Sampling UAV Most Informative Diagnostic Signals

Roberto Pietrantuono†, Massimo Ficco*, Stefano Russo†, and Gabriella Gigante ‡
*Seconda Università degli Studi di Napoli, Via Roma 29, 81031 Aversa, Italy
†Università degli Studi di Napoli Federico II, Via Claudio 21, 80125, Naples, Italy
‡CIRA - Italian Aerospace Research Center, Capua, Italy

***Abstract*—Detecting and diagnosing failures of Unmanned Aerial Vehicles during their mission is a key challenge for their effective deployment. On-board diagnostic systems are able to provide a huge amount of information about the state of the vehicle during the flight, by monitoring sensors, software, and hardware components. However, the ability of processing such data in an online manner is a serious obstacle to a timely detection and diagnosis of failures. This paper proposes a method to progressively focus the data collection on signals providing the most reliable information about the system failure probability, so as to reduce considerably the number of false alarms and/or undetected failures, and to ease the online data processing. We set a simulation experiment showing that the proposed approach is able to select the most informative subset of signals in few iterations in an effective and efficient way.**

***Index Terms*—Diagnosis, FDDR, FDIR, Health Management, UAV, Importance Sampling, Detection, Software Failure.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are highly complex systems whose behavior is dependent on both mechanical and electrical subsystems, and on software-intensive controls, such as the autopilot [1]. They are required to satisfy rigorous safety requirements, as their incorrect operation could be dangerous for other aircrafts, people, and environment [2], [3]. Besides pre-flight certified development and testing activities, an UAV is required to implement appropriate diagnostic and prognostic means to be able to promptly detect mission-time failures, diagnose them, and trigger mitigating actions on time.

Current diagnostic systems in UAVs are based on conventional FDDR (Fault Detection, Diagnosis, and Recovery) tasks jointly with Bayesian reasoning. Despite these provide a valuable support, finer health management systems are needed to "quickly and reliably pinpoint failures", as claimed by NASA researchers [1]. On-board diagnostics could provide huge amount of information about the health of the system at runtime; however, a relevant challenge is the ability of a diagnostic framework to elaborate such information timely and reliably [4]. For instance, while Bayesian methods are deemed indeed valuable because of their ability to perform deep reasoning using probabilistic models, their usage is limited due to the computationally expensive algorithms they adopt.

This paper proposes a method to filter diagnostic information in order to focus the attention on the most informative data while discarding useless information. The method is intended to improve the capability of diagnostic systems to take information only from sources actually contributing to the correct detection of failures, so as to reduce considerably the number of false alarms and/or undetected failures by easing the online data processing. This would greatly relax the burden on health management systems, which could improve the timeliness and reliability of decisions. The method, based on a Montecarlo sampling technique named *importance sampling*, which iteratively indicates the best set of signals able to provide the most of information with few samples. To demonstrate potentials of this technique, we setup a simulative experiment showing that the method can considerably improve the *precision* and *recall* of decisions taken about failures, by analyzing, after few iterations, a reduced set of samples (about 17% of samples examined at the beginning) taken from only the 10% of the initial set of signals.

II. Related work

There exist some frameworks for FDDR used in aerospace industry. They adopt a model-based approach, with different levels of abstraction and of complexity in the techniques used to specify and check the system behavior. TEAMS2[1] enables hierarchical multi-signal diagnosis, but does not model temporal or probabilistic relations. HyDE4[2] exploits models at various abstraction levels, comparing actual signal values with model results. *Livingston5*[3] is a NASA open-source diagnosis and recovery engine using temporal logic to specify expected properties and the SMV model checker to formally verify them. Model checking and temporal logic are also used in [5] for health management of real time systems. Bayesian networks are being adopted in health management systems for their ability to perform probabilistic reasoning; in [6], authors present ProADAPT, a system health management algorithm exploiting compilation of Bayesian networks into arithmetic circuits for efficient computation. The recent work by NASA [1] also adopts Bayesian system health models to reason about correlation of events with system failures, considering both sensors output and software events. Nonetheless, the spread of Bayesian approaches is heavily hindered by the computational effort needed to cope with the increasing amount of information coming from hardware, sensors, and software.

At software level, several papers address the online diagnosis problem, despite there is still a long way to go. For instance, our previous work [7] exploit information at OS level to support online diagnosis of software failures in ATC (Air Traffic Control) systems. A survey of techniques for failure detection and prediction at software level is in [8]. In [9], we adopt dynamic analysis to assess the probability of failure online, exploiting information at application interface level to detect anomalies, other than at OS-level.

[1] http://www.teamqsi.com/products/teams-designer/
[2] http://ti.arc.nasa.gov/tech/dash/diagnostics-and-prognostics/hyde-diagnostics/
[3] http://ti.arc.nasa.gov/opensource/projects/livingstone2/

Software diagnosis has mainly been studied separately so far. The cited work by NASA [1] is one relevant example combining on-board sensors and software signals to detect failures at both hardware and software level. In fact, software undergoes a series of extensive pre-flight V&V activities, ranging from formal verification and code analysis (e.g., [10]) to testing (e.g., [11], [12] up to process-level improvements via planning and measurements based on historical data analysis (e.g., [13]). However, as also remarked in [1], the interactions with sensors and hardware (hence the operating environment) can activate faults in unexpected (thus untested) environment (e.g., environment-dependent faults [14]), and thus should be considered in a unique framework. Indeed, putting together sensors and software signals exacerbate the problem of heavy load for on-board diagnostic and health management systems, which are forced to adopt simplistic rudimentary solutions in order to provide responses on time. What we propose is a way to reduce such a load by pinpointing the most influential monitoring points for a given context.

III. Basic terminology and background

In the following, each component of the system whose behavior is monitored by some (software/hardware) sensor is defined as *Diagnosable Units* (DUs), representing the atomic entities that can be observed for diagnostic purposes. A DU may be in "anomalous" state (where "anomalous" are all the conditions that deviate from specified normal behaviors). Specifically, each DU is associated with *signals* (we assume one signal per unit), which are a 0/1 representation of the correct/incorrect behavior of a DU: a signal notifies an incorrect behavior by an *alarm* whenever one or more of the associated (software/hardware) sensor values are "anomalous", namely, deviate from specified values. A DU failure is seen as an *error* for the system as a whole, which may (or may not) impact the state of the system. During the operational phase of the UAV, the diagnostic system monitors the DUs to detect possible problems. In such a typical detection strategy, not all the anomalies correspond to actual errors, i.e., errors could be signaled even when the system is behaving correctly but that condition has not been recognized as normal. Actually, in many cases, anomalies of DUs either do not correspond to actual DU errors, or, if they are errors, do not lead to a system failure, so the notified event constitutes a *false alarm*from the failure diagnosis perspective. Such a pessimistic strategy leads to a non-negligible amount of false alarms. On the other hand, it aims at minimizing the more serious situation of a DU error not notified by any signal (namely, a *false negative*). his is a more dangerous scenario the diagnostic system is required to cope with.

IV. The proposed sampling method

In an UAV system, there are, potentially, numerous sources of information that could be exploited for identifying problems. However, the diagnostic system is required to have the ability of detecting failures promptly and in an online manner in order to react while the system is flying. This requires a high processing ability, which could be not easily achievable.

Our approach aims at progressively identifying those sources of information that are more relevant to detect actual failures. The method uses historical information about DU signals, trying to shift the focus on those signals that have provided more accurate information in the past.

We consider two different scenarios. In the former, we want the diagnostic system spotting those signals with higher *precision*, i.e., signals that notify a *real* problem whenever an alarm is raised. In the latter scenario, we want the diagnostic system to identify those signals with higher *recall*, namely, which always raise an alarm whenever there is a problem. In both cases, the goal is to allow the diagnostic system to focus only on the most informative signals, in order to increase the number of samples collected from them, while neglecting the big amount of useless information coming from other signals (i.e., taking few or no samples from them). This not only would allow a system to take more accurate decisions with less information, but would also allow refining the diagnostic system development in future versions.

The approach is based on a Monte Carlo sampling method, named *importance sampling*, which is an inference method to approximate the computation of the true distribution of variables of interest. It samples from the true (unknown) distribution, and thus represents the beliefs (i.e., hypotheses) about the state of the system by sets of samples. Each sample is associated with a probability that the belief is true, and at each iteration: (1) the hypotheses are modified to account for changes in the system, (2) the probability of each hypothesis is updated by examining some samples of that hypothesis; and (3) a larger number of samples are drawn from hypotheses with a larger probability, to be analyzed in the next iteration. The goal is to converge, in few iterations, to the true probability distribution over the set of hypotheses, identifying the ones more likely to be true. In our case, samples are the signals taken at each iteration from each DU; the hypotheses are the belief about which signal is more likely to improve the overall *precision* or *recall*, for the two scenarios, respectively; the true unknown distribution to approximate is the optimal number of samples for each signal that maximizes precision (or recall).

1) Initial Assignment: As first step, the importance sampling requires a relatively small set of samples. In this step, we distribute samples uniformly, as no historical information is available:

$$NT^0_{s_i} = \frac{NT^0}{|T|} \tag{1}$$

where NT^0 is the total number of samples to be taken at iteration 0, and $NT^0_{s_i}$ is the resulting number of samples to be assigned to signal s_i. In this case, the same number of samples are drawn from each signal; the initial probabilities p^0_i, representing the importance of the signal, are $p^0_i = \frac{1}{|T|}$.

2) Probability Update: After the first iteration, the algorithm iteratively updates the likelihood that a signal s_i contributes more to precision (or recall), p_i. This probability is based on the proportion of observed *true positives* over the *total number of alarms generated* (or the proportion of observed *true positives* with respect to the *total number of failure-causing DU errors*). The update formulas for *precision* and *recall* scenarios are, respectively:

$$p_i^k = p_i^{k-1} + \frac{(-1+2\cdot TP_i^k/A_i^k)}{\sum_i A_i^k}; \quad p_i^k = p_i^{k-1} + \frac{(-1+2\cdot TP_i^k/E_i^k)}{\sum_i E_i^k} \tag{2}$$

where: p_i^k is the probability assigned to signal s_i at iteration k; p_i^{k-1} is the probability the signal s_i had at the previous iteration; TP_i^k is the number of *true positives* observed for the signal s_i at iteration k (i.e., notifications actually corresponding to failure-causing DU errors); A_i^k are the alarms generated at iteration k; E_i^k are the actual failure-causing DU errors observed at iteration k. The update rule sums up the probability in the previous iteration with an incremental factor. The latter is based on the fraction of relevant notifications of that signal in the current iteration (i.e., true positives over alarms, and true positives over actual errors in the two cases) and it is inversely proportional to the total number of signals considered in that iteration. The numerator is constrained between -1 and 1 to be robust to noisy observations, similarly to [15]. The values of p_i^k are then normalized, since they are probabilities: $p_i^k = (p_i^k)/(\sum_i p_i^k)$. *These probabilities represent the estimate at iteration k of the relative importance of signal s_i.*

3) Importance Sampling Algorithm: The importance sampling algorithm chooses, at each iteration, the number of samples in the next iteration, with the goal of taking more samples from more relevant signals. We use the importance sampling scheme with the KLD variant, which adapts the number of sample in each iteration to the desired error and confidence. The number of samples to generate at iteration $k+1$ is given by [15]:

$$\eta^{k+1} = \frac{1}{2\epsilon}\chi^2_{q-1,1-\delta} \approx \frac{q-1}{2\epsilon}\{1 - \frac{2}{9(q-1)} + \sqrt{\frac{2}{9(q-1)}}z_{1-\delta}\}^3 \tag{3}$$

where: ϵ is the error between the sampling-based estimate and the true distribution that we want to tolerate (in our case $\epsilon = 0.1$); $1-\delta$ is the confidence that we have in this approximation ($\delta = 0.01$); q is the number of singals from which at least one sample has been drawn in iteration k; $z_{1-\delta}$ is the normal distribution evaluated with significance level δ. η^{k+1} is the resulting number of samples to draw in the $(k+1)$-th iteration. Alg. 1 is run at each iteration. It first computes the cumulative probability distribution of signals (suppose m signals). Then it computes the number of samples for the next iteration, and distributes, as output, the samples to take from each signals (denoted as $N_i^{(k+1)}$) proportionally to their relative importance. It is executed until the number of available samples ends. This will progressively shift the sampling towards most relevant signals, considering their performance over iterations. Signals not contributing to improve precision (or recall) will be given less and less samples, saving processing time and increasing the accuracy of failure detection.

The importance sampling procedure. **Inputs**: s_i, $p_i^{(k)}$: $i \in [1,m]$
//sort such that $p_i^k \geq p_{i+1}^{(k)}$
$b_1 = p_1^{(k)}$; *//Initialize Cumulative Distribution*
for i=1 to m
  $N_i^{(k+1)}$=0; *//initialization*
end for
for i=2 to m
  $b_i = b_{i-1} + p_i^{(k)}$; *//Compute Cumulative Distribution*
end for
//Compute $\eta^{(k+1)}$ *according to Eq. 3*
$r_1 \sim U[0, \frac{1}{\eta^{(k+1)}}]$ *//Draw sample from uniform distribution*
//Distribute test cases to each criterion
i = 1;
for j = 1 **to** $\eta^{(k+1)}$
  while $r_j > b_i$ **do** *//Find the bucket to fill*
    $i = i + 1$;
  end while
  $N_i^{(k+1)}$=$N_i^{(k+1)}$ + 1; *//Fill the bucket*
  r_{j+1}= $r_j + \frac{1}{\eta^{(k+1)}}$
end for
//Return re-ordered $\{N_i^{(k+1)}\} : i \in [1,m]$

V. Evaluation

A. Evaluation procedure

We simulate a scenario where the diagnostic system provides samples from a set of signals at a predefined sampling time, assumed to be 1 s. It may happen that:

- a failure-causing DU error occurs, and the associated signal is correctly generated by the diagnostic system (these are the *true positive*s);
- a failure-causing DU error occurs, but the associated signal is not generated (*false negative*s);
- a failure-causing DU error does not occur, and the associated signal is, correctly, not generated (*true negative*s);
- a failure-causing DU error does not occur, and the associated signal is erroneously generated (*false positive*s, also known as false alarms).

The problem we address is how to allow the reasoner focusing only on the most important signals, from two different perspectives: *i)* spot signals with higher *precision* (namely, with many true positives over the total number of alarms generated); *ii)* spot signals with higher *recall* (namely, many true positives with respect to the total number of failure-causing DU errors). These two scenarios lead us to setup an experiment for *precision* and one for *recall*.

1) Exp. 1 - Precision: We consider a list of 100 signals provided by an UAV diagnostic system, ranging from voltage and current signals to temperature and pressure sensor signals, decided jointly with a domain expert at CIRA (Italian Center for Aerospatial Research). Signals are sorted according to their actual "importance". When we consider precision (Exp. 1), we sort signals based on their likelihood of raising few false alarms, assuming for all signals an equal "recall" (in other words: since in this experiment we do not evaluate the ability of signals to raise an alarm every time they should, we assume all signals being equally able to notify an actual problem). The list of signals sorted by precision is referred to as *ground truth* (GT_P) list. Provided that all signals have the same recall, our aim is to identify those with the best precision (with less false alarms).

Suppose that each signal has the same probability of raising an alarm. Then, based on the ground truth list, each signal s is characterized by a "true" probability $P(s)$ that, whenever the signal generates an alarm, it is a true alarm. Thus, $1-P(s)$ is

the probability that, when an alarm is generated, it is a false alarm. The algorithm is required to produce a list of *most important* signals as much similar as possible to the GT_P list.

2) Exp. 2 - Recall: We again sort signals with a domain expert based on their likelihood of not missing real problems (e.g., anomalous events/state), assuming for all the signals an equal "precision", obtaining a ground truth list with respect to recall. The list sorted by recall is referred to as the *ground truth* GT_R list. Provided that all signals have the same precision, our aim is to spot the ones with the best recall.

Suppose that each DU has the same probability of failing. Then, based on the ground truth list, each signal s is characterized by a probability $P'(s)$ that, whenever the unit is in the anomalous state, the signal raises an alarm. Thus $1 - P'(s)$ is the probability that, when a DU fails, the associated signal does not raise any alarm (i.e., a *true negative*). The algorithm is again required to produce a list of *most important* signals as much similar as possible to the GT_R list.

B. Simulation procedure

In both experiments the simulation has the input parameters:

- Number of samples to draw in the first iteration for each signal, K; in the case of Experiment 2, this represents the number of samples to draw from each diagnosable unit. It is set to 15 in our experiment.
- Number of signals to sample, N, set to 100.
- Maximum number of iterations, $Iter$, set to 20.

In the first iteration, K samples are drawn; in Exp. 1, for each sample there is the same probability to generate an alarm; similarly, in Exp. 2, there is the same probability for each DU to fail. Whenever an alarm is generated by a signal s, it is tagged as *true positive* with probability $P(s)$ and as *false positive* with probability $1 - P(s)$ in Exp. 1; it is tagged as *true positive* with probability $P'(s)$ and as *true negative* with probability $1 - P'(s)$ in Exp. 2. At the end of the iteration the real number of true positives is obtained in both experiments. Based on these values, the "importance" for each signal is computed according to Equ. 2, so as to maximize precision (recall). After the first iteration, the importance sampling algorithm determines how many samples are needed in each subsequent iteration, and how many of these should be taken for each signal. The simulation continues until the maximum iteration count is reached.

C. Metrics

During the simulation, we evaluate, *at each iteration*:

- *precision*: number of *true positive* over total number of alarms (*true positives+false positives*), for Exp. 1;
- *recall*: number of *true positive* over total number of failure-causing DU errors (*true positives+false negatives*), for Exp. 2;
- how many samples are required to determine which are the most "important" signals (with an error less than the specified ξ, and a confidence higher than $(1 - \delta)\%$);
- the probability assigned at each signal by Equ. 2 (i.e., the importance of the signal at that iteration).

At the end of the simulation, we derive the *Overlap(GT,IS, T%)* metric computed in this way: consider the list of the actual most important signals (namely, the GT list) and the list of the most important signals as suggested by the importance sampling algorithm – i.e., the IS list. The overlap between GT and IS at *T%* is the fraction of the top *T%* signals of the GT list that is also present in the top *T%* if the IS list.

D. Results

Figures 1 and 2 show the precision and recall over iterations in Experiments 1 and 2, respectively. As the number of iterations grows, the precision increases, getting to 0.8 after 4 iterations; the recall increases with a similar trend in Exp. 2, also getting to 0.8 after 4 iterations. Initially, samples are taken uniformly from all signals. Over the iterations, more samples are taken from more important signals, while fewer samples are taken from the others, remarkably reducing the number of false alarms (precision), as well as the number of undetected DU failures (recall).

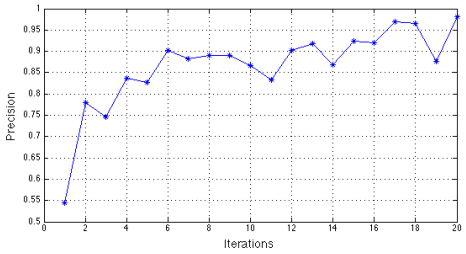

Fig. 1: Precision over iterations

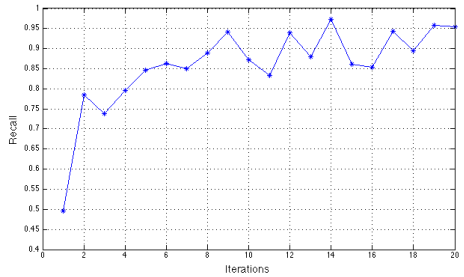

Fig. 2: Recall over iterations

It is important to notice that the increase of precision and recall is obtained by taking less and less samples. Figures 3 and 4 depict the number of samples drawn at each iteration in both experiments. At the first iteration, 15 samples per signal are taken (hence 1500); at subsequent iterations, the IS algorithm suggests to sample less and less samples; at iteration 4, where precision and recall are permanently over 0.8, the number of samples is permanently under 270 in both cases. In other words, *higher precision or recall are obtained despite lower and lower samples are taken form the diagnostic system*.

Figures 5 and 6 show how the importance attributed to each signal varies over iterations. At the first iteration, all signals have the same probability of being important (namely, P= 1/100 = 0.01). The graphs show that the focus is progressively shifted on fewer and fewer signals, deemed to be more important. At iteration 20, *samples will be taken mainly from 11*

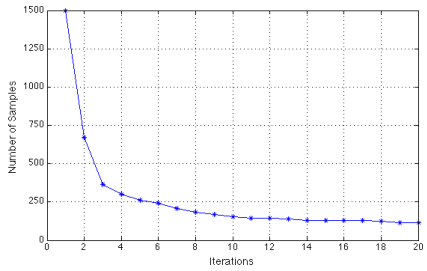

Fig. 3: Number samples required over iterations: Exp. 1 (precision)

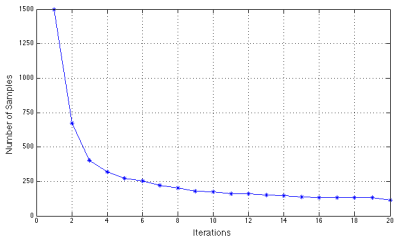

Fig. 4: Number samples required over iterations: Exp. 2 (recall)

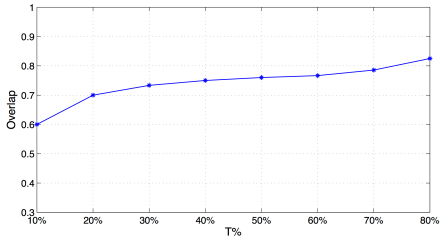

Fig. 7: Overlap: fraction of signals of top T% of the GT list also present in the top T% of the IS list in Exp. 1 (precision)

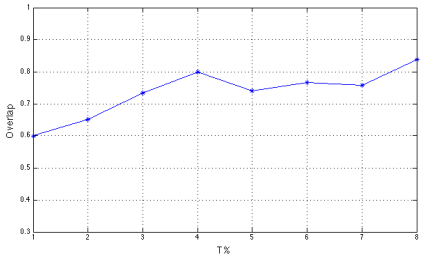

Fig. 8: Overlap: fraction of signals of top T% of the GT list also present in the top T% of the IS list in Exp. 2 (recall)

signals for precision, and 9 for recall, proportionally to their estimated importance. The decrease of the number of total samples indicates that there is progressively less ambiguity about which signals are more important.

Finally, Figures 7 and 8 report the overlap metric, with varying values of T. They show that a fraction between 0.6 and 0.85 of the top T% of GT signals are present also in the suggested IS list, for both Experiments 1 and 2. Interestingly, this is roughly invariable with the increase of T: it means that even when we consider only the top 10% of signals of the GT list, hence 10 signals in our case, we find 6 of these 10 signals in the IS list; similarly, if we consider the most important 90 signals, we find that 70 of these are also suggested by the IS list. High overlap with both small and large value of T indicates that the algorithm not only can spot most of the important signals, but also identify them almost in the same order they appear in the GT list.

VI. Conclusion

We presented an algorithm to reduce the amount of relevant information that a diagnostic system is required to analyze, while contemporary improving the precision and recall of the failure detection means. Preliminary simulations demonstrate the potential of this technique to spot the most important source of information for effective and efficient health management systems. While we have considered precision and recall as separate objectives, Equ. 2 can be unified under some combined metrics (e.g., *F-measure*, or *Balance*) so as to obtain the most "important" signals considering suitable combinations of precision and recall.

Acknowledgment

This work has been supported by the MINIMINDS PON Project (n. B21C12000710005) of the public private laboratory 'COSMIC' (PON02_00669), funded by the Italian MIUR (Ministero dell'Istruzione, dell'Universita' e della Ricerca).

References

[1] J. Schumann, K.Y. Rozier, T. Reinbacher, T. Mbaya, C. Ippolito, "Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems". In Proc. of the Annual Conf. of the Prognostics and Health Management Society, 2013.

[2] G. Gigante, F, Gargiulo, and M. Ficco, "A semantic driven approach for requirements verification". In Studies in Computational Intelligence, vol. 570, 2015, pp. 427-436.

[3] G. Zazzaro, G. Gigante, E. Zaccariello, M. Ficco, and B. Di Martino, "Supporting Development of Certified Aeronautical Components by applying Text Analysis Techniques". In Proc. of the 8th Int. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS 2014), 2014, pp. 602-607.

[4] G. Gigante, F. Gargiulo, M. Ficco, and D. Pascarella, "A Semantic Driven Approach for Consistency Verification between Requirements and FMEA". In Proc. of the 9th Int. Symp. on Intelligent Distributed Computing, 2015, pp. 1-10.

[5] T. Reinbacher, K.Y. Rozier, and J. Schumann, "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems". In Tools and Algorithms for the Construction and Analysis of Systems, LNCS, 8413, 2014, pp. 357-372

[6] B.W. Ricks and O.J. Mengshoel, "The diagnostic challenge competition: Probabilistic techniques for fault diagnosis in electrical power systems". In Proc. of the 20th Int. Work. on Principles of Diagnosis, pp. 415-422.

[7] A. Bovenzi, D. Cotroneo, R. Pietrantuono, G. Carrozza, "Error detection framework for complex software systems". In Proc. of the 13th European Work. on Dependable Computing, 2011, pp. 61-66.

[8] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods". In ACM Comput. Surv., vol. 42, no. 3, pp. 42.

[9] R. Pietrantuono, S. Russo, K.S. Trivedi, "Online Monitoring of Software System Reliability". In Proc. of the European Dependable Computing Conference (EDCC), 2010, pp. 209-218.

[10] I.G. Ferreira, C. Laorden, I. Santos, P.G. Bringas, "A Survey on Static Analysis and Model Checking". In Proc. of the Int. Joint Conf. SOCO'14-CISIS'14-ICEUTE'14, Advances in Intelligent Systems and Computing, vol. 299, 2014, pp. 443-452.

[11] D. Cotroneo, R. Pietrantuono, and S.Russo, "Combining operational and debug testing for improving reliability". In IEEE Transactions on Reliability, vol. 62, no. 2, 2013, pp. 408-423.

[12] P. Baker, Z.R. Dai, J. Grabowski, O. Haugen, I. Schiefer-decker, C. and Williams, "Model-Driven Testing: Using the UML Testing Profile". Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2011.

[13] R. Pietrantuono, S. Russo, and K. Trivedi, "Software reliability and testing time allocation: An architecture-based approach". In IEEE Transactions on Software Engineering, vol. 36, no. 3, 2010, pp. 323-337.

[14] R. Pietrantuono, S. Russo, J. Alonso, K.S. Trivedi, "Emulating Environment-Dependent Software Faults". In Proc. of the 1st Int. Work. on Complex faults and Failures in Large Software Systems, 2015.

[15] M. Sridharan and A.S. Namin, "Prioritizing mutation operators based on probabilistic sampling". In Proc. of ISSRE, 2010, pp. 1-4.

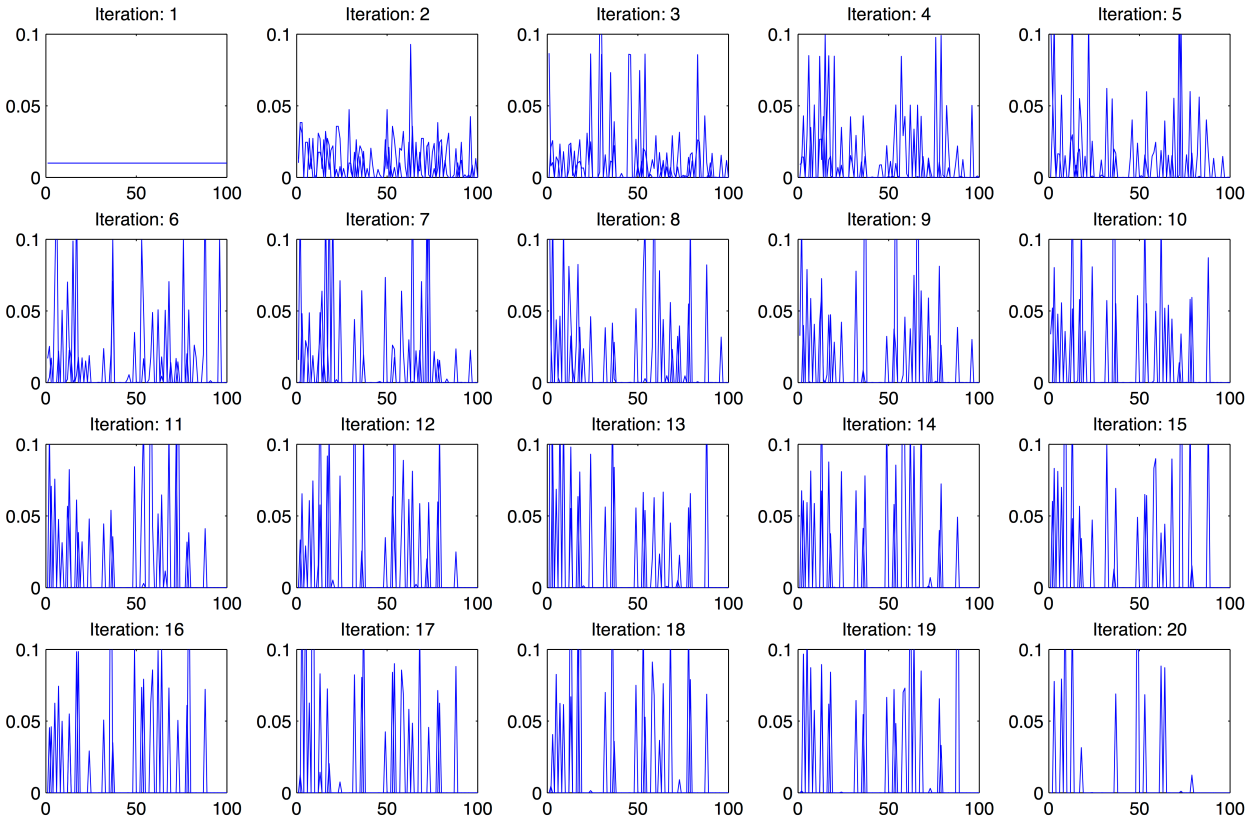

Fig. 5: Number of samples required over iterations: Exp.1 (precision)

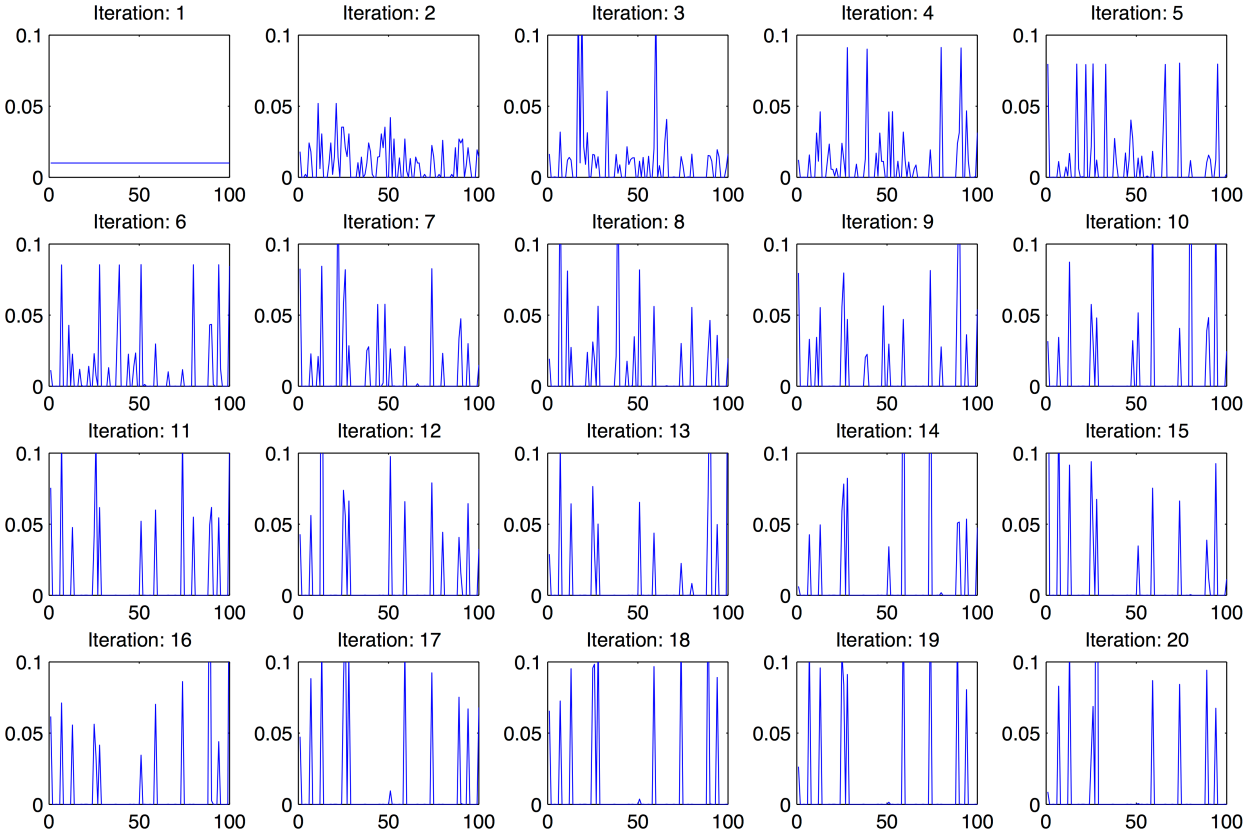

Fig. 6: Number of samples required over iterations: Exp. 2 (recall)