A New Bayesian Approach to Multiple Intermittent Fault Diagnosis*

Rui Abreu and **Peter Zoeteweij** and **Arjan J. C. van Gemund**
Delft University of Technology
Mekelweg 4, 2628 CD, Delft, The Netherlands
{r.f.abreu,p.zoeteweij,a.j.c.vangemund}@tudelft.nl

Abstract

Logic reasoning approaches to fault diagnosis account for the fact that a component c_j may fail intermittently by introducing a parameter g_j that expresses the probability the component exhibits correct behavior. This component parameter g_j, in conjunction with a priori fault probability, is used in a Bayesian framework to compute the posterior fault candidate probabilities. Usually, information on g_j is not known a priori. While proper estimation of g_j can have a great impact on the diagnostic accuracy, at present, only approximations have been proposed. We present a novel framework, BARINEL, that computes exact estimations of g_j as integral part of the posterior candidate probability computation. BARINEL's diagnostic performance is evaluated for both synthetic and real software systems. Our results show that our approach is superior to approaches based on classical persistent fault models as well as previously proposed intermittent fault models.

1 Introduction

In model-based fault diagnosis (MBD) approaches faults are typically assumed to be persistent. However, in many practical situations faults manifest themselves intermittently, such as in copiers where sometimes sheets may be blank, or where a worn roller sometimes slips and causes a paper jam [De Kleer *et al.*, 2008]. Intermittent behavior is also relevant in software fault diagnosis, which is the primary context of this paper. Although software is supposed to be inherently deterministic, intermittent component models are often essential. This can be due to non-determinism (e.g., race conditions) caused by design faults related to properly dealing with concurrency. A more compelling reason is the modeling abstraction typically applied, where, for example, the software component's input and output values are abstracted in the model, such that a component's (abstracted) output may differ for the same (abstracted) input. Although a weak fault model (that doesn't stipulate particular faulty behavior) admits any output behavior, applying classical (persistent fault) diagnosis to software components that do not consistently exhibit failures results in severely degraded diagnostic performance (as is also shown in this paper).

A model for intermittent behavior [De Kleer, 2007] was introduced as an extension of the GDE framework [De Kleer and Williams, 1987; De Kleer *et al.*, 1992]. Essentially, next to the prior probability p_j that a component c_j is at fault, a parameter g_j is used to express the probability that a faulty component exhibits correct (good, hence g) behavior. The model is incorporated into the standard, Bayesian framework that computes the posterior probability of diagnosis candidates based on observations [De Kleer and Williams, 1987; De Kleer, 2006].

The intermittency framework has been shown to yield significantly better results (e.g., in the diagnosis and replanning of paper sheet paths in copiers with intermittent component failures [Kuhn *et al.*, 2008], and in software fault diagnosis [Abreu *et al.*, 2008a]), compared to an approach based on a classical, persistent fault model. An important problem in using the intermittency model, however, is the estimation of g_j, as calibration data on correct and incorrect component behavior is typically not available. Estimating g_j for each component c_j would be straightforward when (sufficient) system observations are available where only that *single*, intermittent component is involved [De Kleer, 2007]. However, in a multiple-fault context usually only system observations are available in which *multiple* components are involved. Consequently, isolating to what extent each individual component contributes to the observed failure behavior is less straightforward. However, as the influence of g_j in the computation of the posterior probability of each diagnostic candidate is significant, exact knowledge of each g_j *can* be critical to overall diagnostic accuracy.

In [De Kleer *et al.*, 2008] as well as in [Abreu *et al.*, 2008a; 2008b] strategies have been proposed to estimate the g_j in a multiple-fault context. However, the approaches are essentially based on an approximation. In this paper, we present a novel approach to the estimation of the g_j in conjunction with a new Bayesian approach towards the computation of the posterior candidate probabilities using an intermittent fault model that generalizes over classical, persistent MBD approaches. The approach represents a departure from the current Bayesian framework as used in current diagnosis approaches (e.g., [De Kleer *et al.*, 2008] and [Abreu *et al.*,

*This work has been carried out as part of the TRADER project under the responsibility of the Embedded Systems Institute.

2008a]) in the sense that (1) the resulting g_j are exact, maximum likelihood estimators instead of approximations, and (2) the computation of the posterior candidate probabilities is an *integral byproduct* of the g_j estimation procedure. The contributions of this paper are threefold: (i) we present our new approach for the candidate probability computation which features the algorithm to compute the g_j. The approach is coined BARINEL, which is the name of the software implementation of our method; (ii) we compare the accuracy and complexity of our method to the current approaches in [De Kleer *et al.*, 2008] and [Abreu *et al.*, 2008a] for observations series that are synthetically generated for known g_j setpoints; (iii) we describe the application of our approach to spectrum-based software multiple-fault diagnosis and evaluate the diagnostic performance using the well-known Siemens suite of benchmark programs and `space`.

To the best of our knowledge, this approach has not been described before. The results from the synthetic experiments, as well as from the application to real software systems, confirm that our new approach has superior diagnostic performance to all Bayesian approaches to intermittent systems known to date.

2 Preliminaries

In this section we introduce existing concepts and definitions.

2.1 Basic Definitions

Definition 1. A diagnostic system DS is defined as the triple $DS = \langle SD, COMPS, OBS \rangle$, 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, 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 healthy (*true*) and faulty (*false*). In Section 3 this definition will be extended.

Definition 2. An h-literal is h_j or $\neg h_j$ for $c_j \in COMPS$.

Definition 3. An h-clause is a disjunction of h-literals containing no complementary pair of h-literals.

Definition 4. A conflict of $(SD, COMPS, OBS)$ is an h-clause of negative h-literals entailed by $SD \cup OBS$.

Definition 5. Let S_N and S_P be two disjoint sets of components indices, faulty and healthy, respectively, such that $COMPS = \{c_j \mid j \in S_N \cup S_P\}$ and $S_N \cap S_P = \emptyset$. We define $d(S_N, S_P)$ to be the conjunction $(\bigwedge_{j \in S_N} \neg h_j) \wedge (\bigwedge_{j \in S_P} h_j)$

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 6. A diagnosis candidate for DS given an observation obs over variables in OBS, is $d(S_N, S_P)$ such that

$$SD \wedge obs \wedge d(S_N, S_P) \not\models \bot$$

In the remainder we refer to $d(S_N, S_P)$ simply as d, which we identify with the set S_N of indices of the negative literals. A minimal diagnosis is a diagnosis that is not subsumed by another of lower fault cardinality (number of negative h-literals, $|d|$).

Definition 7. A diagnostic report $D = \{d_1, \ldots, d_k, \ldots, d_K\}$ is an ordered set of all K diagnosis candidates, for which $SD \wedge obs \wedge d_k \not\models \bot$.

2.2 Computing Diagnoses

The Bayesian approach serves as the foundation for the derivation of diagnostic candidates, i.e., (1) deducing whether a candidate diagnosis d_k is consistent with the observations, and (2) the posterior probability $\Pr(d_k)$ of that candidate being the actual diagnosis. With respect to (1), rather than computing $\Pr(d_k)$ for *all* possible candidates, just to find that most of them have $\Pr(d_k) = 0$, search algorithms are typically used instead, such as CDA* [Williams and Ragno, 2007], SAFARI [Feldman *et al.*, 2008], or just a minimal hitting set (MHS) algorithm when conflict sets are available, e.g. [De Kleer and Williams, 1987], but the Bayesian probability framework remains the basis. In this section we will briefly describe the contemporary approach to the derivation of candidates and their posterior probability. In the following, we assume weak fault models.

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 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 8. Let $S_f = \{c_j | c_j$ involved in a failing process$\}$, and let $S_p = \{c_j | c_j$ involved in a passing process$\}$, denote the *fail set* and *pass set*, respectively.

Approaches for fault diagnosis that assume persistent, weak fault models often generate candidates based on fail sets (aka conflict sets), essentially using an MHS algorithm to derive minimal candidates. Recent approaches that allow intermittency also take into account pass sets. A fail set indicts components, whereas a pass set exonerates components. The extent of indictment or exoneration is computed using Bayes' rule. In the following we assume that a number of pass and fail sets have been collected, either by static modeling (e.g., logic circuits, where each primary output yields a pass or fail set) or by dynamic profiling (e.g., software, where each run yields a pass or fail set, both known as a *spectrum* [Abreu *et al.*, 2007]).

Definition 9. Let N denote the number of passing and failing processes. Let N_f and N_p, $N_f + N_p = N$, denote the number of fail and pass sets, 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 input to the Bayesian probability update process.

Ranking Diagnoses

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

$\Pr(d_k) = \prod_{j \in S_N} \Pr(\{j\}) \cdot \prod_{j \in S_P} (1 - \Pr(\{j\}))$. For each observation $obs_i = (A_{i*}, e_i)$ the posterior probabilities are updated according to Bayes rule (naive Bayes classifying)

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

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

$$\Pr(obs_i|d_k) = \begin{cases} 0 & \text{if } obs_i \wedge d_k \models \bot \\ 1 & \text{if } d_k \rightarrow obs_i \\ \varepsilon & \text{if } d_k \rightarrow \{obs_1, \ldots, obs_i, \ldots, obs_N\} \end{cases}$$

As mentioned earlier, rather than updating each candidate only candidates derived from an MHS algorithm are updated implying that the 0-clause need not be considered.

Many policies exist for ε [De Kleer, 2006]. Three policies can be distinguished. The first policy, denoted $\varepsilon^{(0)}$ equals the classical MBD policy for persistent, weak faults, and is defined as follows

$$\varepsilon^{(0)} = \begin{cases} \frac{E_P}{E_P + E_F} & \text{if } e_i = 0 \\ \frac{E_F}{E_P + E_F} & \text{if } e_i = 1 \end{cases} \quad (1)$$

where $E_P = 2^M$ and $E_F = (2^{|d_k|} - 1) \cdot 2^{M - |d_k|}$ are the number of passed and failed observations that can be explained by diagnosis d_k, respectively. A disadvantage of this classical policy is that pass sets, apart from making single faults more probable than multiple faults, do not help much in pinpointing the faults, in particular for weak fault models which do not rule out any candidates (the 2^M term in Eq. 1). In addition, there is no way to distinguish between diagnoses with the same cardinality, because the terms are merely a function of the cardinality of the diagnosis candidate.

The next two, intermittent policies account for the fact that components of pass sets should to some extent be exonerated. In the following we distinguish between two policies, $\varepsilon^{(1)}$ [De Kleer, 2007] and $\varepsilon^{(2)}$ [Abreu *et al.*, 2008a] which are defined as

$$\varepsilon^{(1)} = \begin{cases} g(d_k) & \text{if } e_i = 0 \\ 1 - g(d_k) & \text{if } e_i = 1 \end{cases} \quad \varepsilon^{(2)} = \begin{cases} g(d_k)^m & \text{if } e_i = 0 \\ 1 - g(d_k)^m & \text{if } e_i = 1 \end{cases}$$

where $m = \sum_{j \in d_k} [a_{ij} = 1]$ is the number of faulty components according to d_k involved in process i. Note that a term $g(d_k)$ is used rather than the real individual component intermittency parameters g_j. As mentioned earlier, this is due to the fact that obtaining g_j from pass and fail sets where multiple intermittent failures are involved was far from trivial. Instead, an "effective" intermittency parameter $g(d_k)$ is estimated for the candidate d_k by counting how many times components of d_k are involved in pass and fail sets. In both strategies $g(d_k)$ is approximated by

$$g(d_k) = \frac{\sum_{i=1..N} [(\bigvee_{j \in d_k} a_{ij} = 1) \wedge e_i = 0]}{\sum_{i=1..N} [\bigvee_{j \in d_k} a_{ij} = 1]}$$

where $[\cdot]$ is Iverson's operator ($[\text{true}] = 1$), $[\text{false}] = 0$]).

Policy $\varepsilon^{(2)}$ is a variant of $\varepsilon^{(1)}$, which approximates the probability $\prod_{j \in d_k} g_j$ that all m components in d_k exhibit good behavior by $g(d_k)^m$ assuming that all components of d_k have equal g values. This takes into account the fact that the failure probability changes when *multiple* intermittent faults are involved.

3 BARINEL Approach

In this section we present our approach to compute the g_j and the associated, posterior candidate probabilities $\Pr(d_k)$ given a set of observations (A, e). In our approach we (1) determine the real g_j instead of $g(d_k)$, and (2) apply the g_j in an improved epsilon policy to compute $\Pr(obs|d_k)$. The key idea underlying our approach is that for each candidate d_k we compute the g_j for the candidate's faulty components that *maximizes the probability Pr*$(e|d_k)$ *of the observations* e *occurring, conditioned on that candidate* d_k (maximum likelihood estimation for naive Bayes classifier d_k). For a given process i, in terms of g_j the epsilon policy is given by

$$\varepsilon = \begin{cases} \prod_{j \in d_k \wedge a_{ij} = 1} g_j & \text{if } e_i = 0 \\ 1 - \prod_{j \in d_k \wedge a_{ij} = 1} g_j & \text{if } e_i = 1 \end{cases}$$

Thus, g_j is solved by maximizing $Pr(e|d_k)$ under the above epsilon policy, according to

$$G = \arg\max_G \Pr(e|d_k)$$

where $G = \{g_j | j \in d_k\}$. This approach implies that for a particular candidate d_k the optimum g_j values may differ with those for another candidate d'_k for the same components.

Generalizing over persistent and intermittent faults, with each candidate d_k each component c_j is associated with a computed g_j value (which from now on we will denote h_j for *health*) which ranges from 0 (persistently failing) to 1 (healthy, i.e., faulty without any failure). Consequently, each candidate diagnosis need only specify the set of component health states h_j, which represents a real-valued generalization over the classical binary "normal/abnormal" entries. For example, for an $M = 4$ component system our framework might yield the double and triple-fault candidates $\{0.33, 1, 1, 0\}$, and $\{0.5, 0.66, 1, 0\}$, respectively, each of which has g_j that optimally explain the observations e, but differ for the same j (e.g., 0.33 vs. 0.5).

Our approach, of which the implementation is coined BARINEL, is described in Algorithm 1 and comprises three main phases. In the first phase (line 2) a list of candidates D is computed from (A, e) using a low-cost, heuristic MHS algorithm called STACCATO that returns an MHS of limited size (typically, 100 multiple-fault candidates), yet capturing all significant probability mass.

In the second phase $\Pr(d_k|(A, e))$ is computed for each $d_k \in D$ (lines 3 to 14). First, GENERATEPR derives for every candidate d_k the probability $\Pr(e|d_k)$ for the current set of observations e. As an example, suppose the following measurements (ignoring healthy components):

c_1	c_2	e	$\Pr(e_i\|\{1,2\})$
1	0	1	$1 - g_1$
1	1	1	$1 - g_1 \cdot g_2$
0	1	0	g_2
1	0	0	g_1

As the four observations are independent, the probability of obtaining e given $d_k = \{1, 2\}$ equals

$$\Pr(e|d_k) = g_1 \cdot g_2 \cdot (1 - g_1) \cdot (1 - g_1 \cdot g_2)$$

Algorithm 1 Diagnostic Algorithm: BARINEL

Inputs: Activity matrix A, error vector e,

Output: Diagnostic Report D

```
1  γ ← ϵ
2  D ← STACCATO((A, e)) ▷ Compute MHS
3  for all d_k ∈ D do
4      expr ← GENERATEPR((A, e), d_k)
5      i ← 0
6      Pr[d_k]^i ← 0
7      repeat
8          i ← i + 1
9          for all j ∈ d_k do
10             g_j ← g_j + γ · ∇expr(g_j)
11         end for
12         Pr[d_k]^i ← EVALUATE(expr, ∀_{j∈d_k} g_j)
13     until |Pr[d_k]^{i−1} − Pr[d_k]^i| ≤ ξ
14 end for
15 return SORT(D, Pr)
```

Subsequently, all g_j are computed such that they maximize $\Pr(e|d_k)$. To solve the maximization problem we apply a simple gradient ascent procedure [Avriel, 2003] (bounded within the domain $0 < g_j < 1$).

In the third and final phase, the diagnoses are ranked according to $\Pr(d_k|(A,e))$, which is computed by EVALUATE according to the usual, posterior update

$$\Pr(d_k|(A,e)) = \frac{\Pr(e|d_k)}{\Pr(obs)} \cdot \Pr(d_k)$$

where $\Pr(d_k)$ is the prior probability that d_k is correct, $\Pr(obs)$ is a normalization factor, and $\Pr(e|d_k)$ is the probability that e is observed assuming d_k correct.

In the following we illustrate that for single-fault candidates, the maximum likelihood estimator for g_1 equals the health state $h_1 = \sum_i e_i/N$, which is the intuitively correct way to estimate g_1 (and has also been the basis for the previous approximation of $g(d_k)$ shown in Section 2). Consider the following (A, e) (only showing columns of c_1 rows where c_1 is hit), e, and the probability of that occurring (Pr):

c_1	e	$\Pr(e_i\|d_k)$
1	0	g_1
1	0	g_1
1	1	$1 - g_1$
1	0	g_1

where g_1 is the true intermittency parameter ($g_1 = \frac{3}{4}$). Averaging e yields the estimate $h_1 = \frac{3}{4}$. To prove this is a perfect estimate, we show that h_1 maximizes the probability of this particular e (or any permutation with 1 fail and 3 passes) to occur. As $\Pr(e|\{1\})$ is given by $\Pr(e|\{1\}) = g_1^3 \cdot (1 - g_1)$, the value of g_1 that maximizes $\Pr(e|\{1\})$ is indeed $\frac{3}{4}$. Consequently, the estimate for g_1 is $h_1 = \frac{3}{4}$.

Finally, to illustrate the benefits of our approach, consider the program spectra in Figure 1 (c_1 and c_2 faulty, '2' means that the component was actually responsible for the overall failure). MHS computation yields $D = \{\{1,2\},\{2,5\},\{1,5\},\{4,5\}\}$. As mentioned in the previous section, $\varepsilon^{(0)}$ does not distinguish between candidates with the same cardinality. Hence, as they rank with the same probability, all candidates would have to be inspected. This also holds for $\varepsilon^{(1)}$ since $\forall_{d_k \in D}\ g(d_k) = \frac{N_p}{N_p+N_f} = \frac{1}{6}$. $\varepsilon^{(2)}$ distinguishes between the probabilities of candidates with same cardinality, but it ranks $\{2,5\}$ at the first place. BARINEL yields better results due to a better estimation of the individual gs, ranking the true fault $\{1,2\}$ at the first position.

c_1	c_2	c_3	c_4	c_5	e
2	1	0	1	0	1
0	2	0	0	1	1
2	0	0	0	1	1
1	2	0	0	1	1
0	2	0	0	1	1
1	0	0	1	1	0

Figure 1: Observation-matrix example

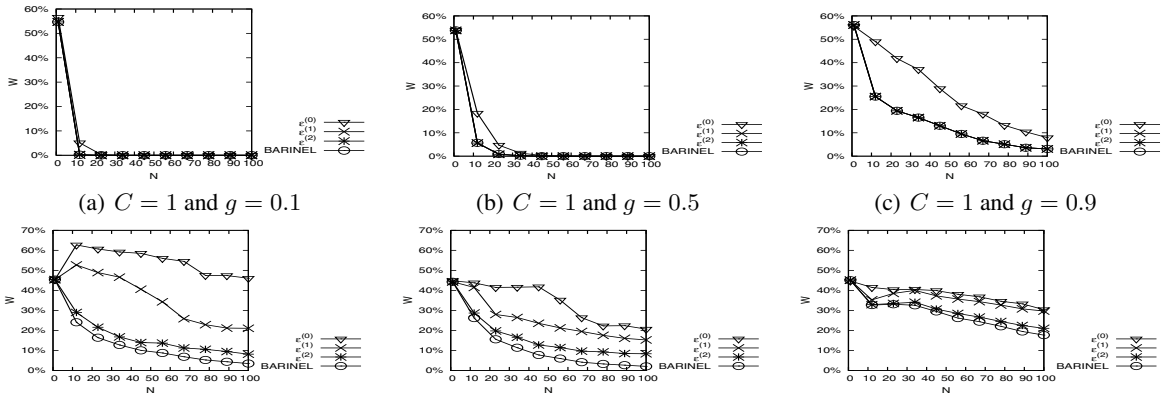
As the formulae that need to be maximized are simple and bounded in the $[0,1]$ domain, the time/space complexity of our approach is identical to the other reasoning policies presented in Section 2 modulo a small, constant factor on account of the gradient ascent procedure, which exhibits rapid convergence for all M and C (see Section 5).

4 Theoretical Evaluation

In order to assess the performance improvement of our framework we generate synthetic observations based on sample (A, e) generated for various values of N, M, and number of injected faults C (cardinality). Component activity a_{ij} is sampled from a Bernoulli distribution with parameter r, i.e., the probability a component is involved in a row of A equals r. For the C faulty components c_j (without loss of generality we select the first C components) we also set g_j. Thus the probability of a component being involved *and* generating a failure equals $r \cdot (1 - g)$. A row i in A generates an error ($e_i = 1$) if at least 1 of the C components generates a failure (noisy-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.

We compare the accuracy of our improved Bayesian framework with the classical framework in terms of a diagnostic performance metric W, that denotes the excess *work* incurred finding the actual components at fault. The metric is an improvement on metrics typically found in software debugging which measure the debugging effort associated with a particular diagnostic method [Abreu *et al.*, 2007; Renieris and Reiss, 2003]. For instance, consider a 4-component program with a unique diagnosis $d_1 = \{1,2,4\}$ with an associated $h_1 = \{0.33, 0.5, 1, 0.25\}$, where c_1, c_2 are actually faulty. The first component to be verified / replaced is the non-faulty c_4, as its health is the lowest. Consequently, W is increased with $\frac{1}{4}$ to reflect that it was inspected in vain.

The graphs in Figure 2 plot W versus N for $M = 20$, $r = 0.6$ (the trends for other M and r values are essentially the same, $r = 0.6$ is typical for the Siemens suite), and different values for C and g. The plots show that W for $N = 1$ is similar to r, which corresponds to the fact that there are on average $(M - C) \cdot r$ components which would have to be



(a) $C = 1$ and $g = 0.1$ (b) $C = 1$ and $g = 0.5$ (c) $C = 1$ and $g = 0.9$

(d) $C = 5$ and $g = 0.1$ (e) $C = 5$ and $g = 0.5$ (f) $C = 5$ and $g = 0.9$

Figure 2: Wasted effort W [%] for several settings

inspected in vain. For sufficiently large N all policies produce an optimal diagnosis, as the probability that healthy diagnosis candidates are still within the hitting set approaches zero.

For small g_j W converges more quickly than for large g_j as computations involving the faulty components are much more prone to failure, while for large g_j the faulty components behave almost similarly, requiring more observations (larger N) to rank them higher. For increasing C more observations are required (N) before the faulty components are isolated. This is due to the fact that failure behavior can be caused by much more components, reducing the correlation between failure and particular component involvement.

The plots confirm that $\varepsilon^{(0)}$ is the worst performing policy, mainly due to the fact that it does not distinguish between diagnosis with the same fault cardinality. Only for $C = 1$ the $\varepsilon^{(2)}$ and $\varepsilon^{(1)}$ policies have equal performance to BARINEL, as for this trivial case the approximation for g_j is equal. For $C \geq 2$ the plots confirm that BARINEL has superior performance, demonstrating that an exact estimation of g_j is quite relevant. In particular, the other approaches steadily deteriorate for increasing C.

5 Empirical Evaluation

In this section we assess the diagnostic capabilities of our approach for real programs. For this purpose, we use the well-known Siemens benchmark set and `space`[1]. The Siemens set contains 132 faulty versions of 7 C programs with extensive test suites. The `space` package provides 38 faulty versions and $13,585$ test cases, as well as $1,000$ test suites of 150 test cases (in our experiments, the test suite used is randomly chosen from the $1,000$ suites provided). See Table 1 for info on the number of components M and test cases N.

For our experiments, we have extended the Siemens set and `space` with program versions where we can activate arbitrary combinations of faults. For this purpose, we limit ourselves to a selection of 130 out of the 172 faults, based on criteria such as faults being attributable to a single line of code, to enable unambiguous evaluation. The observation matrices are obtained using the GNU `gcov` profiling tool.

[1] http://sir.unl.edu for further info.

Using this extended set of programs, the diagnostic quality - quantified by W - of BARINEL is compared to the three ε strategies outlined in Section 2.2. Table 1 presents a summary of the diagnostic quality of the different techniques.

Similar to Section 4, we aimed at $C = 5$ for the multiple fault-cases, but for `print_tokens` insufficient faults are available. All measurements except for the four-fault version of `print_tokens` are averages over 100 versions, or over the maximum number of combinations available, where we verified that all faults are active in at least one failed run.

In agreement with in the previous section, the results for software systems confirm that BARINEL outperforms the other approaches, especially considering the fact that the variance of W is higher (coefficient of variance up to 0.5 for `schedule2`) than in the synthetic case (1,000 sample matrices versus up to 100 matrices in this case). Only in 4 out of 24 cases, BARINEL is not on top. Apart from the obvious sampling noise, just mentioned, this is due to particular properties of the programs. For `schedule2` with $C = 2$ and $C = 5$, $\varepsilon^{(0)}$ is better due to the fact that almost all failing runs involve all faulty components (highly correlated occurrence). Hence, the program effectively has a single fault spreading over multiple lines, which favors $\varepsilon^{(0)}$ since it ranks candidates with cardinality one first. For `tcas` with $C = 2$ and $C = 5$, $\varepsilon^{(2)}$ marginally outperforms BARINEL (by less than 0.5%) which we attribute to the fact that the program is almost branch-free and small ($M = 174$) combined with the sampling noise (σ_W of 5% for `tcas`).

In these experiments the average time cost of BARINEL is merely 2.8 ms/obs, the difference with the other methods (0.2 ms/obs) being due to the (unoptimized) gradient ascent procedure.

6 Related Work

As mentioned earlier, in many model-based diagnosis approaches (e.g., GDE [De Kleer and Williams, 1987] GDE+ [Struss and Dressler, 1989], CDA* [Williams and Ragno, 2007], SAFARI [Feldman *et al.*, 2008]) faults are modeled to be persistent. Consequently, they may not work optimally when components fail intermittently. Recently, a model for intermittent behavior was introduced as an exten-

	print_tokens			print_tokens2			replace			schedule			schedule2			tcas			tot_info			space		
M / N	539 / 4,130			489 / 4,115			507 / 5,542			397 / 2,650			299 / 2,710			174 / 1,608			398 / 1,052			9,564 / 150		
C	1	2	4	1	2	5	1	2	5	1	2	5	1	2	5	1	2	5	1	2	5	1	2	5
#matrices	4	6	1	10	43	100	23	100	100	7	20	11	9	35	91	30	100	100	19	100	100	28	100	100
$\varepsilon^{(0)}$	13.7	18.2	22.8	21.6	26.1	30.8	16.2	25.1	33.8	17.2	23.5	28.6	29.3	**26.6**	**28.9**	28.0	26.9	28.7	14.0	18.2	21.5	19.5	25.2	34.3
$\varepsilon^{(1)}$	1.2	2.4	5.0	4.2	7.6	14.5	3.0	5.2	12.5	0.8	1.6	3.0	22.8	31.4	38.3	16.7	24.2	30.5	5.1	8.7	17.4	2.2	3.6	9.5
$\varepsilon^{(2)}$	1.2	2.4	4.8	5.1	8.9	15.5	3.0	5.2	12.4	0.8	1.5	3.1	21.5	29.4	35.6	16.7	**24.1**	**30.5**	6.1	11.7	20.9	2.2	3.7	9.9
BARINEL	**1.2**	**2.4**	**4.4**	**1.9**	**3.4**	**6.6**	**3.0**	**5.0**	**11.9**	**0.8**	**1.5**	**3.0**	**21.5**	28.1	34.9	**16.7**	24.5	30.7	**5.0**	**8.5**	**15.8**	**1.7**	**3.0**	**7.4**

Table 1: Wasted effort W [%] on combinations of $C = 1 \ldots 5$ faults for the Siemens set and space

sion of the GDE framework [De Kleer, 2007], later extended by [Abreu *et al.*, 2008a; 2008b]. Our approach improves on the approximations within these works, providing superior results. Furthermore, unlike MBD approaches we only assume a very abstract component model without even considering static system interconnection topology.

In logic (model-based) reasoning approaches to automatic software debugging, the model is typically generated from the source code - see [Mayer and Stumptner, 2008] for an evaluation of several models. The model is generated by means of static analysis techniques and is extremely complex. While at this detailed level intermittency is not an issue, the level of detail is such that the associated diagnostic complexity prohibits application to programs larger than a few hundred lines of code. As an indication, the largest program used in [Mayer and Stumptner, 2008] is tcas (172 lines of code only).

Our dynamic approach towards determining component involvement and system failure (i.e., through (A, e)) is inspired by statistical approaches to automatic software debugging, known as spectrum-based fault localization (each row in A is a spectrum). Well-known examples include the Tarantula tool [Jones and Harrold, 2005], the Nearest Neighbor technique [Renieris and Reiss, 2003], and the Ochiai coefficient [Abreu *et al.*, 2007]. These approaches rank components in terms of the statistical similarity of component involvement and observed program failure behavior. While attractive from complexity-point of view, the approaches do not consider multiple faults. Furthermore, the similarity metric has little value other than for ranking, in contrast to our probability metric.

7 Conclusions

Intermittent fault models can be crucial when modeling complex systems. Estimating the probability that a faulty component exhibits correct behavior is an important step for logic reasoning approaches to properly handle intermittent failures. In contrast to previous work, which merely approximates such probabilities for particular diagnosis candidates, in this paper we present a novel approach (BARINEL) to compute the exact probabilities per component at a complexity that is only a constant factor greater than previous approaches.

We have compared the diagnostic performance of BARINEL with the classical (Bayesian) reasoning approach, as well as with three intermittent reasoning approaches. Synthetic experiments have confirmed that our approach consistently outperforms the previous approaches, demonstrating the significance of maximum likelihood estimation over approximation. Application to the Siemens benchmark and space also suggest BARINEL's superiority (20 wins out of 24 trials), while the exceptions are caused by component clustering in combination with sampling noise.

Future work includes (1) extending the activity matrix from binary to integer, allowing us to exploit component involvement frequency (e.g., program loops), (2) reducing the cost of gradient ascent by introducing quadratic convergence, and (3) applying BARINEL to intermittent hardware.

References

[Abreu *et al.*, 2007] R. Abreu, P. Zoeteweij, and A. J. C. van Gemund. On the accuracy of spectrum-based fault localization. In *Proc. TAIC PART*, 2007.

[Abreu *et al.*, 2008a] R. Abreu, P. Zoeteweij, and A. J. C. van Gemund. A dynamic modeling approach to software multiple-fault localization. In *Proc. DX*, 2008.

[Abreu *et al.*, 2008b] R. Abreu, P. Zoeteweij, and A. J. C. van Gemund. An observation-based model for fault localization. In *Proc. WODA*, 2008.

[Avriel, 2003] M. Avriel. *Nonlinear Programming: Analysis and Methods*. 2003.

[De Kleer and Williams, 1987] J. De Kleer and B. C. Williams. Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130, 1987.

[De Kleer *et al.*, 1992] J. De Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.

[De Kleer *et al.*, 2008] J. De Kleer, B. Price, L. Kuhn, M. Do, and R. Zhou. A framework for continuously estimating persistent and intermittent failure probabilities. In *Proc. DX*, 2008.

[De Kleer, 2006] J. De Kleer. Getting the probabilities right for measurement selection. In *Proc. DX*, 2006.

[De Kleer, 2007] J. De Kleer. Diagnosing intermittent faults. In *Proc. DX*, 2007.

[Feldman *et al.*, 2008] A. Feldman, G. Provan, and A. J. C. van Gemund. Computing minimal diagnoses by greedy stochastic search. In *Proc. AAAI*, 2008.

[Jones and Harrold, 2005] J. A. Jones and M. J. Harrold. Empirical evaluation of the Tarantula automatic fault-localization technique. In *Proc. ASE*, 2005.

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

[Mayer and Stumptner, 2008] Wolfgang Mayer and Markus Stumptner. Evaluating models for model-based debugging. In *Proc. ASE*, 2008.

[Renieris and Reiss, 2003] M. Renieris and S. P. Reiss. Fault localization with nearest neighbor queries. In *Proc. ASE*, 2003.

[Struss and Dressler, 1989] P. Struss and O. Dressler. "Physical Negation" - Integrating fault models into the general diagnostic engine. In *Proc. IJCAI*, 1989.

[Williams and Ragno, 2007] B. Williams and R. Ragno. Conflict-directed A* and its role in model-based embedded systems. *Discrete Applied Mathematics*, 155(12):1562–1595, 2007.