Delft University of Technology
Software Engineering Research Group
Technical Report Series

Diagnosing Intermittent Faults

Arjan J.C. van Gemund, Rui Abreu, and Peter Zoeteweij

Report TUD-SERG-2008-041

TU Delft

SERG

TUD-SERG-2008-041

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
http://www.se.ewi.tudelft.nl/techreports/

For more information about the Software Engineering Research Group:
http://www.se.ewi.tudelft.nl/

Diagnosing Intermittent Faults

Arjan J.C. van Gemund Rui Abreu Peter Zoeteweij

November 5, 2008

In this working report we outline how to determine the intermittency parameters g_j from the activity matrix A (context: DX'08 paper Abreu, Zoeteweij, Van Gemund). We start with the single fault (SF) case and show that averaging over the error vector e is the exact way. We also show that in this way the probability of obtaining exactly this e vector in A is optimal. This is the key insight that allows us to determine g in the general multiple-fault (MF) case. We formulate the g_j problem as a (probability) maximization problem, which we solve using a simple gradient ascent technique.

1 Single fault case

In the following, we determine the optimal g_j in the simple case where $C = 1$. Consider A (only showing columns of c_1 and e and rows where c_1 is hit):

1	0
1	0
1	1
1	0

The probability of this occurring (Pr) is given by

		$\Pr(e_i)$
1	0	g
1	0	g
1	1	$1-g$
1	0	g

where g is the true intermittency parameter (that we cannot directly measure). Now we intuitively know that to derive g from A we simply average over e. Thus we compute g as

$$g = \frac{n_{10}}{n_{10} + n_{11}} = \frac{2}{3}$$

But why is this procedure the right one? The key insight is that **this** value of g **maximizes** the **probability** of **these** observations of e_i ($\Pr(e)$) to occur!

Given g, $\Pr(e)$ is given by $\Pr(e) = g^3 \cdot (1-g)$. Since we may assume that each individual outcome of $\Pr(e)$ is independent the value of g that maximizes $\Pr(e)$ is indeed $\frac{2}{3}$.

In general, for $N_F \to \infty$, $n_{10} = N_F \cdot g$ and $n_{11} = N_F \cdot (1-g)$. Thus $\Pr(e)$ is given by

$$\Pr(e) = g^{N_F \cdot g} \cdot (1-g)^{N_F \cdot (1-g)}$$

Why is this probability maximal? Let x denote our estimation of g. The value of x for which

$$\Pr(e) = x^{N_F \cdot g}(1-x)^{N_F \cdot (1-g)}$$

is maximal is $x = g$.

Proof $x^{N_F \cdot g} \cdot (1-x)^{N_F \cdot (1-g)}$ maximal implies $x^g \cdot (1-x)^{1-g}$ maximal
find x by derivative to x equals zero:
$g \cdot x^{g-1} \cdot (1-x)^{1-g} - x^g \cdot (1-g) \cdot (1-x)^{(1-g-1)} = 0$
$g \cdot (1-x) = x \cdot (1-g)$
$x = g$ ∎

As

$$g = \frac{n_{10}}{n_{10} + n_{11}}$$

it follows that

$$x = \frac{n_{10}}{n_{10} + n_{11}}$$

is the perfect estimator for g.

In summary, g is found by maximizing $\Pr(e)$, i.e., $g = \arg\max_g \Pr(e)$

2 Multiple fault case

Instead of generalizing to the C fault case, we just treat the $C = 2$ case for ease of exposition, as generalization to C faults is trivial.

Consider A given by (again, ignoring healthy columns)

			$\Pr(e_i)$
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 there is a row that involves both c_1 and c_2 we cannnot just estimate g_j through the above, single-fault approach due to the dependency between g_1 and g_2. Again, we need to find x that maximizes

$$\Pr(e) = x_1 \cdot x_2 \cdot (1 - x_1) \cdot (1 - x_1 \cdot x_2)$$

In general, for given g_j, A will contain

a	10 0	entries ($\Pr = g_1$)
b	10 1	entries ($\Pr = 1 - g_1$)
c	01 0	entries ($\Pr = g_2$)
d	01 1	entries ($\Pr = 1 - g_2$)
e	11 0	entries ($\Pr = g_1 \cdot g_2$)
f	11 1	entries ($\Pr = 1 - g_1 \cdot g_2$)

where a, ..., f are samples from the binomial distributions ($\mu = N_F \cdot g_1$, $\sigma = N_F \cdot g_1 \cdot (1-g_1)$), ..., ($\mu = N_F \cdot g_1 \cdot g_2$,$\sigma = N_F \cdot g_1 \cdot g_2(1 - g_1 \cdot g_2)$), respectively.

Consequently, we need to find the g_j that maximizes

$$\Pr(e) = g_1^a \cdot (1-g_1)^b \cdot g_2^c \cdot (1-g_2)^d \cdot g_1^e \cdot g_2^e \cdot (1 - g_1 \cdot g_2)^f$$

The above easily generalizes to the multiple-fault case although the formulae become more complex. Later on, we will outline a simple algorithm to solve this argmax problem.

3 A new diagnosis algorithm

Note that the above expression for $\Pr(e)$ is similar to the epsilon strategy, so apparantly the optimal g_j also maximizes the Bayesian ranking, i.e., determining the opimal g_j also generates the top ranking probability (if we ignore the priors). This implies that we adopt the above argmax problem as key step in the diagnosis algorithm. Previously, for every candidate we used the same g_j and plugged that into a Bayesian probability computation to derive the ranking. In our new approach, for every candidate we compute the optimal g_j that best explains the observations, which immediately gives us the highest probability in the Bayesian ranking (for a certain cardinality). Instead of computing the probabilities of the remaining candidates down in the ranking (based on this g_j), we only need consider the one top probability, and compare it to the top probability of another ranking that is based on the optimal g_j when assuming another candidate. The reason is that there is actually no point in going down a particular ranking if we know that the g_j are only valid for the top candidate in that ranking. So the true candidate is the one that produces a ranking of which the top probability is highest over all rankings (as this is also the probability of that candidate). Thus, instead of having two loops (the argmax loop and the Bayesian loop) we only consider an argmax loop as in the following:

```
for each MF candidate (that survives the MHS filter)
   solve the argmax problem yielding  g_j and Pr(e)
sort the candidates in terms of Pr(e) . p^C
```

In terms of the resulting ranking probabilities this algorithm deviates from the traditional approach in that in each individual probability is now taken into account the g_j that corresponds to that candidate!

Continuing with the above example A:

1	0	1	$1 - g_1$
1	1	1	$1 - g_1 g_2$
0	1	0	g_2
1	0	0	g_1

After the HS filter (no MHS), 2 candidates remain, which we code using bits for each c_j (e.g., $d_1 0 = c_1$ faulty, c_2 not, so single fault)

d_{10} : only c_1 faulty so $\Pr(e)$ is given by $\Pr(e) = (1-g_1)^2 \cdot g_1$ which maximizes for $g_1 = \frac{1}{3}$ yielding $\Pr(e) = \frac{4}{27}$

d_{11} : both c_1, c_2 faulty so $\Pr(e) = x_1 \cdot x_2 \cdot (1-x_1) \cdot (1-x_1 x_2)$ which maximizes for $g_1 = \frac{1}{3}$, $g_2 = 1$ yielding $\frac{4}{27}$.

As $g_2 = 1$ indicates c_2 is not at fault we reject this solution. Actually, c_1, c_2 is subsumed by c_1 so considering $d_1 1$ is pointless but we just want to show how this works.

Thus the diagnosis equals d_{10} with $\Pr(d_{10}) = \Pr(e) \cdot p$, $g_j = \arg\max_{g_j} \Pr(e)$ where p is the prior fault prob.

4 The argmax problem

For every candidate we can compile the expression $pr(e)$ to be maximized. Since the expression is differentiable we can apply a simple Gradient Ascent technique (bounded wiithin the domain $0 < g_j < 1$), or, better, a Newton algorithm that exploits the Hessian operator to achieve quadratic convergence. The compilation of the Nabla operator needed in both approaches is relatively straightforward, but requires quite some paperwork to derive the expression in the general case for $C > 2$. The Hessian is even more complex, so we would first stick to simple Gradient Ascent. Appendix A outlines the general approach.

5 Conclusion

The proposed algorithm presents a radical departure from the standard Bayesian update regimen as formulated by Dekleer for non-intermittent systems, extended by both Dekleer and Abreu et al. with epsilon policies to account for various intermittency models. Recognizing that intermittency behavior is component-specific, the algorithm takes into account the interplay between deriving the g_j from A that optimally explain A given a candidate hypothesis and deriving the probability ranking of the diagnostic candidates. Experiments are underway to evaluate the performance of the algorithm compared to the earlier epsilon policies.

A Gradient Ascent Procedure

We outline the computational aspects of the algorithm for $C = 2$ for ease of exposition. The approach easily generalizes for $C > 2$.

Consider our running example A:

$$\begin{array}{c|c} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{array}$$

Let our minimal hitting set algorithm (e.g. STACCATO) yield the diagnosis D comprising the candidates $h^{(1)}, .., h, .., h^{(k)}$ e.g., $h = (1, 1, 0, \ldots, 0)$ means $1, 2$ h_1 means 1st bit of h. For A we only have $D = (1, 0), (1, 1)$.

For each h we define $\Pr(h)$. Compiled in order of the rows of A we obtain:

$$\begin{aligned} \Pr((1,0)) &= (1-g_1)(1-g_1)1g_1 \\ \Pr((1,1)) &= (1-g_1)(1-g_1g_2)g_2g_1 \end{aligned}$$

Generalized per row i we obtain

$$\Pr(h,i) = g_1^{(h_1 A_{i,1})} ... g_M^{(h_M A_{i,M})}$$

for $e_i = 0$ or

$$\Pr(h,i) = (1 - g_1^{(h_1 A_{i,1})} .. g_M^{(h_M A_{i,M}}$$

for $e_i = 1$.

In our example:

$$\begin{aligned} \Pr((1,0),1) &= 1 - g_1^{(1\cdot 1)} g_2^{(0\cdot 0)} = 1 - g_1 \\ \Pr((1,0),2) &= 1 - g_1^{(1\cdot 1)} g_2^{(0\cdot 1)} = 1 - g_1 \\ \Pr((1,0),3) &= g_1^{(1\cdot 0)} g_2^{(0\cdot 1)} = 1 \\ \Pr((1,0),4) &= g_1^{(1\cdot 1)} g_2^{(0\cdot 0)} = g_1 \end{aligned}$$

(note that there are no more g_j involved than g_1).

$$\begin{aligned} \Pr((1,1),1) &= 1 - g_1^{(1\cdot 1)} g_2^{(1\cdot 0)} = 1 - g_1 \\ \Pr((1,1),2) &= 1 - g_1^{(1\cdot 1)} g_2^{(1\cdot 1)} = 1 - g_1g_2 \\ \Pr((1,1),3) &= g_1^{(1\cdot 0)} g_2^{(1\cdot 1)} = g_2 \\ \Pr((1,1),4) &= g_1^{(1\cdot 1)} g_2^{(1\cdot 0)} = g_1 \end{aligned}$$

(note that there are no more g_j involved than g_1, g_2).

In general, the expression for Pr(h) has a complex form with many terms (for C=2):

$$g_1^{n_{10}}, g_2^{n_{01}}, (g_1g_2)^{n_{11}}, (1-g_1)^{n'_{10}}, (1-g_2)^{n'_{01}}, (1-g_1g_2)^{n'_{11}}$$

where n, n' are counters, coded as bit strings that encode the involvement of individual components (like h), accumulated while scanning the rows like above. In the above example for candidate (1,1) we have $n_{10} = 1, n_{01} = 1, n_{11} = 0, n'_{10} = 1, n'_{01} = 0, n'_{11} = 1$.

Instead of expressing Pr we will only consider $\mathrm{d}(\log \Pr(e))/\mathrm{d}g_j$ as maximizing $\log \Pr(e)$ gives us the same g_j and computing ∇ from the log is much more convenient.

$\mathrm{d}(\log \Pr(e))/\mathrm{d}g_1$ has the form (for C=2, per term):

$$
\begin{aligned}
&(\frac{1}{g_1})^{n_{10}} \cdot n_{10} \cdot g_1^{(n_{10}-1)} \cdot 1 + 0 + (\frac{1}{g_1 g_2})^{n_{11}} \cdot n_{11} \cdot (g_1 g_2)^{(n_{11}-1)} \cdot g_2 + \\
&(\frac{1}{1-g1})^{n'_{10}} \cdot n'_{10}(1-g_1)^{(n'_{10}-1)} \cdot (-1) + 0 + \\
&(\frac{1}{(1-g_1 g_2)})^{n'_{11}} \cdot n'_{11}(1-g_1 g_2)^{(n'_{11}-1)} \cdot (-g_2)
\end{aligned}
$$

which reduces to

$$\frac{n_{10}}{g_1} + 0 + \frac{n_{11}}{g_1} - \frac{n'_{10}}{(1-g_1)} - 0 - n'_{11}\frac{g_2}{(1-g_1 g_2)}$$

Similarly, $\mathrm{d}(\log \Pr(e))/\mathrm{d}g_2$ equals

$$0 + \frac{n_{01}}{g_2} + \frac{n_{11}}{g_2} - 0 - \frac{n'_{01}}{(1-g_2)} - n'_{11}\frac{g_1}{(1-g_1 g_2)}$$

Consequently $\mathrm{d}(\log \Pr(e))/\mathrm{d}g_j$ contains all terms which involve g_j. It follows

$$\frac{\mathrm{d}\log \Pr}{\mathrm{d}g_j} = \sum_{\text{all n involving j}} \frac{n}{g_j} + \sum_{\text{all n' involving j}} -n'\frac{f'(n')}{(1-f(n'))}$$

where $f(n) = g_1^{n_1} \ldots g_M^{n_M}$ and $f'(n) = f(n)/g_j$.

In the above example for candidate (1,1) $n_1 0 = 1, n_0 1 = 1, n_1 1 = 0, n'_1 0 = 1, n'_0 1 = 0, n'_1 1 = 1$ as $\Pr((1,1) = g_1^1 g_2^1 (g_1 g_2)^0 (1-g_1)^1 (1-g_2)^0 (1-g_1 g_2)^1 = g_1 g_2 (1-g_1)(1-g_1 g_2)$.

Thus it follows

$$
\begin{aligned}
\frac{\mathrm{d}\log \Pr}{\mathrm{d}g_1} &= \sum_{(10,11)} \frac{n}{g_1} - \sum_{(10,11)} n'\frac{f'(n')}{(1-f(n')} \\
&= \frac{(n_{10}+n_{11})}{g_1} - n'_{10}\frac{f'(10)}{(1-f(10))} - n'_{11}\frac{f(11)}{1-f(11))} \\
&= \frac{(n_{10}+n_{11})}{g_1} - n'_{10}\frac{1}{(1-g_1)} - n'_{11}\frac{g_2}{(1-g_1 g_2)}
\end{aligned}
$$

and

$$\begin{aligned}\frac{\mathrm{d}\log \Pr}{\mathrm{d}g_2} &= \sum_{(01,11)} \frac{n}{g_2} - \sum_{(10,11)} n' \frac{f'(n')}{(1-f(n'))} \\ &= \frac{(n_{01}+n_{11})}{g_2} - n'_{01}\frac{f'(01)}{(1-f(01))} - n'_{11}\frac{f(11)}{(1-f(11))} \\ &= \frac{(n_{01}+n_{11})}{g_2} - n'_{01}\frac{1}{(1-g_2)} - n'_{11}\frac{g_1}{(1-g_1g_2)}\end{aligned}$$

When substituting the n and n' counters for the above example, the proper expressions are derived. The above expressions are the terms used for ∇. The use of ∇ in the gradient ascent iteration is straightforward.

TUD-SERG-2008-041
ISSN 1872-5392