ECAI 2014
T. Schaub et al. (Eds.)
© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-419-0-1053

Heuristics to Increase Observability in Spectrum-based Fault Localization

Claudio Landi[1] and **Arjan van Gemund**[2] and **Marina Zanella**[3]

Abstract. The high abstraction level of Spectrum-based Fault Localization (SFL) reasoning, on the one hand, offers the advantage of a model-free approach to diagnosis, while, on the other, reduces the inherently limited testability of many hardware and software systems. Thus, along with substantial complexity gains, SFL exhibits limited diagnostic performance, compared to Model-Based Diagnosis. This paper describes two algorithms (Lion and Tiger) that exploit low cost heuristics to determine the best location to insert additional test oracles (monitors, probes, invariants) so as to increase the observability within the systems. Experiments show that even simple algorithms can considerably improve SFL's diagnostic accuracy.

1 AMBIGUITY REDUCTION

In SFL [1, 5] the following is given:

- A finite set $C = \{c_1, \ldots, c_j, \ldots, c_M\}$ of M components of which M_f are faulted.
- A finite set $T = \{t_1, \ldots, t_i, \ldots, t_N\}$ of N tests with binary outcomes $O = (o_1, \ldots, o_i, \ldots, o_N)$, where $o_i = 1$ if test t_i failed, and $o_i = 0$ otherwise.
- A $N \times M$ (test) coverage matrix, $A = [a_{ij}]$, where $a_{ij} = 1$ if test t_i involves component c_j, and 0 otherwise.

The *health* of c_j is denoted $h_j \in \mathbb{R}$, where $h_j = 1$ represents full health, $h_j = 0$ represents faulted in all circumstances, while $0 < h_j < 1$ represents the case where c_j returns a failure in fraction h_j of the cases. The result of SFL is a *component ranking* $R =< c_{r(1)}, \ldots, c_{r(j)}, \ldots, c_{r(M)} >$, ordered non-increasingly in terms of the likelihood $\mathsf{Pr}(c_j)$ that c_j is at fault. For the purpose of this paper we equate $\mathsf{Pr}(c_j)$ to the Ochiai [2] similarity coefficient s_j,

$$s_j = \frac{n_{11}(j)}{\sqrt{(n_{11}(j) + n_{01}(j))(n_{11}(j) + n_{10}(j))}}$$

where $n_{pq}(j)$ is given by $n_{pq}(j) = |\{i \,:\, a_{ij} = p \wedge o_i = q\}|$.

The diagnostic utility (accuracy, performance) of R is measured in terms of the *identification cost* C_d, which models the verification effort of a diagnostician, going down the ranking R searching for the actual faults (true positives). In particular, we measure the identification effort *wasted* on false positives (i.e., excluding the components found to be faulted) in order to obtain a metric that is independent of M_f. Let r denote the index in R of the faulted component with the lowest likelihood. Then the identification cost for all M_f faults equals r and the wasted cost is $r - M_f$. We will consider a normalized value $C_d = (r - M_f)/(M - M_f)$ which ranges from 0 to 1 (0 indicates no identification effort, i.e., all faulted components are ranked at the top, whereas 1 indicates maximum identification effort, i.e., all faulted components are at the bottom of the list).

Consider the circuit topology shown on the left of Fig. 1.

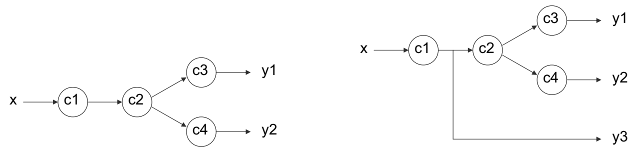

Figure 1. Example system (left) extended with monitor (right)

In SFL, each primary output observation is interpreted as one test. Thus one input vector (x) represents two tests, one involving c_1, c_2, and c_3 (the cone of y_1), and one involving c_1, c_2, and c_4 (the cone of y_2). Two test oracles readily monitor y_1 and y_2. Assume that c_1 is at fault, and that we observe y_1 correct [4] (pass), and y_2 incorrect (fail). In terms of A and O we have

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

where '+' and '-' denote pass and fail, respectively, to distinguish O from A in the figure. The associated similarity coefficients for $c_1, \ldots, c_4$ are $0.71, 0.71, 0, 1$, respectively. In terms of diagnostic performance, $C_d = 0.5$ since, after inspecting c_4 (and finding out that it is normal), half of the times the diagnostician will hit on c_1. SFL's disappointing diagnostic performance ($C_d = 0.5$, no better than random) is due to the *ambiguity group* (c_1, c_2), that have equal columns in A and therefore end up in the ranking with equal similarity coefficient. Suppose that, as shown on the right of Fig. 1, in order to improve the diagnosis, we choose the output of c_1 to place the new monitor so as to break the ambiguity group. If we run x again we now obtain

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

The output of c_1 (y_3) turns out to be incorrect. The associated similarity coefficients for $c_1, \ldots, c_4$ are now $0.82, 0.50, 0, 0.71$, respectively. Consequently, c_1 is now ranked on top and $C_d = 0$, i.e., perfect diagnostic performance.

[1] University of Brescia, Italy, email: claudio.landi24@gmail.com
[2] Delft University of Technology, The Netherlands, email: a.j.c.vangemund@gmail.com
[3] University of Brescia, Italy, email: marina.zanella@unibs.it

[4] Note that a fault has not to manifest itself at the component's output, nor propagate to a system output.

2 ALGORITHMS

Both algorithms for oracle placement sketched in the following are based on a simple, myopic heuristic, their space complexity is negligible compared with that of the coverage matrix, and they are intended to be called successively, until sufficient diagnosability is reached. The actual algorithms can be found in [4].

Algorithm 1 is intended for a static, design for SFL-diagnosability context, where a system topology and associated test suite are available (i.e., a coverage matrix A), but where the tests are not yet executed (or executable). Its rationale is to search for the largest ambiguity group (AG) in A: if one exists, it adds a monitor at a location that minimizes its size. Parameter $P \subseteq C$ denotes the subset of components whose output is (already) monitored by a test oracle. Function GETLARGESTAG returns the largest AG, denoted C', from the components selected as parameter (initially C). If there exist several largest AGs, one of them is randomly chosen. Function UPDATEA adds the new rows to A that are generated by those tests in the test suite that exercise c. Lion selects the component c' that minimizes A's new AG size (s), and adds it to P. Note that each time a monitor is added, it is possible that other AGs are also split, which produces a further ambiguity decrease. The algorithm returns the updated A and P, which can be used when Lion is recursively applied. The ambiguity size is typically used as termination criterion.

GETLARGESTAG takes $O(M^2 \cdot N)$ operations. The search for component c' costs $O(M)$ (size of C') times the sum of the costs of both the matrix update ($O(N)$ partial row copies) and the identification of the largest AG ($O(M^2 \cdot N)$), which totals $O(N \cdot M^3)$.

Algorithm 1 LION

```
function LION($A, P$)
    $C' \leftarrow$ GETLARGESTAG($A, C$)
    if $|C'| = 1$ then
        return $NULL$
    end if
    $s = |C'|$
    for each $c \in C',\ c \notin P$ do
        $A' \leftarrow$ UPDATEA($A, c$)
        $C'' \leftarrow$ GETLARGESTAG($A', C'$)
        if $|C''| < s$ then
            $s \leftarrow |C''|$
            $c' \leftarrow c$
        end if
    end for
    $A' \leftarrow$ UPDATEA($A, c'$)
    $P' \leftarrow P \cup \{c'\}$
    return $(A', P')$
end function
```

Algorithm 2 is intended for a dynamic operational context where the outcomes of run-time tests are available. Such outcomes are exploited so as to assist SFL in minimizing the ambiguity of R (in probabilistic sense). To this end, Tiger selects the component having the highest likelihood among those whose output is not monitored yet. Function SFL computes ranking R given coverage matrix A and test outcomes O. After the new component c is selected for monitoring, Tiger recomputes the matrix, retests the system (function TEST), and returns the new A, P, O to allow recursive application. The ranking index of the component selected for monitoring and/or the size of the largest subset of components that have the same value of the similarity coefficient can be used as termination criterion.

The time complexity of Tiger is entirely determined by the SFL algorithm ($O(M(N + \log M))$), the matrix update ($O(N \cdot M)$), and executing the test suite (amounting to $O(N \cdot M)$).

Algorithm 2 TIGER

```
function TIGER($A, P, O$)
    $R \leftarrow$ SFL($A, O$)
    $R' \leftarrow R \setminus P$
    if $|R'| = 0$ then
        return $NULL$
    end if
    $c \leftarrow r'_1$
    $A' \leftarrow$ UPDATEA($A, c$)
    $O' \leftarrow$ TEST($T$)
    $P' \leftarrow P \cup \{c\}$
    return $(A', P', O')$
end function
```

3 EXPERIMENTS

The algorithms have been tested by using a simulator based on a real situations awareness system developed at Thales Naval Systems. The original DAG topology of the system was modified to include some more components and random execution paths (tests) to make the diagnosis task more challenging. The system comprises $M = 74$ components. There are 10 test suites available, each providing $N = 33$ distinct execution paths. The diagnostic performance of Lion and Tiger are measured in terms of the metric C_d, obtained from injecting M_f faults, running the tests and performing diagnosis, while increasing the number of monitors as subsequently computed by either Lion or Tiger. In order to provide a reference, the performances of the algorithms are compared to those of Random monitor placement (upper bound on C_d) as well as Brute-Force placement (lower bound on C_d), enumerating over all $M - |P|$ added monitors.

Each of the 10 test suites is executed, and the C_d results are averaged over the 10 test suites. Within each test suite the C_d results are averaged over random fault injections, executing each test multiple times to sample from the random faulty component health (h). In total, per test suite the number of tests executed is 450 per monitor placement. Due to the complexity of brute-force computation, the experiments have been restricted to single fault injections ($M_f = 1$). Experiments offer evidence that, even with its low-cost heuristic, Tiger is capable of approaching optimum brute-force placement, roughly within a factor close to 3 in terms of monitors required. Lion performs less, as it does not exploit the test outcomes. Nevertheless, Lion performs far better than random placement, as the AG size in A is a good predictor of diagnostic performance.

In the future, time efficiency and quality of results achieved by SFL when assisted by either Lion or Tiger should be compared with those relevant to a Model-Based Diagnosis solver that, as SFL, can diagnose multiple persistent and intermittent faults [3].

REFERENCES

[1] R. Abreu and A.J.C. van Gemund, 'Diagnosing intermittent faults using maximum likelihood estimation', *Artificial Intelligence*, (2010).

[2] R. Abreu, P. Zoeteweij, and A.J.C. van Gemund, 'On the accuracy of spectrum-based fault localization', in *Proceedings TAIC-PART'07*, (2007).

[3] J. de Kleer, 'Diagnosing multiple persistent and intermittent faults', in *Proceedings IJCAI'09*, (2009).

[4] C. Landi. Algoritmi per la riduzione dell'ambiguità di diagnosi topologiche, Università degli Studi di Brescia, Italy 2012. MSc thesis.

[5] A.J.C. van Gemund, S. Gupta, and R. Abreu, 'The ANTARES approach to automatic systems diagnosis', in *Proceedings DX'11*, (2011).