Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence

Spectrum-Based Sequential Diagnosis

Alberto Gonzalez-Sanchez
Department of Software Technology
Delft University of Technology, The Netherlands

Rui Abreu
Department of Informatics Engineering
University of Porto, Portugal

Hans-Gerhard Gross and **Arjan J. C. van Gemund**
Department of Software Technology
Delft University of Technology, The Netherlands

Abstract

We present a spectrum-based, sequential software debugging 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, due to a novel, approximate diagnostic entropy estimation approach, which considers the subset of diagnoses that cover almost all Bayesian posterior probability mass. Synthetic experiments show that SEQUOIA achieves much better diagnostic uncertainty reduction compared to random test sequencing. Real programs, taken from the Software Infrastructure Repository, confirm SEQUOIA's better performance, with a test reduction up to 80% compared to random test sequences.

Introduction

Multiple observations (tests) are often required to achieve acceptable diagnostic certainty. In *sequential diagnosis* (SD) one computes the optimal sequence of tests that (on average) minimizes both *testing cost* C_t, and *diagnostic cost* C_d[1], i.e., the subsequent testing effort by the diagnostician to find the actual diagnosis within the list of candidates returned after expending C_t of testing. 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, Provan, and van Gemund 2010)). Due to complexity problems, in the software domain, the domain considered in this paper, most model-based approaches are limited to small programs and/or must take a single-fault assumption, and statistical debugging approaches have been typically used instead. To further complicate matters, in (Pattipati and Alexandridis 1988) it was proven that obtaining an optimal sequence of tests for SD, even for a single-fault, is an NP-Hard optimization problem and can only be approximated in practice.

[1]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.

Recently, a Bayesian reasoning approach has been described that considers the program's component involvement in tests on the one hand (expressed in terms of so-called *spectra*), and the test outcomes (pass/fail) on the other hand. In (Abreu and van Gemund 2010) it is shown that this spectrum-based reasoning approach offers high accuracy (i.e., low C_d) provided sufficient tests are available. While the above approaches do involve multiple observations (e.g., an entire regression test suite) they do not qualify as SD since the choice of tests is *random* rather than generating/selecting the next best test based on the diagnosis obtained thus far. Hence, the trade-off between C_d and C_t for random strategies 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 of polynomial complexity, which provides the required accuracy at acceptable cost. 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, Baysian approach can handle very large system sizes and test set sizes. Furthermore, unlike MBD approaches such as (Kuhn et al. 2008; Feldman, Provan, and van Gemund 2010) no system information is required, other than component test coverage, which is typically known from test execution profiling. 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. Furthermore, we evaluate SEQUOIA for real-world programs taken from the SIR benchmark suite of programs (Do, Elbaum, and Rothermel 2005). Our results show that especially for the first (most important) tests the decay of diagnostic cost C_d per test of SEQUOIA is significantly better than random.

Spectrum-based Fault Diagnosis

Definition A diagnostic system DS is defined as the triple $DS = \langle SDT, COMPS, OBS\rangle$, where SDT is a propositional theory describing the behavior of the system, $COMPS = \{c_1, \ldots, c_m, \ldots, c_M\}$ is a set of components in SDT, of which $M_f = 0, \ldots, M$ can be simultaneously faulty, and OBS is a set of observable variables in SDT.

With each component $c_m \in COMPS$ we associate a *health variable* h_m which denotes component health. The health states of a component are either healthy (h_m = *true*) or faulty (h_m = *false*).

Definition An h-literal is h_m or $\neg h_m$ for $c_m \in COMPS$.

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

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

Definition Let I_N and I_P be two disjoint sets of components indices, faulty and healthy, respectively, such that $COMPS = \{c_m \mid m \in I_N \cup I_P\}$ and $I_N \cap I_P = \emptyset$. We define $d(I_N, I_P)$ as the conjunction $(\bigwedge_{m \in I_N} \neg h_m) \wedge (\bigwedge_{m \in I_P} h_m)$

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 diagnosis candidate $d(I_N, I_P)$ for DS given an observation *obs* over variables in OBS is such that

$$SDT \wedge obs \wedge d(I_N, I_P) \not\models \perp$$

In the remainder we refer to $d(I_N, I_P)$ simply as d, which we identify with the set I_N of indices of the negative literals.

Definition A diagnostic report is an ordered set $D =< d_1, \ldots, d_k, \ldots, d_K >$ of K diagnosis candidates, sorted by posterior probability $\Pr(d_k)$, for which $SDT \wedge obs \wedge d_k \not\models \perp$.

Diagnostic performance is expressed in terms of a cost metric C_d that measures the excess *effort* incurred in finding all components at fault (residual diagnostic effort). C_d measures *wasted* effort, independent of the number of faults M_f in the program, to enable an unbiased evaluation of the effect of M_f on C_d. Thus, regardless of M_f, $C_d = 0$ represents an ideal diagnosis technique (all M_f faulty components on top of the ranking, no effort wasted on testing other components to find they are not faulty), while $C_d = M - M_f$ represents the worst case (testing all $M - M_f$ healthy components until arriving at the M_f faulty ones). This model is inspired by similar ones used in previous work (Abreu and van Gemund 2010). For example, consider an $M = 5$ component program with the following diagnostic report $D = \langle\{4, 5\}, \{4, 3\}, \{1, 2\}\rangle$, while c_1 and c_2 are actually faulty. The first diagnosis candidate leads the developer to inspect c_4 and c_5. As both components are healthy, C_d is increased by 2. The new information that $\Pr(\{4, 5\}) = 0$ leads to discarding $\{4, 3\}$ (c_4 can no longer be part of a multiple fault), The next components to be inspected are c_1 and c_2. As they are both faulty, no more effort is wasted, and consequently, $C_d = 2$.

Spectrum-based Candidate Generation

In the production of the consistent candidates for D, typical search algorithms are, e.g., GDE (de Kleer and Williams 1987), SAFARI (Feldman, Provan, and van Gemund 2010). In the case of spectrum-based candidate generation for SD we use a minimal hitting set algorithm (MHS) (e.g. (Abreu and van Gemund 2009; de Kleer and Williams 1987)) to compute a set of *minimal diagnoses* (candidates that are not subsumed by another of lower fault cardinality, i.e., number of negative h-literals $M_f = |d|$. A minimal diagnosis is a minimal hitting set over all conflicts).

In software, a test produces a sequence of component activity (e.g., statement execution) that results in a particular return value. The result of a process is either nominal ("pass") or an error ("fail").

Definition Let $S_f = \{c_m | c_m$ involved in a failing process$\}$, and let $S_p = \{c_m | c_m$ 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, Zoeteweij, and van Gemund 2007; Harrold et al. 1998), and originate from dynamically profiling the software components (e.g., statements or modules) during each program run, hence the name *spectrum-based* diagnosis.

Definition Let N denote the number of passing and failing processes (tests). Let N_f and 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_{nm} denotes whether component m was involved in process n ($a_{nm} = 1$) or not ($a_{nm} = 0$). Let e denote the *error vector*, where e_n signifies whether process n has passed ($e_n = 0$) or failed ($e_n = 1$).

In spectrum-based diagnosis, each component c_m is modeled by the abstract, *weak fault model* (WFM)

$$h_m \implies (ok_{m,inp} \implies ok_{m,out})$$

where $ok_{m,inp}$ and $ok_{m,out}$ denote the (binary) correctness of the component's input and output. If the output is not correct, we can be sure that c_m is not healthy. However, the fact that a the output is correct does not guarantee that c_m is healthy.

In our software setting, we know test inputs are always correct, but output correctness depends on the presence of bugs in the program and whether that specific input triggers the bug or not. Some inputs can be coincidentally correct, depending on the nature of the program and the test. For example, 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 ($15/10 = 15/15 = 1$), whereas in the second case the component fails ($20/10 = 2 \neq 20/15 = 1$). Hence, the division component (statement) exhibits intermittent failure behavior.

We compute the candidates by applying the 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 λ MHS solutions in D, where even small λ ($O(M)$) produces no difference in terms of C_d compared to a non-truncated diagnosis D (Abreu and van Gemund 2009).

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 $\Pr(m) = p_m$ denote the prior probability that a component c_m is at fault. Assuming components fail independently the prior probability of a candidate d_k is given by

$$\Pr(d_k) = \prod_{m \in I_N} \Pr(\{m\}) \cdot \prod_{m \in I_P} (1 - \Pr(\{m\})) \tag{1}$$

For each observation $obs_n = (A_{n*}, e_n)$ the posterior probabilities are updated according to Bayes' rule

$$\Pr(d_k|obs_n) = \frac{\Pr(obs_n|d_k)}{\Pr(obs_n)} \cdot \Pr(d_k|obs_{n-1}) \tag{2}$$

The initial value $\Pr(d_k|obs_0)$ (i.e., no observations are available) corresponds to the prior $\Pr(d_k)$. The denominator $\Pr(obs_n)$ is a normalizing term that is identical for all d_k and thus needs not be computed directly. $\Pr(obs_n|d_k)$ is defined for intermittent components by

$$\Pr(obs_n|d_k) = \begin{cases} \displaystyle\prod_{c_m \in d_k \wedge a_{nm}=1} g_m & \text{if } e_n = 0 \\ 1 - \displaystyle\prod_{c_m \in d_k \wedge a_{nm}=1} g_m & \text{if } e_n = 1 \end{cases} \tag{3}$$

where g_m 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). This assumes an or-model: the test will fail when either faulty component fails (an or-model) (Kuhn et al. 2008; Abreu, Zoeteweij, and van Gemund 2009). In this paper we will assume the g_m are available, either by definition, or from prior mutation analysis (Voas 1992).

Example We will illustrate Bayesian diagnosis with one example. Let us assume the function in Table 1, composed by 8 source code statements. We have a set of 6 available test inputs, that cover different statements of the program, as per matrix A. For simplicity, we will assume that all statements have identical prior probability $p_m = 0.1$ (this prior is much lower in real programs, but allows the numbers in the example to be easier to understand), and that there is no intermittency ($g_m = 0$).

Given the activity matrix A and observations e in Table 1, the MHS algorithm would produce the candidate set $D = \{\{1\}, \{2\}, \{4\}, \{8\}, \{3, 5\}, \{3, 6\}, \{5, 6\}, \{5, 7\}\}$. The initial probability of the single-fault candidates correspond to $\Pr(d_k) = p_m \cdot (1 - p_m)^7 = 0.05$, whereas for the double-fault candidates it is $\Pr(d_k) = p_m^2 \cdot (1 - p_m)^6 = 0.005$. But since by the MHS we assume the candidates in D are the only ones possible, their prior probabilities are normalized to $\Pr(d_k) = 0.23$ and $\Pr(d_k) = 0.02$ for single- and

	Program: Character Counter	A: t_1	t_2	t_3	t_4	t_5
	`function count(char * s) {`					
	`int let, dig, other, i;`					
c_1	`while(c = s[i++]) {`	1	1	1	1	1
c_2	`if ('A'<=c && 'Z'>=c)`	1	1	1	1	0
c_3	`let += 1;`	1	1	0	0	0
c_4	`elsif ('0'<=c && '9'>=c)`	1	1	1	1	0
c_5	`dig += 2; /* FAULT */`	0	1	0	1	0
c_6	`elsif (isprint(c))`	1	0	1	1	0
c_7	`other += 2; /* FAULT */`	1	0	0	0	0
c_8	`printf("%d %d %d\n",`	1	1	1	1	1
	`let, dig, others);}`					
	Test case outcomes (e)	1	1	0	1	0

Table 1: Example program and activity matrix A (transposed for readability)

double-fault candidates respectively. During the Bayesian update, the posterior probabilities of each candidate would evolve, until only $\{3, 5\}$ and $\{5, 7\}$ remain as candidates with non-null probability $\Pr(\{3, 5\}) = \Pr(\{5, 7\} = 0.5$. The remaining MHS candidates are discarded at some test. For example, $\{4\}$ gets discarded in t_3 since $\Pr(0|\{4\}) = 0$.

The final ranking is $D = \{\{5, 7\}, \{3, 5\}\}$. Since both candidates have equal probability, there is a 50% chance of inspecting c_3, therefore the residual diagnostic effort value is $C_d = 0.5 \cdot 1 + 0.5 \cdot 0 = 0.5$.

SEQUOIA

The diagnostic procedure outlined in the previous section 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. From SD perspective, this would correspond to a *random* ordering (of rows), i.e., a passive approach.

SEQUOIA is a spectrum-based SD algorithm that greedily selects the next best test (row) from A based on the diagnosis obtained thus far. As our principal optimization target C_d cannot be computed (since we do not know the actual health state d_*), we can only reason in terms of the *expectation* $\mathrm{E}[C_d]$. Experiments have shown that the entropy H of D is a very good predictor for $\mathrm{E}[C_d]$ as it expresses the uncertainty in D[2]. Consequently, we shall use the entropy drop $\Delta\mathrm{H}$ (aka information gain (Johnson 1960)) of a diagnosis D as test selection criterion. The next best test is the one that yields the highest $\Delta\mathrm{H}$ averaged over the two possible test outcomes (pass/fail), according to

$$\begin{aligned} \Delta\mathrm{H}(D_{n-1}, i) = \quad & H(D_{n-1}) - \\ & \Pr(e_i = 0) \cdot \mathrm{H}(D_n|e_i = 0) - \\ & \Pr(e_i = 1) \cdot \mathrm{H}(D_n|e_i = 1) \end{aligned}$$

where $D_n|e_i = 0$ represents the updated probabilities of the current diagnosis D if test i passes, $D_n|e_i = 1$ if it fails. Both $D_n|e_i = 0$ and $D_n|e_i = 1$ consider only the bayesian update, without re-calculating the MHS, since otherwise the

[2] D need not be a minimal diagnosis. H is defined for any D as long as posteriors are associated with the d_k.

algorithm would require N^2 MHS calculations instead of N. $\mathrm{H}(D_n)$ is defined as

$$\mathrm{H}(D_n) = -\sum_{d_k \in D_n} \Pr(d_k|obs_n) \cdot \log_2(\Pr(d_k|obs_n)) \quad (4)$$

Conceptually, when considering all possible test outcome combinations, a test suite prioritized for diagnosis is a binary tree with $O(2^N)$ nodes. However, if performed on line, only the $O(N)$ nodes corresponding to the current test path are expanded. In theory, given our Bayesian approach to multiple, intermittent fault diagnosis, computing D has $O(2^M)$ cost. 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.

Example Let us calculate ΔH for test t_3. In our example system, after t_1 and t_2 have been executed, the MHS algorithm and Bayesian update produce $D_2 = \{\{1\},\{2\},\{3\},\{4\},\{8\},\{5,6\},\{5,7\}\}$, with $\Pr(d_k) = 0.199$ for the single-fault candidates, and $\Pr(d_k) = 0.002$ for the double-fault candidates. $H(D_2) = 2.35$

Should t_3 fail, candidates $\{3\}$ and $\{5,7\}$ would be dropped from D since $\Pr(e = 1|\{3\}) = \Pr(e = 1|\{5,7\}) = 0$. The probability of a failure is $\Pr(e = 1) = 4 \cdot 0.199 + 0.002 = 0.798$. The updated probability of $\{1\}$, $\{2\}$, $\{4\}$, and $\{8\}$, is $\Pr(d_k) = 0.249$ and $\Pr(\{5,6\}) = 0.004$. The entropy $H(D_3|e = 1) = 2.06$

On the other hand, should t_3 pass, the only remaining candidates in D would be precisely $\{3\}$ and $\{5,7\}$. The probability of the test passing is $\Pr(e = 0) = 1 - \Pr(e = 1) = 0.202$. The updated candidate probabilities would be $\Pr(\{3\}) = 0.98$, $\Pr(\{5,7\}) = 0.02$, and the entropy $H(D_3|e = 0) = 0.14$

Known all these values, we can calculate the information gain of t_3 applied to D_2 as

$$\Delta\mathrm{H}(D_2, 3) = 2.35 - 0.798 \cdot 2.06 - 0.202 \cdot 0.14 = 0.677$$

If instead of choosing t_3 we chose t_4, the information gain would be 0, since we are sure that t_4 will fail (hence no update of D will happen),

$$\Delta\mathrm{H}(D_2, 5) = 2.35 - 1 \cdot 2.35 - 0.0 \cdot 0.00 = 0.00$$

Approximate Information Gain

From our choice of STACCATO, it would seem that this MHS algorithm 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 in our example system in Table 1. After the first test observation, the activity matrix and error vector are composed by only one observation

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	e
1	1	1	1	0	1	1	1	1

An MHS algorithm will return the diagnosis $D = \{\{1\},\{2\},\{3\},\{4\},\{6\},\{7\},\{8\}\}$, where all candidates have probability 0.142. This diagnosis has a number of problems.

1. c_5 is missing in all the candidates, and the actual candidate, {5,7}, is not even being considered. This severely affects diagnostic effort.
2. The entropy of the minimal D is $H = 2.8$, when in reality, if we considered a full D with all 2^8 candidates, the entropy would be $H = 4.85$.
3. Since information gain is computed without recalculating the MHS, if the test fails, many of the low-cardinality candidates disappear from the MHS. They should be replaced by higher cardinality candidates, but the IG calculation does not consider these candidates and will produce large errors.

In a full D setting, $H(D) = 4.85$ as we have seen. If t_2 passes, $H(D|e = 0) = 1.24$, with $\Pr(e = 0) = 0.19$. If t_2 fails, $H(D|e = 1) = 4.83$, with $\Pr(e = 1) = 0.81$. The information gain value is then $IG_{full} = 0.70$.

In an MHS setting, $H(D) = 2.8$. If t_2 passes, $H(D|e = 0) = 1.00$, with $\Pr(e = 0) = 0.28$. If t_2 fails, $H(D|e = 1) = 2.32$, with $\Pr(e = 1) = 0.72$. The information gain value is then $IG_{MHS} = 0.84$. It can be seen how this test is assigned an IG value with 20% error.

It is clear that both the quality of the diagnosis and H computation are compromised due to the fact that essential, *non-minimal* candidates in D are missing. While MHS is appropriate in a passive diagnosis setting where many observations are already available, in an active diagnosis setting, especially at the beginning, non-minimal diagnosis candidates, which still contain a large probability mass, are crucial.

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 initial MHS would be extended to contain also additional double-faults, and the value of entropy would be corrected to $H(D) = 4.31$. If t_2 passes, $H(D|e = 0) = 1.24$, with $\Pr(e = 0) = 0.21$. If t_2 fails, $H(D|e = 1) = 4.19$, with $\Pr(e = 1) = 0.79$. The information gain value is then $IG_{HS1} = 0.74$, which, while still not totally precise, has only 5% error.

Optimality Figure 1 shows the evolution of H versus i for a randomly generated A with $N = 150$, $M = 10$, $M_f = 3$, $g_m = U(0,1)$. Three plots are shown, i.e., optimal H for the complete D with all non-minimal candidates ("OPT"), the first-order approximation ("HS1"), and the original MHS-based D ("MHS"), respectively. The results clearly show that a first-order approximation already delivers good accuracy, nearly as good as the optimal. 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).

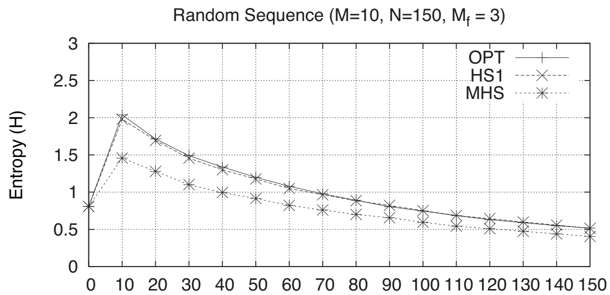

Figure 1: Evolution of H

Algorithm

Algorithm 1 describes the SEQUOIA algorithm. The algorithm takes as arguments the set of components c_m and the associated priors p_j and failure intermittencies g_m, the test matrix A, and test costs, C_n, and two entropy estimation parameters: the maximum number of minimal diagnosis candidates λ as generated by STACCATO, and the maximum number of non-minimal candidates γ subsequently generated by the 1st-order hitting set extension.

Algorithm 1 SEQUOIA

procedure SEQUOIA(COMPS, A, C, λ, γ)
  $T \leftarrow \{1, \ldots, N\}$
  $A' \leftarrow \emptyset$
  $e \leftarrow \emptyset$
  D, PrD $\leftarrow$ INITIALD(COMPS, γ)
  while $T \neq \emptyset$ **do**
    $i \leftarrow \arg\max_{i \in \text{SAMPLE}(T)} \dfrac{\Delta\text{H}(D, i)}{C_i}$
    $e_i \leftarrow$ RUNTEST(i)
    $A' \leftarrow A' || A_{i*}$
    $e \leftarrow e || e_i$
    $D \leftarrow$ STACCATO(A', e, λ)
    $D \leftarrow$ EXPANDSUBSUMED(D, COMPS, γ)
    for all $d_k \in D$ **do**
      PrD[d_k] $\leftarrow$ BAYESUPDATE(d_k, A', e)
    $T \leftarrow T \setminus \{i\}$

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 γ 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 σ tests (SAMPLE) that yields the highest information gain per unit of cost (C_i represents the cost of each test). Merely considering a subset (e.g., $\sigma = 100$) instead of all available tests ($O(N)$) yields a substantial gain, while the sacrifice in optimality is small[3]. 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 $||$ 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 EXPANDSUBSUMED. In order to keep complexity under control, only the first λ minimal candidates are generated. Likewise, only the first γ 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.

[3]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).

Time 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, ΔH has to be calculated for the σ test cases in the sample, with a cost of $O(\gamma M)$. The STACCATO algorithm has a complexity of $O(M_f \cdot M \cdot (N + \log M))$ (Abreu and van Gemund 2009) although in practical situations, its complexity can be considered $O(M \cdot N)$ since M_f is small and $N \gg \log M$. The Bayesian update of a candidate accounts $O(\gamma \cdot M \cdot N)$.

The combined complexity of each of the steps in the inner loop is, in light of these results, $O(M \cdot N)$, albeit with a significant factor. Our timing experiments show that 1000×1000 matrices require 11s per test choice on average.

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 to study the effect of the various parameters in a controlled setting.

Component activity a_{nm} is sampled from a Bernoulli distribution with parameter ρ, i.e., the probability a component is involved in a row of A equals ρ. All test costs are equal. For the M components we also set $g_m = g$. Thus the probability of a component being involved *and* generating a failure equals $\rho \cdot (1 - g)$. A row n in A generates an error ($e_n = 1$) if at least 1 of the M_f components generates a failure (or-model). Measurements for a specific scenario are averaged over $1,000$ sample matrices, yielding a coefficient of variance of approximately 0.02.

Figure 2 shows the evolution of relative diagnostic cost $C_d/(M - M_f)$ of SEQUOIA (SEQ) and random sequencing (RND) versus the number of tests, for $M = 100$ components with $M_f = 3, 5$, respectively. The matrix comprises $N = 500$ tests, although we prioritize only the first 100. SEQUOIA's parameters are set to $\lambda = M$ and $\gamma = 10 \cdot M$.

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 reduction of tests required for acceptable diagnostic accuracy is dramatically less.

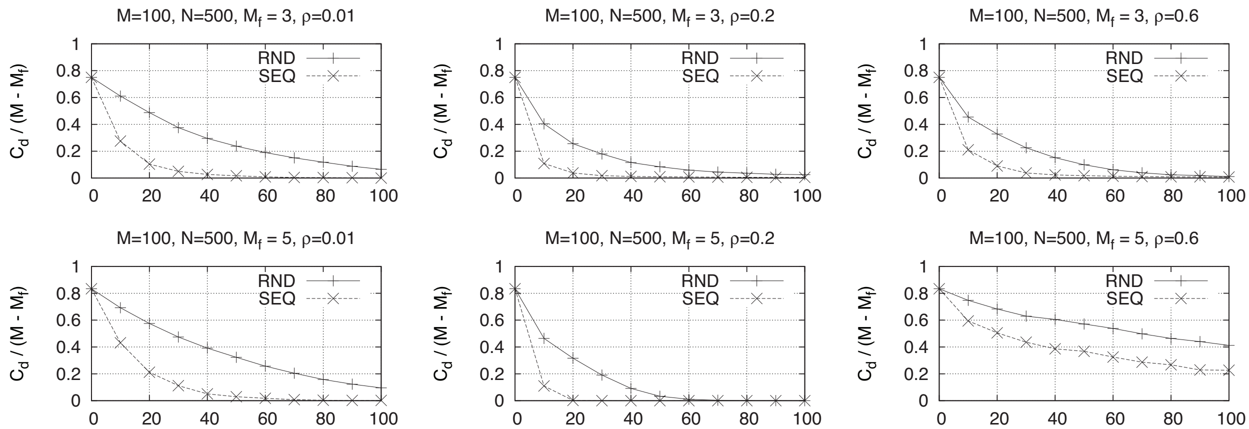

Figure 2: C_d evolution for synthetic A

Empirical Evaluation

In this section, we evaluate SEQUOIA for the Siemens benchmark set (Hutchins et al. 1994) and three Unix programs from the SIR (Do, Elbaum, and Rothermel 2005) repository. Table 2 lists some information about the programs, where M corresponds to the number of lines of code (components in this context).

Experimental Setup

For our experiments, we inject 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 either 1 or 3 faults to avoid raising the fault density to unrealistic levels in software (above 10 faults per KLOC). We compare random ordering (RND), SEQUOIA with minimal diagnoses (MHS), and SEQUOIA with first-order subsumption (SEQ).

The coverage matrices A are obtained by recording test coverage, and the output of a reference implementation used as test oracle to determine the test outcome. The test costs C_n correspond to the execution time of tests. The prior fault probability of statements in our experiments is set to $p_m = 0.01$, corresponding to the aforementioned 10 faults per KLOC fault density. The intermittency g_m values (per program the average g value over the various seeded faults is shown in Table 2) were calculated using a mutation study similar to (Rothermel et al. 2001). To discard the effect of errors in the mutation study we performed a second experiment with simulated error vectors (described later on). As with our theoretical evaluation, SEQUOIA's parameters are set to $\lambda = M$ and $\gamma = 10 \cdot M$. Each point is averaged over 100 different runs.

Performance Results

Figure 3 shows the evolution of the relative diagnostic cost for `schedule2` for $M_f = 1$ and $M_f = 3$. It is clear that the decay of C_d for SEQUOIA is much better than for random test selection. MHS is slightly better than SEQ for $M_f = 1$ since single faults are better diagnosed without D expansion. However, the moment M_f increases, it can be seen that MHS performs even worse than RND. Due to space limitations, 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 diagnostic algorithm), according to

$$S = \sum_{n=1}^{N} \left(\frac{C_d(n-1) - C_d(N)}{M - M_f} \cdot C_i \right)$$

Table 2 shows the results for $M_f = 1$, and $M_f = 3$, for (1) simulated test outcomes, and (2) actual outcomes (according to the test suite's test oracle). In the simulated case, the test outcome is computed (drawn) according to the g_m estimates used in SEQUOIA in conjunction with the or-model (Eq. 3), in order to elliminate the effects of imperfect intermittency estimation.

For the simulations, SEQUOIA clearly improves over random ordering. However, this is not always true when the real test case outcomes are used. For $M_f = 3$ we see how the improvements achieved by SEQUOIA is consistently lower than in the simulations, and how in the real cases of `tcas` and `tot_info` SEQUOIA performs worse. In these cases the estimation of g_m is poor and causes SEQUOIA to make wrong decisions. Since RND is able to attain a good final C_d, these two cases highlight the fact that sequencing performance (the IG heuristic) is affected much more by intermittency estimation errors than the Bayesian update itself.

For `tcas` the performance is even worse because D contains too large a quantity of equal minimal diagnoses (so-called ambiguity groups) that have relatively high posterior probability. Even the 1st-order expansion of D cannot avoid a substantial loss of information that causes frequent under-estimation of the failure probabilities. As a result, SEQUOIA will erroneously compensate by selecting tests that include too many components, causing loss of diagnostic power. Note, however, that this situation can be easily detected, and

				Simulated $M_f = 1$					Real $M_f = 1$					Simulated $M_f = 3$					Real $M_f = 3$				
Program	M	N	g	RND	MHS		SEQ		RND	MHS		SEQ		RND	MHS		SEQ		RND	MHS		SEQ	
`print_tokens`	539	4130	0.20	2.80	**1.08**	-61%	**1.35**	-52%	4.63	**1.82**	-61%	**2.44**	-47%	8.86	17.08	+93%	**2.33**	-74%	**5.16**	17.49	+239%	**4.94**	-4%
`print_tokens2`	489	4115	0.32	2.21	1.58	-28%	**1.45**	-34%	4.56	**2.51**	-45%	2.84	-38%	9.57	21.08	+120%	**5.37**	-44%	6.37	12.90	+102%	**3.55**	-44%
`replace`	507	5542	0.35	2.07	**1.09**	-48%	**1.24**	-40%	4.43	**1.64**	-63%	4.61	+4%	14.51	32.51	+124%	**6.30**	-57%	10.32	30.18	+193%	**4.49**	-56%
`schedule`	397	2650	0.28	2.79	**1.07**	-62%	**1.35**	-52%	7.89	**1.35**	-83%	8.01	+1%	22.40	27.45	+23%	**6.84**	-69%	11.55	25.72	+123%	**4.29**	-63%
`schedule2`	299	2710	0.32	4.01	**1.43**	-64%	1.54	-62%	9.03	**1.70**	-81%	7.64	-15%	24.56	28.79	+17%	**6.49**	-74%	11.78	22.03	+87%	**2.70**	-77%
`tcas`	174	1608	0.77	4.33	**3.27**	-25%	**3.45**	-20%	**2.51**	6.66	+165%	9.53	+279%	6.63	24.35	+267%	**4.37**	-34%	**4.44**	14.80	+233%	12.53	+182%
`tot_info`	398	1052	0.27	2.76	**1.36**	-51%	1.61	-42%	**2.40**	**2.20**	-8%	6.71	+179%	11.81	27.04	+129%	**4.67**	-60%	**2.46**	7.63	+210%	3.89	+58%
`space`	9126	500	0.17	4.89	13.29	+171%	**1.91**	-61%	7.13	13.70	+92%	**3.06**	-57%	7.05	28.28	+301%	**2.85**	-59%	7.40	40.51	+447%	**5.26**	-29%
`gzip`	6708	211	0.11	6.29	24.70	+293%	**1.16**	-82%	7.26	21.45	+196%	**1.42**	-80%	12.27	34.74	+183%	**2.52**	-79%	16.00	23.32	+46%	**3.67**	-77%
`sed`	9014	184	0.20	3.98	5.30	+33%	**1.60**	-60%	4.78	5.32	+11%	**0.99**	-79%	7.30	30.50	+318%	**2.40**	-67%	10.77	35.40	+229%	**3.38**	-69%

Table 2: Aggregate diagnostic cost (S) for RND, MHS, and SEQ (statistical winners in bold)

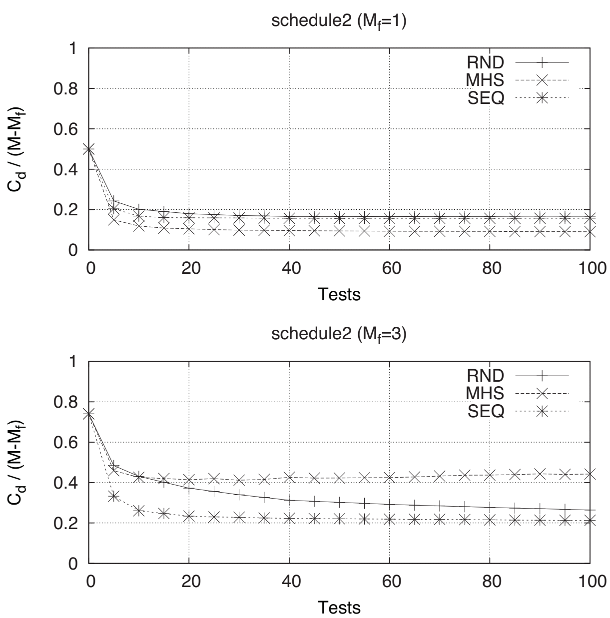

Figure 3: $C_d/(M - M_f)$ evolution for `schedule2`

switching at this point from IG to RND mode completely solves this performance problem. In fact, a hybrid solution where IG is used for the first tests (where C_d decay is large), eventually switching to random mode, offers the best of both worlds.

Related Work

An early example related to SD is found in (de Kleer and Williams 1987) which presents a sequential probing strategy to reduce diagnostic entropy. Apart from the fact that our entropy computation is less complex, we consider tests rather than probes, typical of SD, which assumes that applying test vectors to a closed system are the only way of accessing system information (a test *can* be seen as a probe, but computing a test that will manifest a certain internal variable is not the same problem).

Classical examples of SD are found in (Shakeri et al. 2000; Raghavan, Shakeri, and Pattipati 1999; Tu and Pattipati 2003; Kundakcioglu and Ünlüyurt 2007). Similar to spectrum-based approaches, a fault matrix is used that encodes per test which components may be at fault when the test fails (essentially A, extended with a cost per test). While the work includes non-unit test cost, it focuses on single-fault diagnosis, which is highly unrealistic considering the large systems we are targeting. Some extensions to multiple-faults are known, but no solutions to either intermittent behavior and the associated, exponential explosion have been published so far.

Recently, two model-based approaches to SD have been published, referred to as Active Diagnosis (Kuhn et al. 2008), and Active Testing (Feldman, Provan, and van Gemund 2009). Both approaches differ from ours that system information needs to be coded in terms of a model instead of just a matrix that represents a projection in terms of test coverage information. As a result, the tests that can be computed are not restricted to the N tests possible in matrix approaches. Another advantage is that the observations generated by the tests offer more information than the binary pass/fail outcomes, offering potentially higher decay rates. Being full-fledged, model-based approaches, however, for the very large (software) systems we consider, the computational costs would be prohibitive.

Conclusion

In this paper we have presented 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. To achieve polynomial complexity SEQUOIA uses an approximate information gain computation approach. Synthetic data shows, that the dynamic selection of the next best test based on the observations made so far, allows SEQUOIA to achieve much better decay of diagnostic uncertainty compared to random test sequencing. Real programs also show that SEQUOIA has better performance, except when the input intermittency parameters are erroneous or the diagnosis involves ambiguity sets that are too large for the entropy estimation to handle. Future work therefore includes solving this problem by expanding the MHS in order of posterior probability (which is currently ignored in our approach).

Supported by the Poseidon project of the Embedded Systems Institute (ESI), The Netherlands and the Dutch Ministry of Economic Affairs (BSIK03021).

References

Abreu, R., and van Gemund, A. J. C. 2009. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *Proc. SARA'09*.

Abreu, R., and van Gemund, A. J. C. 2010. Diagnosing multiple intermittent failures using maximum likelihood estimation. *Artif. Intell.*

Abreu, R.; Zoeteweij, P.; and van Gemund, A. 2007. On the accuracy of spectrum-based fault localization. In *Proc. TAIC PART'07*.

Abreu, R.; Zoeteweij, P.; and van Gemund, A. 2009. A new Bayesian approach to multiple intermittent fault diagnosis. In *Proc. IJCAI'09*.

David, H. A. 1970. *Order Statistics*. John Wiley & Sons.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artif. Intell.*

Do, H.; Elbaum, S. G.; and Rothermel, G. 2005. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Emp. Soft. Eng. J.*

Feldman, A.; Provan, G. M.; and van Gemund, A. J. C. 2009. Fractal: Efficient fault isolation using active testing. In *Proc. IJCAI'09*.

Feldman, A.; Provan, G. M.; and van Gemund, A. J. C. 2010. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research*.

Harrold, M.; Rothermel, G.; Wu, R.; and Yi, L. 1998. An empirical investigation of program spectra. *ACM SIGPLAN Notices*.

Hutchins, M.; Foster, H.; Goradia, T.; and Ostrand, T. 1994. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proc. ICSE '94*.

Johnson, R. 1960. An information theory approach to diagnosis. In *Symposium on Reliability and Quality Control*.

Kuhn, L.; Price, B.; de Kleer, J.; Do, M.; and Zhou, R. 2008. Pervasive diagnosis: Integration of active diagnosis into production plans. In *Proc. AAAI'08*.

Kundakcioglu, O. E., and Ünlüyurt, T. 2007. Bottom-up construction of minimum-cost and/or trees for sequential fault diagnosis. *IEEE TSMC*.

Pattipati, K., and Alexandridis, M. 1988. Application of heuristic search and information theory to sequential fault diagnosis. In *Proceedings IEEE International Symposium on Intelligent Control*.

Raghavan, V.; Shakeri, M.; and Pattipati, K. R. 1999. Optimal and near-optimal test sequencing algorithms with realistic test models. *IEEE TSMC*.

Rothermel, G.; Untch, R. H.; Chu, C.; and Harrold, M. J. 2001. Prioritizing test cases for regression testing. *IEEE TSE*.

Shakeri, M.; Raghavan, V.; Pattipati, K. R.; and Patterson-Hine, A. 2000. Sequential testing algorithms for multiple fault diagnosis. *IEEE TSMC*.

Tu, F., and Pattipati, K. R. 2003. Rollout strategies for sequential fault diagnosis. *IEEE TSMC*.

Voas, J. M. 1992. Pie: A dynamic failure-based technique. *IEEE TSE*.