Diagnosing Analogue Linear Systems Using Dynamic Topological Reconfiguration

Alexander Feldman
General Diagnostics
Burgwal 47
2611GG, Delft, The Netherlands
email: alex@general-diagnostics.com

Gregory Provan
University College Cork
College Road, Cork, Ireland
email: g.provan@cs.ucc.ie

Abstract

Fault diagnosis of analogue linear systems poses many challenges, such as the size of the search space that must be explored and the possibility of simulation instabilities introduced by particular fault classes. We study a novel algorithm that addresses both problems. This algorithm dynamically modifies the simulation model during diagnosis by pruning parametrized components that cause discontinuity in the model. We provide a theoretical framework for predicting the speedups, which depends on the topology of the model. We empirically validate the theoretical predictions through extensive experimentation on a benchmark of circuits.

Introduction

Fault diagnosis of analogue systems is an important problem, given their prevalence in synthetic and natural domains. For example, one heavily-studied domain is that of analogue circuits, and enormous research has been directed towards the testing and diagnosis of electronic devices (Bandler and Salama 1985; Kabisatpathy, Barua, and Sinha 2005; Milor 1998).

We focus on *hard (catastrophic) faults*, which characterize when the system exhibits significant (and often abrupt) behavioral changes. For example, a hard fault in a circuit occurs when the terminals of a component become stuck-open or stuck-short. Hard faults contrast with *parametric faults*, which characterize deviations of component parameters that result in performance beyond acceptable limits.

Hard faults can be difficult to diagnose for two reasons. First, the space of failure behaviours may be very large and difficult to search. Second, hard faults can cause inference techniques to display numerical instability: e.g., open circuits may cause singular matrices for the matrix inversion operator, and short circuits may cause infinite conductance values. As a consequence, numerical simulators, e.g., SPICE (Nagel and Pederson 1973) or MODELICA, may become unstable under particular fault conditions. To avoid these problems, such modern solvers replace short-/open-circuits with small/large resistances, respectively. In this paper, resistances close to zero and large resistances are denoted as ε and $\mathcal{E}$, respectively. While replacing zeroes and infinity with small and large numbers is less of a problem for a single simulation, it introduces a significant computational overhead for the thousands of simulations necessary for examining many combinations of parameter values. We show how we can remove such components from the model, both speeding up simulation and removing the source of instability.

This paper proposes a novel method for diagnosis of hard multiple-fault instances in analogue systems. We adopt a model for analogue systems based on a graphical topological structure, in which nodes represent measurement points and edges the components. We propose an approach for diagnosing hard faults that performs topological reductions of the model during runtime inference, which results in significant speedups compared to state-of-the-art approaches, none of which employ such reductions (Korzybski 2008; Liu and Starzyk 2002; Tadeusiewicz and Hałgas 2006).

This graphical model enables us to capture a wide range of domains, e.g., mechanical, electrical, and thermal systems, through using a bond graph interpretation for the network model (Karnopp, Margolis, and Rosenberg 2006).

The contributions of this paper are as follows: (1) we design an optimization method that dynamically optimizes the diagnosis model during diagnostics inference based on removing redundant system component models; (2) we theoretically characterize the speedups of our algorithm based on system topology for a class of networks; (3) we empirically validate our predictions using a novel benchmark of analogue electrical circuits that represents a large class of circuit topologies that occur in real-world systems; (4) we show that the proposed optimization method improves the diagnostic performance as predicted by the analytical model by up to 2 orders of magnitude.

Concepts and Definitions

Figure 1 shows an electrical circuit that we use for illustrating the algorithms in this paper.

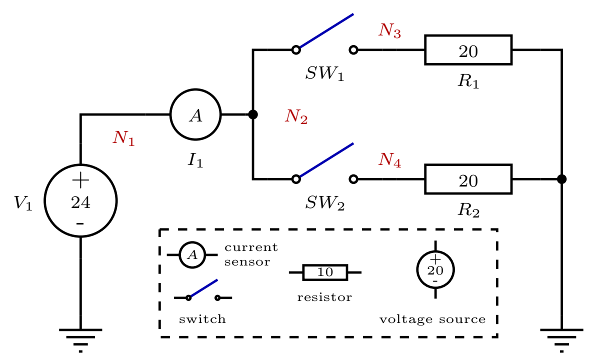


Figure 1: Simple circuit used as a running example

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The circuit in figure 1 consists of a voltage source V_1, two switches SW_1 and SW_2, two resistors R_1 and R_2, and a current sensor I_1. The two switches SW_1 and SW_2 are modeled as resistors whose values (0 for short-circuited and ∞ for open-circuited) is specified by the user as user-defined input (command).

An analogue linear system consists of an inter-connected set of components. For example, in a circuit a component can be a resistor or capacitor, and connections are wires. We represent a system model M in terms of the system variables ξ, the component equations Σ, and the connection topology G. We describe each in turn below.

We define a variable for each system component $c_i \in \mathcal{C}$ and component-connection $z_i \in \mathcal{Z}$, as follows.

Definition 1 (Variables)**.** We represent variable i, ξ_i, as a 3-tuple $\langle \tau_i, \mathcal{D}_i, \omega_i \rangle$, where

- $\tau_i \in \tau$ represents the variable's type, which can be *effort*, *flow*, or *connection*;
- $\mathcal{D}_i \in \mathcal{D}$ represents the (continuous-valued) variable's domain, i.e., the possible values the variable can take on;
- $\omega_i \in \Omega$ represents the variable's health status, or failure mode.

A health assignment (or mode) $\omega \in \Omega$ is an assignment of nominal or faulty operating conditions to all variables in ξ.

We represent the equations as follows.

Definition 2 (Equations Σ)**.** We represent a linear analogue system in terms of a set Σ of relations between effort $\vec{x}$ and flow $\vec{z}$ vectors of continuous-valued variables, using $T\vec{x} = \vec{z}$, where T is an $n \times n$ non-singular matrix and $\vec{x}$, $\vec{z}$ are collections of variables in ξ.

For each health assignment (or mode) we assume that we can specify a distinct set of equations. Hence, for each $\omega \in \Omega$ we specify an equation set Σ_ω given by $T_\omega \vec{x_\omega} = \vec{z}_\omega$, such that $\Sigma = \bigcup_{\omega \in \Omega} \Sigma_\omega$.

For example, for circuits $\vec{x} \in \mathbb{R}^n$ is an (unknown) nodal voltage vector, and $\vec{z} \in \mathbb{R}^m$ is a measurable current-source vector.

We represent the topology of a system using a graph G.

Definition 3 (Topology Graph)**.** Given a model M with components $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$, and component connections (junctions) $\mathcal{Z} = \{z_1, z_2, \cdots, z_l\}$, where component c_i occurs between junctions z_j, z_k, we represent the topology graph $G(V, E)$ of M such that the vertices V correspond to connections in M and edges E correspond to components in M.

Edges in G are also labeled with the type of the corresponding components and their parameters.

We assume that the system can operate in a set Ω of health states (modes), which can consist of nominal or faulty operating conditions. Given a system that consists of a discrete set of components with a corresponding set of health parameters COMPS, a health assignment $\omega \in \Omega$ is an assignment to all variables in COMPS.

Further, for each health mode we assume that we can specify a distinct set of equations. Hence, for each $\omega \in \Omega$ we specify an equation set Σ_ω given by $T_\omega \vec{x_\omega} = \vec{z}_\omega$, such that $\Sigma = \bigcup_{\omega \in \Omega} \Sigma_\omega$.

Given a flow vector $\vec{z}$, *simulation* of $\vec{x_\omega}$ involves computing the effort vector via $\vec{x_\omega} = T_\omega^{-1} \vec{z}_\omega$. Since Σ is linear, we can simulate efficiently.

Given an observation $\vec{\alpha}$, we estimate the health assignment (i.e., solve a diagnostic problem) by computing an optimal solution of a health estimation problem where the health variables are discrete and the problem is split in two parts: simulation and residual analysis.

In real-world applications, a straightforward simulation function is not sufficient to adequately solve the diagnostic problem. This is because models are imprecise, there is sensor noise, health parameters are discrete, etc. Instead, we compute a difference between $\vec{\alpha}$ and a simulation $\hat{\vec{z}}$ (using the residual function of Definition 4), and then identify the mode that minimises this function.

Definition 4 (Residual Function)**.** Given two m-dimensional real vectors $\hat{\vec{z}}, \vec{\alpha} \in \mathbb{R}^m$, a residual function $R : \{\hat{\vec{z}}, \vec{\alpha}\} \mapsto R(\hat{\vec{z}}, \vec{\alpha})$ maps $\hat{\vec{z}}$ and $\vec{\alpha}$ into the real interval $[0; \infty)$.

For the residual function R we typically use a statistical estimator. In this paper we use the absolute residual function:

$$R_{\text{abs}}(\hat{\vec{z}}, \vec{\alpha}) = \sum_{i=1}^{m} w_i \, |\hat{z}_i - \alpha_i| \quad (1)$$

where $\hat{z}_i$ and α_i are the i^{th} components of the $\hat{\vec{z}}$ and $\vec{\alpha}$ vectors, respectively, and w_i are weights (parameters).

Definition 5 (Health Estimation Problem)**.** Given a diagnostic model M, an observation $\vec{\alpha}$ and a residual function R, the health estimation problem is to compute an assignment $\omega_{\min}$ to all variables in COMPS such that:

$$\omega_{\min} = \operatorname*{argmin}_{\omega} R(\vec{z}_\omega, \vec{\alpha})$$

Efficiently solving the above health estimation problem is the main goal of our framework.

Algorithms

In what follows we outline an algorithm for solving the problem given in definition 5. Our algorithm consists of (1) a search algorithm that iterates over a subspace of the possible combinations of discrete parameter values, (2) a model optimization algorithm that improves the model complexity, and (3) simulation and residual algorithms that rank the candidates and compute the optimal parameter assignment. The primary goal of this section is to show how model optimization (step 2) significantly improves the performance of the simulation (step 3).

To illustrate the possibility to reduce complexity, consider the running example where the simulator assumes a short-circuit in R_2 (see fig. 2). While a purely numerical simulation approach is agnostic toward the existence of the short-circuit (it is typically represented as a small resistor), a symbolic preprocessing step can easily discover that a short-circuit in R_2 leads to the voltage source V_1 being shorted, thus making the whole simulation trivial and, hence, faster.

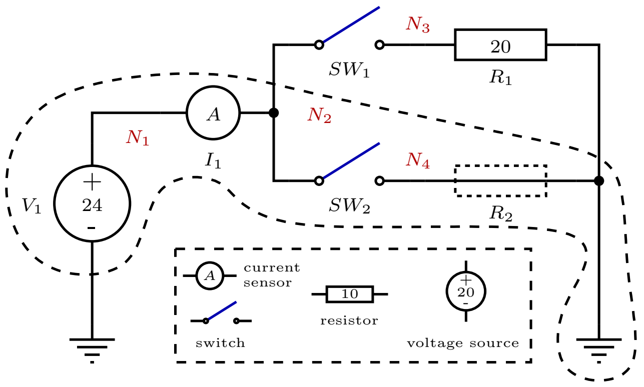


Figure 2: Shorted R_2 makes simulation of the circuit in figure 1 trivial as it shorts the voltage source V_1

Algorithm 1 provides a generic search method for solving the parameter estimation problem in definition 5. The idea is to search the space of all possible health parameter combinations. This space is exponential in the number of components (health variables) and normally the search is limited to a subset of all combinations. Examples of such subsets are all k-fault combinations or the assignment sets generated by a greedy search (Feldman, Provan, and van Gemund 2010b).

```
Algorithm 1: DiagnosisSearch($M, \alpha$)
Input: $M$, model
Input: $\alpha$, observation
Result: $\omega$, diagnosis
Local variable: $\mathbf{h}$, health assignment, initially $\{\}$
Local variable: $\vec{p}$, simulation vector
Local variables: $r, r_{\min}$, residuals, $r, r_{\min} \in [0; \infty)$

1 $r \leftarrow \infty$
2 repeat
3     $\vec{p} \leftarrow$ SimulateSystem(MakeGraph($M, \mathbf{h}$))
4     $r \leftarrow$ ComputeResidual($\vec{p}, \alpha$)
5     if $r < r_{\min}$ then
6         $\omega \leftarrow \mathbf{h}$
7         $r \leftarrow r_{\min}$
8 until $\mathbf{h} \leftarrow$ NextHealthAssignment($\mathbf{h}, r$) $\neq \varnothing$;
9 return $\omega$
```

Algorithm 1 starts by assuming the user-defined default parameter assignment $\mathbf{h} = \{\}$ which, in diagnostic context, represents the *all-okay* status of the system. Successive health assignments are generated by the NextHealthAssignment subroutine.

Depending on the implementation of NextHealthAssignment, algorithm 1 can perform different types of search such as breadth-first, depth-first, iterative deepening, greedy (in this case NextHealthAssignment uses the r parameter), random, and many others. Which policy is optimal depends, i.a., on the simulation and residual functions (see def. 4) and the topology of the model. Choosing a search policy (or a combination of such) is a topic of its own and is not discussed in this paper. Our main focus is how the complexity of the simulation function SimulateSystem in line 3 affects the overall complexity of the optimization algorithm.

The function MakeGraph takes a set of component models, a topology, and a parameter guess $\mathbf{h}$ and generates a graph G. G is fed to the simulation function SimulateSystem. The graph represents a system of linear equations and the simulation subroutine, described in the section that follows, uses linear algebra tools to compute the unknown simulation variables.

Algorithm 2 outlines a system simulator that supports only linear elements: dissipative elements (resistors), effort (voltage) and flow (current) sources. Our approach is similar to the one used in SPICE (Nagel and Pederson 1973). The implementation of Algorithm 2 uses the Modified Nodal Analysis (MNA) of Ho, Ruehli, and Brennan (1975), generating nodal matrices of size $n \times n$ where n is the number of nets in the input netlist L.

```
Algorithm 2: SimulateSystem($L$)
Input: $G$, graph
Result: $\vec{z}$, voltage vector
Local variable: $G$, graph
Local variable: $N$, nodal matrix
Local variable: $\vec{j}$, current vector

1 $G \leftarrow$ OptimizeGraph($G$)
2 $N \leftarrow$ MakeNodalMatrix($G$)
3 $\vec{j} \leftarrow$ MakeCurrentVector($G$)
4 $\vec{z} \leftarrow N^{-1}\vec{j}$
5 return $\vec{z}$
```

The remaining steps of the algorithm are concerned with performing the simulation. First, we generate the matrix N from the graph G (e.g., as described by Kielkowski (1994, p. 18)). Note that this is the phase in which instability can occur due to particular fault classes, e.g., open-/short-circuits.

The time complexity of algorithm 2 is dominated by the matrix inversion in line 3. The matrix inversion operation is as complex as matrix multiplication (Cormen et al. 2001, p. 757) and its complexity is $O(n^3)$ or $O(n^{2.807})$ when using the Strassen algorithm[1] (Press et al. 2002, pp. 105–107) where the matrix N is of size $n \times n$.

The most important addition to algorithm 2 is the OptimizeGraph call in line 1. The purpose of OptimizeGraph is to reduce the number of nodes in G, thus reducing the size of the nodal matrix N.

The task of algorithm 3 is to reduce the complexity of simulation by removing from the model components (with specific parameters) corresponding to hypothesized faults. Algorithm 3 has two main parts: first all components that are faulty (for example open- and short-circuited resistors) are removed (lines 3–10) and, second, only those systems that are closed through a effort source are retained (lines 12–24).

The auxiliary function IsDissipativeElement(e) returns true if and only if its argument e is a dissipative element edge (resistor in the electrical domain). The function

[1]There exist faster methods for sparse matrices but they are of no practical significance for our algorithms.

Algorithm 3: OPTIMIZEGRAPH(G)

```
   Input: $G = \langle V, E \rangle$, graph
   Result: $G$, optimized graph
   Local variables: $v$, vertex, $e$, edge
   Local variables: $S$, edge stack, $P$, set of edges
   Local variable: $stop$, Boolean flag
1  repeat
2      $stop \leftarrow$ true
3      foreach $e \in E$ : ISDISSIPATIVEELEMENT($e$) do
4          if $\Omega(e) < \varepsilon \vee$ SRC($e$) = DST($e$) then
5              $V \leftarrow V \setminus$ DST($e$)
6              $E \leftarrow E \setminus e$
7              RECONNECT($E$, SRC($e$), DST($e$))
8              $stop \leftarrow$ false
9          if $\Omega(e) = \infty$ then
10             $E \leftarrow E \setminus e$
11 until $stop$ = false;
12 PUSH($s$, EFFORTSOURCE($E$))
13 while $e \leftarrow$ POP($s$) $\neq \varnothing$ do
14     PUSH($s$, ADJACENTEDGES($G, e$))
15     $P \leftarrow P \cup e$
16 repeat
17     $stop \leftarrow$ true
18     foreach $e \in E$ do
19         if $|e| < 2$ or $e \notin P$ then
20             $V \leftarrow V \setminus$ DST($e$)
21             $E \leftarrow E \setminus e$
22             $stop \leftarrow$ false
23 until $stop$ = false;
24 REMOVEORPHANEDVERTEXES($V, E$)
25 return $G = \langle V, E \rangle$
```

$\Omega(e)$ returns the resistance of e. The functions SRC(e) and DST(e) return the two respective vertices that are connected to an edge e. Notice that it is customary that the representation of the simulation graph V is directed, although the orientation of the edges produces no difference in the simulation results.

While edges that represent open-circuited resistors can be simply removed from the graph (lines 9–10), after removing short-circuited resistors (lines 4–8), the adjacent wires have to be reconnected. This is done by the RECONNECT subroutine and the process is illustrated in figure 3. As removing a short-circuit can cause another short-circuit (see again fig. 3), the component removal loop in lines 3–10 has to be repeated until no more removals are performed. This is achieved by using the $stop$ flag.

The second part of algorithm 3 (lines 12–22), removes all components that are not doubly-connected to an effort source. This process first performs a Depth-First Search (DFS) on the graph G (Sedgewick 2002, pp. 81–99). In our case, the DFS starts from all voltage sources. The subroutine EFFORTSOURCE returns all edges that are connected to a positive terminal of an effort (voltage) source. These edges are added to the stack s in line 12. The result of the DFS is that all edges that can be reached from effort sources are added to the set of edges P. The DFS uses the function ADJACENTEDGES to generate all edges that are neighbors of an

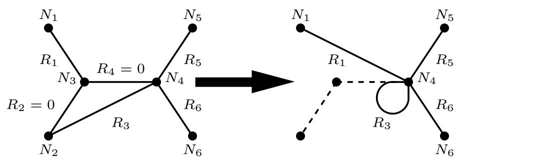


Figure 3: Removing the short-circuited resistors R_2 and R_4 in the left graph makes nodes N_2 and N_3 unnecessary. To preserve the circuit we have to disconnect R_1 from N_2 and to connect it to N_4. Similarly, both ends of R_3 have to be connected to N_4. At this moment, N_4 shorts R_3 and R_3 can be removed.

edge e.

The loop in lines 16–23 removes all edges that are not in P, i.e., they are not connected to an effort source. This loop also removes all hanging edges. The condition for a hanging edge is in line 19. The expression $|e|$ denotes the sum of the degrees of the two vertexes incident to e. Similarly to the first part of the algorithm, the loop in lines 18–22 is repeated until no further truncation of the graph is possible.

Finally, all orphaned nodes (zero-degree nodes) are removed by the helper subroutine REMOVEORPHANEDVERTEXES.

Proposition 1 (Correctness)**.** *Given a graph G @@@ TBD @@@*

Proof (Sketch). □

The worst-case time complexity of algorithm 3 is $O(|E|^2)$, i.e., it is polynomial in the number of edges. Some time complexity in algorithm 3 can be traded for memory, thus achieving near linear performance. The outer loops in lines 1–11 and 16–23, for example, can be removed at the cost of managing data structures that keep track of all graph nodes that have to be merged and all orphaned or hanging paths.

Predicted Topological Reduction

This section examines the type of topological reduction that is possible for a typical model. We focus on two key topologies, serial and parallel sub-graphs, as shown in figure 4(a).[2] We assume we have a topology graph in which parallel edges are allowed, e.g., for system redundancy. The size of a graph $G(V, E)$ is denoted in terms of the number of edges, i.e., $|G| = |E|$. The size of $G(V, E)$ is essential to the algorithmic performance of algorithm 3 as it is equal to the size of the nodal matrix.

Definition 6 (Contraction Operators)**.** Given a graph $G(V, E)$ and an edge $e = \{v, w\}$, the graph $G'(V', E') = G(V, E) \searrow e$ is such that $E' = E \backslash \{e \cup F\}$, $V' = V \backslash v$, F is the set of all edges that are parallel to $\{v, w\}$, and each edge $\{x, v\} \in E$ such that $x \neq w$ is replaced with an edge $\{x, w\} \in E'$.

Given a graph $G(V, E)$ and an edge e, the graph $G'(V', E') = G(V, E) \nwarrow e$ is such that $E' = E \backslash e$, $V' =$

[2]We plan to address additional sub-graphs in future work.

$V \setminus \{W \cup v\}$ where v is incident to e and W is the set of singly connected vertices in $E \setminus v$.

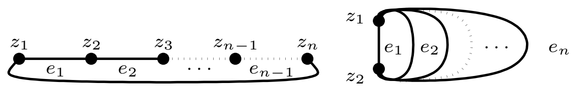
We have chosen the symbols $\searrow$ and $\nwarrow$ to visually resemble the set exclusion operator $\setminus$ as the two contraction operators decrease the size of G. The graph contraction ratio, for graphs with at least on edge, is:

$$\rho = \frac{\sum_{e \in E} \left[|G \searrow e| + |G \nwarrow e| \right]}{2\,|G|^2} \tag{2}$$

The variable ρ in eq. 2 adds-up the effect of applying $\searrow$ and $\nwarrow$ to each edge in E. Note that $\rho \in [0;1]$. The denominator in eq. 2 is simply $|G(V,E)|$-times the number of edges in $G(V,E)$. The coefficient is 2 in the denominator because we have two contraction operators: $\searrow$ and $\nwarrow$. In graphs with a single loop, $\rho = 0$, i.e., the sum of the sizes of all contracted graphs is zero.

Lemma 1. *The following two statements are true:*

1. *The total graph contraction for a serial graph $G(V,E)$ with edges $E = \{\{z_1, z_2\}, \{z_2, z_3\}, \ldots, \{z_{n-1}, z_n\}\}$ is $\rho = \frac{1}{2}$.*
2. *The total graph contraction for a parallel graph $G(V,E)$ with edges $E = \{\{z_1, z_2\}, \{z_1, z_2\}, \ldots, \{z_1, z_2\}\}$ is $\rho = \frac{1}{2}$.*



(a) serial (b) parallel

Figure 4: Serial and parallel graph topologies

Proof (Sketch). In a graph $G(V,E)$ with serial topology, there are exactly $G(V,E) = |E| = n$ edges as illustrated in figure 4(a). Choose any edge e. Applying $\searrow$ to e removes only e from the original graph $G(V,E)$ and results in:

$$\sum_{e \in E} |G \searrow e| = n\,(n-1) \tag{3}$$

On the other hand, applying $\nwarrow$ to e, regardless of the choice of e results in a singly-connected graph:

$$\sum_{e \in E} |G \nwarrow e| = n \tag{4}$$

Substituting eq. 3 and eq. 4 into eq. 2 gives us $\rho = \frac{n + n(n-1)}{2n^2}$ which simplifies to $\rho = \frac{1}{2}$. The correctness of statement 2 can be proven in a similar manner. □

We now prove a theorem that predicts how algorithm 3 affects a class of random graphs; in the following section we empirically validate our theoretical predictions. Our random graphs depict serially connected circuits with random connections. We generate our random graphs by adding random edges to a serially connected graph such as the one shown in figure 4(a). This structure of augmenting regular graphs with random edges is the standard method for random-graph generators, and is known to closely match the structure of real-world systems (da F. Costa et al. 2007).

In the following, we denote the average graph degree of a graph $G(V,E)$ as $\bar{d} = 2|E|/|V|$.

Theorem 1. *The total graph contraction for a serially-connected generated graph $G(V,E)$ is $\rho = k\bar{d}$.*

Proof (Sketch). We decompose $G(V,E)$ into two graphs: $G(V,E')$ and $G(V,E'')$ where E' is the set of random edges and $G(V,E'')$ is a serially connected graph. This decomposition is possible due to the way $G(V,E)$ is constructed. Let $G(V,E'')$ consists of C components (the computation of C is shown in Erdős and Rényi (1960)). We can now show that the overall reduction $\rho = k_1\rho' + k_2\rho''$ will be the average of reducing $G(V,E')$ and $G(V,E'')$. From lemma 1, $\rho'' = \frac{1}{2}$. Finally, both k_1 and k_2 are proportional to $\bar{d}$, hence $\rho = k_1'\bar{d}\rho' + k_2''\bar{d}\rho''$. As there is no dependency between k_1', k_1'', ρ', and ρ'' (ρ'' depends on C only), then we can conclude that ρ is proportional to the average graph degree. □

Although there is an analytical method to obtain ρ for generated graphs, we compute it experimentally which also shows the absolute improvement in computational complexity due to the symbolic preprocessing. We do this in the following section.

Experiments

Algorithms 1–3 are implemented as a part of the (deleted for anonymity) diagnostic framework.

Benchmark

No benchmark for analogue linear circuits exists, to the best of our knowledge, unlike digital circuit benchmarks, e.g., (Brglez and Fujiwara 1985). Consequently, we have generated the circuits that we need.

Table 1 shows a number of regular circuits. These topologies can be scaled by setting a variable N, producing a range of circuit sizes.

Name	N	variables	|COMPS|
N-SERIAL	3–202	11–608	3–202
N-PARALLEL	3–202	9–407	3–202
N-MIXED	3–102	18–513	6–204
N-MESH	3–12	48–723	18–288
N-TREE	3–5	57–6253	27–3125

Table 1: Circuit benchmark

All benchmark circuits have a single voltage source that cannot fail. The N-SERIAL circuits, shown in figure 5(a), consist of N resistors connected in series. Similarly, the N-PARALLEL circuits in figure 5(b) consist of N resistors connected in parallel. The N-MIXED topology is a combination of N-SERIAL and N-PARALLEL as shown in figure 5(c). The N-MESH circuits consist of $2 \times N^2$ resistors arranged in a rectangular grid as shown in figure 5(d). Finally, the N-TREE circuits are complete N-ary trees of depth N. They are shown in figure 5(e).

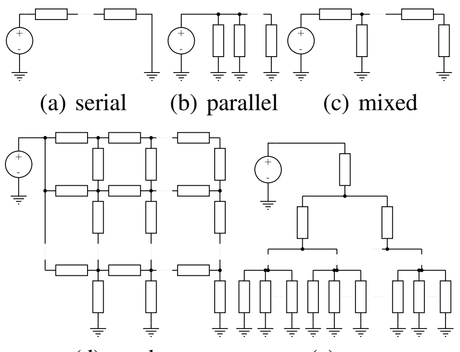


Figure 5: Scalable circuits with regular topology used for performance analysis

For N-Serial, $\bar{d} = 2$. For N-Parallel, $\bar{d} = N + 1$. For N-Mixed, $\bar{d} = (4N + 2)/(N + 2)$ which approaches 4 for larger N. The N-Mesh circuits have average graph degree $\bar{d} = (4N^2+2)/(N^2+2)$ which approaches 4 when increasing N even faster than N-Mixed.

In addition to the regular circuits from the preceding section we have created 6 370 "generated" circuits that have realistic topological properties. We generate these circuits by adapting the random graph algorithm proposed by Sedgewick (2002, p. 42). We modify the original algorithm to ensure multiple graph components for sparse graphs.

We generate graphs by specifying the vertex/edge ratio r. The relation between the graph generation parameter r and the average graph density $\bar{d}$ is given by $|E| = rN + C - 1$, where C is the number of connected components in G. From the fact that $|V| = N$ it follows that $\bar{d} = 2Nr + C - 1/N$. For our experiments we have chosen $0.1 \leq r \leq 0.4$ which translates to $2.06 \leq \bar{d} \leq 2.9$.

Results

Figure 6 shows the ratio of the CPU time of the optimized versus the non-optimized algorithm. The performance gain due to the graph preprocessing is close to 2.02, does not depend on the circuit size, and is almost constant (there is some measurement noise).

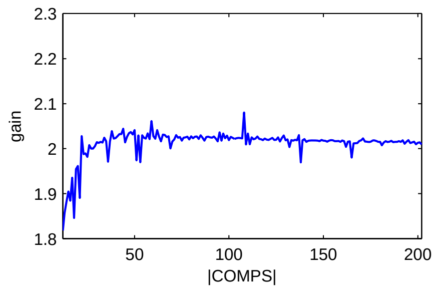


Figure 6: N-Serial performance gain

The performance gain for N-Parallel is the same as for N-Serial. Not surprisingly, algorithm 3 helps only a little in the connected topologies of figure 5(c) and figure 5(d). For these two dense topologies, we have measured a performance gain of at most 9%. The reason for this is that a single faulty component decreases the size of the nodal matrix by one.

The most significant benchmark performance gain due to algorithm 3 is for k-fault simulations where $k > 1$. The performance of the diagnostic search decreases exponentially with k, except when k approaches N because then the nodal matrix is almost degenerate, hence easier to decompose. On the other hand, multiple faults increase the frequency with which significant parts of the nodal graphs are pruned. For example, when $k = 2$, an open-circuit close to the voltage source in N-Serial makes N simulations use a 2×2 matrix instead of the normal $N \times N$.

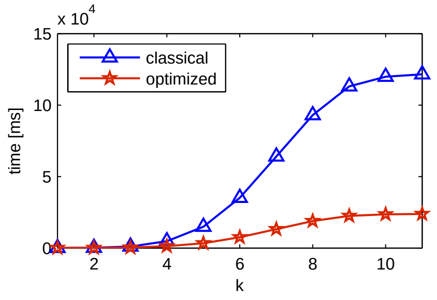


Figure 7: 11-Serial k-fault performance

The performance gain due to k-fault simulations is shown in figure 7. In order to simulate k-fault combinations, we have chosen a small $N = 11$. The performance gain increases for larger k and reaches a factor of 5.12 for $k = 12$.

Figure 8 shows the average algorithm complexity with random circuits. On the x-axis we have the number of nodes in the topology graph. The y-axis is the number of edges coefficient r. The z-axis shows the ratio of the time of the diagnosis with and without algorithm 3.

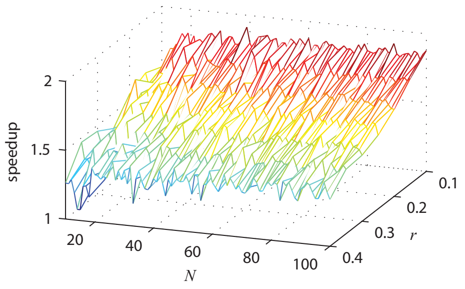


Figure 8: Performance gain for random circuits of various size and with varying density

Figure 8 shows that the performance gain due to algorithm 3 does not depend on the circuit size, i.e., it is almost constant. What determines the performance gain is the edge density. For $r = 0.1$ we have almost twice speed improvement and it decreases to ≈ 1.25 for $r = 0.4$. The performance gain is bounded from below by the performance gain of the N-Serial topology due to the nature of random graph generation algorithm.

The linear correlation coefficient of the speedup shown in figure 8 and the average graph degree is -0.89 which validates the analysis in section . The negative sign in the correlation is due to the fact that r defined as the number of graph vertices per edge while $\bar{d}$ is reciprocal to r, i.e., $\bar{d}$ is defined as the number of edges per graph vertex.

We have a modeled the Electrical Power System (EPS) of a real-world satellite (Feldman et al. 2013). It consists of, amongst others, 96 heating elements (modeled as resistors), 112 switches (modeled as resistors whose resistor changes as a result of a user-supplied command), and approx. 1000 equations. Most components have at least two fault modes, in addition to the nominal one. The many switching components and the designed cold-redundancy lead to a large number of normally open-circuit or short-circuit elements at any given time. As a result, algorithm 3 leads to a performance speedup factor of at-least 38.9, and that is for the single fault assumption. For the double-fault assumption we have a performance gain of 194 times.

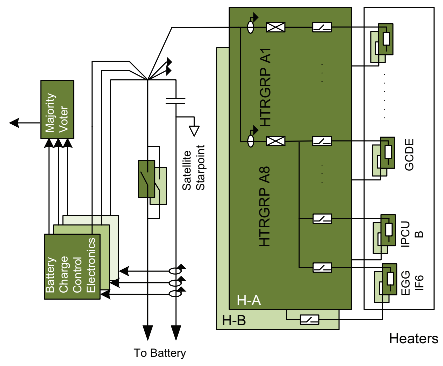


Figure 9: The design of the satellite EPS is highly redundant to increase reliability

The reason for the very good performance of the symbolic reduction is in the highly regular and redundant design of the satellite EPS as is illustrated in figure 9. Consider, for example, the topology of the power distribution and heater network. The electrical heaters provide double redundancy and are grouped in lines of six. A fault assumption eliminates a whole line of resistors which, given, the cubic complexity of the simulation algorithm translates to the substantial savings that we have measured in our experiments.

The performance gain of symbolically pruning faulty components is even more expressed when assuming double-fault hypotheses (the latter increases diagnostic accuracy which is outside the scope of this work) or when using diagnostic inference as a subroutine in a more complex algorithm such as active testing (Feldman, Provan, and van Gemund 2010a). The double-fault assumptions decrease the size of the nodal matrix with more than 20% and there are many cases in which the nodal matrix becomes of trivial size (e.g., two faulty main distribution switches).

Related Work

Analog systems diagnosis is a well-studied area, and a variety of approaches have been developed, based on methods such as off-line numerical test-generation (Duhamel and Rault 1979), bond graph approaches (Samantaray et al. 2006), machine learning (Aminian and Aminian 2000), "intelligent" methods (Dague et al. 1992; Fenton, McGinnity, and Maguire 2001), and MODELICA fault models (Minhas et al. 2014). The domain of analogue systems that has received the most attention is that of fault diagnosis of analog circuits, e.g., (Bandler and Salama 1985; Kabisatpathy, Barua, and Sinha 2005; Milor 1998). Although much progress has been made, most prior work addresses only single-fault cases. Achievements towards automatic multiple-fault diagnosis are documented in, i.a., (Korzybski 2008; Liu and Starzyk 2002); however, many aspects of the problem are still open, and no fully-automatic method exists for multiple-fault generic analog circuit diagnosis.

In general, parametric-fault simulation-after-test (SAT) methods must be used to address the non-linear diagnostic equations. When the parameters differ only slightly beyond their tolerance ranges, the equation can be linearized (Tadeusiewicz and Hałgas 2006). For more significant differences, more accurate results for this task are needed. These include methods for solving nonlinear equations, e.g., the Newton-Raphson method or its variants (Cherubal and Chatterjee 1999; Navid and Willson, Jr. 1979), modified nodal approaches (Ho, Ruehli, and Brennan 1975), evolutionary algorithms (Augusto and Almeida 2000), or least square methods (Hongkui and Pengnian 1988).

Researchers have studied dynamic discontinuities in bond graphs (Mosterman and Biswas 1996). This paper complements the approach of Mosterman and Biswas by providing an algorithmic framework for dealing with structural (or parametric) discontinuities. Our approach is fully applicable to bond graphs (Mosterman and Biswas 1999).

Conclusions

In this paper we study the advantages of dynamically modifying the simulation model for steady-state analysis in diagnosis. We show that speedups due to pruning of parametrized components that cause discontinuity in the model depend on the topology of the model, and empirically justify this theoretical prediction through extensive experimentation on a benchmark of circuits. Further, our method does not decrease the diagnostic accuracy.

We observe most computational savings in real-world circuits where, due to standard redundancy for fault-tolerance, there are many unpowered and shorted sub-circuits that can be pruned. We also observe that the computational performance increases with the number k of k-fault combinations.

References

Aminian, M., and Aminian, F. 2000. Neural-network based analog-circuit fault diagnosis using wavelet transform as preprocessor. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47(2):151–156.

Augusto, J. A. S., and Almeida, C. B. 2000. Analog fault diagnosis in nonlinear DC circuits with an evolutionary algorithm. In *Proc. IEEE CEC'00*, volume 1, 609–616.

Bandler, J. W., and Salama, A. E. 1985. Fault diagnosis of analog circuits. *Proc. of the IEEE* 73(8):1279–1325.

Brglez, F., and Fujiwara, H. 1985. A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In *Proc. ISCAS'85*, 695–698.

Cherubal, S., and Chatterjee, A. 1999. Parametric fault diagnosis for analog systems using functional mapping. In *Proc. DATE'99*, 195–200.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms*. The MIT Press.

da F. Costa, L.; Rodrigues, F. A.; Travieso, G.; and Boas, P. R. V. 2007. Characterization of complex networks: A survey of measurements. *Advances in Physics* 56(1):167–242.

Dague, P.; Devés, P.; Luciani, P.; and Taillibert, P. 1992. Analog systems diagnosis. In *Readings in Model-Based Diagnosis*. Morgan Kaufmann. 229–234.

Duhamel, P., and Rault, J.-C. 1979. Automatic test generation techniques for analog circuits and systems: A review. *IEEE Transactions on Circuits and Systems* 26(7):411–440.

Erdős, P., and Rényi, A. 1960. On the evolution of random graphs. In *Publication of The Mathematical Institute of the Hungarian Academy of Sciences*, 17–61.

Feldman, A.; de Castro, H. V.; van Gemund, A.; and Provan, G. 2013. Model-based diagnostic decision-support system for satellites. In *Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, USA*, 1–14.

Feldman, A.; Provan, G.; and van Gemund, A. 2010a. Stochastic algorithms for sequential model-based diagnosis. *Journal of Artificial Intelligence Research* 39:301–334.

Feldman, A.; Provan, G. M.; and van Gemund, A. J. C. 2010b. Approximate model-based diagnosis using greedy stochastic search. *J. Artif. Intell. Res. (JAIR)* 38:371–413.

Fenton, W.; McGinnity, M.; and Maguire, L. 2001. Fault diagnosis of electronic systems using intelligent techniques: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 31(3):269–281.

Ho, C.-W.; Ruehli, A. E.; and Brennan, P. A. 1975. The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems* 22(6):504–509.

Hongkui, Y., and Pengnian, Q. 1988. A new approach to analog fault diagnosis and an experimental diagnostic system. In *Proc. ISCAS'88*, 1175–1178. IEEE.

Kabisatpathy, P.; Barua, A.; and Sinha, S. 2005. *Fault Diagnosis of Analog Integrated Circuits*, volume 30. Springer.

Karnopp, D. C.; Margolis, D. L.; and Rosenberg, R. C. 2006. *System Dynamics: Modeling and Simulation of Mechatronic Systems*, volume 3. John Wiley & Sons New Jersey.

Kielkowski, R. M. 1994. *Inside SPICE: Overcoming the obstacles of circuit simulation*. McGraw-Hill.

Korzybski, M. 2008. Dictionary method for multiple soft and catastrophic fault diagnosis based on evolutionary computation. In *Proc. ICSES'08*, 553–556. IEEE.

Liu, D., and Starzyk, J. A. 2002. A generalized fault diagnosis method in dynamic analogue circuits. *International Journal of Circuit Theory and Applications* 30(5):487–510.

Milor, L. S. 1998. A tutorial introduction to research on analog and mixed-signal circuit testing. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 45(10):1389–1407.

Minhas, R.; Kleer, J. D.; Matei, I.; Saha, B.; Janssen, B.; Bobrow, D.; and Kurtoglu, T. 2014. Using fault augmented modelica models for diagnostics. In *Proc. 10th International Modelica Conference*.

Mosterman, P. J., and Biswas, G. 1996. A formal hybrid modeling scheme for handling discontinuities in physical system models. In *Proc. AAAI/IAAI'96*, 985–990.

Mosterman, P. J., and Biswas, G. 1999. Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 29(6):554–565.

Nagel, L. W., and Pederson, D. O. 1973. SPICE (simulation program with integrated circuit emphasis). Technical Report ERL-M382, EECS Department, University of California, Berkeley.

Navid, N., and Willson, Jr., A. N. 1979. A theory and an algorithm for analog circuit fault diagnosis. *IEEE Transactions on Circuits and Systems* 26(7):440–457.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. 2002. *Numerical Recipes in C++*. Cambridge University Press.

Samantaray, A. K.; Medjaher, K.; Bouamama, B. O.; Staroswiecki, M.; and Dauphin-Tanguy, G. 2006. Diagnostic bond graphs for online fault detection and isolation. *Simulation Modelling Practice and Theory* 14(3):237–262.

Sedgewick, R. 2002. *Algorithms in C - Part 5: Graph Algorithms*. Addison-Wesley.

Tadeusiewicz, M., and Hałgas, S. 2006. An algorithm for multiple fault diagnosis in analogue circuits. *International Journal of Circuit Theory and Applications* 34(6):607–615.