Optimizing Model-Based Diagnosis Complexity for Analogue Linear Systems

Alexander Feldman and **Gregory Provan**
University College Cork, Cork, Ireland
e-mail: a.feldman@ucc.ie, g.provan@cs.ucc.ie

Abstract

Fault diagnosis of analogue systems is a challenging task and no fully automated solution exists. We address the task of diagnosing hard faults in linear analogue systems. We develop an algorithm that dynamically reduces the size of the simulation model during diagnostics inference. We compare our simulation approach to the approach used by industry and academia and empirically demonstrate that our algorithm provides significant speedups on a benchmark of analogue circuits with regular and semi-random topologies. We apply our algorithm to the model of a real-world satellite and demonstrate 40 times speedup. We measure a significant performance increase (200 times) for a configuration of the diagnostic search that achieves higher diagnostic accuracy, optimizing a trade-off in computational complexity versus diagnostic accuracy. Our framework can be directly applied to any system for which the bond graph energy formulation applies and the model optimization part of the algorithm can be used to improve the computational complexity and numerical stability of a large class of numerical solvers that require steady-state simulation as a part of their convergence process.

1 Introduction

Fault diagnosis of analogue systems is an important problem, given their prevalence in synthetic and natural domains. The most 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].

We can separate three sub-tasks of analogue fault diagnosis: detecting faulty behaviors, isolating the faulty components, and determining the parameters of the faulty components. The success of this process depends on accurate definitions of fault models. We can classify fault models for analogue systems into two categories: hard (catastrophic) faults and soft (parametric) faults. A *hard (catastrophic) fault model* characterizes 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. A *parametric fault model* characterizes when there exist deviations of component parameters that result in performance beyond acceptable limits.

No fully-automatic method exists for multiple-fault diagnosis of analogue systems, even for the heavily-studied domain of circuit diagnosis. This paper proposes a 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 existing approaches that do not employ such reductions [Korzybski, 2008].

The contributions of this paper are as follows: (1) we introduce a simulation-based search framework for diagnosis of analogue systems; (2) we design an optimization method that significantly improves the simulation performance in the case of failures; (3) we apply this approach to the domain of analogue electrical circuits; (4) we design a benchmark of analogue electrical circuits that represents a large class of circuit topologies; (5) we propose an analytical model that predicts speedups based on system topology; and (6) we experimentally show that the proposed optimization method improves the diagnostic performance as predicted by the analytical model.

2 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], and AI-based methods [Dague *et al.*, 1992]. 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]. 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]; however, many aspects of the problem are still open, and no fully-automatic method exists for multiple-fault generic analog circuit diagnosis.

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].

This work differs from the circuit topology-modification algorithms based on qualitative circuit models (e.g., [Hotz *et al.*, 1997]). In this qualitative circuit work, a qualitative structural model of a circuit is transformed (using star-delta transformations and series-parallel reductions) to generate a structure more suitable for fault simulation. The original approach, the SDSP method [Hotz *et al.*, 1997], was limited to resistive networks that consist of one voltage source and an unlimited number of resistors, but was generalized in [Snooke and Lee, 2013]. The circuit topology transformations in the SDSP method are performed *prior to* any inference (as opposed to during the course of diagnostics inference in our approach), and do not cover the cases addressed in our approach, in which extremal values (0 or ∞) may occur in voltage or current. Further, the class of transformations, because they are for a different purpose, are quite different than the transformations employed in this article.

3 Concepts and Definitions

In what follows we present the basic concepts of circuit diagnosis.

3.1 System Description

A 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 our model in terms of two parts, the connection topology, and the component equations. We represent the topology of a system using a graph G.

Definition 1 (Topology Graph)**.** Given a model M with components $COMPS = \{c_1, c_2, \cdots, c_n\}$, and component connections (junctions) $Z = \{z_1, z_2, \cdots, z_l\}$, 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 represent the equations as follows. We represent a generic linear analogue system in terms of a relation between effort $\vec{x}$ and flow $\vec{z}$ vectors of variables, using $T\vec{x} = \vec{z}$, where T is an $n \times m$ matrix. 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 adopt a simulation approach called Modified Nodal Analysis (MNA) [Ho *et al.*, 1975]. This approach converts the component equations, together with the system topology, into a matrix representation that incorporates both equations and topology using the Kirchoff Current Law (KCL), in order to enable efficient simulation. Due to space constraints, we refer the reader to [Ho *et al.*, 1975] for a full description of MNA; here we focus on optimizations of MNA when using this approach for hard-fault diagnostics inference.

As an example of problem formulation using MNA, consider a simple circuit, as shown in Figure 1. In the circuit we identify 3 nodes at which we measure voltage, denoted $v_a,\ v_b,\ v_c$. This circuit has 2 voltage sources, $V_1,\ V_2$, and two current flows $i_{v1},\ i_{v2}$. The 3 nodes and 2 voltage sources, ($n = 3,\ m = 2$), result in a system with system equation $T\vec{x} = \vec{z}$ and characteristic nodal matrix T of size $(n+m) \times (n+m)$. The resulting matrix is shown in equation 1. For example, the first of 5 possible equations describing the system is obtained by applying KCL at v_a, i.e., $\frac{1}{R_1}v_a - i_{v1} = 0$. This is captured by the first row of the T matrix, and the first element of the x and z vectors.

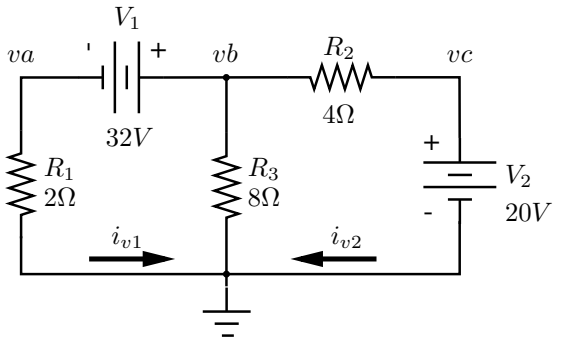


Figure 1: Simple circuit example

$$\begin{bmatrix} \frac{1}{R_1} & 0 & 0 & -1 & 0 \\ 0 & \frac{1}{R_2}+\frac{1}{R_3} & -\frac{1}{R_2} & 1 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \\ i_{v1} \\ i_{v2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V_1 \\ V_2 \end{bmatrix} \quad (1)$$

From our general system equation $T\vec{x} = \vec{z}$, our example maps to equation 1 as follows. The T matrix denotes only known quantities; in our circuit example it denotes the values of the passive elements (the resistors). The vector $\vec{x}$ denotes the unknown quantities (efforts and flows); for our circuit these correspond to node voltages and the currents through the independent voltage sources. The vector $\vec{z}$ denotes only known quantities (efforts and flows).

3.2 Model-Based Diagnosis

We will adopt a mode-based representation, i.e., we assume that the system can operate in a set Ω of 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 mode $\omega \in \Omega$ is an assignment to all variables in COMPS.

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

Definition 2 (Diagnostic Model)**.** Given a system that consists of a discrete set of components with a corresponding set of health parameters COMPS, a diagnostic model $M = \langle \mathrm{SD}, \mathrm{COMPS} \rangle$ is specified using a function $\mathrm{SD} = \bigcup_{\omega \in \Omega} \mathrm{SD}_\omega$.

In this article, we are typically given the flow vector $\vec{z}$ and must compute the effort vector via $\vec{x_\omega} = T_\omega^{-1}\vec{z}_\omega$, a process we call *simulation* of $\vec{x_\omega}$. Since SD is linear, we can simulate efficiently.

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

In real-world applications, straightforward simulation function (from definition 2) 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 3), and then identify the mode that minimises this function.

Definition 3 (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 some statistical estimator. In this paper we use the absolute residuals function:

$$R_{\text{abs}}(\hat{\vec{z}}, \vec{\alpha}) = \sum_{i=1}^{m} w_i \left| \hat{z}_i - \alpha_i \right| \tag{2}$$

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

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

Solving the above health estimation problem, while maintaining computational efficiency is the main goal of our framework.

4 Algorithms

In what follows we outline an algorithm for solving the problem given in definition 4. 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 (2) significantly improves the performance of the simulation step (3).

Algorithm 1 provides a generic search method for solving the parameter estimation problem in definition 4. 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 *et al.*, 2010].

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. 3) 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

Algorithm 1: DIAGNOSISSEARCH(M, α)

Input: M, model
Input: α, observation
Result: ω, 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 ← ∞
2 repeat
3     p⃗ ← SIMULATESYSTEM(MAKENETLIST(M, h))
4     r ← COMPUTERESIDUAL(p⃗, α)
5     if r < r_min then
6         ω ← h
7         r ← r_min
8 until h ← NEXTHEALTHASSIGNMENT(h, r) ≠ ∅;
9 return ω
```

in line 3 affects the overall complexity of the optimization algorithm.

The function MAKENETLIST takes a set of component models, a topology, and a parameter guess $\mathbf{h}$ and generates a netlist. This netlist is fed to the simulation function SIMULATESYSTEM. The netlist 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 circuit 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 *et al.* [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: L, netlist
Result: $\vec{z}$, voltage vector
Local variable: G, graph
Local variable: N, nodal matrix
Local variable: $\vec{j}$, current vector

```
1 G ← MAKEGRAPH(L)
2 G ← OPTIMIZEGRAPH(G)
3 N ← MAKENODALMATRIX(G)
4 j⃗ ← MAKECURRENTVECTOR(G)
5 z⃗ ← N^{-1} j⃗
6 return z⃗
```

Algorithm 2 starts by converting the input netlist L into a graph G. This is achieved by calling the MAKEGRAPH subroutine. The implementation of MAKEGRAPH is not discussed in this text because a netlist, describing a linear circuit, is already almost a graph, and constructing G from L is straightforward.

As is customary in SPICE, elements in L are represented as edges in G and nets in L are nodes in G. Edges in G are labeled with the type of the corresponding elements and their parameters.

The matrix N is directly generated from the graph G as described by Kielkowski [1994, p. 18]. The MNA approach, however, does not tolerate short-circuits as they lead

to infinite conductances. Further, open-circuits may cause singular matrices. To avoid these problems, even modern solvers replace short-circuits with small and open-circuits with large resistances. 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 increases significantly the overall complexity when solving thousands of times for various combinations of parameter values.

The time complexity of algorithm 2 is dominated by the matrix inversion in line 4. 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 2. 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 components with specific parameters from the model. 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 circuits that are closed through a voltage source are retained (lines 12–24).

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 ← true
3      foreach e ∈ E : ISDISSIPATIVEELEMENT(e) do
4          if Ω(e) < ε ∨ SRC(e) = DST(e) then
5              V ← V ∖ DST(e)
6              E ← E ∖ e
7              RECONNECT(E, SRC(e), DST(e))
8              stop ← false
9          if Ω(e) = ∞ then
10             E ← E ∖ e
11 until stop = false;
12 PUSH(s, EFFORTSOURCE(E))
13 while e ← POP(s) ≠ ∅ do
14     PUSH(s, ADJACENTEDGES(G, e))
15     P ← P ∪ e
16 repeat
17     stop ← true
18     foreach e ∈ E do
19         if |e| < 2 or e ∉ P then
20             V ← V ∖ DST(e)
21             E ← E ∖ e
22             stop ← false
23 until stop = false;
24 REMOVEORPHANEDVERTEXES(V, E)
25 return G = ⟨V, E⟩
```

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

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 $\Omega(e)$ returns the resistance of e. The functions SRC(e) and DST(e) return the two respective vertexes 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 a 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 2. As removing a short-circuit can cause another short-circuit (see again fig. 2), 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.

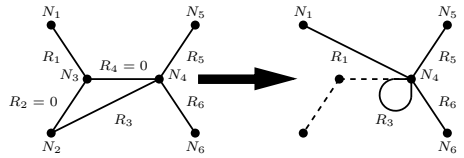


Figure 2: Removing the short-circuited resistors R_2 and R_4 in the left graph makes nodes N_2 and N_3 unnecessary. To preserve the circuit we have to disconnect R_1 from N_2 and to connect it to N_4. Similarly, both ends of R_3 have to be connected to N_4. At this moment, N_4 shorts R_3 and R_3 can be removed.

The second part of algorithm 3 (lines 12–22), removes all circuit elements that are not doubly-connected to a voltage 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 a voltage sources are added to the set of edges P. The DFS uses the function ADJACENTEDGES to generate all edges that are neighbors of an edge e.

The loop in lines 16–23 removes all edges that are not in P, i.e., they are not connected to a voltage 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.

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.

5 Average-Case Analysis

This section examines the type of complexity reduction that is possible for a typical model.

Definition 5 (Topology Graph)**.** Given a model M with components COMPS = $\{c_1, c_2, \ldots, c_n\}$, and junctions Z = $\{z_1, z_2, \ldots, z_l\}$, where component c_i occurs between junctions $z_j,\ z_k$, we map M into a graph $G(V, E)$ of vertices V and edges E by mapping each component to an edge and each junction to a vertex.

Parallel edges are allowed. The size of a graph $G(V, E)$ is denoted[2] as $|G(V, E)|$ where $|G(V, E)| = |E|$. 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 $E' = E \backslash e$, $V' = V \backslash \{W \cup v\}$ where v is incident to e and W is the set of singly connected vertices in $E \backslash v$.

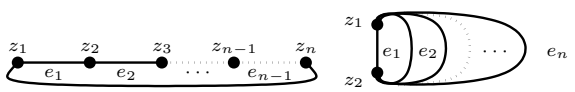
We have chosen the symbols $\searrow$ and $\nwarrow$ to visually resemble the set exclusion operator $\backslash$ as the two contraction operators lead to a decrease in the size of G. The graph contraction ratio is:

$$\rho = \frac{\sum_{e \in E} \left[|G(V, E) \searrow e| + |G(V, E) \nwarrow e| \right]}{2 |G(V, E)|^2} \tag{3}$$

The variable ρ in eq. 3 adds-up the effect of applying $\searrow$ and $\nwarrow$ to each edge in E. Note that $\rho \in [0; 1]$. The denominator in eq. 3 is simply $|G(V, E)|$-times the number of edges in $G(V, E)$. The coefficient 2 in the denominator is because we have two contraction operators: $\searrow$ and $\nwarrow$. In the empty graph, $\rho = 1$ by definition, i.e., no contraction is possible. 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 3: Boundary graph topologies

[2]This is an abuse of notation. Bollobás [1998], for example, uses $|G|$ to measure the order of $G(V, E)$, while the graph size is denoted as $e(G)$.

Proof (Sketch). We show the correctness of statement 1 and statement 2 can be proven in a simmilary way.

In a graph $G(V, E)$ with serial topology, there are exactly $G(V, E) = |E| = n$ edges as illustrated in figure 3(a). Choose any edge e. Applying $\searrow$ to e, removes only e from the original graph $G(V, E)$ and results in:

$$\sum_{e \in E} |G(V, E) \searrow e| = n\,(n-1) \tag{4}$$

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

$$\sum_{e \in E} |G(V, E) \searrow e| = n \tag{5}$$

Adding eq. 4 to eq. 5 gives us $\rho = \frac{n+n(n-1)}{2n^2}$ which simplifies to $\rho = \frac{1}{2}$. □

We next continue with analyzing the effect of algorithm 3 on random graphs. The subject of this analysis are random graphs, of the Erdős and Rényi type [1960], well below the phase transition, where the components are serially connected. Due to serially connecting all components, we refer to these graphs as "semi-random".

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

Theorem 1. *The total graph contraction for a serially-connected semi-random 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 such as the one shown in figure 3(a). 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 the paper of 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 we have that $\rho'' = \frac{1}{2}$. 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 semi-random graphs, we compute it experimentally which also shows the absolute improvement in computational complexity due to the symbolic preprocessing. This we do in the section that follows next.

6 Experiments

We have run a big number of experiments to characterize the performance of the algorithms described in section 4.

Algorithms 1–3 are implemented as a part of the (deleted for anonymity) diagnostic framework. The implementation is in C++ and uses the BOOST C++ library collection [Schäling, 2011].

6.1 Benchmark

Unlike with digital circuits [Brglez and Fujiwara, 1985] and to the best of our knowledge, there is no linear SPICE benchmark, so 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 4(a), consist of N resistors connected in series. Similarly, the N-PARALLEL circuits in figure 4(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 4(c). The N-MESH circuits consist of $2 \times N^2$ resistors arranged in a rectangular grid as shown in figure 4(d). Finally, the N-TREE circuits are complete N-ary trees of depth N. They are shown in figure 4(e).

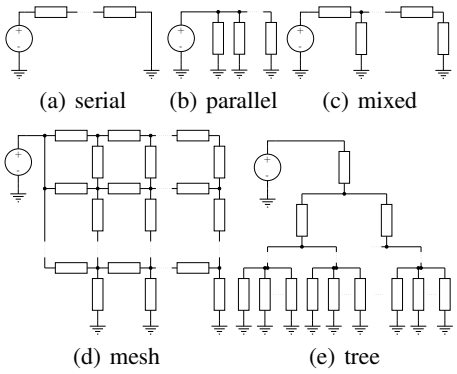


Figure 4: Scalable circuits with regular topology used for performance analysis

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 semi-random circuits. These circuits are generated by algorithm that is an adaptation of the random graph algorithm proposed by Sedgewick [2002, p. 42]. The reason for the modification is that the original algorithm produced multiple graph components for sparse graphs.

We generate random graphs by specifying the vertex/edge ratio r. The relation between the semi-random graph generation paramter 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 $0.1 \le r \le 0.4$ which translates to $2.06 \le \bar{d} \le 2.9$.

6.2 Results

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 4(c) and figure 4(d). For these two dense topologies, we have measured a performance gain of at most 9%. This is because 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 $N \times N$.

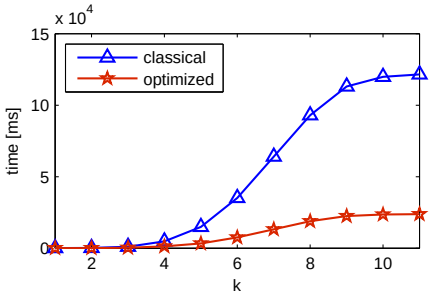


Figure 5: 11-SERIAL k-fault performance

The performance gain due to k-fault simulations is shown in figure 5. 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 6 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.

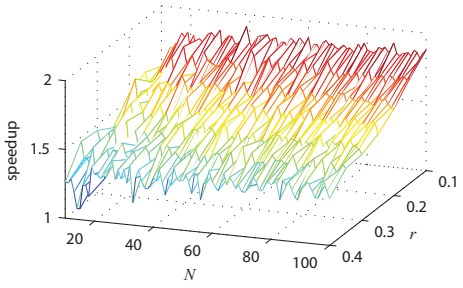


Figure 6: Performance gain for random circuits

Figure 6 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 6 and the average graph degree is -0.89 which validates the analysis in section 5. 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 of a real-world satellite [deleted, 1900]. It consists of, a.o, 96 heating elements (modeled as resistors) and 112 switches (modeled as resistors whose resistor changes as a result of a user-supplied command). The large number of switching components and the designed cold-redundancy leads 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.

7 Conclusions

In this paper we study the advantages of preprocessing the simulation model for steady-state analysis in diagnosis. The speedup due to pruning of parametrized components that cause discontinuity in the model depends on the topology of the model and we have established that throughout extensive experimentation on a benchmark of circuits. 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 for higher k when considering k-combination of faults.

Acknowledgments

One of the authors has been supported in the writing of this paper by SFI grant 12/RC/229.

References

[Aminian and Aminian, 2000] Mehran Aminian and Farzan Aminian. 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, 2000.

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

[Bollobás, 1998] Bélla Bollobás. *Modern Graph Theory*. Springer, 1998.

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

[Cormen *et al.*, 2001] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[Dague *et al.*, 1992] Philippe Dague, Philippe Devés, Pierre Luciani, and Patrick Taillibert. Analog systems diagnosis. In *Readings in Model-Based Diagnosis*, pages 229–234. Morgan Kaufmann, 1992.

[deleted, 1900] deleted. Deleted for anonymity. 1900.

[Duhamel and Rault, 1979] Pierre Duhamel and Jean-Claude Rault. Automatic test generation techniques for analog circuits and systems: A review. *IEEE Transactions on Circuits and Systems*, 26(7):411–440, 1979.

[Erdős and Rényi, 1960] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. In *Publication of The Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[Feldman *et al.*, 2010] Alexander Feldman, Gregory M. Provan, and Arjan J. C. van Gemund. Approximate model-based diagnosis using greedy stochastic search. *J. Artif. Intell. Res. (JAIR)*, 38:371–413, 2010.

[Ho *et al.*, 1975] Chung-Wen Ho, Albert E. Ruehli, and Pierce A. Brennan. The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems*, 22(6):504–509, June 1975.

[Hotz *et al.*, 1997] Lothar Hotz, Heiko Milde, Ralf Möller, and Bernd Neumann. Resistive networks revisited: Exploitation of network structures and qualitative reasoning about deviations is the key. In *Proc. 8th Int. Workshop on Principles of Diagnosis,(DX97)*, 1997.

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

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

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

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

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

[Press *et al.*, 2002] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[Samantaray *et al.*, 2006] Arun K. Samantaray, Kamal Medjaher, Belkacem Ould Bouamama, Marcel Staroswiecki, and Geneviéve Dauphin-Tanguy. Diagnostic bond graphs for online fault detection and isolation. *Simulation Modelling Practice and Theory*, 14(3):237–262, 2006.

[Schäling, 2011] Boris Schäling. *The Boost C++ Libraries*. XML Press, 2011.

[Sedgewick, 2002] Robert Sedgewick. *Algorithms in C - Part 5: Graph Algorithms*. Addison-Wesley, 2002.

[Snooke and Lee, 2013] Neal Snooke and Mark Lee. Qualitative order of magnitude energy-flow-based failure modes and effects analysis. *Journal of Artificial Intelligence Research*, 46:413–447, 2013.