Spectrum-Based Fault Localisation for Multi-Agent Systems

Lúcio S. Passos*,† and Rui Abreu*,‡§ and Rosaldo J. F. Rossetti*,†

*DEI/FEUP †Artificial Intelligence and INESC TEC ‡HASLab §Palo Alto
University of Porto Computer Science Lab. University of Porto Braga, Portugal
Porto, Portugal Porto, Portugal 3333 Coyote Hill Rd
lucio.san.passos@gmail.com, rui@computer.org, and rossetti@fe.up.pt

Abstract

Diagnostic undesired behavior in Multi-Agent Systems (MASs) is crucial to ascertain agents’ correct operation. However, generation of MAS models is both error-prone and time-intensive, as it exponentially increases with the number of agents and their interactions. In this paper, we propose a light-weight, automatic debugging-based technique, coined ESFL-MAS, which shortens the diagnostic process, while only relying on minimal information about the system. ESFL-MAS uses a heuristic that quantifies the suspiciousness of an agent to be faulty; therefore, different heuristics may have different impact on the diagnostic quality. Our experimental evaluation shows that 10 out of 42 heuristics yield the best diagnostic accuracy (96.26% on average).

1 Introduction

Previous approaches to ascertain nominal behavior of Multi-Agent Systems (MASs) (see [Nguyen et al., 2011; Fisher et al., 2007]) assume a priori knowledge (i.e., model) to diagnose observed failures. This knowledge can be appropriately built when designers fully understand the environment upon which agents act as well as agents’ state space.

However, in practice, due to (i) the complexity of MASs, (ii) dynamism of the environment, and (iii) presence of legacy systems, MAS and/or agent models are rather laborious to build. As a consequence, building the model is an error-prone task. Any knowledge not included in the built model by designers may therefore prevent the capability of model-based fault diagnosis to effectively recognize faults.

To address this issue, this paper considers a model-less approach to pinpoint behavioral faults in MASs. Spectrum-based Fault Localisation (SFL) is a promising technique that does not rely on an explicit model of the system under analysis and has been shown to yield good diagnostic accuracy for software systems [Hofer et al., 2015; Abreu et al., 2009].

The diagnosis process in SFL is based on the analysis of the differences in the so-called program spectra (abstraction over program traces) for passed and failed runs. SFL isolates the faulty component, using a similarity coefficient as heuristic, whose activity most correlates with observed failures. More importantly, SFL can be applied to resource-constrained environments due to its relatively low computational overhead [Abreu et al., 2009]. Such properties suggest that SFL is a well-suited technique for MASs.

Literature has shown that there is no standard similarity coefficient that yields the best result for SFL [Yoo et al., 2014; Hofer et al., 2015; Le et al., 2013]. Empirical evaluation is therefore essential to establish which set of heuristics excels for the specific context to which SFL is being applied. To the best of our knowledge, SFL has not as yet been applied to diagnose behavioral faults in MASs; there is hence the need to empirically evaluate different formulae using known faults to compare the performance yielded by several coefficients.

This paper makes the following contributions:

- We discuss the limitations of applying SFL with commonly used block hit spectra for time-persistent entities such as agents;
- We describe the Extended Spectrum-based Fault Localisation for Multi-Agent Systems (ESFL-MAS) to diagnose agent behavioral faults when testing the system as a whole;
- We present an experimental study on the impact of 42 heuristics in the ESFL-MAS diagnostic accuracy using the well-known and real-world representative Pickup and Delivery Problem as test suite;
- We show that for ESFL-MAS the Accuracy, Coverage, Jaccard, Laplace, Least Contradiction, Ochiai, Rogers and Tanimoto, Simple-Matching, Sorensen-Dice, and Support outperform the remainder coefficients across the entire quantity and quality data space (yielding 96.26% accuracy) in the specific conditions of our test suite.

2 Related Work

There is a wide set of approaches to increase the reliability in MASs. Formal verification, with model checking and theorem proving (see [Fisher et al., 2007]), has received great attention by the community. Albeit exhaustive and automated, such approaches are computationally costly, despite employing reduction techniques, as well as rely solely on the model of the system under test to certify correct functioning.

As for diagnosis in the scope of MASs, Dellarocas and Klein [2000] propose a diagnosis based on a fault-model of
agents. Fault-based diagnosis is not recommended for MASs due to agents interactions. Micalizio [2013] proposes a diagnostic system that pinpoint the set of erroneous actions integrated with recovery processes. Plan-diagnosis techniques depend on given agents’ plans curbing their usage at system-level testing. Concerned with determining coordination failures within team of agents Kalech [2012] proposes the social diagnosis. These works are robust and scale well in collaborative MASs; however, they focus only on the coordination failures, neglecting their influence on MAS overall performance. We refer the interested reader to Passos et al.’s [2015a] work for further analysis.

Therefore, to the best of our knowledge, all state-of-the-art approaches to ensure MAS correct behaviour rely on a priori model of the system to identify flaws. This work goes further and extends the use of SFL for MASs, collecting system dynamics information rather than using predefined models.

3 Preliminaries

Spectrum-based Fault Localisation

SFL is a dynamic program analysis technique, which requires minimal information about the system to be diagnosed. The SFL abstracts the system in terms of two general concepts: components and transaction. The former is an element of the system that, for diagnosis purposes, is considered to be atomic. Such entities could be individual statements, blocks, and so forth. The latter is a set of component information, whose correctness of output can be verified.

SFL relies on a set of test to produce a sequence of component activities that results in a particular output. The result of a process is either nominal (“pass”) or an error (“fail”). These fail and pass sets are also known as spectra, and originate from the collection of transactions. Additionally, the overall results of tests are called error vector. Given the hypothesis that closely correlated components are more likely to be relevant to an observed failure, the basic idea of SFL is that comparing the transactions over multiple runs and then computing the suspiciousness values of components can indicate which of these is the most likely to be the faulty one.

Several SFL methods use different formulae of similarity coefficients to compute such suspiciousness values. In this paper, an exhaustive list of 42 heuristics [Hofer et al., 2015] has been studied focusing on the context of fault localisation in software agents. These coefficients are: Accuracy ($C_1$), Added Value ($C_2$), Anderberg ($C_3$), Certainty Factor ($C_4$), Collective Strength ($C_5$), Confidence ($C_6$), Conviction ($C_7$), Coverage ($C_8$), Example and Counterexample ($C_9$), Gini Index ($C_{10}$), Goodman and Kruskal ($C_{11}$), Information Gain ($C_{12}$), Interest ($C_{13}$), Interestingness Weighting Dependency ($C_{14}$), J-Measure ($C_{15}$), Jaccard ($C_{16}$), Kappa ($C_{17}$), Klosgen ($C_{18}$), Laplace ($C_{19}$), Least Contradiction ($C_{20}$), Leveraging ($C_{21}$), Loefinger ($C_{22}$), Normalized Mutual Information ($C_{23}$), Ochiai ($C_{24}$), Ochiai II ($C_{25}$), Odd Multiplier ($C_{26}$), Odds Ratio ($C_{27}$), One-way Support ($C_{28}$), Piatesky-Shapiro ($C_{29}$), Relative Risk ($C_{30}$), Rogers and Tanimoto ($C_{31}$), Sebag-Schoenaauer ($C_{32}$), Simple-Matching ($C_{33}$), Sorensen-Dice ($C_{34}$), Support ($C_{35}$), Tarantula ($C_{36}$), Two-way Support ($C_{37}$), Two-way Support Variation ($C_{38}$), Yule’s Q ($C_{39}$), Yule’s Y ($C_{40}$), Zhang ($C_{41}$), $\phi$-coefficient ($C_{42}$). For the sake of organisation, we refer the interested reader to Hofer et al.’s [2015] work for further details regarding these formulae.

Concepts and Definitions

In this paper we consider that faults in agents’ behaviour depend on a given context, i.e. on how each agent interprets that particular situation; and, mainly, these faults are a systemic matter as it might affect the overall performance [Platton et al., 2007]. More specifically, the term “faulty agent” is used to refer that the agent either is unhealthy and needs to be repaired or has been induced to a failure state (known as cascading effect).

Diagnosis is the task of pinpointing the faulty component that led to symptoms (failure/error). In software, the set of components can be defined at any level of granularity: a class, a function, or a block. The lower the level of granularity gets, the more focused is the diagnosis, even though such low granularity requires more computational effort [Zamir et al., 2014]. The diagnosis problem for MASs can be defined as follows.

Definition 1 (Multi-Agent Diagnosis Problem) Given a multi-agent system MAS that has a set of agents AGS and a set of observations OBS for a test case, then the diagnosis problem is to find the faulty agent which is responsible for the mismatch between the expected MAS performance and the observed one.

The multi-agent diagnosis problem is defined with granularity at the agent level and thus considers agents as black boxes. This is a fair assumption when different parties implement agents reasoning and do not completely share their knowledge and/or architecture. On the one hand, the proposed technique is not able to identify the specific bug inside the code of the faulty agent; on the other hand, however, it has the advantage of not being either programming language or agent architecture specific.

4 ESFL-MAS

In this section we describe our approach, called Extended Spectrum-based Fault Localization for MAS (ESFL-MAS). The first step is to map concepts of MASs into the aforementioned elements of SFL. As stated by Definition 1, this work deals with agent-level diagnosis and therefore components of the system are the agents themselves. As for transactions, since MASs must run during some period of time (several time frames if considering discrete time) to observe their emerging behaviour, ESFL-MAS considers the error status of an agent’s behaviour at time $n$ as a transaction. We use the terms time frame and time step interchangeably.

The far most commonly used type of spectra is called hit spectra [Abreu et al., 2009]. It encodes the involvement (success/failure) of each component of the system in terms of involved/not involved in a given test case. The constraint of using the hit spectra is related to the lack of useful information about state of the agent execution. Since agents are time-persistent entities, they are always active and acting upon the environment; this creates spectra with very high entropy.
High entropy means that there is less useful information in the spectra which, consequently, decreases the diagnostic quality of SFL [Campos et al., 2013]. Therefore, block hit spectra is not suitable in the context of MASs.

The solution proposed to overcome such a limitation is to encode the performance of each agent in terms of being expected/unexpected at a determined time step. A detection of symptoms (also called error detection) phase is responsible to infer any behavioural violation from observations of the system [de Kleer and Williams, 1987]. Specifically for agents, this can be done using several methods from monitoring agent’s utility to applying anomaly detection techniques. Note that detecting an unexpected behaviour does not necessarily mean that one has identified the agent that is causing the system failure to occur. ESFL-MAS pinpoints the faulty agent so the designer is able to fix it, which is also essential to improve reliability of MASs. Given this performance-oriented perspective, we propose the performance spectra, where the error detection phase generates the set of data composing the spectra. It is worthwhile mentioning that error detection mechanisms are outside the scope of this paper.

Definition 2 Let $N$ denote the number of passing and failing time frames. 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 $N \times M$ performance matrix, where $a_{nm}$ denotes whether agent $m$ performed an unexpected behaviour at time $n$ ($a_{nm} = 1$) or not ($a_{nm} = 0$). Let $e_n$ denote the error vector, where $e_n$ implies whether the MAS has passed ($e_n = 0$) or failed ($e_n = 1$) the test case at time $n$.

Given that the pair $(A, e)$ highly depends on both the environment’s settings and the agents’ autonomy, SFL is limited to catch multiple instances of both dependencies. This limitation is addressed as follows. First, to solve the problem of multiple environment’s settings, one must run the MAS for different environment and agent settings; thus, the ESFL-MAS collects performance spectra referring to several test cases. Second, to solve the agent’s autonomy problem, one must execute the MAS $J$ rounds of the same test case to ensure that the collected spectra cover as many agents’ activation paths (i.e., choices) as possible.

For each agent, a dichotomy matrix is then created (see Table 1). One dimension of this matrix is related to the amount of time steps in which the agent had an unexpected behaviour detected, and the other is the passed/failed MAS status determined by the expected output of a test case.

We have observed that agents are constantly monitored over time but do not consistently fail. For this reason, the collected performance spectra have several time frames in which either every agents performed an expected behaviour or every agent performed an unexpected behaviour. Both events contain no information for SFL because only the variability of transactions in the spectrum contributes towards improving diagnostic quality. A proposed optimisation to ESFL-MAS, named MAS-Filter, recognises these aforementioned events and filters them from the performance spectra to increase quality of the diagnosis process. Conceptually, when excluding the non-useful time frames, the entropy value of the spectra tends to its optimal value and consequently increases diagnostic accuracy. The MAS-Filter is defined as follows.

### MAS-Filter

$$\text{MAS-Filter}((A, e)) = (A, e)^F \leftarrow (A, e) - \{A_n, e_n) : (A_n, e_n) \in (A, e) \land (\forall a \in A_n (a = 1) \lor \forall a \in A_n (a = 0))\}$$

where $(A, e)^F$ is the filtered spectra. This operation is executed when the value of $a_{nm}$ is equal to 1 or 0 for all agents in the MAS. In Table 2, we highlight the lines that would be deleted by MAS-Filter.

**Running Example.** It is borrowed from our experimental setup described in Section 5. We have reduced the number of agents of this example. Let us assume that agent Ag5 erroneously compute its distance from a gold nugget because of an unforeseen bug in the reasoning process unintentional left by the designer/programmer. Because of this fault Ag5 presents lower performance given some specific situation. The faulty agent will be diagnosed using Algorithm 1.

The MAS is executed twice and assuming that there is a mechanism able to detect unexpected behaviour in the agents, the pairs $(A, e)$ shown in Table 2 are built (ignoring the highlighted columns). MAS-Filter filters the highlighted columns and generates the pairs $(A, e)^F$. They are depicted for illustration purposes only as they are not taken into account in the computation of the dichotomy matrices.

ESFL-MAS uses the information of the dichotomy matrices from Table 2 to compute the suspiciousness values using the similarity coefficient. In this example, we choose the Jaccard coefficient ($C_{16}$). ESFL-MAS computes the suspiciousness value by inserting the information of the dichotomy matrix into the formula, e.g., for agent Ag5:

$$C_{16} = \frac{c_{11}}{c_{11} + c_{10} + c_{01}} = \frac{3}{3 + 0 + 1} = 0.75$$

The process is repeated for every agent until it obtains the values shown in Table 3. Afterwards, the list of agents is sorted in descending order of the coefficient value. As expected, the

<table>
<thead>
<tr>
<th>MAS status</th>
<th>Behaviour of $A_{gm}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failed ($e_n = 1$)</td>
<td>$c_{11}$</td>
</tr>
<tr>
<td>Passed ($e_n = 0$)</td>
<td>$c_{10}$</td>
</tr>
</tbody>
</table>

Table 1: Dichotomy table for performance spectrum

<table>
<thead>
<tr>
<th>Agent</th>
<th>$N$</th>
<th>$c_{11}$</th>
<th>$c_{10}$</th>
<th>$c_{01}$</th>
<th>$c_{00}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{g1}$</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$A_{g2}$</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$A_{g3}$</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$A_{g4}$</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$A_{g5}$</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

Table 2: The performance spectra, error vector, and the values of the dichotomy matrix for the running example.
faulty agent $Ag_5$ was the highest ranked by ESFL-MAS in the end of the process.

5 Experimental Setup

5.1 Test Suite

We use an instance of the Pickup and Delivery Problem (PDP) [Savelsbergh and Sol, 1995] to test our approach because (i) it is well-known and (ii) it is a real-world representative problem. MASs offer an interesting solution for PDP [Fischer et al., 1996].

The Second Edition of the Multi-Agent Programming Contest (MAPC) [Dastani et al., 2007] provides an instance of PDP known as the GoldMiners scenario. GoldMiners implements fundamental concepts of MASs, such as autonomy, team-work coordination, high-level interaction, as well as partial and local perception of the environment. We chose the MAS programmed in AgentSpeak, an agent-oriented programming language, using Jason [Bordini et al., 2007], an interpreter for an extended version of AgentSpeak, given our previous experience using AgentSpeak [Rossetti et al., 2002]. Additionally, MAPC’s implementations were previously tested and validated by the MAS community.

The Jason implementation aims to find a schedule that delivers as many items as possible at a lowest cost relying on a twofold strategy: first, a priori allocation of agents

Table 3: The Jaccard coefficient values and ranking.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Coefficient Value</th>
<th>Ranking $(D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Ag_1$</td>
<td>0.40</td>
<td>2</td>
</tr>
<tr>
<td>$Ag_2$</td>
<td>0.33</td>
<td>3</td>
</tr>
<tr>
<td>$Ag_3$</td>
<td>0.20</td>
<td>4</td>
</tr>
<tr>
<td>$Ag_4$</td>
<td>0.17</td>
<td>5</td>
</tr>
<tr>
<td>$Ag_5$</td>
<td>0.75</td>
<td>1</td>
</tr>
</tbody>
</table>

5.2 Data Acquisition

A two-step process (depicted in Figure 1) generates the spectra required by the experiments.

Collecting Logs A MAS initially configured according to a given test case is executed to obtain the logs both from agents and from the overall system. For the experimental setup, we randomly generated 5 test cases and each of them corresponded to a set of initial positions for: agents, the depot, and gold nuggets. To collect information to generate spectra, the MAS with 25 agents was executed 75 times for each test case recording 1000 time steps.

Expected MAS Performance and Error Detection Both used test cases and the generation of performance spectra de-
Diagnostic quality is defined as the ratio of correct decisions to the total number of decisions made by the agents. A higher diagnostic quality indicates a more accurate decision-making process. Diagnostic performance is expressed in terms of diagnostic quality for each group of agents, as well as the standard deviation ($\sigma$). Note that the number of groups (and their elements) is not the same in every type of MAS. Only Group 01 has the same similarity coefficients and consistently yields the best accuracy (average of 96.26%) for our benchmark.

### 5.3 Evaluation Metric

Diagnostic performance is expressed in terms of diagnostic quality (also referred to as accuracy) that evaluates how many agents need to be inspected before the faulty agent is found. If other agents have the same similarity coefficient as the faulty agent, we use the average ranking position for these agents. Diagnostic quality is defined as [Steimann et al., 2013]

$$Q = \left(1 - \frac{|\{j | S_j > S_f\}| + |\{j | S_j \geq S_f\}| - 1}{2(M-1)}\right) \times 100\%$$

where $S_j$ and $S_f$ denote the suspiciousness value for agent $j$ and for the faulty agent respectively, and $M$ is the total number of agents. Intuitively, the $|\{j | S_j > S_f\}|$ term represents the number of agents ranked in front of the faulty agent whereas $|\{j | S_j \geq S_f\}|$ represents the number of agents with same or higher suspiciousness compared to the faulty one.

### 6 Experimental Results

The resulting groups for our benchmark are shown in Figure 2; moreover, Table 5 presents the average of diagnostic quality for each group of coefficients, as well as the standard deviation ($\sigma$). Note that the number of groups (and their elements) is not the same in every type of MAS. Only Group 01 has the same similarity coefficients and consistently yields the best accuracy (average of 96.26%) for our benchmark.

![Figure 1: Experimental Phases](image1)

Figure 1: Experimental Phases

![Figure 2: Similarity coefficients grouped by their accuracy](image2)

Figure 2: Similarity coefficients grouped by their accuracy; each node corresponds to a group; edges indicate relationships between groups such that A $\rightarrow$ B means “group A requires less effort to diagnose than group B”; those with the same vertical alignment present less than 1% difference in the mean accuracy.

After carefully analysing the group compositions, two fundamental aspects can explain them. First, from the mathematical formulae of coefficients, there is a logical causal relationship between accuracy and the dichotomy matrix: the more significant a coefficient assigns to anomalous behaviour for which a system’s error has been detected (represented by $c_{11}$), the better the diagnostic quality. Second, the test suite has reduced capacity of representing cascading effects of undesired behaviour. Thus, coefficients with greater emphasis on $c_{10}$ have lower diagnostic quality.

Second, the test suite has reduced capacity of representing cascading effects of undesired behaviour; for instance, the faulty agent sending the wrong location of a gold nuggets for a correct agent inducing the latter in failure. Thus, coefficients that give greater emphasis on time steps where MASs are correct and an agent have performed unexpectedly have lower diagnostic quality.

### On the Impact of Observation Quantity

In the previous results, we have assumed that the designer/tester has time to run the MAS several times under different conditions to collect a considerable amount of measurements. In practice, the tester works under short-time constraints. To investigate how the amount of data influences ESFL-MAS performance, we evaluate $Q$ while varying the number of passed ($N_p$) and failed time steps ($N_f$) encom-

<table>
<thead>
<tr>
<th>Group</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCNO</td>
<td>96.25</td>
<td>73.96</td>
<td>53.17</td>
<td>47.90</td>
<td>43.54</td>
<td>36.80</td>
<td>22.08</td>
</tr>
<tr>
<td>NCO</td>
<td>95.16</td>
<td>67.74</td>
<td>53.27</td>
<td>47.87</td>
<td>46.19</td>
<td>37.47</td>
<td>23.32</td>
</tr>
<tr>
<td>CNO</td>
<td>97.08</td>
<td>55.83</td>
<td>54.10</td>
<td>50.00</td>
<td>8.358</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CO</td>
<td>96.54</td>
<td>66.23</td>
<td>49.95</td>
<td>44.92</td>
<td>27.36</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5: Mean accuracy for each similarity coefficient
passed in the diagnosis process. We study this influence on the diagnostic accuracy $Q$ throughout all the range of available data in the range of 0.001-100%.

Figure 3 shows such evaluations of Group 01 and 02 for NCNO and CO versions. We can see that $N_c$ changes the diagnostic quality, however, as agents are more organised and coordinated, this effect becomes insignificant for all groups. This happens because, when agents work as teammates in an organised manner, the MAS performs as a “well-oiled machine” and no agent fails when the system is performing properly, otherwise the system itself would fail. Concerning the number of erroneous time steps $N_f$, we confirm from Figure 3 that adding failed time steps improves the diagnostic quality. The benefit of inducing more than 200 steps is marginal on average.

**On the Impact of Error Detection Precision**

This experiment discloses how precision in detecting error affects diagnostic quality for each similarity coefficient. For any realistic system and practical error-detection mechanism, there will very likely exist errors that go undetected. They can go undetected because of two reasons. First, the fault agent only jeopardises system’s operation under specific conditions. For instance, let us assume that a Miner agent, erroneously, is not able to perceive gold nuggets; no error is detected unless the agent is near a gold nugget. Second, analogously to faults in software, errors induced by agents might not propagate all the way to system failures and thus would go undetected.

Consequently, the number of rows in spectra, in which both faulty agent and system fail, will only be a fraction of the total rows in which the agent fails. More intuitively, this proportion represents the Error Detection Precision ($EDP$), that is, how precisely the error detection phase is able to correlate a system failure with the faulty agent. Using the prevision notation, we define

$$EDP = \left( \frac{c_11(f)}{c_11(f) + c_{10}(f)} \right) \times 100\%$$

where $f$ is the location of the faulty agent.

Each faulty version of our benchmark has an inherent value for $EDP$ fluctuating from 3.31% to 97.77%. We vary $EDP$ by (1) excluding time steps that activate the faulty agent, but for which no system error has been detected decreasing $c_{10}(f)$, and increasing $EDP$; and (2) excluding time steps that activate the faulty agent and for each an system error has been detected decreasing $c_{11}(f)$, and decreasing $EDP$.

Figure 4 shows for all cases how the diagnostic quality changes with respect to the error detection precision. We see that, on average for all cases, a detection precision more than 40% has marginal contribution to a better fault diagnosis. This does not mean that the community needs to give up improving error detection techniques; this means that, when coupled with a diagnosis phase, error detection needs a solid (not necessarily optimal) performance. Moreover, we confirm the Group 01 as the best set of similarity coefficients for MASs also regarding the $EDP$ variation. Foremost, we show that ESFL-MAS can maintain high accuracy even for low error detection precision being the borderline $EDP \geq 10\%$.

**7 Conclusions**

We proposed a novel approach, called ESFL-MAS, to localise faults in MASs that is able to identify agents that may jeopardise the overall performance through run-time profiles of the system. We argued that SFL needs be extended to support agent-specific features (such as autonomy) and then we proposed such extensions.

The empirical evaluation produced prominent results, giving a good prospect for the application of ESFL-MAS. Results show that Accuracy, Coverage, Jaccard, Laplace,
Least Contradiction, Ochiai, Rogers and Tanimoto, Simple-Matching, Sorensen-Dice, and Support yield the best diagnostic accuracy for the used benchmark. They yield roughly 96.26% diagnostic accuracy and are stable when varying either error detection precision and quantity of observations. However, from the experiments we also observed that ESFL-MAS’ accuracy might be jeopardised by cascading faults, produced by highly interacting agents.

Future work will address the improvement in our experimental setup to support cascading faults. Afterwards, we intend to compare ESFL-MAS to other diagnosis approaches including Kalech’s [2012] and Micalizio’s [2013]; as well as experiment with the spectrum-based reasoning technique [Abreu et al., 2009] which reasons in terms of multiple faults. Finally, we plan to integrate ESFL-MAS with an agent-oriented methodology [Passos et al., 2015b].

Acknowledgement

Authors would like to thank Anabela N. Fernandes, Leonardo A. Piedade, Nuno Cardoso, and Zafeiris Kokkinogenis for the useful discussions during the development of this work. This work is supported by Fundação para a Ciência e a Tecnologia (FCT), grant SFRH/BD/66717/2009, and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), grant 9382-13-5.

References


