Spectrum-Based Sequential Diagnosis

Alberto Gonzalez-Sanchez\(^1\), Rui Abreu\(^2\), Hans-Gerhard Gross\(^1\), and Arjan J.C. van Gemund\(^1\)

\(^1\) Department of Software Technology, Delft University of Technology, The Netherlands
\{a.gonzalezSanchez, h.g.gross, a.j.c.vangemund\}@tudelft.nl
\(^2\) Department of Informatics Engineering, University of Porto, Portugal
rui@computer.org

ABSTRACT

Often multiple observations are required to achieve acceptable diagnostic certainty. We present a spectrum-based sequential diagnosis approach coined SEQUOIA, that greedily selects tests out of a suite of tests to narrow down the set of diagnostic candidates with a minimum number of tests. SEQUOIA handles multiple faults, that can be intermittent, at polynomial time and space complexity. This is due to a novel, approximate diagnostic entropy estimation approach, which is based on a relatively small subset of diagnoses that cover almost all Bayesian posterior probability mass. Synthetic data shows, that the dynamic selection of the next best test based on the test results measured so far, allows SEQUOIA to achieve much better decay of diagnostic uncertainty compared to random test sequencing. Real programs, taken from the Siemens set, also show that SEQUOIA has better performance, except in a few cases where the diagnosis includes large fault sets, which affects the entropy estimation quality.

1 INTRODUCTION

In fault diagnosis an important performance metric is the accuracy of the multiple-fault diagnosis \(D\) returned by a diagnostic algorithm (DA). In many cases diagnostic accuracy is insufficient (\(D\) comprises too many candidates). Diagnostic uncertainty can be reduced by supplying more modeling information, and more observation information. Observation information can be in spatial sense (more probing points), and in temporal sense (more tests), the latter of which is the focus of this paper. Ignoring the computational cost of the DA itself, the diagnostic process may be viewed as (1) expending costs to generate observation input to the DA (testing cost \(C_t\)), and (2) expending costs to find the actual diagnosis from \(D\) returned by the DA by the diagnostician (diagnostic cost \(C_d\)). The latter is inverse to diagnostic accuracy (utility) as false positives and negatives in \(D\) result in wasted effort by the diagnostician finding the actually faulted components.

In sequential diagnosis (SD) one computes the optimal sequence of tests (observations) that, on average, minimizes \(C_t\) and \(C_d\). SD approaches are typically found in the (hardware) systems testing domain where system knowledge is encoded in terms of predetermined test matrices (e.g., (Shakeri et al., 2000)), or in the model-based diagnosis domain where system models are available (e.g., (Kuhn et al., 2008; Feldman et al., 2010)). In the software domain, the domain considered in this paper, the problem of diagnosis has received considerable attention. Due to complexity problems most model-based approaches are limited to small programs and/or must take a single-fault assumption. Typically, statistical approaches are used, that abstract from the program code and compute the correlation between component involvement in tests on the one hand (expressed in terms of a so-called spectra), and the test outcomes (pass/fail) on the other, deriving a ranking of suspect components. While these approaches essentially involve multiple observations (e.g., an entire regression test suite) they do not qualify as SD since the choice of tests is random (a passive diagnosis approach), rather than generating/selecting the next best test based on the diagnosis obtained thus far (an active diagnosis approach). Hence, the decrease of \(C_d\) per test is far from optimal.

In this paper we present a spectrum-based SD approach and an associated algorithm dubbed SEQUOIA (SEQuencing fOr dIAGnosis) that greedily computes a sequence out of a large set of given tests that delivers near-optimal diagnostic performance in terms of the decay of \(C_d\) as function of the number of tests. Unlike statistical techniques, which have insufficient diagnostic accuracy to select the next best test, we use an approximate, Bayesian reasoning approach which

\footnote{Not necessarily in terms of the algebraic sum as testing cost and diagnostic cost can have different dimensions, e.g., CPU time, and human labor, respectively.}
provides the required accuracy. Similar to approaches based on a test coverage matrix such as (Shakeri et al., 2000), the tests are selected from a fixed test set. Unlike multiple-fault, matrix-based approaches, however, our approximate, Bayesian approach can handle large system sizes and test set sizes. Furthermore, unlike MBD approaches such as (Kuhn et al., 2008; Feldman et al., 2010) no system information is required, other than component test coverage, which is typically known from test execution profiling.

The paper makes the following contributions.

- We present spectrum-based SD, and associated concepts and terminology;
- We describe our greedy SD algorithm SEQUOIA, featuring a novel, approximate diagnosis kernel that provides good test selection capabilities at polynomial cost;
- We evaluate SEQUOIA for synthetic tests which allow us to assess its performance potential for parameters such as system size, test set size, number of faults, component coverage probability, component fault intermittency, etc.;
- We evaluate SEQUOIA for real-world programs taken from the SIR benchmark suite of programs (Do et al., 2005), namely the Siemens suite.

Our results show that especially for the first tests the decay of diagnostic cost \( C_d \) per test of SEQUOIA is much better than random.

The paper is organized as follows. In Section 2 we present standard concepts and notations in spectrum-based fault diagnosis. Section 3 presents SEQUOIA. In Sections 4 and 5 we evaluate SEQUOIA using synthetic and real programs, respectively. Section 6 reviews related work on sequential fault diagnosis, while Section 7 concludes the paper.

2 PRELIMINARIES

In this section we describe spectrum-based fault diagnosis.

2.1 Basic Definitions

Definition A diagnostic system \( DS = (SD, COMPS, OBS) \), where \( SD \) is a propositional theory describing the behavior of the system, \( COMPS = \{c_1, \ldots, c_M\} \) is a set of components in \( SD \), of which \( M_f = 0, \ldots, M \) can be simultaneously faulty, and \( OBS \) is a set of observable variables in \( SD \).

With each component \( c_j \in COMPS \) we associate a health variable \( h_j \), which denotes component health. The health states of a component are either healthy (\( h_j \text{ true} \)) or faulty (\( h_j \text{ false} \)).

Definition An h-literal is \( h_j \) or \( \neg h_j \) for \( c_j \in COMPS \).

Definition An h-clause is a disjunction of h-literals containing no complementary pair of h-literals.

Definition A conflict of \((SD, COMPS, OBS)\) is an h-clause of negative h-literals entailed by \( SD \cup OBS \).

Definition Let \( J_N \) and \( J_P \) be two disjoint sets of components indices, faulty and healthy, respectively, such that \( COMPS = \{c_j \mid j \in J_N \cup J_P\} \) and \( J_N \cap J_P = \emptyset \). We define \( d(J_N, J_P) \) to be the conjunction \((\bigwedge_{j \in J_N} \neg h_j) \land (\bigwedge_{j \in J_P} h_j)\).

A diagnosis candidate is a sentence describing one possible state of the system, where this state is an assignment of the status healthy or not healthy to each system component.

Definition A diagnostic candidate \( d(J_N, J_P) \) for \( DS \) given an observation \( obs \) over variables in \( OBS \) is such that

\[
SD \land obs \land d(J_N, J_P) \neq \bot
\]

In the remainder we refer to \( d(J_N, J_P) \) simply as \( d \), which we identify with the set \( J_N \) of indices of the negative literals.

A minimal diagnosis is a diagnosis that is not subsumed by another of lower fault cardinality (i.e., number of negative h-literals \( M_f = |d| \)). A minimal diagnosis is the minimal hitting set over all conflicts.

Definition A minimal hitting set of a collection \( S \) is a set \( d \) such that

\[
\forall \forall s_i \in S, s_i \cap d \neq \emptyset \land \exists d' \subset d : s_i \cap d' = \emptyset
\]

i.e., \( d \) contains at least one element from each subset in \( S \), and no proper subset of \( d \) is a hitting set. There may be several minimal hitting sets for \( S \), which constitutes a collection of minimal hitting sets.

Definition A diagnostic report \( D \) = \(<d_1, \ldots, d_k, \ldots, d_K>\) is an ordered set of all \( K \) minimal diagnosis candidates, for which \( SD \land obs \land d_k \neq \bot \).

The candidates are ordered by posterior probability.

The diagnostic accuracy of a diagnostic report \( D \) depends on the ranking of the actual system’s fault state \( d^* \) within \( D \). Assuming a diagnostician traverses \( D \) top to bottom, a diagnostic approach that produces a \( D \) where \( d^* \) is ranked on top has higher accuracy (i.e., generates less cost \( C_d \)) than an approach that ranks \( d^* \) lower.

The derivation of diagnostic candidates is based on a Bayesian approach, i.e., (1) deducing whether a candidate diagnosis \( d_k \) is consistent with the observations, and (2) the posterior probability \( \text{Pr}(d_k) \) of that candidate being the actual diagnosis. With respect to (1), rather than computing \( \text{Pr}(d_k) \) for all possible candidates, just to find that most of them have \( \text{Pr}(d_k) = 0 \), search algorithms are typically used instead, such as CDA* (Williams and Rago, 2007), SAFARI (Feldman et al., 2010), or just a minimal hitting set (MHS) algorithm when conflict sets are available (e.g. (Abreu and van Gemund, 2009; de Kleer and Williams, 1987)), but the Bayesian probability framework remains the basis.

In this section we will briefly describe the contemporary, spectrum-based approach to the derivation of candidates, and their posterior probability.

Spectrum-based Candidate Generation

Consider a particular process, involving a set of components, that either yields a nominal result or a failure. For instance, in a logic circuit a process is the
sub-circuit (cone) activity that results in a particular primary output. In a copier a process is the propagation of a sheet of paper through the system. In software a process is the sequence of software component activity (e.g., statements) that results in a particular return value. The result of a process is either nominal ("pass") or an error ("fail").

**Definition** Let \( S_f = \{c_j|c_j\text{ involved in a failing process}\} \), and let \( S_p = \{c_j|c_j\text{ involved in a passing process}\} \), denote the fail set and pass set, respectively. In software fail and pass sets are also known as (execution) spectra (Abreu et al., 2007; Harrold et al., 1998), and originate from dynamically profiling the software components (e.g., statements or modules) during each program run. As, next to the pass/fail feedback, the spectra are the only input to the diagnosis, the diagnostic approach is termed spectrum-based diagnosis.

**Definition** Let \( N \) denote the number of passing and failing processes. Let \( N_f = N_p \), \( N_f + N_p = N \), denote the number of fail and pass sets (spectra), respectively. Let \( A \) denote the \( N \times M \) activity matrix of the system, where \( a_{ij} \) denotes whether component \( j \) was involved in process \( i \) (\( a_{ij} = 1 \)) or not (\( a_{ij} = 0 \)). Let \( e \) denote the error vector, where \( e_i \) signifies whether process \( i \) has passed (\( e_i = 0 \)) or failed (\( e_i = 1 \)).

The observations \((A,e)\) are the only input to the diagnosis process (see Figure 1).

Each component \( c_j \) is associated with a weak model

\[
h_j \Rightarrow (ok_{j,\text{inp}} = ok_{j,\text{out}})
\]

where \( ok_{j,\text{inp}} \) and \( ok_{j,\text{out}} \) denote the (binary) correctness of the component’s input and output, respectively. The correctness of each program run is modeled by a chain of the involved component models, where, by definition, \( ok_{\text{inp}} \) of the first component in the chain is true, and where \( ok_{\text{out}} \) of the last component reflects whether the run passes or fails. In the former case, a pass set is recorded, whereas in the latter case a fail set (i.e., a conflict set) is recorded.

In our abstract, spectrum-based approach to software diagnosis, we must take into account the fact that our software component model must express intermittent failure behavior. Consider, the integer division \( x / 10 \) where \( 10 \) is a fault that should have been 15. Consider two input values \( x = 15 \), and \( x = 20 \), respectively. In the first case, the component produces a correct output, whereas in the second case the component fails. If both inputs were modeled as correct (e.g., because they were produced by nominal, components) the division component exhibits intermittent failure behavior. Although a weak fault model (that does not stipulate particular fault behavior) admits any output behavior, modeling inconsistently failing (software) components merely in terms of weak models would result in degraded diagnostic performance (Abreu et al., 2008).

Next to the fail sets (conflicts), from which the diagnostic candidates \( d_k \) are derived (e.g., applying a minimal hitting set algorithm (de Kleer and Williams, 1987), recent approaches that allow intermittency also take into account the pass sets. We compute the candidates by applying our STACCATO MHS algorithm (Abreu and van Gemund, 2009) to the fail and the pass sets. STACCATO computes MHS solutions in approximate order of posterior probability, exploiting the pass sets to optimize search order. As a result, all probability mass is concentrated in the first \( \lambda \) MHS solutions in \( D \), where even small \( \lambda (O(M)) \) produces no difference in terms of \( C_d \) compared to a non-truncated diagnosis \( D \) (Abreu and van Gemund, 2009).

Due to this feature the time complexity of STACCATO is merely \( O(M \cdot M \cdot (N + \log M)) \). The performance benefits of the approximate approach is exemplified by the fact that matrices with \( M = 1,000,000 \), \( N = 1,000 \), and \( C = 1,000 \) are processed with an average solution rate of 88.6 ms (2.3 GHz Intel Pentium-6 PC with 4 GB memory).

**Candidate Probability Computation**

Given the multitude of candidates that are typically generated, the ranking within \( D \) induced by posterior probability computation is critical to diagnostic accuracy. Let \( \Pr(j) = p_j \) denote the prior probability that a component \( c_j \) is at fault. Assuming components fail independently the prior probability of a candidate \( d_k \) is given by

\[
\Pr(d_k) = \prod_{j \in S_{N_f}} \Pr(\{j\}) \cdot \prod_{j \in S_{N_p}} (1 - \Pr(\{j\}))
\]

For each observation \( obs_i = (A_{ik}, e_i) \) the posterior probabilities are updated according to Bayes’ rule

\[
\Pr(d_k|obs_i) = \frac{\Pr(obs_i|d_k) \cdot \Pr(d_k)}{\Pr(obs_i)}
\]

The denominator \( \Pr(obs_i) \) is a normalizing term that is identical for all \( d_k \) and thus needs not be computed directly. \( \Pr(obs_i|d_k) \) is defined as

\[
\Pr(obs_i|d_k) = \begin{cases} 0 & \text{if} \ obs_i \land d_k \text{ inconsistent;} \\ 1 & \text{if} \ obs_i \text{ is unique to } d_k; \\ \varepsilon & \text{otherwise.}
\end{cases}
\]

As mentioned earlier, rather than updating each candidate only candidates derived from the diagnostic search algorithm are updated, implying that the 0-clause need not be considered. For the large majority of cases, the \( \varepsilon \)-clause applies. Many policies exist for \( \varepsilon \) (de Kleer, 2006). For intermittent components the epsilon policy is given by

\[
\varepsilon = \begin{cases} \prod_{c_j \in d_k \land a_{ij} = 1} g_j & \text{if} \ e_i = 0 \\ 1 - \prod_{c_j \in d_k \land a_{ij} = 1} g_j & \text{if} \ e_i = 1
\end{cases}
\]

where \( g_j \) models the component’s failure intermittency, i.e., probability that a faulty component will not
cause a failure when covered in a test (a false negative). The epsilon policy assumes an or-model: the test will fail when either faulty component fails (an or-model) (Kuhn et al., 2008; Abreu et al., 2009b).

In hardware, intermittency is an established concept (Abreu et al., 2009a; de Kleer, 2009b). In software, intermittency is related to the concepts of coincidental correctness (Wang et al., 2009), failure exposing potential (Rothermel et al., 2001), and testability as determined through mutation analysis (Voas, 1992). Many ways exist to approximate $g_j$ (de Kleer, 2009a; Abreu et al., 2009b; Rothermel et al., 2001; Voas, 1992). In this paper we will assume the $g_j$ are available, either by definition (in Section 4), or from prior mutation analysis (in Section 5).

2.2 Sequential Diagnosis

Sequential diagnosis (SD) is the process of computing the sequence of observations that will optimally reduce diagnostic uncertainty. As in this paper we shall consider unit test cost, the SD problem can be solved by a myopic strategy, i.e., finding the next best test. In terms of $C_d$ SD aims to find the next test that achieves the highest decay of $C_d$. In a spectrum-based perspective, each test corresponds to a row $i$ in $A$, and $A$ is referred to as test (coverage) matrix of the program and associated test suite. The SD problem corresponds to finding the next best test $i$ in $A$, yielding an optimal ordering of rows in $A$.

The diagnostic procedure outlined in the previous sections does not consider how $A$ is composed. $A$ is merely processed in terms of MHS candidate generation and the subsequent, Bayesian posterior probability computation per candidate, yielding the diagnostic report $D$. From SD perspective, this would correspond to a random ordering (of rows), i.e., a passive approach to diagnosis.

In spectrum-based SD we compute a row ordering, by actively computing the next best row $i$, that, on average, will optimally reduce $C_d$, given the diagnosis computed so far, based on the outcomes of the tests executed so far. The performance indicator of SD is $C_d$'s decay rate, which should be higher than for random ordering, reflecting the added value of actively selecting tests based on the diagnosis obtained thus far in the test sequence.

3 SEQUOIA

SEQUOIA is a spectrum-based SD algorithm that greedily selects the next best test (row) from $A$ based on the diagnosis obtained thus far. Since our principal optimization target $C_d$ cannot be computed (since we don’t know the actual health state $d_k$) we can only reason in terms of the expectation $E[C_d]$. Experiments have shown that the entropy $H$ of $D$ is a very good predictor for $E[C_d]$ as it expresses the uncertainty in $D$. Consequently, we shall use the entropy drop $\Delta H$ (aka information gain (Johnson, 1960)) of a diagnosis $D$ as test selection criterion. The next best test is the one that yields the highest $\Delta H$ averaged over the two possible test outcomes (pass/fail), according to

$$\Delta H(D, i) = \Pr(e_i = 0) \cdot H(D|e_i = 0) + \Pr(e_i = 1) \cdot H(D|e_i = 1)$$

where $D|e_i = 0$ represents the updated diagnosis if test $i$ passes, $D|e_i = 1$ if it fails, and $H(D)$ is defined as

$$H(D) = - \sum_{d_k \in D} \Pr(d_k) \cdot \log_2(\Pr(d_k))$$

For a matrix $A$ that accommodates all $2^N$ possible tests, and for one, persistent fault ($M_f = 1$, $g_j = 0$), information gain-based sequencing effectively performs a binary search, bisecting the set of candidate components after each test. Hence, the decay rate is optimal ($C_d$ is halved after each test, which is the best achievable for tests that have a binary outcome).

Conceptually, when considering all the possible test outcome combinations, a test suite prioritized for diagnosis is a binary tree with $O(2^N)$ nodes. However, as only the nodes corresponding to the current test path are expanded, only the $O(N)$ test sequence is returned. In theory, given our Bayesian approach to multiple, intermittent fault diagnosis, computing $H$ has exponential cost. Hence, computing the information gain at each step in SD has exponential cost. Low-cost, statistical approaches to fault diagnosis would solve this problem. However, experiments have shown that their diagnostic accuracy is far from sufficient to serve as sufficiently accurate predictors of information gain. Especially for systems where $g_j$ is high (i.e., many false negatives), the differences in information gains of tests can be small, requiring a diagnostic technique that provides a highly accurate diagnosis $D$ and hence $H$ to select the right tests. Bearing in mind the fact that each step in SD involves selecting the best test out of $N$ candidates (i.e., computing $N$ diagnoses per step), finding a diagnostic technique that pairs accuracy with sufficiently low complexity is a formidable challenge in SD.

3.1 Approximate Information Gain

In this section we address the above challenge. From Section 2 it would seem that STACCATO is a good candidate to generate $D$ as, in combination with the Bayesian posterior update scheme, it pairs good diagnostic accuracy with low cost. However, while diagnostic reports comprising minimal diagnoses generally have high practical utility in fault localization, they have limited use in SD. Consider the matrix $A$ corresponding to a single triple-fault $D = \{\{1, 2, 3\}\}$. After the first test, an MHS algorithm will return the singleton $D = \{\{1\}\}$ with $H = 0$. Regardless whether we consider test $i = 2$ or $i = 3$ the information gain will be zero since $D$ will return a single double-fault, with, again, $H = 0$.

While the diagnostic accuracy of $D$ is not the issue (minimal diagnoses are often preferred), the quality of the $H$ computation is compromised due to the fact that essential, non-minimal candidates in $D$ are missing. Consider the same $A$ but with $D$ containing all candidates. After the first test we have $D = \{\{1\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$, which now yields $H = 0.94$ (for $g_j = 0$, $p_j = 0.1$). As the next
two tests provide equal information gain we will consider the second row as best next test, which yields \( D = \{\{1, 2\}, \{1, 2, 3\}\} \), with \( H = 0.46 \). Only after the third test we obtain \( H = 0 \) as only the triple fault is left in \( D \).

As computing all non-minimal hitting sets in \( D \) is exponentially complex, we present an approximation, based on extending the MHS solutions returned by STACCATO with a limited set of candidates that are subsumed by the MHS candidates. The candidates that are added are the MHS candidates extended with one additional component (high posteriors). Because of this limitation, we refer to this extension as first-order approximation of the entire non-minimal hitting set in \( D \). In the above example, after the first test the single MHS solution \( \{1\} \) in \( D \) would be extended to \( D = \{\{1\}, \{1, 2\}, \{1, 3\}\} \), which yields \( H = 0.86 \), which is a much better approximation to \( H = 0.94 \) than \( H = 0 \).

Figure 2 shows the evolution of \( H \) versus \( i \) for a randomly generated \( A \) with \( N = 150, M = 10 \), \( M_f = 3, g_j = U(0, 1) \). Three plots are shown, i.e., \( H \) for the perfect \( D \) with all non-minimal candidates ("Full \( D \"), the first-order approximation ("MHS+1"), and the original MHS-based \( D \"MHS\"\), respectively. The results clearly show that a first-order approximation already delivers good accuracy. This is significant since the cost of generating first-order subsumptions is non-negligible (in this example approximately 100 additional candidates are generated per MHS).

3.2 Algorithm

Algorithm 1 describes the SEQUOIA algorithm. The algorithm takes as arguments the set of components \( c_i \) and the associated priors \( p_i \) and failure intermitteness \( g_j \), the test matrix \( A \), and two entropy estimation parameters: the maximum number of minimal diagnosis candidates \( \lambda \) as generated by STACCATO, and the maximum number of non-minimal candidates \( \gamma \) subsequently generated by the 1st-order hitting set extension.

\( T \) represents the index set of the available tests. \( A' \) (initially empty) represents the version of \( A \) whose rows are sorted by the sequencing algorithm, and \( e \) stores the pass/fail outcomes of the tests. The INITIALD function initializes \( D \) to the first \( \gamma \) candidates in descending order of posterior probability, each \( d_k \) probability is stored in the array PrD.

The main loop performs the test sequencing. First, a test is selected from a subset of available tests \( T \) (SAMPLE) that yields the highest information gain. Merely considering a subset (e.g., 100) instead of all available tests \( O(N) \) yields a substantial gain, while the sacrifice in optimality is small. Next, the test is executed based on the current \( D \) and Pr. The selected test and its outcome are appended to \( A' \) and \( e \) (the \( \cup \) operators denote appending a matrix row, and vector element, respectively). Second, \( D \) is updated by the STACCATO algorithm, and subsequently extended to contain the first order subsumed (non-minimal) candidates by EXPANSUBSUMED. The combined complexity of these steps is \( O(M \cdot N) \). To keep complexity under control, only the first \( \lambda \) minimal candidates are generated. Likewise, only the first \( \gamma \) first-order subsumed candidates are generated. Third, the probabilities of the newly obtained candidates are calculated by the BAYESUPDATE function. Finally, the selected test (index) \( i \) is removed from the set of available tests \( T \).

Time/Space Complexity

To minimize computational cost SEQUOIA only stores a limited number of candidates in \( D \). As a general guideline from our experiments, \( \lambda = M \) and \( \gamma = 10 \cdot M \) provide good performance.

For every iteration of the main loop, the information gain \( \Delta H \) has to be calculated for each of the test cases in SAMPLE(T). This has a \( O(|D| \cdot \sigma) \) cost, where \( \sigma \) is the size of the sample. Given the current definition of \( \gamma \) in our experiments, the cost of the information gain phase corresponds to \( O(M \cdot \sigma) \), which in the worst case is \( O(M \cdot N) \) (no sampling).

The STACCATO algorithm has a complexity of \( O(MN + M \log M) \) (Abreu and van Gemund, 2009), and the subsequent expansion in EXPANSUBSUMED is \( O(|D|) \), therefore \( O(M) \). In all practical situations, the complexity of STACCATO can be considered \( O(MN) \) as the effect of the logarithmic term is only noticeable for extremely large \( M \) relative to \( N \) (\( M > 2^N \)).

The Bayesian update of a candidate in BAYESUPDATE has a complexity of \( O(N) \) per candidate, which in total accounts for \( O(|D| \cdot N) \), and, given the defined value of \( \gamma = 10 \cdot M \), yields \( O(MN) \).

The combined complexity of each of the steps in the inner loop is, in light of these results, \( O(MN) \), albeit with a significant factor.

\(^2\)The information gain selected by the argmax is the highest order statistic from the information gain distribution in the test set. Its mean value grows less than logarithmic with the sample size (David, 1970).
4 THEORETICAL EVALUATION

In order to assess the performance potential of our SD approach we generate synthetic observations based on sample $(A, e)$ generated for various values of $N, M$, and number of injected faults $M_f$. The motivation for using synthetic data next to data from real programs is the ability to study the effect of the various parameters in a controlled setting whereas real programs only represent a few parameter settings in the multi-dimensional parameter space.

Component activity $a_{ij}$ is sampled from a Bernoulli distribution with parameter $\rho$, i.e., the probability a component is involved in a row of $A$ equals $\rho$. For the $M_f$ faulty components $c_j$ (without loss of generality we select the first $M_f$ components) we also set $g_j$. Thus the probability of a component being involved and generating a failure equals $\rho \cdot (1 - g_j)$. A row $i$ in $A$ generates an error ($\epsilon_i = 1$) if at least 1 of the $M_f$ components generates a failure (or-model). Measurements for a specific $(N, M, C, r, g)$ scenario are averaged over 1,000 sample matrices, yielding a coefficient of variance of approximately 0.02.

Figure 3 shows the evolution of relative diagnostic cost $C_d(M - M_f)$ of SEQUOIA (IG) and random sequencing (RND) versus the number of tests, for $M = 100$ components with $M_f = 3, 5$, respectively. The matrix comprises $N = 250$ tests. SEQUOIA’s parameters are set to $\lambda = M$ and $\gamma = 10 \cdot M$.

For more faults ($M_f$) it generally takes more tests to achieve diagnostic accuracy. The plots clearly show the advantage of SD compared to random selection (i.e., just selecting the next row in the matrix without applying any knowledge on the current diagnosis and information gain per test). In particular, the amount of tests required for an acceptable diagnosis is far less. Note that in the $M_f = 3$ case the advantage seems less. However, this is due to the fact that only $N = 250$ tests are available, and the zero asymptote for $M_f = 5$ is not yet reached at $N = 250$.

5 EMPIRICAL EVALUATION

In this section, we evaluate SEQUOIA for the Siemens benchmark set (Do et al., 2005) of programs.

5.1 Experimental Setup

The Siemens suite is composed of seven programs. Every single program has a correct version and a set of faulty versions of the same program. Each faulty version contains exactly one fault. For each program a set of inputs is also provided, which were created to provide full component coverage overall.

Table 1 lists some information about the programs used in our experiments, where $M$ corresponds to the number of lines of code (components in this context).

For our experiments, we have extended the subject programs with program versions where we can activate arbitrary combinations of multiple faults. For this purpose, we generate random faults in the code by applying mutation operators used in mutation testing. We will restrict our experiments to systems with 3 faults. The reason to this is twofold. First, the difficulty to generate faulty versions which exercise all of their faults increases with $M_f$ (we require all the seeded faults to be executed in the matrices used in our experiment, and for these small programs faults often lead to other faults not being executed). Secondly, given the small size of the programs, using more than 3 faults would increase the fault density to unrealistic levels in software (above 10 faults per KLOC).

The coverage matrices $A$ are obtained using the profiling facility within Zoltar, a fault localization tool (Janssen et al., 2009a). As each program suite includes a correct version, we use the output of the correct version as reference. We characterize a run as failed if its output differs from the corresponding output of the correct version, and as passed otherwise. The intermittency $g_j$ values (per program the average $g$ value over the various seeded faults is shown in Table 1) were calculated using a mutation study similar to (Rothermel et al., 2001). As with our theoretical evaluation, SEQUOIA’s parameters are set to $\lambda = M$ and $\gamma = 10 \cdot M$.

5.2 Performance Results

Figure 4 shows the evolution of the relative diagnostic cost for $schedule$ for $M_f = 1$ and $M_f = 3$. It is

![Figure 3: $C_d$ evolution for synthetic A](image)

![Table 1: The subject programs](image)

![Figure 4: $C_d$ evolution for schedule](image)
clear that the decay of $C_d$ for SEQUOIA is much bet-
ter than for random test selection. Due to space lim-
itations, we do not show the plots for all programs.

Rather, we summarize the performance results in terms
of the area under the SEQUOIA and RND curve for $C_d$
above the asymptote for large $N$ (which is determined
by the program and fault seeding, rather than a DA),
according to

$$\frac{1}{N} \sum_{i=1}^{N} (C_d(i) - C_d(N))$$

Table 2 shows the results for $M_f = 1$, and $M_f = 3$.
For $M_f = 1$ the first five programs show signif-
ificant improvements for SEQUOIA, while for tcas and
tot_info SEQUOIA performs worse. For $M_f = 3$
we see the same trend, while also for print_tokens
SEQUOIA performs worse. In these cases it turns out
that $D$ contains a too large quantity of equal minimal
diagnoses (so-called ambiguity groups), the 1st-order
expansion of which results into a number of candi-
dates that by large exceeds $\gamma$. Consequently, too many
(unequal) minimal diagnoses are ignored, resulting in an
tropy estimation that is too inaccurate to select the
right tests.

6 RELATED WORK

Software fault diagnosis has received quite some atten-
tion. However, apart from statistical, spectrum-based
approaches (Jones et al., 2002; Abreu et al., 2007; Lib-
lit et al., 2005; Renieris and Reiss, 2003), which are es-
entially examples of passive SD, most (model-based)
work considers a single observation (run) (Mayer and
Stumptner, 2008; Groce, 2004; Wotawa et al., 2002;
Peischl and Wotawa, 2003) in the context of single-
fault diagnosis. An exception is the multiple-fault rea-
dition. However, apart from statistical, spectrum-based
approaches, a fault matrix is used that

$$\sum_{i=1}^{N} (C_d(i) - C_d(N))$$

Table 2: Sequencing Performance

<table>
<thead>
<tr>
<th></th>
<th>$M_f = 1$</th>
<th>$M_f = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>print_tokens</td>
<td>0.0303 0.0060</td>
<td>0.0432 0.0574</td>
</tr>
<tr>
<td>print_tokens2</td>
<td>0.0280 0.0016</td>
<td>0.0435 0.0260</td>
</tr>
<tr>
<td>replace</td>
<td>0.0283 0.0046</td>
<td>0.0588 0.0169</td>
</tr>
<tr>
<td>schedule</td>
<td>0.0288 0.0172</td>
<td>0.0636 0.0234</td>
</tr>
<tr>
<td>schedule2</td>
<td>0.0111 0.0064</td>
<td>0.0683 0.0132</td>
</tr>
<tr>
<td>tcas</td>
<td>0.0362 0.1526</td>
<td>0.0269 0.1190</td>
</tr>
<tr>
<td>tot_info</td>
<td>0.0024 0.0559</td>
<td>0.0174 0.0324</td>
</tr>
</tbody>
</table>

...


