On Finding Minimum Deadly Sets for Directed Networks

Norbert Zeh* Nicola Santoro*

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 in the weighted and unweighted scenarios. For directed acyclic graphs (DAGs), we show that a weighted minimum deadly set can be found in $O(MC(n,m))$ time, where $MC(n,m)$ is the time required to compute a minimum cut for a DAG with n vertices and m edges. 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.

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 [23]; 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 initally 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 can 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, 12, 17, 26]).

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.

*School of Computer Science, Carleton Unviersity, Ottawa, Canada, {nzeh,santoro}@scs.carleton.ca

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

In terms of system dynamics, these are the patterns for which the system converges to a *monochromatic* fixed point. These patterns have been called *dynamos* by Peleg who introduced their study in [25]. Most of the results are known for the static version of this process, that is, considering only one or two steps in the evolution [2, 3, 15, 23]. In the dynamic case, results on dynamos exist for a variety of topologies, including chordal rings, tori, and most interconnection networks [8, 9, 10, 11, 16, 25]. Recently nondeterministic rules and weighted majority functions have been investigated in [13, 14, 18].

All these studies have almost exclusively considered *undirected networks*, assuming the neighborhood relationship to be symmetric. 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 [21]. 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 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, on the complexity of recognizing whether a set of faults is deadly, on the efficient generation of such CFP, on how to route if the pattern is not catastrophic, etc. [5, 6, 19, 20, 21, 22].

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 our study, we model a given directed network as a directed graph G. The vertices represent the elements of the network. There is an edge from node v to node w in G if and only if there is a link *from v to w* in the network. Given a set of faulty nodes in G, a non-faulty node becomes faulty if all its in-neighbors or all its out-neighbors are faulty. We call a set S of nodes *deadly* if repeated application of this rule, starting with the elements in S being faulty, eventually causes all nodes in G to become faulty. Set S is *minimum* if it has minimum cardinality. A more general version assigns real weights to the nodes of G; in this case a minimum deadly set is a deadly set of minimum weight. The (unweighted or weighted) *deadly set problem* is the problem of finding a minimum deadly set for a given graph. In many applications a node becomes faulty already if a certain percentage ε, $0 < \varepsilon < 1$, of its in-neighbors or out-neighbors is faulty. In the unweighted case this percentage is determined w.r.t. the cardinality of the in-neighborhood or out-neighborhood of each vertex; in the weighted case it is determined w.r.t. the total weight of the in-neighbors or out-neighbors of each vertex. We call a set of nodes ε-*deadly* if repeated application of this ε-majority rule, starting with the elements in S being faulty, eventually causes all nodes in G to become faulty. The (weighted or unweighted) ε-*deadly set problem* is the problem of finding a minimum ε-deadly set for a given directed graph.

In Section 3, we show that the weighted and unweighted deadly and ε-deadly set problems are NP-hard on arbitrary directed graphs. Due to the hardness of the general case, we then consider the class of directed acyclic graphs. These graphs are the least powerful from a reliability point of view because intuitively the absence of cycles should make "deadly" events more likely to occur. Hence, it should be simpler to solve the deadly set problem on DAGs. The results of Section 4 support this intuition only partially. In particular, we show in Section 4.1 that the weighted and unweighted deadly set problems can be solved in $O(MC(n,m))$ time on DAGs, where $MC(n,m)$ is the time required to compute a minimum cut for a DAG with n vertices

[2] The initially faulty elements are dead by definition; we are using the original terminology of [21] here.

and m edges. In Section 4.2 on the other hand, we show that the weighted ε-deadly set problem is NP-hard for DAGs with bounded in-degree and out-degree and with weights bounded by some constant, provided that the permissible ranges for vertex degrees and weights are not too restrictive at the same time. This implies that the weighted ε-deadly set problem is NP-hard for DAGs whose vertices have in-degree and out-degree at most 3. It also implies that the unweighted ε-deadly set problem is NP-hard for DAGs of bounded degree.

Definitions A *directed graph* $G = (V, E)$ 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 vertex v its *source vertex*, w its *target vertex*; both v and w are *endpoints* of (v, w). Edge (v, w) is said to be *incident* to v and w. Vertices 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 $1 \le i \le k$. 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.

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 and 4. Given an assignment $\omega : V \to \mathbb{R}^+$ of weights to the vertices of G, 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 $0 < \varepsilon \le 1$, where a vertex $v \in V$ is contained in set S_i, $i > 0$, if and only if at least one of the following conditions is true:

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

A set S is ε-*deadly* if there exists some $k \ge 0$ such that $S_i = V$, for all $i \ge k$. We call a subgraph I of G ε-*immortal* if $I \cap S = \emptyset$ implies that $I \cap S_i = \emptyset$, for all $i \ge 0$. Setting $\varepsilon = 1$, the above definitions are equivalent to the following.

A subset $S \subseteq V$ of vertices of a graph G induces a *propagation sequence* $S = S_0 \subseteq S_1 \subseteq S_2 \subseteq \ldots$ of subsets of V, where a vertex $v \in V$ is contained in set S_i, $i > 0$, if and only if at least one of the following conditions is true:

- $v \in S_{i-1}$,
- $\mathcal{N}^-(v) \neq \emptyset$ and $S_{i-1} \cap \mathcal{N}^-(v) = \mathcal{N}^-(v)$ (all in-neighbors of v are in S_{i-1}), or
- $\mathcal{N}^+(v) \neq \emptyset$ and $S_{i-1} \cap \mathcal{N}^+(v) = \mathcal{N}^+(v)$ (all out-neighbors of v are in S_{i-1}).

Set S is *deadly* if there exists some $k \ge 0$ such that $S_i = V$, for all $i \ge k$. We call a subgraph I of G *immortal* if $I \cap S = \emptyset$ implies that $I \cap S_i = \emptyset$, for all $i \ge 0$.

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:*

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

Proof. See Appendix. □

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 $S = S_0, S_1, S_2, \dots$ 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 $I_0, \dots, I_j$, 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. □

Depending on the choice of ε and the weight function ω, we define four variants of the deadly set problem:

Unweighted deadly set problem: Find a deadly set of minimum cardinality for graph G.

Weighted deadly set problem: Find a deadly set of minimum weight for graph G.

Unweighted ε-deadly set problem: Find an ε-deadly set of minimum cardinality for graph G, assuming that all vertices in G have unit weight.

Weighted ε-deadly set problem: Find an ε-deadly set of minimum weight for graph G.

3 Deadly Sets for Arbitrary Directed Graphs

In this section, we prove our negative result for arbitrary directed graphs. We do this by providing a linear time reduction from the vertex cover problem to the unweighted deadly set problem. As this is the easiest variant of the deadly set problem, the other three variants have to be NP-hard as well.

Given an undirected graph $G = (V, E)$, a *vertex cover* of G is a subset $C \subseteq V$ such that for every edge $\{v, w\} \in E$, $\{v, w\} \cap C \neq \emptyset$. The *vertex cover problem* is the problem of finding a vertex cover C such that there is no vertex cover C' of smaller cardinality. The following result together with our linear-time reduction implies the NP-hardness of the deadly set problem for directed graphs.

Theorem 1 [7] *The vertex cover problem is NP-complete.*

Theorem 2 *The unweighted and weighted deadly and* ε*-deadly set problems,* $0 < \varepsilon < 1$*, are NP-hard on directed graphs.*

Proof sketch. Given an undirected graph G, we construct a directed graph $G' = (V, E')$ with edge set $E' = \{(v,w),(w,v) : \{v,w\} \in E\}$. As every cycle of G' contains at least one edge of G, and there are no sources or sinks in G', every vertex cover of G must be deadly for G' by Lemmas 1 and 2. Conversely, every edge $\{v,w\}$ in G corresponds to a cycle (v,w,v) in G'. Hence, by Lemma 2, every ε-deadly set of G' must contain at least one of v and w and is thus a vertex cover for G. Thus, there is a one-to-one correspondence between vertex covers of G and ε-deadly sets of G'. Hence, a vertex cover for G is minimum if and only if it is a minimum deadly set for G'. $\square$

4 Deadly and ε-Deadly Sets for Directed Acyclic Graphs

Having proved a negative result for directed graphs in general, we now turn to the more restricted class of directed acyclic graphs. We show that the deadly set problem can be solved efficiently on these graphs, while the ε-deadly set problem remains NP-hard even if we restrict the class of graphs still further.

4.1 Deadly Sets

We provide an efficient solution for the deadly set problem by reducing the problem to that of finding a minimum cut in an *st*-graph whose size is linear in the size of the given DAG. Given a DAG $G = (V,E)$, a *separator* S of G is a set of vertices such that any source-to-sink path in G contains at least one vertex in S. A *cut* of an *st*-graph $G = (V,E)$ is a set $C \subseteq E$ of edges such that any *st*-path contains at least one edge in C. The following theorem is an immediate consequence of Lemmas 3 and 4 below.

Theorem 3 *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.*

Lemma 3 *Given a directed acyclic graph* $G = (V,E)$*, a subset* $S \subseteq V$ *of vertices in* G *is deadly for* G *if and only if it is a separator of* G*.*

Proof. Assume that S does not separate G. Then there is a path $P = (v_0, v_1, \dots, v_t)$ in G such that v_0 is a source, v_t is a sink, and $v_i \notin S$, for $0 \le i \le t$. Path P satisfies the conditions of Lemma 1 for $\varepsilon = 1$ and is thus immortal. As $P \cap S = \emptyset$, this implies that S is not deadly for G, by Lemma 2.

So assume that S is not deadly. By Lemma 2, there has to be an immortal subgraph I of G such that $I \cap S = \emptyset$. If $\deg_G^-(v) > 0$, for all vertices $v \in I$, I must contain a cycle because this implies that $\deg_I^-(v) > 0$, for all $v \in I$. Since I is a subgraph of G, this implies that G contains a cycle, contradicting the acyclicity of G. Thus, there has to be a vertex $v_0 \in I$ with $\deg_G^-(v_0) = 0$. We construct a path in I from v_0 to a vertex v_t with $\deg_G^+(v_t) = 0$. Assume that we have constructed a path $(v_0, \dots, v_i)$ in I. If $\deg_G^+(v_i) > 0$, then there has to be a vertex $v_{i+1} \in \mathcal{N}^+(v_i) \cap I$. As I is finite, we reach a situation where either $v_{t+1} = v_j$, for $j \le t$ or $\deg^+(v_t) = 0$. The former cannot happen because G is acyclic. Hence, $\deg^+(v_t) = 0$, and the path $(v_0, \dots, v_t)$ is a path from a source v_0 to a sink v_t in G such that $S \cap \{v_0, \dots, v_t\} = \emptyset$. Thus, S is not a separator. $\square$

Lemma 4 *Given a weighted DAG* $G = (V,E)$ *with* n *vertices and* m *edges and a function* $\omega : V \to \mathbb{R}^+$ *assigning weights to the vertices in* G*, it takes* $O(MC(n,m))$ *time to find a minimum weight separator of* G*, where* $MC(n,m)$ *denotes the time to compute a minimum cut for 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 separator S of G from a cut C of G' and that S is a minimum weight separator of G if an only if C is a minimum cut for G'.

Given a cut C of G', we construct a separator S of 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. Otherwise, $e' = (s,(w,0))$, and we add w to S.

First we prove that S is a separator of G. Assume for the sake of contradiction that S is not a separator. That is, there is a path $P = (v_0,\ldots,v_k)$ in G that does not contain a vertex in S; v_0 is a source, and v_k is a sink. 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 $\omega(S) \le \omega'(C)$. Thus, the weight of a minimum separator 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 separator S' 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, thereby contradicting the minimality of C.

Assume that there is 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 $(v_0,v_1,\ldots,v_k)$ in G cannot contain a vertex in S', a contradiction.

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 of G. Thus, it is sufficient to find a deadly set for H. Let $n' \le n$ be the number of vertices in H, and $m' \le m$ be the number of edges in H. Then $m' \ge n'/2$. 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 weight separator for G takes $O(n+m)+MC(O(n),O(m)) = O(MC(O(n),O(m)))$ time. $O(MC(O(n),O(m))) = O(MC(n,m))$ if $MC(n,m)$ is polynomial in n and m, which is the case. □

4.2 ε-Deadly Sets

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 4 *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 $D \ge \max(\lceil \frac{2w}{\varepsilon W} \rceil, \lceil \frac{\varepsilon w}{(1-\varepsilon)W} \rceil) + 2$.*

We prove Theorem 4 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 $G(F,\varepsilon)$ has size polynomial in the size of F and ensure that it satisfies the conditions of Theorem 4.

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

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

A Boolean formula $F(x_1,\dots,x_n)$ is satisfiable if there is a truth assignment $\beta : X \to \{\mathbf{true}, \mathbf{false}\}$ such that $F(\beta(x_1),\dots,\beta(x_n)) = \mathbf{true}$. 3-SAT is the problem of deciding for a given formula F in 3-CNF whether or not it is satisfiable.

Theorem 5 [4] *3-SAT is NP-complete.*

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 decomposed into $2n$ identical *layers* $L_1,\dots,L_k$ (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 $G(F,\varepsilon)$. If $\varepsilon > \frac{1}{2}$, we will also need a *booster gadget*, which is connected to all layers of $G(F,\varepsilon)$. Before describing these gadgets, we give some intuition about their respective purposes.

Every *literal gadget* represents one of the $2n$ literals $x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, x_n, \bar{x}_n$. The purpose of these gadgets is to provide an ε-deadly set of low weight for $G(F,\varepsilon)$ if F is satisfiable.

Every *multiplier gadget* is connected to one of the literal gadgets, say the one corresponding to x_i. The task of this gadget is to take the value of x_i and distribute it over several nodes, each of which represents the value of x_i in one of the clauses C_j.

Each *negator gadget* is connected to two multiplier gadgets corresponding to literals x_i and $\bar{x}_i$. Its purpose is to guarantee that an ε-deadly set of low weight contains at least one of the two literal nodes x_i and $\bar{x}_i$. That is, together all negator gadgets guarantee that an ε-deadly set of low weight corresponds to a valid assignment of truth values to the variables in X, as we will define "low weight" so that we cannot afford to add both x_i and $\bar{x}_i$ to the ε-deadly set.

Every *clause gadget* corresponds to a clause C_j in F. Its purpose is to guarantee that an ε-deadly set of low weight satisfies C_j. That is, together all clause gadgets guarantee that an ε-deadly set of low weight corresponds to an assignment of truth values to the variables in X that satisfies F.

The purpose of the *final gadget* is to link negator gadgets and clause gadgets in such a way that a set of low weight that is ε-deadly for the final gadget has the properties enforced by all negator and clause gadgets.

The *feedback gadgets* guarantee that an ε-deadly set of low weight that is deadly for the final gadgets in all layers, i.e., corresponds to an assignment of truth values to the variables in X that satisfies F, is ε-deadly for the whole graph $G(F,\varepsilon)$.

Some of the nodes in negator and clause gadgets have to be included in set S_{i+1} of an ε-propagation sequence even if only one of its regular input nodes is in S_i. This creates problems if $\varepsilon > \frac{1}{2}$. The *booster gadget* provides extra input nodes to all these nodes and guarantees that these input nodes can be included in S_i by adding only a single node of low weight to S.

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 and booster 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 $x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n$ over X (see Figure 5c). 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 $x_1, \ldots, x_n$ over X (see Figure 4b). 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 4c). 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 5b). 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. If $\varepsilon \leq \frac{1}{2}$, 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 5a). 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 $\bar{X}_{i,j}$ 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 $\psi_{i,1}, \ldots, \psi_{i,k}$.

Apart from layers $L_1, \ldots, L_k$, graph $G(F, \varepsilon)$ contains literal gadgets and possibly a booster gadget:

There is one *literal gadget* for each of the $2n$ literals over the set X of variables in F (see Figure 4a). 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. The internal nodes in Γ have $\lceil \frac{2}{\varepsilon W} \rceil$ additional children of weight W each. If $\varepsilon \leq \frac{1}{2}$, the internal nodes in Δ have a number of additional children each. Their weights and number are determined as for the extra children of internal nodes in tree Ψ_i in a final gadget. Again, these additional children are *not* considered regular nodes of Δ and Γ.

If $\varepsilon > \frac{1}{2}$, we need an additional *booster gadget* to make the negator and clause gadgets work (see Figure 5d). This booster gadget consists of a binary tree Ξ with root ξ. There are $2n(n+2m)\Omega$ leaves of weight ζ in Ξ, where $\Omega = \lceil \frac{2\varepsilon-1}{(1-\varepsilon)W} \rceil$ and $\zeta = W$ if $W \leq \frac{1}{1-\varepsilon}$, $\Omega = 2$ and $\zeta = \frac{2\varepsilon-1}{2-2\varepsilon}$ if $\frac{1}{1-\varepsilon} < W < \frac{2\varepsilon-1}{1-\varepsilon}$, and $\Omega = 1$ and $\zeta = \frac{2\varepsilon-1}{1-\varepsilon}$ if $W \geq \frac{2\varepsilon-1}{1-\varepsilon}$. The edges in Ξ are directed from the root towards the leaves.

To finish the construction of $G(F, \varepsilon)$, we have to connect the literal and booster gadgets with layers $L_1, \ldots, L_k$. To do that we add the following edges. We add edges connecting the leaves of all feedback gadgets to the leaves of trees Δ in the literal gadgets, so that a leaf in a feedback gadget corresponding to a literal λ is connected to exactly one leaf in the tree Δ of the literal gadget corresponding to λ, and every leaf in a literal gadget is connected to exactly one leaf in a feedback gadget. Similarly, we add edges from the leaves of trees Γ in the literal gadgets to the roots of the trees in all multiplier gadgets. In particular, a leaf in a tree Γ corresponding to literal λ is connected to exactly one node $\hat{\lambda}_i$, and every such node $\hat{\lambda}_i$ is connected to exactly one leaf in a tree Γ. If $\varepsilon > \frac{1}{2}$, let Ω be the constant computed in the description of the booster gadget. Then we add edges from the leaves of the booster gadget to all negator nodes $\chi_{i,j}$ and all nodes $\gamma_{i,j}$ and $\hat{\gamma}_{i,j}$ in clause gadgets so that every leaf in the booster gadget is connected to exactly one node, and every node $\chi_{i,j}$, $\gamma_{i,j}$, or $\hat{\gamma}_{i,j}$ is connected to Ω leaves in Ξ. Also, we ensure that if ξ_1 and ξ_2 are the children of ξ in Ξ, then ξ_1 and ξ_2 have at least one descendant leaf each that is connected to a negator node $\chi_{i,j}$.

To illustrate this construction at an example, Figure 3 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 4. Next we prove a number of properties of the graph $G(F,\varepsilon)$ which, together with Theorem 5, provide the proof for Theorem 4.

Lemma 5 *Every ε-deadly set of $G(F,\varepsilon)$ of weight at most $n + \lceil 2\varepsilon \rceil - 1$ contains exactly one of x_j and $\bar{x}_j$, for all $x_j \in X$, and the node ξ if $\varepsilon > \frac{1}{2}$.*

Proof. We prove the lemma for $\varepsilon > \frac{1}{2}$. (The case $\varepsilon \leq \frac{1}{2}$ is easier.) Assume that there exists an ε-deadly set S of weight at most $n+1$ that does not satisfy the conditions of the lemma. We consider all possible cases and show that in each of them, 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_1 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_1 be the graph induced by all nodes in H_1 and all sinks and sources adjacent to nodes in H_1. Then it is easy to verify that I_1 satisfies the conditions of Lemma 1 and that $I_1 \cap S = \emptyset$. Thus, by Lemmas 1 and 2, S cannot be ε-deadly.

Thus, S contains at least one node in the literal gadgets for x_j and $\bar{x}_j$, for every variable $x_j \in X$. Now assume that $\xi \notin S$. Then there has to be a layer L_i and a variable x_j such that $L_i \cap S = \emptyset$, one of the literal gadgets for x_j and $\bar{x}_j$ does not contain a node in S, and the path P from ξ to $\chi_{i,j}$ does not contain a node in S. Assume w.l.o.g. that the literal gadget for x_j does not containa node in S. Then let H_2 be the subgraph consisting of P, the path from ϕ_i to x_j, the path from 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_2 be the subgraph of $G(F,\varepsilon)$ induced by the nodes in H_2 and all sources and sinks adjacent to nodes in H_2. As above it is easy to verify that I_2 is ε-immortal and $S \cap I_2 = \emptyset$. Thus, S cannot be ε-deadly.

Thus, S contains node ξ and exactly one node in the literal gadgets corresponding to x_j and $\bar{x}_j$, for every variable $x_j \in X$. Assume that there exists a variable x_j such that $x_j, \bar{x}_j \notin X$. Then there has to be a layer L_i such that the paths 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, $I_1 \cap S = \emptyset$, for the same subgraph I_1 as constructed above, and S cannot be ε-deadly. □

Lemma 5 implies that $G(F,\varepsilon)$ does not have an ε-deadly set of weight less than $n + \lceil 2\varepsilon \rceil - 1$. It also implies that every ε-deadly set S of $G(F,\varepsilon)$ of weight $n + \lceil 2\varepsilon \rceil - 1$ corresponds to an assignment of truth values to the variables in X. In particular, we define $\beta(x_j) = \mathbf{true}$ if $x_j \in S$, and $\beta(x_j) = \mathbf{false}$ if $\bar{x}_j \in S$. Conversely, every assignment $\beta : X \to \{\mathbf{true}, \mathbf{false}\}$ of truth values to the variables in X corresponds to a set $S_{\beta,\varepsilon} = \{x_j : \beta(x_j) = \mathbf{true}\} \cup \{\bar{x}_j : \beta(x_j) = \mathbf{false}\} \cup \Xi_\varepsilon$, where $\Xi_\varepsilon = \emptyset$ if $\varepsilon \leq \frac{1}{2}$, and $\Xi_\varepsilon = \{\xi\}$ if $\varepsilon > \frac{1}{2}$. The following lemma can be shown using similar arguments as those in the proof of Lemma 5 (see Appendix).

Lemma 6 *The set $S_{\beta,\varepsilon}$ is ε-deadly for $G(F,\varepsilon)$ if and only if $F(\beta(x_1), \ldots, \beta(x_n)) = $ **true**.*

Lemmas 5 and 6 together imply that $G(F,\varepsilon)$ has a minimum ε-deadly set of weight $n+\lceil 2\varepsilon\rceil-1$ if and only if F is satisfiable. If F is not satisfiable, every ε-deadly set of $G(F,\varepsilon)$ has weight greater than $n+\lceil 2\varepsilon\rceil-1$. Thus, we can decide whether F is satisfiable only by deciding whether $G(F,\varepsilon)$ has a minimum ε-deadly set of weight $n+\lceil 2\varepsilon\rceil-1$. An algorithm that computes a minimum 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, this proves Theorem 4. The following two corollaries are immediate consequences of Theorem 4.

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 $\varepsilon'=\min(\varepsilon,(2-2\varepsilon)/\varepsilon)$.*

5 Conclusion

Our negative results of Sections 3 and 4.2 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 4.2, 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.

However, we leave a number of interesting questions unanswered. In particular, our NP-hardness proof in Section 4.2 requires that $k=\Omega(\varepsilon n)$. For constant k, on the other hand, a simple algorithm takes $O(n^k(n+m)\log n)$ time[3] to compute a minimum ε-deadly set of weight at most k if it exists. This is sufficient to decide whether the given graph has an ε-deadly set of weight at most k; but it leaves a large gap between $k=O(1)$ and $k=\Omega(\varepsilon n)$.

Our construction in Section 3 produces the rather pathological situation where the constructed graph contains the reverse edge (w,v) for every edge (v,w) in the graph. Although we do not believe that the deadly set problem becomes solvable in polynomial time if we disallow this, we were not able to provide an NP-hardness proof under this weak restriction.

In terms of positive results, the most important open question is to find out whether the deadly set problem on DAGs is in fact easier than computing a minimum cut for an *st*-graph. We were able to reduce the problem of finding a minimum cut in an *st*-graph with n vertices and m edges to that of finding a deadly set in another DAG; but the number of vertices in the constructed DAG is not necessarily linear in the number of vertices of the given *st*-graph. Even if the weighted version of the two problems turn out to be equivalent, the unweighted deadly set problem on DAGs may be easier than finding a minimum cut.

References

[1] Z. Agur, A. S. Fraenkel, S. T. Klein. The number of fixed points of the majority rule. *Discrete Mathematics*, 70:295–302, 1988.

[2] J.C. Bermond, J. Bond, D. Peleg, S. Perennes. Tight bounds on the size of 2-monopolies. In *Proc. of 3rd Colloquium on Structural Information and Communication Complexity*, 170–179. 1996.

[3] J.C. Bermond, D. Peleg. The power of small coalitions in graphs. In *Proc. of 2nd Colloquium on Structural Information and Communication Complexity*, 173–184. 1995.

[3] Here, m is the number of edges in the graph, not the number of clauses in formula F.

[4] T.H. Cormen and C.E. Leiserson and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[5] R. De Prisco, A. De Santis. Catastrophic faults in reconfigurable VLSI linear arrays. *Discrete Applied Mathematics*, 75:105–123, 1997.

[6] R. De Prisco, A. Monti, L. Pagli. Efficient testing and reconfiguration of VLSI linear arrays. *Theoretical Computer Science*, 197:171-188, 1998.

[7] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[8] P. Flocchini, F. Geurts, N. Santoro. Irreversible dynamos in chordal rings. *Discrete Applied Mathematics*, to appear, 2001.

[9] P. Flocchini, R. Kralovic, A, Roncato, P. Ruzicka, N. Santoro. On time versus size for monotone dynamic monopolies. In *Proc. of 7th Colloquium on Structural Information and Communication Complexity*, 111–126, 2000.

[10] P. Flocchini, E. Lodi, F. Luccio, L. Pagli, N. Santoro. Irreversible dynamos in tori. *Proc. EUROPAR 98*, 554-562,1998.

[11] P. Flocchini, E. Lodi, F. Luccio, L. Pagli, N. Santoro. Monotone dynamos in tori. In *Proc. of 6th Colloquium on Structural Information and Communication Complexity*, 152-165, 1999.

[12] E. Goles, J. Olivos. Periodic behavior of generalized threshold functions. *Discrete Mathematics*, 30:187–189, 1980.

[13] Y. Hassin, D. Peleg. Distributed probabilistic polling and applications to proportionate agreement. In *Proc. of 26th International Colloquium on Automata, Languages and Programming*, 402–411, 1999.

[14] Y. Hassin, D. Peleg. Extremal bounds for probabilistic polling in graphs. In *Proc. of 7th Colloquium on Structural Information and Communication Complexity*, 167–180, 2000.

[15] N. Linial, D. Peleg, Y. Rabinovich, M. Sachs. Sphere packing and local majority in graphs In *Proc. of 2nd ISTCS*, 141-149, 1993.

[16] F. Luccio, L. Pagli, H. Sanossian. Irreversible dynamos in butterflies. In *Proc. of 6th Colloquium on Structural Information and Communication Complexity*, 204-218, 1999.

[17] G. Moran. The r-majority vote action on 0–1 sequences. *Discrete Mathematics*, 132:145–174, 1994.

[18] T. Nakata, H. Imahayashi, M. Yamashita. Probabilistic local majority voting for the agreement problem in finite graphs. In *Proc. of the 5th Computing and Combinatorics Conference*, 330–338, 1999.

[19] A. Nayak, L. Pagli, N. Santoro. On testing for catastrophic faults in reconfigurable arrays with arbitrary link redundancy. *INTEGRATION, the VLSI Journal*, 20:327–342, 1996.

[20] A. Nayak, J. Ren, N. Santoro. An improved testing scheme for catastrophic fault patterns. In *Information Processing Letters*, 73:199-206, 2000.

[21] A. Nayak, N. Santoro, and R. Tan. Fault intolerance of reconfigurable systolic arrays. In *Proc. of the 20th International Symposium on Fault-Tolerant Computing*, 202–209, 1990.

[22] L. Pagli, G. Pucci. Counting the number of fault patterns in redundant VLSI arrays. *Information Processing Letters*, 50:337–342, 1994.

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

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

[25] D. Peleg. Size bounds for dynamic monopolies *Discrete Applied Mathematics*, 86:263–273, 1998.

[26] S. Poljak. Transformations on graphs and convexity. *Complex Systems*, 1:1021–1033, 1987.

Appendix

Omitted Proofs

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:*

1. *Either $\mathcal{N}^-(v) = \emptyset$ or $\omega(\mathcal{N}^-(v) \cap I) > (1-\varepsilon)\omega(\mathcal{N}^-(v))$, and*
2. *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 $\omega(\mathcal{N}^-(v) \cap I) > (1-\varepsilon)\omega(\mathcal{N}^-(v))$, and $\mathcal{N}^+(v) \neq \emptyset$ implies that $\omega(\mathcal{N}^+(v) \cap I) > (1-\varepsilon)\omega(\mathcal{N}^+(v))$ because I is assumed to satisfy the conditions of the lemma. As $S_k \cap I = \emptyset$, this implies that $\omega(\mathcal{N}^-(v) \cap S_k) < \varepsilon \cdot \omega(\mathcal{N}^-(v))$ and $\omega(\mathcal{N}^+(v) \cap S_k) < \varepsilon \cdot \omega(\mathcal{N}^+(v))$, a contradiction.

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 6 *The set $S_{\beta,\varepsilon}$ 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 $S_{\beta,\varepsilon}$ is ε-deadly for $G(F,\varepsilon)$ if $F(\beta(x_1), \ldots, \beta(x_n)) =$ **true**. So assume that $F(\beta(x_1), \ldots, \beta(x_n)) =$ **false**. Then there has to be a clause $C_j = \lambda_{j,1} \vee \lambda_{j,2} \vee \lambda_{j,3}$ such that $\beta(C_j) = \beta(\lambda_{j,1}) \vee \beta(\lambda_{j,2}) \vee \beta(\lambda_{j,3}) =$ **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 all vertices in I satisfy the conditions of Lemma 1 and that $I \cap S_{\beta,\varepsilon} = \emptyset$. Thus, $S_{\beta,\varepsilon}$ cannot be ε-deadly for $G(F,\varepsilon)$, by Lemma 2. □

Figures

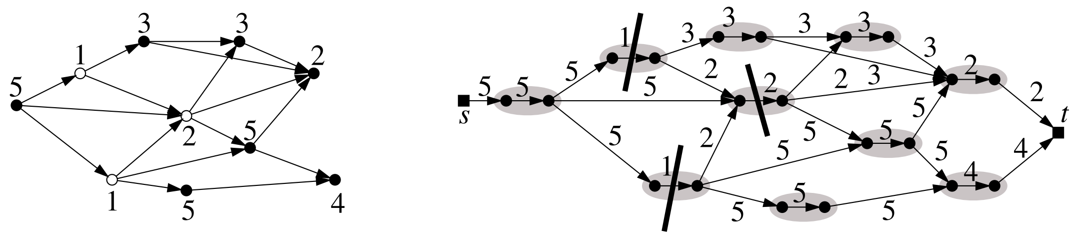

Figure 1: A weighted DAG $G = (V,E)$ and the st-graph $G' = (V',E')$ constructed from G as in the proof of Lemma 4. Vertices in G' corresponding to the same vertex in G are circled. A minimum cut of G' is shown as fat lines such that any st-path has to cross one of these lines. The white vertices in G represent the corresponding minimum weight separator of G.

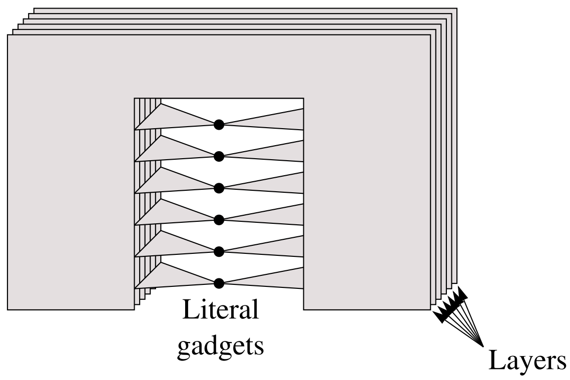

Figure 2: The global structure of $G(F,\varepsilon)$.

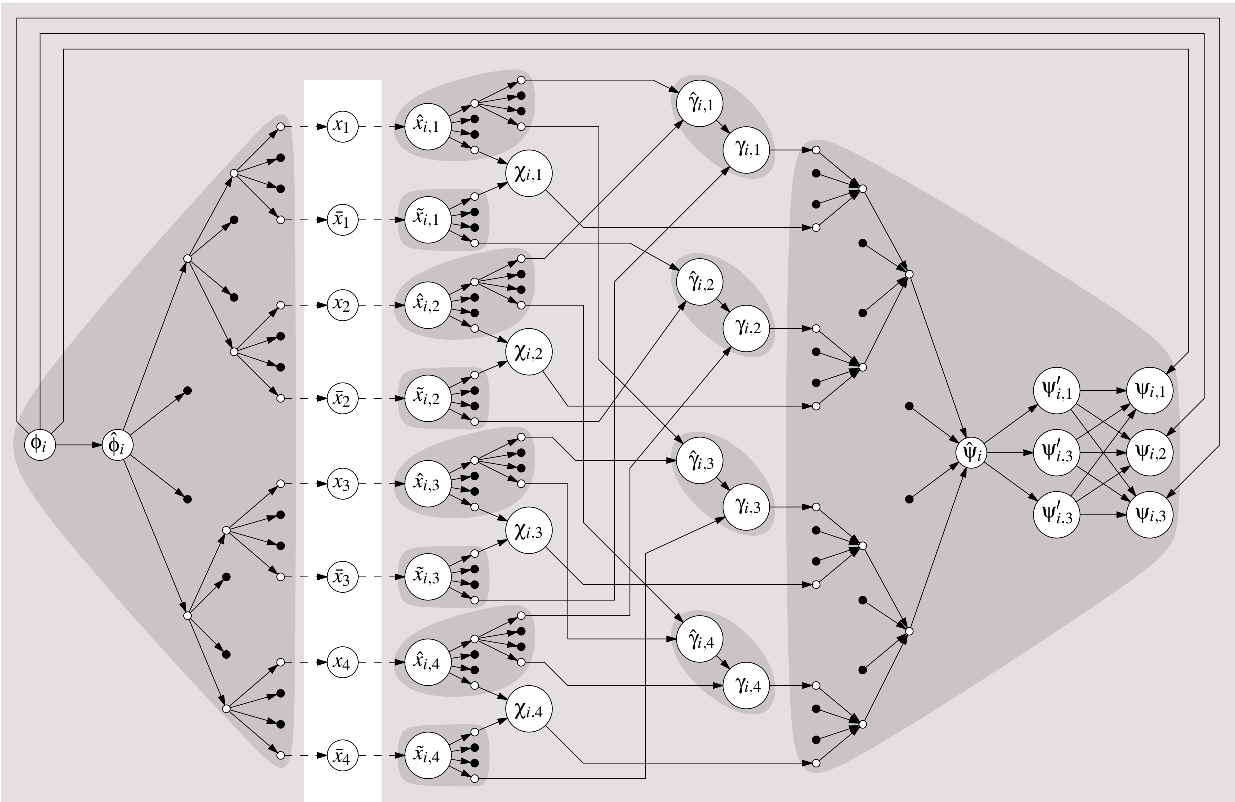

Figure 3: This figure shows a layer L_i of the graph $G(F,\varepsilon)$ constructed for formula $F = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4) \wedge (x_1 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \vee x_3 \vee x_4)$ and $\varepsilon \leq \frac{1}{2}$, and its connections to the literal gadgets. Edges to and from literal nodes have been dashed to signify that they correspond to paths in the corresponding trees Δ and Γ. Feedback, multiplier, clause, and final gadgets are shaded. Feedback blockers and the extra leaves in Ψ_i are not shown in their correct numbers, as these depend on ε. The same is true for vertices $\psi'_{i,h}$ and $\psi_{i,h}$.

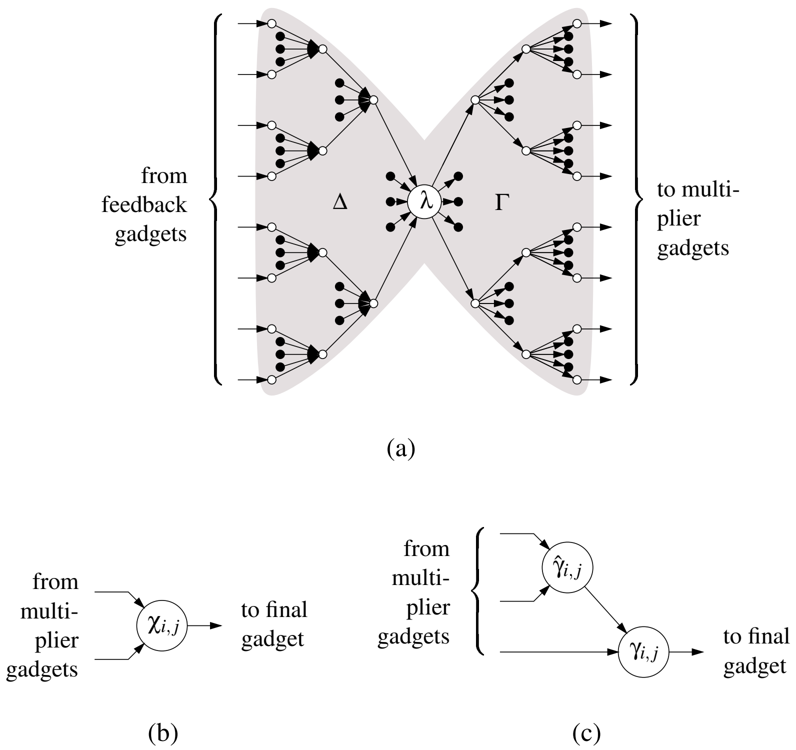

Figure 4: (a) A literal gadget. Regular nodes are shown as white discs. They form a binary tree. Black discs are additional nodes that serve as feedback blockers or enforce a logical AND behavior on Δ. (b) A negator gadget. (c) A clause gadget.

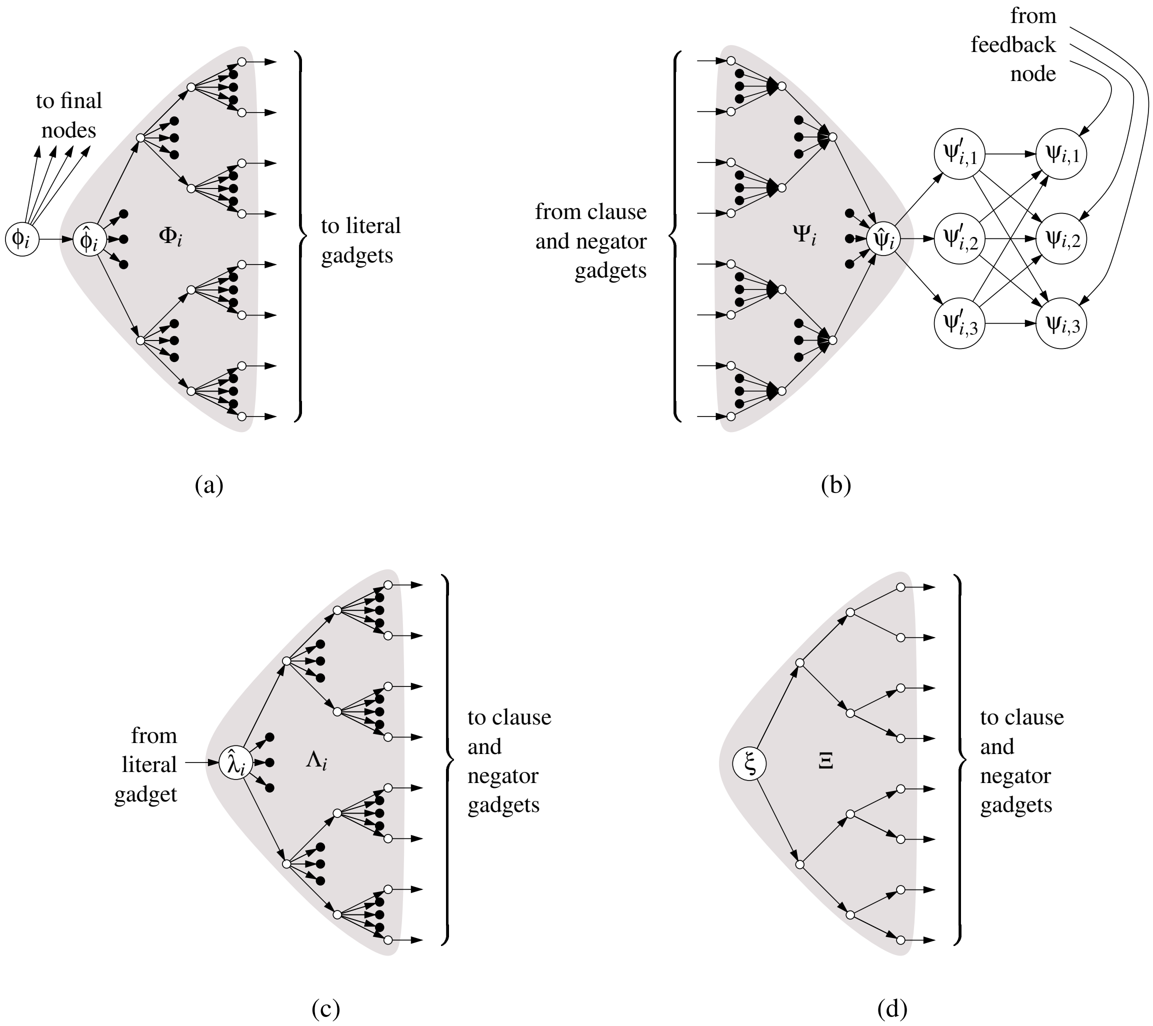

Figure 5: (a) A feedback gadget. Regular nodes of Φ_i are shown as white discs. These nodes form a binary tree. The black discs are feedback blockers. (b) A final gadget. Regular nodes of Ψ_i are shown as white discs. These nodes form a binary tree. The black discs are additional children of the nodes in Ψ_i required to enforce a logical AND behavior of Ψ_i. (c) A multiplier gadget. Regular nodes of Λ_i are shown as white discs. These nodes form a binary tree. The black discs are feedback blockers. (d) The booster gadget.