that either all vertices in I_j satisfy the conditions of Lemma 1, or we can add a vertex not in S_{k+1} to I_j to obtain a proper supergraph I_{j+1} of I_j with $I_{j+1} \cap S_{k+1} = \emptyset$. As we can repeat this augmentation process only a finite number of times, we must finally obtain an ε-immortal subgraph I_r of G with $I_r \cap S_{k+1} = \emptyset$. As $S \subseteq S_{k+1}$, $I_r \cap S = \emptyset$, as desired.

So assume that there is a vertex $v \in I_j$ which does not satisfy the first condition of Lemma 1. That is, $\mathcal{N}^-(v) \neq \emptyset$ and $\omega(\mathcal{N}^-(v) \cap I_j) \leq (1-\varepsilon)\omega(\mathcal{N}^-(v))$. As $v \notin S_{k+1}$, $\omega(\mathcal{N}^-(v) \cap S_k) < \varepsilon \cdot \omega(\mathcal{N}^-(v))$. Thus, $\omega(\mathcal{N}^-(v) \cap I_j) + \omega(\mathcal{N}^-(v) \cap S_k) < \omega(\mathcal{N}^-(v))$, and there has to be a vertex $u \in \mathcal{N}^-(v) \setminus (I_j \cup S_k)$. Since $S_k = S_{k+1}$, $u \notin I_j \cup S_{k+1}$. We obtain I_{j+1} by adding u to I_j. If vertex v does not satisfy the second condition of Lemma 1, a similar procedure finds a vertex $w \in \mathcal{N}^+(v) \setminus (I_j \cup S_{k+1})$ to be added to I_j. □

3 Deadly Sets for Arbitrary Directed Graphs

In order to prove the NP-hardness of all variants of the deadly set problem for arbitrary directed graphs, we provide a linear time reduction of the vertex cover problem to the unweighted deadly set problem. A *vertex cover* of an undirected graph G is a set C of vertices of G such that every edge has at least one endpoint in C. It has been shown that finding such a set C of minimum cardinality is NP-hard (e.g., see [6]).

Theorem 1 *The unweighted and weighted deadly and ε-deadly set problems, $0 < \varepsilon < 1$, are NP-hard on directed graphs.*

Proof. Given an undirected graph G, we construct a directed graph $G' = (V, E')$ with edge set $E' = \{(v,w),(w,v) : \{v,w\} \in E\}$. Every edge in G corresponds to a cycle in G'. Thus, every deadly set for G' covers every edge in G because cycles are immortal. Conversely, if a vertex cover does not contain a vertex v of G, all neighbors of v must be in the vertex cover. Thus, every vertex cover for G is deadly for G'. □

4 Deadly Sets for Directed Acyclic Graphs

Next we provide a linear time reduction of the problem of finding a minimum deadly set for a given DAG to that of finding a minimum cut in an *st*-graph constructed from the given DAG. A *cut* of an *st*-graph $G = (V,E)$ is a set $C \subseteq E$ of edges such that any path from s to t contains at least one edge in C. As such a minimum cut can be found in $O(nm\log(n^2/m))$ time [2], this provides an efficient solution for the deadly set problem on DAGs.

Theorem 2 *It takes $O(MC(n,m))$ time to solve the weighted deadly set problem on directed acyclic graphs, where $MC(n,m)$ is the time required to compute a minimum cut in an st-graph with n vertices and m edges.*

Proof. Let a DAG $G = (V,E)$ and a function $\omega : V \to \mathbb{R}^+$ assigning weights to the vertices of G be given. Assume that G does not contain isolated vertices. We show how to deal with the general case at the end of the proof. We construct an *st*-graph $G' = (V', E')$ with vertex set $V' = (V \times \{0,1\}) \cup \{s,t\}$ and edge set

$$\begin{aligned} E' = &\{(s,(v,0)) : v \text{ is a source in } G\} \cup \\ &\{((w,1),t) : w \text{ is a sink in } G\} \cup \\ &\{((v,0),(v,1)) : v \in V\} \cup \\ &\{((v,1),(w,0)) : (v,w) \in E\}. \end{aligned}$$

(See Figure 1.) We define a function $\omega' : E' \to \mathbb{R}^+$ assigning weights to the edges of G' as

$$\omega'(e) = \begin{cases} \omega(v) & \text{if } e = (s,(v,0)) \\ \omega(w) & \text{if } e = ((w,1),t) \\ \omega(v) & \text{if } e = ((v,0),(v,1)) \\ \max(\omega(v),\omega(w)) & \text{if } e = ((v,1),(w,0)) \end{cases}.$$

We show how to construct a deadly set S of G from a cut C of G' and that S is a minimum deadly set of G if an only if C is a minimum cut for G'.

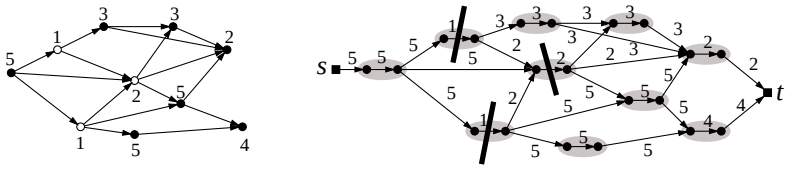

Figure 1: A graph $G = (V,E)$ (left) and its corresponding graph $G' = (V',E')$ constructed in the proof of Theorem 2. Any st-path in G' must cross one of the three solid lines representing a minimum st-cut of G'. The white vertices in G represent the corresponding minimum deadly set of G.

Given a cut C of G', we construct a deadly set S for G as follows: For every edge $e' = (v',w') \in C$, we add v to S if $v' = (v,x)$ for some $v \in V$ and $x \in \{0,1\}$. Otherwise, $e' = (s,(w,0))$, and we add w to S.

First we prove that S is deadly for G. Assume the contrary. That is, there is an immortal subgraph I of G such that $I \cap S = \emptyset$. A simple inductive argument shows that I must contain a source-to-sink path $P = (v_0,\ldots,v_k)$ as a subgraph. Then the corresponding st-path $P' = (s,(v_0,0),(v_0,1),(v_1,0),(v_1,1),\ldots,(v_k,0),(v_k,1),t)$ in G' cannot contain an edge in C, a contradiction.

It is easy to see that $\omega(S) \leq \omega'(C)$. Thus, the weight of a minimum deadly set of G does not exceed the weight of a minimum cut in G'. Now assume that C is a minimum cut of G', and there exists a minimum deadly set S' of G such that $\omega(S') < \omega'(C)$. Let C' be the edge set $\{((v,0),(v,1)) : v \in S'\}$. Thus, $\omega'(C') = \omega(S') < \omega'(C)$. We show that C' is a cut for G', thereby contradicting the minimality of C.

So assume that there exists an st-path $(s,(v_0,0),(v_0,1),(v_1,0),(v_1,1),\ldots,(v_k,0),(v_k,1),t)$ in G' that does not contain a single edge in C'. Then the corresponding source-to-sink path $P' = (v_0,v_1,\ldots,v_k)$ in G cannot contain a vertex in S'. But this is a contradiction, as P' is immortal.

If G contains isolated vertices, let H be the subgraph of G obtained by removing all isolated vertices. By Lemmas 1 and 2, every isolated vertex has to be contained in any deadly set for G. Thus, it is sufficient to find a deadly set for H. Let $n' \leq n$ be the number of vertices in H, and $m' \leq m$ be the number of edges in H. Then $m' \geq n'/2$. The graph H' constructed from H by the above procedure has $2n' + 2 = O(n)$ vertices and at most $3n' + m' = O(m)$ edges. Thus, finding a minimum deadly set for G takes $O(n+m) + MC(O(n),O(m)) = O(MC(n,m))$ time. □

5 ε-Deadly Sets for Directed Acyclic Graphs

We now show that the ε-deadly set problem is NP-hard even for a restricted subclass of directed acyclic graphs whose vertices have weights within a restricted range.

Theorem 3 *The weighted ε-deadly set problem, $0 < \varepsilon < 1$, is NP-hard for directed acyclic graphs whose nodes have in-degree and out-degree bounded by some constant D and weights between w and W, provided that $D \geq 2 + \max(\lceil \frac{2w}{\varepsilon W} \rceil, \lceil \frac{\varepsilon w}{(1-\varepsilon)W} \rceil)$.*

We prove Theorem 3 by constructing a graph $G(F,\varepsilon)$ for a given Boolean formula F in 3-CNF such that F is satisfiable if and only if $G(F,\varepsilon)$ has an ε-deadly set of low weight. We show that $G(F,\varepsilon)$ can be constructed in time polynomial in the size of F and ensure that it satisfies the conditions of Theorem 3.

We need the following definitions and results. Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of Boolean variables, a Boolean formula $F(x_1, \ldots, x_n)$ is in conjunctive normal form with clauses of size exactly 3 (3-CNF) if

1. $F(x_1, \ldots, x_n) = C_1 \wedge C_2 \wedge \cdots \wedge C_m$,
2. $C_i = \lambda_{i,1} \vee \lambda_{i,2} \vee \lambda_{i,3}$, for $1 \leq i \leq m$, and
3. For all pairs (i, j), $1 \leq i \leq m$, $1 \leq j \leq 3$, there exists a k, $1 \leq k \leq n$, such that $\lambda_{i,j} = x_k$ or $\lambda_{i,j} = \overline{x_k}$.

A Boolean formula $F(x_1, \ldots, x_n)$ is satisfiable if there is a truth assignment $\beta : X \to \{\textbf{true}, \textbf{false}\}$ such that $F(\beta(x_1), \ldots, \beta(x_n)) = \textbf{true}$. It has been shown that the problem of deciding whether a given formula F in 3-CNF (3-SAT) is satisfiable is NP-hard (e.g., see [5]).

Given formula F, the graph $G(F,\varepsilon)$ consists of a number of subgraphs, called *gadgets*. Most of these gadgets are duplicated, so that we can think of $G(F,\varepsilon)$ as being composed of $2n$ identical *layers* $L_1, \ldots, L_k$ (see Figure 2). Each of these layers is comprised of $2n$ *multiplier gadgets*, n *negator gadgets*, m *clause gadgets*, one *final gadget*, and one *feedback gadget*. We add $2n$ *literal gadgets* to $G(F,\varepsilon)$, which are connected to all layers of $G(F,\varepsilon)$. We describe $G(F,\varepsilon)$ for the case $\varepsilon \leq \frac{1}{2}$. For $\varepsilon > \frac{1}{2}$, $G(F,\varepsilon)$ contains an additional *booster gadget*. Details will appear in the full paper.

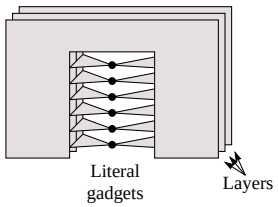

Figure 2: The global structure of $G(F,\varepsilon)$.

We first describe the gadgets in one layer L_i and show how they are connected to form layer L_i. The other layers are identical. Then we describe the literal gadgets and show how they are linked to the layers. Assume that we have normalized w and W so that $w = 1$. All nodes in $G(F,\varepsilon)$ have weight 1 unless stated otherwise.

There is one *multiplier gadget* in L_i for each of the $2n$ literals $x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}$ over X (see Figure 3a). The multiplier gadget for literal λ is a binary tree Λ_i with root $\hat{\lambda}_i$. The edges of Λ_i are directed from the root towards the leaves. There is one more leaf in Λ_i than clauses in F containing λ. We augment Λ_i with additional nodes. In particular, every internal node of Λ_i has $\lceil \frac{2}{\varepsilon W} \rceil$ additional leaves of weight W as children. These extra children serve as *feedback blockers* and are *not* considered regular leaves of Λ_i.

There is one *negator gadget* in L_i for each of the n variables in X (see Figure 3b). The negator gadget for variable x_j consists of a single *negator node* $\chi_{i,j}$.

There is one *clause gadget* per clause C_j in F (see Figure 3c). Each such clause gadget consists of two nodes $\gamma_{i,j}$ and $\hat{\gamma}_{i,j}$ and an edge from $\hat{\gamma}_{i,j}$ to $\gamma_{i,j}$. Node $\gamma_{i,j}$ is called the *clause node* for clause C_j in layer L_i.

There is one *final gadget* in L_i (see Figure 3f). It consists of a binary tree Ψ_i with root $\hat{\psi}_i$, $k = \lceil \frac{\varepsilon}{(1-\varepsilon)W} \rceil$ nodes $\psi'_{i,1}, \ldots, \psi'_{i,k}$ of weight W each, and k *final nodes* $\psi_{i,1}, \ldots, \psi_{i,k}$ of weight W each. We add edges from $\hat{\psi}_i$ to all nodes $\psi'_{i,1}, \ldots, \psi'_{i,k}$ and edges $(\psi'_{i,j}, \psi_{i,h})$, $1 \leq j,h \leq k$. There is one leaf in Ψ_i for every clause in F and every variable in X, $n+m$ in total. The edges in Ψ_i are directed from the leaves towards the root. Every internal node of Ψ_i has a number of additional children; the purpose of these children is to increase the in-weight of every regular node in Ψ_i, in order to enforce a logical AND behavior of the nodes in Ψ. These additional children are *not* considered regular leaves of Ψ_i. Their number depends on W and ε. In particular, if $W \leq \frac{1}{\varepsilon}$, every node has $\lfloor \frac{2-2\varepsilon}{\varepsilon W} \rfloor$ additional children of weight W each. If $\frac{1}{\varepsilon} < W < \frac{2-2\varepsilon}{\varepsilon}$, every node in Ψ_i has 2 additional children of weight $\frac{1-\varepsilon}{\varepsilon}$. If $W \geq \frac{2-2\varepsilon}{\varepsilon}$, every node in Ψ_i has one extra child of weight $\frac{2-2\varepsilon}{\varepsilon}$.

There is one *feedback gadget* in L_i (see Figure 3d). It consists of a binary tree Φ_i with root $\hat{\phi}_i$ and a *feedback node* ϕ_i. There is an edge from ϕ_i to $\hat{\phi}_i$. Tree Φ_i has one leaf for each of the $2n$ literals over X. The edges in Φ_i are directed from the root towards the leaves. Every internal node of Φ_i has $\lceil \frac{2}{\varepsilon W} \rceil$ additional children of weight W that serve as feedback blockers and are *not* considered to be regular leaves of Φ_i.

We connect these gadgets to form layer L_i. We add edges from the leaves in the multiplier gadgets to the nodes in the clause gadgets as follows: Consider a clause gadget corresponding to clause $C_j = \lambda_{j,1} \vee \lambda_{j,2} \vee \lambda_{j,3}$. Then we add edges from two leaves in trees $\Lambda_{i,j,1}$ and $\Lambda_{i,j,2}$ to $\hat{\gamma}_{i,j}$ and an edge from a leaf in tree $\Lambda_{i,j,3}$ to $\gamma_{i,j}$. Every leaf in each tree Λ_i is connected to at most one node in a clause gadget. This leaves one regular leaf per tree Λ_i with out-degree 0. We add edges from the remaining two leaves in trees $X_{i,j}$ and $\overline{X}_{i,j}$ to the negator node $\chi_{i,j}$, for every variable $x_j \in X$. We add edges from the clause nodes in all clause gadgets and the negator nodes in all negator gadgets to the leaves of the final gadget, so that every leaf in the final gadget is connected to exactly one node in a clause or negator gadget, and vice versa. We add edges from the feedback node ϕ_i to all final nodes $\psi_{i,1}, \ldots, \psi_{i,k}$.

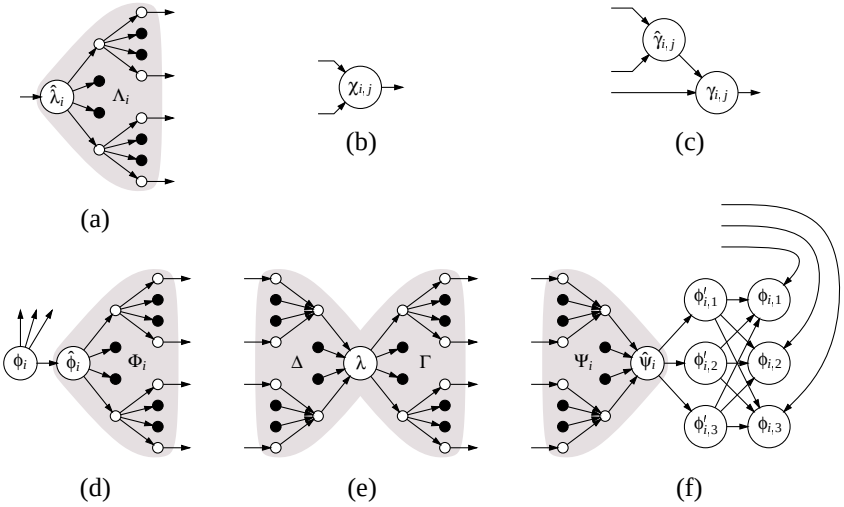

Figure 3: The gadgets in $G(F,\varepsilon)$. Regular nodes are white. Additional nodes such as feedback blockers are black. (a) A multiplier gadget. (b) A negator gadget. (c) A clause gadget. (d) A feedback gadget. (e) A literal gadget. (f) A final gadget.

Apart from layers $L_1,\ldots,L_k$, graph $G(F,\varepsilon)$ contains a number of literal gadgets. There is one *literal gadget* for each of the $2n$ literals over the set X of variables in F (see Figure 3e). The literal gadget for literal λ consists of two binary trees Δ and Γ sharing the same root λ. The edges in Δ are directed from the leaves towards the root; the edges in Γ are directed from the root towards the leaves. Both trees have $2n$ leaves. Let μ_1 and μ_2 be the children of λ in Δ, and ν_1 and ν_2 be the children of λ in Γ. Then we guarantee that each of μ_1, μ_2, ν_1, and ν_2 has n leaves as descendants. Let the descendant leaves of μ_1 and ν_1 be numbered 1 through n, and the descendant leaves of μ_2 and ν_2 be numbered $n+1$ through $2n$. The internal nodes in Γ have $\lceil \frac{2}{\varepsilon W} \rceil$ additional children of weight W each. The internal nodes in Δ have the same number of additional children as the internal nodes in tree Ψ_i in a final gadget. Again, these additional children are *not* considered to be regular nodes of Δ and Γ.

To finish the construction of $G(F,\varepsilon)$, we have to connect the literal gadgets with layers $L_1,\ldots,L_k$. To do that we add edges between the leaves in all feedback gadgets and the leaves in trees Δ, and between the leaves in trees Γ and the roots of the trees in all multiplier gadgets. More precisely, if $\lambda = x_j$, we connect the leaf corresponding to λ in tree Φ_i to the i-th leaf of tree Δ, and the i-th leaf of tree Γ to node $\hat{\lambda}_i$; if $\lambda = \overline{x}_j$, we connect the leaf corresponding to λ in tree Φ_i to the $((i+n/2) \bmod 2n)$-th leaf of tree Δ, and the $((i+n/2) \bmod 2n)$-th leaf of tree Γ to node $\hat{\lambda}_i$.

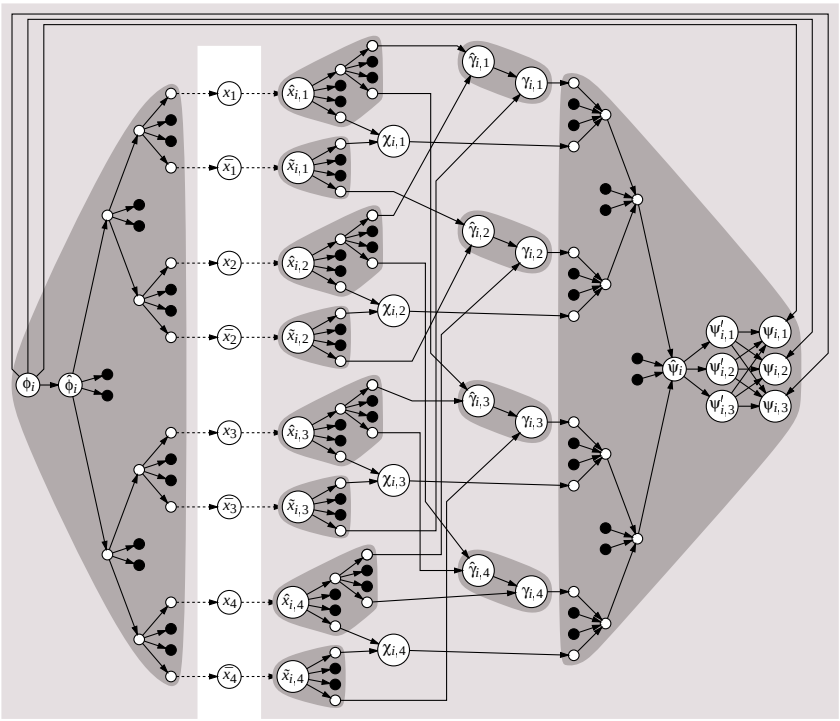

Figure 4: One layer L_i of graph $G(F,\varepsilon)$, for $F = (x_1 \vee x_2 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_4) \wedge (x_1 \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee x_3 \vee x_4)$ and $\varepsilon \leq \frac{1}{2}$. The connections of nodes in L_i to literal gadgets are shown as dashed lines. Feedback, multiplier, clause, and final gadgets are shaded.

To illustrate this construction at an example, Figure 4 shows one layer of the graph $G(F,\varepsilon)$ constructed for the formula $F = (x_1 \vee x_2 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_4) \wedge (x_1 \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee x_3 \vee x_4)$ and $\varepsilon \leq \frac{1}{2}$. Clearly, $G(F,\varepsilon)$ can be constructed in $O(n(n+m)/\varepsilon')$ time from F, where $\varepsilon' = \min(1-\varepsilon,\varepsilon)$. Also, it is easily verified that $G(F,\varepsilon)$ satisfies the conditions of Theorem 3. Next we prove a number of properties of the graph $G(F,\varepsilon)$ which, together with the NP-hardness of 3-SAT, provide the proof for Theorem 3.

Lemma 3 *Every* ε*-deadly set of* $G(F,\varepsilon)$ *of weight at most* n *contains exactly one of the nodes* x_j *and* $\bar{x_j}$*, for all* $x_j \in X$.

Proof. Assume that there exists an ε-deadly set S of weight at most n that does not satisfy the condition of the lemma. We show that there is an ε-immortal subgraph I with $I \cap S = \emptyset$, thereby contradicting the ε-deadliness of S.

First assume that there exists a variable $x_j \in X$ such that none of the literal gadgets for x_j and $\bar{x_j}$ contains a node in S. As there are $2n$ layers, and no node has weight less than 1, there has to be a layer L_i such that $L_i \cap S = \emptyset$. Let H be the subgraph of $G(F,\varepsilon)$ consisting of the paths from ϕ_i to x_j and $\bar{x_j}$, the paths from x_j and $\bar{x_j}$ to $\chi_{i,j}$, and the paths from $\chi_{i,j}$ to all final nodes $\psi_{i,h}$ in L_i. Let I be the graph induced by all nodes in H and all sinks and sources adjacent to nodes in H. Then it is easy to verify that I satisfies the conditions of Lemma 1 and that $I \cap S = \emptyset$. Hence, by Lemma 2, S cannot be ε-deadly.

Thus, S contains exactly one node in the literal gadgets for x_j and $\bar{x_j}$, for every variable $x_j \in X$. Now assume that there exists a variable x_j such that $x_j, \bar{x_j} \notin S$. Then there has to be a layer L_i such that the paths from ϕ_i to x_j and $\bar{x_j}$ and from x_j and $\bar{x_j}$ to $\chi_{i,j}$ do not contain a node in S. Also, $L_i \cap S = \emptyset$. Thus, $I \cap S = \emptyset$, for the same subgraph I as constructed above, and S cannot be ε-deadly. □

Lemma 3 implies that $G(F,\varepsilon)$ does not have an ε-deadly set of weight less than n. It also implies that every ε-deadly set S of $G(F,\varepsilon)$ of weight n corresponds to an assignment of truth values to the variables in X. In particular, we define $\beta(x_j) = \textbf{true}$ if $x_j \in S$, and $\beta(x_j) = \textbf{false}$ if $\bar{x_j} \in S$. Conversely, every assignment $\beta : X \to \{\textbf{true}, \textbf{false}\}$ of truth values to the variables in X corresponds to a set $S_\beta = \{x_j : \beta(x_j) = \textbf{true}\} \cup \{\bar{x_j} : \beta(x_j) = \textbf{false}\}$.

Lemma 4 *The set S_β is ε-deadly for $G(F,\varepsilon)$ if and only if $F(\beta(x_1),\ldots,\beta(x_n)) =$* **true**.

Proof. It is easy to verify that the set S_β is ε-deadly for graph $G(F,\varepsilon)$ if $F(\beta(x_1),\ldots,\beta(x_n)) = \textbf{true}$. So assume that $F(\beta(x_1),\ldots,\beta(x_n)) = \textbf{false}$. Then there has to be a clause $C_j = \lambda_{j,1} \vee \lambda_{j,2} \vee \lambda_{j,3}$ such that $\beta(C_j) = \beta(\lambda_{j,1}) \vee \beta(\lambda_{j,2}) \vee \beta(\lambda_{j,3}) = \textbf{false}$. Let H be the subgraph of $G(F,\varepsilon)$ induced by the paths from ϕ_i to nodes $\lambda_{j,1}$, $\lambda_{j,2}$, and $\lambda_{j,3}$, the paths from $\lambda_{j,1}$, $\lambda_{j,2}$, $\lambda_{j,3}$ to $\gamma_{i,j}$, and the paths from $\gamma_{i,j}$ to all final nodes $\psi_{i,h}$ in layer L_i. Let I be the subgraph of $G(F,\varepsilon)$ induced by all nodes in H as well as all sources and sinks adjacent to vertices in H. Then it is easy to verify that I satisfies the conditions of Lemma 1 and that $I \cap S_\beta = \emptyset$. Thus, S_β cannot be ε-deadly for $G(F,\varepsilon)$, by Lemma 2. □

Lemmas 3 and 4 together imply that $G(F,\varepsilon)$ has an ε-deadly set of weight n if and only if F is satisfiable. If F is not satisfiable, every ε-deadly set of $G(F,\varepsilon)$ has weight greater than n. Thus, we can decide whether F is satisfiable only by deciding whether $G(F,\varepsilon)$ has an ε-deadly set of weight n. An algorithm that computes an ε-deadly set of this weight actually gives a truth assignment β which satisfies F. As the size of $G(F,\varepsilon)$ is polynomial in the size of F, and it takes time polynomial in the size of F to construct $G(F,\varepsilon)$, this proves Theorem 3. The following two corollaries are immediate consequences of Theorem 3.

Corollary 1 *The weighted ε-deadly set problem, $0 < \varepsilon < 1$, is NP-hard for directed acyclic graphs whose nodes have in-degree and out-degree at most 3.*

Corollary 2 *The unweighted ε-deadly set problem, $0 < \varepsilon < 1$, is NP-hard for directed acyclic graphs whose vertices have in-degree and out-degree at most $2 + \lceil 2/\varepsilon' \rceil$, where $\varepsilon' = \min(\varepsilon, (2-2\varepsilon)/\varepsilon)$.*

6 Conclusion

Our negative results of Sections 3 and 5 are very strong in the following sense. Our proof in Section 3 implies that for any class of undirected graphs for which the vertex cover problem is NP-hard, the deadly set problem is NP-hard for the corresponding class of directed graphs. As mentioned at the end of Section 5, the construction of that section implies that in general even just deciding whether a given DAG has an ε-deadly set of weight (or size) at most k is NP-hard. An algorithm that computes a minimum deadly set is also able to compute a satisfying truth assignment for a given formula F in 3-CNF.

From a practical point of view, the most important open question is the existence of efficient approximation algorithms. We did not investigate this question.

References

[1] AGUR, Z., FRAENKEL, A. S., AND KLEIN, S. T. The number of fixed points of the majority rule. *Discrete Mathematics 70* (1988), 295–302.

[2] AHUJA, R. K., ORLIN, J. B., AND TARJAN, R. E. Improved time bounds for the maximum flow problem. *SIAM J. Comput. 18* (1989), 939–954.

[3] BERMOND, J. C., BOND, J., PELEG, D., AND PERENNES, S. Tight bounds on the size of 2-monopolies. In *Proc. of 3rd Colloquium on Structural Information and Communication Complexity* (1996), pp. 170–179.

[4] BERMOND, J. C., AND PELEG, D. The power of small coalitions in graphs. In *Proc. of 2nd Colloquium on Structural Information and Communication Complexity* (1995), pp. 173–184.

[5] CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. *Introduction to Algorithms*. MIT Press, 1990.

[6] EVEN, S. *Graph Algorithms*. Computer Science Press, 1979.

[7] FLOCCHINI, P., GEURTS, F., AND SANTORO, N. Irreversible dynamos in chordal rings. *Discrete Applied Mathematics* (2001). to appear.

[8] FLOCCHINI, P., KRALOVIC, R., RONCATO, A., RUZICKA, P., AND SANTORO, N. On time versus size for monotone dynamic monopolies. In *Proc. of 7th Colloquium on Structural Information and Communication Complexity* (2000), pp. 111–126.

[9] FLOCCHINI, P., LODI, E., LUCCIO, F., PAGLI, L., AND SANTORO, N. Irreversible dynamos in tori. In *Proc. EUROPAR 98* (1998), pp. 554–562.

[10] FLOCCHINI, P., LODI, E., LUCCIO, F., PAGLI, L., AND SANTORO, N. Monotone dynamos in tori. In *Proc. of 6th Colloquium on Structural Information and Communication Complexity* (1999), pp. 152–165.

[11] GOLES, E., AND OLIVOS, J. Periodic behavior of generalized threshold functions. *Discrete Mathematics 30* (1980), 187–189.

[12] HASSIN, Y., AND PELEG, D. Distributed probabilistic polling and applications to proportionate agreement. In *Proc. of 26th Int. Colloquium on Automata, Languages, and Programming* (1999), pp. 402–411.

[13] HASSIN, Y., AND PELEG, D. Extremal bounds for probabilistic polling in graphs. In *Proc. of 7th Colloquium on Structural Information and Communication Complexity* (2000), pp. 167–180.

[14] LINIAL, N., PELEG, D., RABINOVICH, Y., AND SACHS, M. Sphere packing and local majority in graphs. In *Proc. of 2nd ISTCS* (1993), pp. 141–149.

[15] LUCCIO, F., PAGLI, L., AND SANOSSIAN, H. Irreversible dynamos in butterflies. In *Proc. of 6th Colloquium on Structural Information and Communication Complexity* (1999), pp. 204–218.

[16] MORAN, G. The r-majority vote action on 0–1 sequences. *Discrete Mathematics 132* (1994), 145–174.

[17] NAKATA, T., IMAHAYASHI, H., AND YAMASHITA, M. Probabilistic local majority voting for the agreement problem in finite graphs. In *Proc. of 5th Computing and Combinatorics Conference* (1999), pp. 330–338.

[18] NAYAK, A., PAGLI, L., AND SANTORO, N. On testing for catastrophic faults in reconfigurable arrays with arbitrary link redundancy. *INTEGRATION, the VLSI Journal 20* (1996), 327–342.

[19] NAYAK, A., REN, J., AND SANTORO, N. An improved testing scheme for catastrophic fault patterns. *Information Processing Letters 73* (2000), 199–206.

[20] NAYAK, A., SANTORO, N., AND TAN, R. Fault intolerance of reconfigurable systolic arrays. In *Proc. of 20th Int. Symposium on Fault-Tolerant Computing* (1990), pp. 202–209.

[21] PAGLI, L., AND PUCCI, G. Counting the number of fault patterns in redundant VLSI arrays. *Information Processing Letters 50* (1994), 337–342.

[22] PELEG, D. Local majority voting, small coalitions and controlling monopolies in graphs: A review. In *Proc. of 3rd Colloquium on Structural Information and Communication Complexity* (1997), pp. 152–169.

[23] PELEG, D. Size bounds for dynamic monopolies. *Discrete Applied Mathematics 86* (1998), 263–273.

[24] PELEG, D. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, 2000.

[25] POLJAK, S. Transformations on graphs and convexity. *Complex Systems 1* (1987), 1021–1033.

[26] DE PRISCO, R., MONTI, A., AND PAGLI, L. Efficient testing and reconfiguration of VLSI linear arrays. *Theoretical Computer Science 197* (1998), 171–188.

[27] DE PRISCO, R., AND DE SANTIS, A. Catastrophic faults in reconfigurable VLSI linear arrays. *Discrete Applied Mathematics 75* (1997), 105–123.

Norbert Zeh is a PhD student at the School of Computer Science, Carleton University, 1125 Colonel By Dr, Ottawa, Ontario, K1S 5B6, Canada.
Email: nzeh@scs.carleton.ca

Nicola Santoro is a faculty member at the School of Computer Science, Carleton University, 1125 Colonel By Dr, Ottawa, Ontario, K1S 5B6, Canada.
Email: santoro@scs.carleton.ca