Improving the Stop-Test Decision When Testing Data are Slow to Converge

Mahnaz Malekzadeh[1] and Iain Bate[1,2]

[1]Mälardalen Real-Time Research Centre, Mälardalen University, Västerås, Sweden
[2]Department of Computer Science, University of York, York, UK
email: mahnaz.malekzadeh@mdh.se and iain.bate@york.ac.uk

Abstract. Testing of safety-critical systems is an important and costly endeavor. To date work has been mainly focusing on the design and application of diverse testing strategies. However, they have left the important decision of "when to stop testing" as an open research issue. In our previous work, we proposed a convergence algorithm that informs the tester when it is concluded that testing for longer will not reveal sufficiently important new findings, hence, should be stopped. The stop-test decision proposed by the algorithm was in the context of testing the worst-case timing characteristics of a system and was evaluated based on the *As Low As Reasonably Practicable (ALARP)* principle. The ALARP principle is an underpinning concept in many safety standards which is a cost-benefit argument. ALARP implies that a tolerable risk should be reduced to a point at which further risk-reduction is grossly disproportionate compared to the benefit attained. An ALARP stop-test decision means that the cost associated with further testing, after the algorithm stops, does not justify the benefit, i.e., any further increased in the observed worst-case timing.
In order to make a stop-test decision, the convergence algorithm used the *Kullback-Leibler DIVergence (KL DIV)* statistical test and was shown to be successful while being applied on system's tasks having similar characteristics. However, there were some experiments in which the stop-test decision did not comply to the ALARP principle, i.e., it stopped sooner than expected by the ALARP criteria. Therefore, in this paper, we investigate whether the performance of the algorithm could be improved in such experiments focusing on the KL DIV test. More specifically, we firstly determine which features of KL DIV could adversely affect the algorithm performance. Secondly, we investigate whether another statistical test, i.e., the *Earth Mover's Distance (EMD)*, could potentially cover weaknesses of KL DIV. Finally, we experimentally evaluate our hypothesis of whether EMD does improve the algorithm where KL DIV has shown to not perform as expected.

1 Introduction

Safety-critical systems are those in which failure can lead to loss of people's lives or catastrophic damage to the environment. Timeliness is an important concept

in a safety-critical system and relates to the notion of *response time*, which is the time a system takes to respond to stimuli from the environment. If the response time exceeds a specific time, called *deadline*, a catastrophe might occur. Testing is an important process in a safety-critical system and can be used to assure that timing requirements are met. However, it is also one of the most expensive parts. Therefore, a key issue for testers is to determine when to stop testing as stopping too early may result in defects remaining in the system, or a catastrophe due to high severity level of undiscovered defects; and stopping too late will result in waste of time and resources. Researchers and practitioners, so far, have mainly focused on the design and application of diverse testing strategies, leaving the stop-test decision largely an open research issue.

Our previous work [1], [2], [3], was about developing a convergence algorithm to make a stop-test decision in the context of testing the worst-case timing characteristics of systems. The algorithm informs the tester whether further testing would reveal significant new information about timing behaviour; and if not, suggests testing to be stopped. In order to make such a decision, the algorithm looks into the testing data, i.e., response times of system's tasks, and (i) determines whether the *Maximum Observed Response Time (MORT)* has recently increased and, when this is no longer the case, (ii) it investigates if the distribution of response times has significantly changed. When no significant new information about the system is revealed during a given period of time, it is concluded, with some statistical confidence, that more testing of the *same nature* is not going to result in significant new insight. However, some other testing techniques may still achieve significant new findings.

The evaluation of the algorithm was based on the *As Low As Reasonably Practicable (ALARP)* principle which is an important concept in many safety standards, and involves weighting benefit against the associated cost [4]. We showed that, by further testing, the sacrifice would be grossly disproportionate compared to the benefits attained, i.e., any significant increase in the observed MORT, after stopping the test, needs a disproportionate amount of testing.

Our evaluations based on simulation showed that the convergence algorithm was successful while being applied across task sets having similar characteristics, e.g., all trials consisted of 10 tasks with task set periods ranging from 200 μs to 400 μs. However, there were some experiments in which the algorithm could not suggest a proper stop-test decision according to ALARP. In order to deal with such situations, in this paper, we investigate whether the algorithm performance could be improved focusing on the statistical test used in the algorithm. The algorithm suggests testers to stop testing when it gets statistically confidence that further testing is not going to reveal significant new findings using the *Kullback-Leibler DIVergence (KL DIV)* [5], [6] statistical test. In this paper, firstly, we determine features of the KL DIV test that may adversely affect the algorithm. Secondly, we look into a similar test, called *Earth Mover's Distance (EMD)*, which seems to cover some weaknesses of KL DIV, thus, improving the convergence algorithm. Finally, we experimentally investigate whether EMD does improve the algorithm compared to KL DIV.

The remainder of this paper is structured as follows. Section 2 describes the related work. The convergence algorithm and the simulation environment in which we run our experiments are stated in 3. The evaluation criteria is presented in Section 4 followed by a comparison of KL DIV and EMD statistical tests in Section 5. The experimental results are shown in Section 6. Section 7 finally states the conclusions and future work.

2 Related Work

There are stringent timing requirements imposed on safety-critical systems which make testing an important and necessary process with which not only the correct system functionality must be tested, but also the timing characteristics.

Testing of a safety-critical system involves reliable timing analysis in the form of the *Worst-Case Execution Time (WCET)* estimation and *Response Time Analysis (RTA)* to assure that tasks in the system meet their timing requirements. However, there are variations in software execution times, caused by some software and hardware characteristics, which make timing analysis a complex and difficult process. Sources such as varying input sets to the software, the software layout in the memory for the code and data may cause execution times variation on the software side. Whereas, features that improve the average performance by exploiting properties of execution history, e.g., caches and pipelines, may cause execution times variations on the hardware side. If all characteristics of software and the underlying hardware are thoroughly understood, accurate WCET estimation would be possible. However, the dependence of hardware and software timing characteristics on the history of execution is one of the main factors that makes the cost of acquiring knowledge needed to perform timing analysis challenging.

WCET estimation is carried out using two main approaches: *Deterministic Timing Analysis (DTA)* and *Probabilistic Timing Analysis (PTA)*. DTA and PTA approaches are performed through two categories: static and measurement-based methods.

2.1 Static Timing Analysis (STA)

Conventional static timing analysis techniques require the system under analysis to be *time deterministic*. In a time deterministic system, for a specified input set, the sequence of events that will occur is completely known, as well as the time after which the output of an event will be generated. This analysis depends on a very detailed model of the system as well as accurate information of dependence between significant events. The growing complexity in safety-critical systems, at hardware and software level, makes the extent of required knowledge, time, effort and cost to acquire and understand all the relevant details, unaffordable. Therefore, as long as current analysis techniques are incapable of covering challenges within increased hardware complexity, there will be a significant degradation in the quality of the resulting products.

To sum up, industry demands higher level of performance with reduced cost and power dissipation which can be provided by advanced hardware features. However, complex hardware features are difficult to deal with by DTA techniques as the amount of required information is becoming challenging.

2.2 Probabilistic Timing Analysis (PTA)

The cost of acquiring knowledge to carry out correct timing analysis can be significantly reduced by a hardware/software architecture in which the execution time behaviour does not depend on execution history [7]. This can be achieved by introducing *randomisation* in the timing behaviour of hardware and software while functional behaviour is kept unchanged which is coupled with new PTA techniques.

Static Probabilistic Timing Analysis (SPTA) [7] statically derives a-priori probabilities from a model of the system. Tight WCET estimates made with SPTA have less dependence on complete knowledge than conventional DTA methods, and lack of information in analysis has less impact on the WCET estimation. In fact, WCET estimates provided by SPTA react much more smoothly to the lack of knowledge, with a gradual shift towards a maximum value as knowledge is reduced. Although the amount of information required about the hardware/software under analysis is reduced, SPTA stills needs a lot of information about the internal behaviour of the architectural components.

Measurement-Based Probabilistic Timing Analysis (MBPTA) requires much less information rather than SPTA which makes it more attractive for industry. MBPTA derives probabilities from end-to-end runs of the software under analysis on the target platform. These runs provide data about the timing behaviour of the software when executing on the proposed hardware with randomised timing behaviour. MBPTA derives probabilities by intensively stress testing the real system through high-coverage input data, then, recording the longest execution time, called *High WaterMark (HWM)*; and adding to it an engineering margin to make safety allowances for the unknown.

A MBPTA technique based on the *Extreme Value Theory (EVT)* [8] is proposed by Cucu-Grosjean et al. [9] which addresses the problems with the way EVT is applied [10], [11]; and derives probabilistic WCET (pWCET) estimates. The authors collect execution times achieved by end-to-end runs of the system under analysis on the target platform. In order to apply EVT, they collect from the original distribution, the values which fit into the tail while discarding the rest of observations. Then, by means of a convergence algorithm, they assure whether enough number of observations is gained to obtain statistical significance; and for EVT to provide high quality pWCET estimates. Their convergence algorithm compares tails of two distributions depicted by $N_{current}$ and $N_{current} + N_{delta}$ where N determines the number of runs and decides to stop observing WCETs if the difference between the distributions' tail becomes below a given threshold for some specified consecutive iterations.

The work of Cucu-Grosjean et al. [9] is similar to ours in the sense that both are based on a convergence algorithm to decide whether enough observations

from testing have been gained to predict future timing behaviour of the system. However, the two methods involve the following differences:

- Our observations relate to the *response times* of tasks in the system without knowledge of the exact WCETs, thus, is based on a *black-box* approach. While their method is based on the *WCETs* obtained from end-to-end runs of the software using a *white-box* technique.
- To decide the sufficiency of observations, our method looks into the *whole distribution* of testing data while theirs just considers the *distribution tail*. In our view, it would not be sufficient to look into the distribution tail as it might happen that while there is no significant change in the distribution tail, the rest of the distribution is still significantly changing which indicates that there is a high chance of observing new and important information by further testing. On the other hand, if the whole distribution is converged, future timing behaviour is not expected to significantly change and even if there would be such a change, its cost is not justified.
- In the work by Cucu-Grosjean et al. [9], the correct use of EVT imposes that observed execution times can be described by *independent and identically distributed (i.i.d.)* random variables. This also places strong requirements on the processor hardware and how end-to-end execution times are taken such that they can be modeled by *i.i.d.* random variables. Our work is neither based on EVT nor assumes hardware dependencies. However, hardware dependencies are a part of our future work.

3 Convergence Algorithm

This section, firstly, describes the system simulator model on which our analysis is based. Further, it describes our proposed convergence algorithm to decide when to stop testing for worst-case timing characteristics of systems.

3.1 Task Set Simulator

Testing data, in this work, is generated using an in-house task set simulator which is a simplified version of what was originally used for the ball and beam example in [12]. The overall scheme of the simulator is shown in Figure 1. The simulator generates a set of tasks and executes them for a specified duration (*SimDur*). No task dependency, overhead and context switch time is considered in the simulator. The simulator neither involves the underlying hardware characteristics. Tasks are prioritized based on the *Deadline Monotonic Priority Ordering (DMPO)*, i.e., the shorter a task's deadline, the highest priority it takes. Periods are randomly generated and each task deadline is equal to its period. Each task is assigned a random execution time in the range [*Best-Case Execution Time (BCET)*, WCET]. The BCET and WCET are the shortest and the longest execution times of a task respectively. The scheduler monitors each task's status: *delayed*, *in run queue* or *executed* and performs preemptive *Deadline Monotonic Scheduling Theory (DMST)* based on tasks' fixed priorities, i.e.,

the task with the highest priority is executed first. The simulator also generates static WCRT of the tasks.

The reason to use the simulator is that, in this way, we have careful control over the task set characteristics and a *ground truth* for evaluations is established. Two ground truths are provided by the simulator: (i) WCRT achieved by static analysis which provides an exact safe result, and (ii) a HWM achieved by significantly longer simulation. Longer simulation is possible due to the nature of the simulator used. However, such increased testing would be prohibitively expensive in a real system.

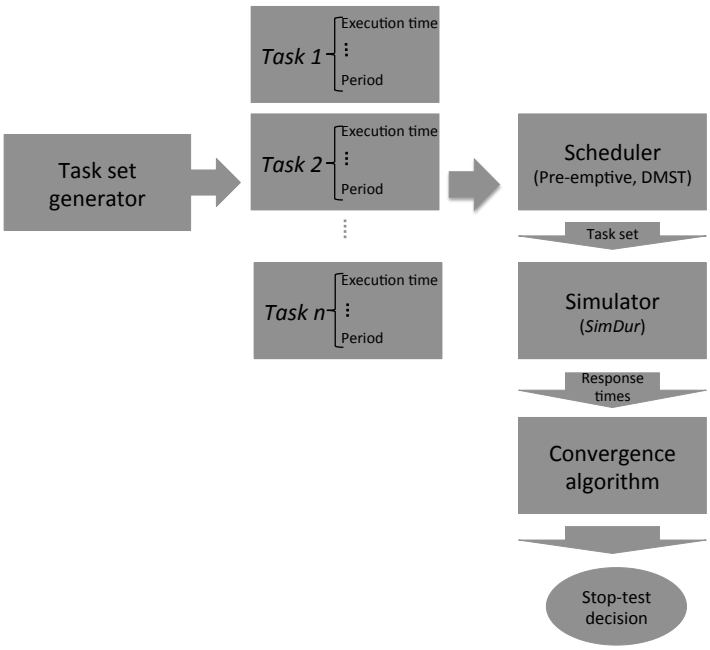

Fig. 1: Overall scheme of task set simulator

3.2 The Convergence Algorithm

As stated earlier in 1, in order to decide when to stop testing, we proposed a convergence algorithm in [1] which was further developed it in [2], [3]. The algorithm makes statistical analysis on the observed response times to determine whether the *Maximum Observed Response Time (MORT)* has recently increased and the distribution of response times has significantly changed using the following tests respectively:

– *High WaterMark (HWM)* which stands for the MORT during testing and determines whether the MORT has recently increased. In order to run the

HWM test, the algorithm compares HWMs corresponding to two successive response times distributions, $T_{current}$ and $T_{current} + T_{\Delta}$, over a couple of iterations. If the HWM does not increase for a specific number of consecutive iterations, the algorithm analyses the distributions in more details using a statistical test. The HWM test increases the scalability of the convergence algorithm by reducing the number of statistical tests that are needed.

– When the MORT values have no increase for a specific time, i.e., the HWM test is passed, the algorithm applies the KL DIV test to examine whether the distribution of response times has significantly changed. The KL DIV test is applied on $T_{current}$ and $T_{current} + T_{\Delta}$ histograms and the result shows the difference between these two. If the test result falls below a specified threshold at a required confidence level, it is concluded that $T_{current}$ and $T_{current} + T_{\Delta}$ have converged. The convergence indicates that no significant new information has been observed, thus, future timing behaviour is not going to be significantly different. At this point, the convergence algorithm proposes to stop testing.

Figure 2 shows the steps within the convergence algorithm along with its parameters. The convergence algorithm takes each task's response times (testing

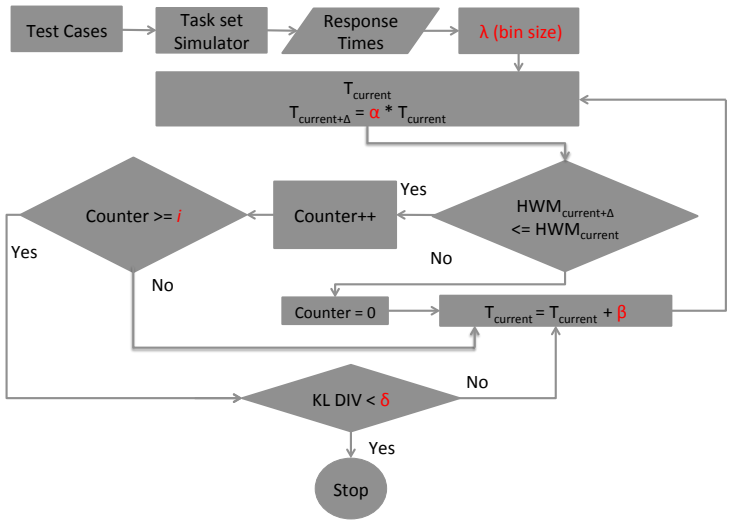

Fig. 2: Convergence algorithm

data) as input, analyses them and proposes a *StopPoint*. In order to improve the scalability of our method, a parameter called λ is introduced that keeps track of frequencies of response times falling between time t and $t + \lambda$. This process is called *binning* where λ defines the *bin size*. Consequently, the algorithm compares histograms of binned response times rather than distributions of indi-

vidual values which saves us memory space (Figure 3). In each round of analysis,

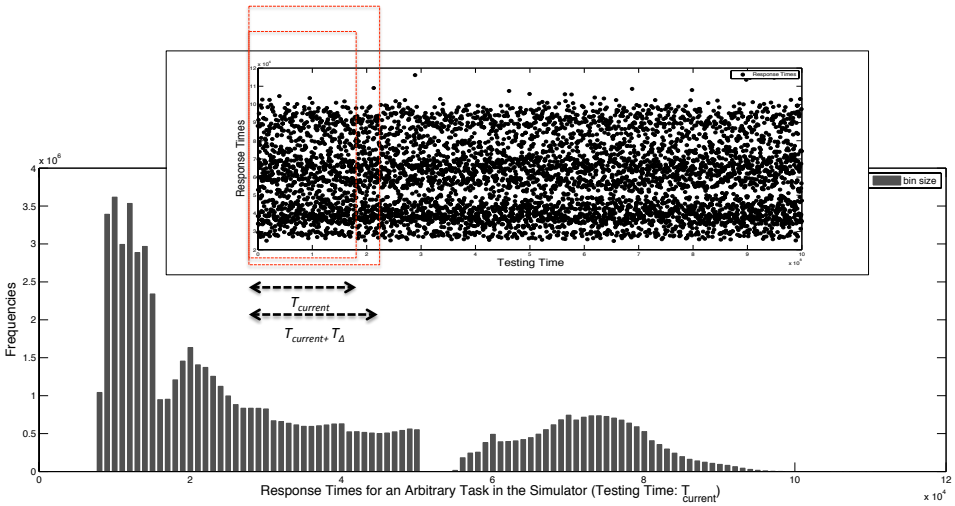

Fig. 3: Binned vs. not binned response times

two histograms $T_{current}$ and $T_{current} + T_{\Delta}$ are compared where $T_{current} + T_{\Delta}$ is equal to α * $T_{current}$ and α forms another parameter in the algorithm. At the next step, the HWM test is applied on $T_{current}$ and $T_{current} + T_{\Delta}$ and if there has been no increase in the HWM for i successive iteration, then, the KL DIV test would be applied. If the KL DIV test's result falls below a threshold, depicted by δ, the algorithm concludes the distributions of response times have converged, thus, suggests to stop testing. i and δ are the other two parameters of the convergence algorithm. If either tests is not passed, $T_{current}$ is increased by β μs and the next analysis round starts. These steps are taken for each task in the simulator and the algorithm would stop when the latest task in the task set passes the HWM and KL DIV tests.

4 Evaluation

As stated earlier, the algorithm is designed and evaluated with a particular focus on the *As Low As Reasonably Practicable (ALARP)* principle which involves weighting benefit against the associated cost [4]. In order to evaluate the convergence algorithm, it is shown that the sacrifice would be grossly disproportionate compared to the benefits attained, i.e., any further significant increase in the observed worst-case timing, after stopping the test, needs a disproportionate amount of testing time.

Three criteria for evaluations are defined as follows.

– The first criterion relates to the closeness of the HWM when the algorithm suggests testing to be stopped (depicted by *HWM at StopPoint*) to the last observed HWM, i.e., the ground truth (depicted by $GTHWM$), and is formulated as follows.

$$\frac{GTHWM - HWMatStopPoint}{GTHWM} \tag{1}$$

The smaller the equation result, the better performance in terms of MORT is achieved at *StopPoint*.

– The second criterion relates to testing effort at *StopPoint* relative to testing effort at GTHWM which is as follows.

$$\frac{TestingEffortatStopPoint}{TestingEffortatGTHWM} \tag{2}$$

Similar to Equation 1, smaller result indicates better performance of the algorithm in terms of testing effort.
The first and second criteria try to compare the gain at *StopPoint* (closeness of the HWM at *StopPoint* to GTHWM) with the associated cost, i.e., cost-benefit analysis. It is ideal to get as close as possible to GTHWM with as low as possible cost when we stop testing.

– A quantified MORT, here called $ALARPHWM$, is also defined so that if the HWM at *StopPoint* falls between $ALARPHWM$ and $GTHWM$, it is considered as an acceptable performance. As all MORT values in the range [$ALARPHWM$, $GTHWM$] are acceptable in terms of benefit, the *StopPoint* with less effort is more desirable. The $ALARPHWM$ is defined as follows.

$$ALARPHWM = GTHWM - \xi\% * GTHWM \tag{3}$$

The value of ξ can be set according to requirements which is 5% in our evaluations. Consequently, the third criteria is defined as in Equation 4.

$$\frac{HWMatStopPoint - ALARPHWM}{HWMatStopPoint} \tag{4}$$

These criteria help to evaluate the performance of the algorithm in terms of benefit at *StopPoint* (closeness to the GTHWM) against the associated cost (testing time) whilst the least acceptable benefit has to be met (ALARPHWM).

5 The Convergence Algorithm Statistical Test

This section starts with a motivational example of implementing the convergence algorithm with a new statistical test called EMD. Further, KL DIV and EMD are compared to explain how EMD could improve the algorithm performance where KL DIV is not successful.

5.1 Motivational Example

The convergence algorithm introduced in Section 3 has shown to be successful so far, i.e., it makes a stop-test decision in compliance to the ALARP criteria introduced in Section 4. Figure 4 shows multiple algorithm's suggested *StopPoint*s and shows how they conforms to our ALARP criteria. For example, the algorithm gets the same benefit in terms of MORT at *StopPoint1* and *StopPoint2* according to Equation 1. However, testing effort at *StopPoint1* is less than *StopPoint2*. So, it is concluded that *StopPoint1* results in better performance than *StopPoint2*. At *StopPoint3*, both HWM and testing effort are less then *StopPoint1* which means that the algorithm has better performance in terms of cost and poorer performance in terms benefit at *StopPoint3* rather than *StopPoint1*. However, as all MORT values in the range $[ALARPHWM, GTHWM]$ are acceptable in terms of benefit (Equation 3), *StopPoint3* with less effort is more desirable.

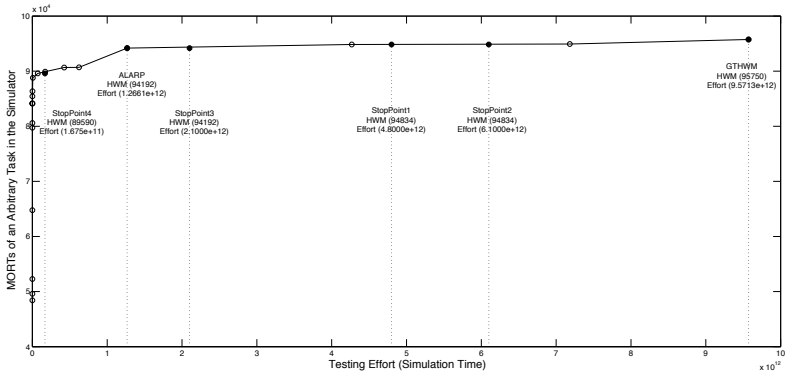

Fig. 4: Evaluation of the convergence algorithm

Figure 4 also shows a situation when the algorithm stop-test decision occurs before ALARPHWM which is not desirable (*StopPoint4*). It can be also seen that the HWM is going to increase soon after *StopPoint4*, i.e., the algorithm *StopPoint* still can be improved, in terms of the observed HWM, with a reasonable amount of further testing. In order to deal with such a situation, we try to investigate whether any feature of KL DIV could have a potential impact on the algorithm and if such features are improved in another statistical test.

5.2 Earth Mover's Distance (EMD)

There were some experiments in which the convergence algorithm based on KL DIV did not result in a desirable stop-test decision. Therefore, we focus on KL DIV features to determine whether any of them could affect the algorithm and if another test can do better. One of the issues with the KL DIV test is that

large quantities of testing data can affect the test results, i.e, subtle changes are harder to be observed with lots of data which might be an issue in our algorithm as it also deals with large amount of WCRTs. Therefore, we are interested in a statistical test which is more appropriate with respect to the large amount of testing data.

Moreover, the KL DIV test measures *bin-by-bin dissimilarity* which means only pairs of bins in two histograms that have the same index are matched [13]. In another word, the dissimilarity between two histograms is a combination of all the pairwise differences. A ground distance is only implicitly used by these measures and in an extreme form, i.e., features that fall into the same bin are close enough to each other to be considered the same, while those that do not are too far apart to be considered similar. In this sense, bin-by-bin measures imply a binary ground distance with a threshold depending on bin size. The major drawback of such a measure is that it only accounts for the correspondence between bins with the same index, and does not use information across bins.

Cross-bin dissimilarity measures, on the other hand, use the information across bins as well and deal better with large amount of data, e.g., the *Earth Mover's Distance (EMD)* test. Such a measure may improve our convergence algorithm performance, i.e., as stated in 3.2, the algorithm compares two successive histograms $T_{current}$ and $T_{current} + T_{\Delta}$ while a big part of these two histograms intersects and there is subtle changes observed in $T_{current} + T_{\Delta}$ compared to $T_{current}$, e.g., the zoomed area in Figure 5 shows how delicate are the changes between the red bars ($T_{current}$) and the black bars ($T_{current} + T_{\Delta}$). Thus, the more sensitive the algorithm to the extra observed testing data in $T_{current} + T_{\Delta}$, the more accurate dissimilarity between histograms could be measured.

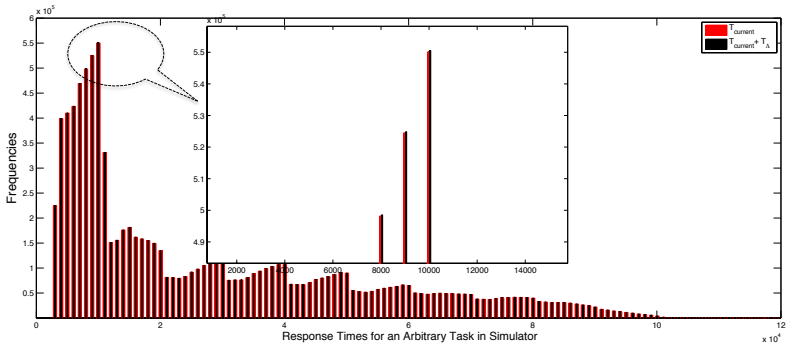

Fig. 5: Two comparing histograms in the convergence algorithm

Therefore, the EMD test hypothetically seems to be a good substitute for the KL DIV test in our algorithm and is described as follows: given two distributions, one can be seen as a mass of earth properly spread in space and the other as a collection of holes in that same space. Then, the EMD measures the least amount

of work needed to fill the holes with earth where a unit of work corresponds to transporting a unit of earth by a unit of ground distance [13].

6 Experimental Results

This section describes the experimental framework used to run the simulation. Then, the experimental results are presented that compare two versions of the algorithm based on: (i) the KL DIV test, and (ii) the EMD test.

6.1 The Experimental FrameWork

The evaluation phase is based on 20 trials, each consisting of 10 tasks, in the simulator. The simulator runs for $SimDur = 10000000$ sec in each trial and generates the following task set characteristics according to pseudo code presented in Algorithm 1 (All timing in microsecond unless otherwise stated):

- Non-harmonic *periods* which are randomly generated within the input domain [50000, 200000] for tasks which are not part of the control software,
- Execution times which are randomly generated between the BCET and the WCET where BCET and WCET are calculated as shown in Algorithm 1, so that the overall task set utilisation is within [80%, 100%],
- *Deadlines* which are equal to periods,
- *Priorities* which are assigned based on DMPO.

Algorithm 1: The *Task Set Generation Pseudo Code*

Input: $NumberOfTasks, MinUTIL, MaxUTIL, UTILStep, MinPeriod, MaxPeriod, PeriodStep, MaxBCET, BCETStep$

Output: $TaskSetCharacteristics$

```
1  NumberOfTasks = 10;
2  foreach Task ∈ {TaskSet} do
3  |  TaskPeriod ← Rand(MinPeriod, MaxPeriod, PeriodStep);
4  |  TaskDeadline ← TaskPeriod;
5  |  TaskBCET ← Rand(1, MaxBCET, BCETStep);
6  |  TaskSetUTIL ← Rand(MinUTIL, MaxUTIL, UTILStep);
7  |  TaskUTIL ← UUniFast(TaskSetUTIL);
8  |  WCET ← TaskPeriod * TaskUTIL;
9  |  TaskPriority ← DMPO(TaskSet);
10 end
```

The *Rand* function in Algorithm **??** is initialized by a random seed and returns a value in the range [MinValue, MaxValue] with coefficient *ValueStep* in which *Value* can stand for period, BCET or UTIL. UTIL corresponds to the utilisation and the UUniFast algorithm generates each task's utilisation with uniform distributions [14]. The *DMPO* function returns each task priority based on the DMPO.

6.2 Results

Results in this section correspond to the comparison of the KL DIV with the EMD test in terms of performance. Evaluation of these two tests is based on the benefit and cost metrics achieved from Equation 1 and 2 respectively.

Figure 6 shows the performance of the convergence algorithm using KL DIV vs. EMD test. The horizontal axis corresponds to the KL DIV and EMD test results for different tasks within the task set while each axis tick shows the task with its priority, e.g., KLDIV1 shows the KL DIV result for the highest priority task. The convergence algorithm parameters had the same tunings within both tests which are as follows: $\{\alpha = 2,\ i = 10,\ \delta = 0.001,\ \lambda = 200,\ \beta = 1250$ sec$\}$. It can bee seen that for the first two highest priority tasks, KL DIV and EMD get almost similar performance in terms of benefit while from the third task, as the task priority decreases, the KL DIV results in poorer benefit rather than the EMD test. However, the EMD test is passed in fewer cases rather than the EMD. It could be argued that EMD is more sensitive to subtle changes in distributions compared to KL DIV. Higher priority tasks tend to converge soon in simulation, thus, result in KL DIV and EMD having similar performance. Lower priority tasks, however, tend to converge later, thus, EMD results in either late or no convergence rather than KL DIV. The tests performance in terms of cost is quite similar as shown in 6 (bottom graph).

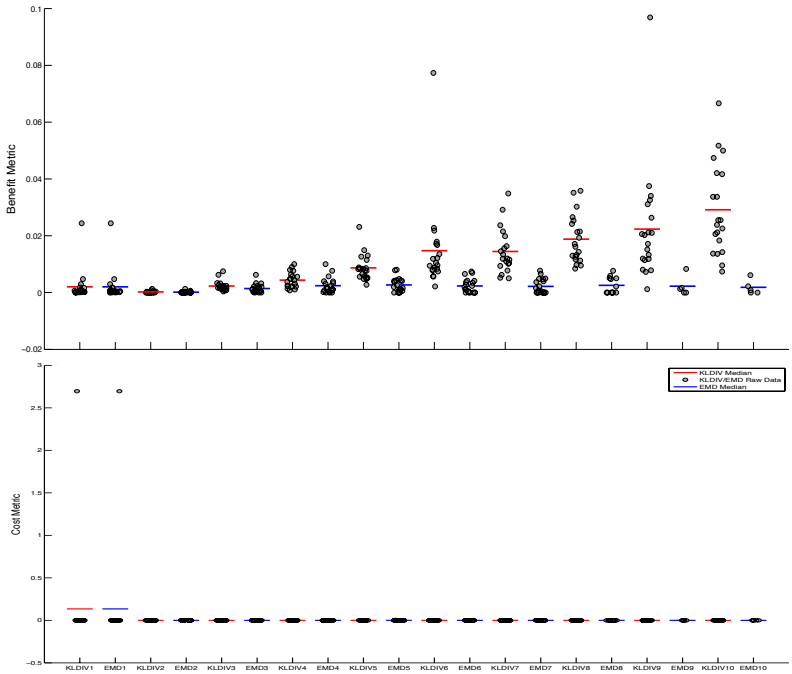

Fig. 6: KL DIV vs. EMD for trials of 10 tasks

From Figure 6, it can be concluded that with a choice of small δ, EMD converges later than KL DIV as the task priority decreases. This might also help the situations where KL DIV stops sooner than expected according to the ALARP criteria. Figure 7 shows such a situation, for a task with low priority, where *StopPointKLDIV* and *StopPointEMD* show the algorithm *StopPoint* for KLDIV and EMD respectively. The parameter ξ in Equation 3 is set to 1%. It can be seen that KL DIV stops sooner than *ALARP* while EMD stops later spending a reasonable amount of further testing time, thus, it improves the performance of the algorithm compared to KL DIV.

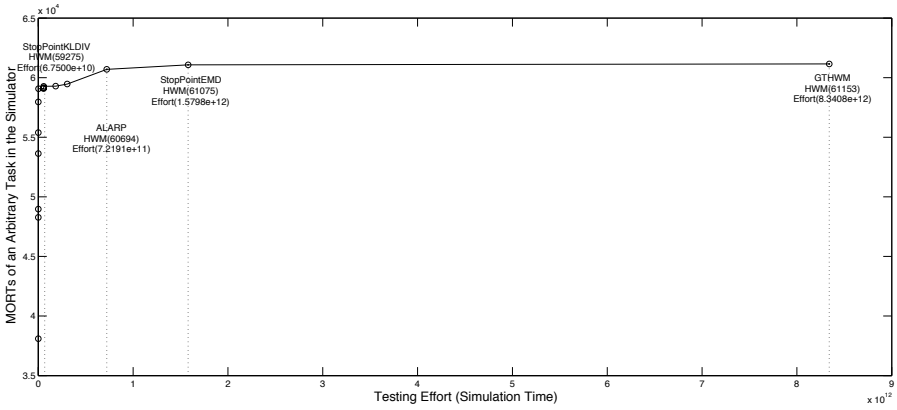

Fig. 7: EMD outperforming KLDIV with $\delta = 0.001$

In the next round of analysis, we increased δ to see how EMD performance would change. We used the following set of tunings through KL DIV and EMD: $\{\alpha = 2,\ i = 10,\ \delta = 0.1,\ \lambda = 200,\ \beta = 1250$ sec$\}$. The results are shown in Figure 8. It can be seen that with bigger δ, the EMD test performance (benefit and cost) is similar to the KL DIV and more tasks of lower priority passed EMD compared to Figure 6. So, it can be concluded that the EMD test is more sensitive to dissimilarity between histograms, i.e., with lower threshold δ, less tasks of lower priority pass the test compared to the KL DIV test.

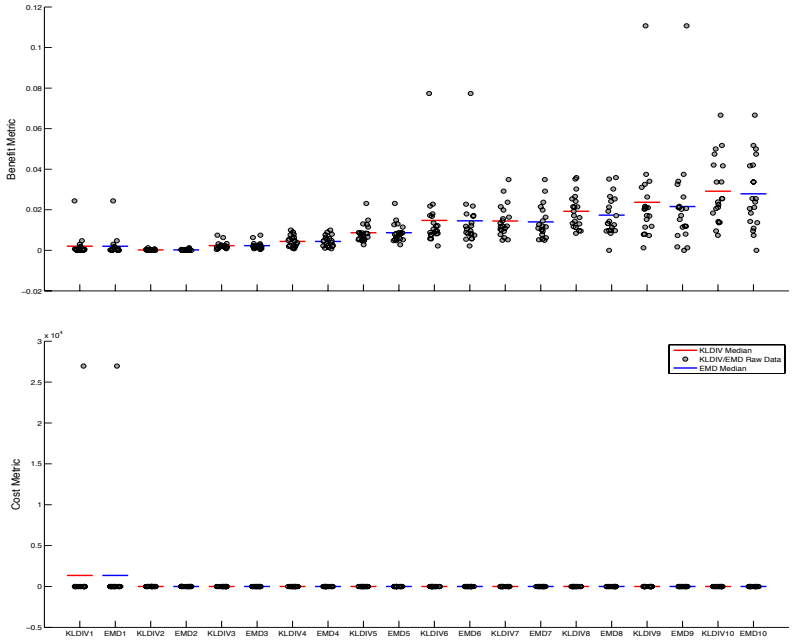

Fig. 8: KL DIV vs. EMD for trials of 10 tasks

In order to determine how λ affects the KL DIV and EMD tests, we used the following tunings: $\{\alpha = 2,\ i = 10,\ \delta = 0.001,\ \lambda = 1400,\ \beta = 1250$ sec$\}$ where λ is bigger than tunings used in Figure 6. As shown in Figure 9, the two tests have similar cost metric values. However, the EMD test has better benefit from task 3 to task 7. For the last three lowest priority tasks, i.e., task 8 to task 10, no task pass the EMD test while the KL DIV results for them is similar to tunings with $\lambda = 200$. So, it can be concluded that more bins, which result in smaller bin size, make differences between histograms more apparent and the EMD test becomes more sensitive using smaller bins. According to [13], cross-bin dissimilarity measures, e.g., EMD, yield better results with smaller bins. Then, the important issue would be to decide the bin size so that a balance between quality of the results and scalability of the algorithm could be made, i.e., smaller bins result in less scalable algorithm in terms of physical memory occupied by testing data.

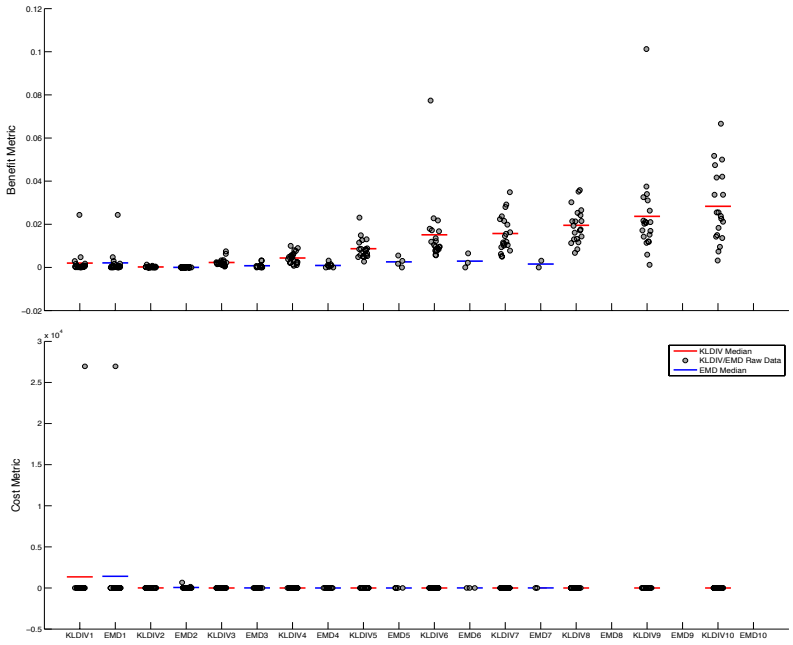

Fig. 9: KL DIV vs. EMD for trials of 10 tasks

From Figure 9 discussion, it can be concluded that the smaller bin size, the more sensitive EMD test becomes to the subtle changes in distributions which is not the case for KL DIV. This might also improve the convergence algorithm where KL DIV does not make an ALARP-based decision. Figure 10 shows such a situation, for a task with mid priority in the task set. The parameter ξ in Equation 3 is set to 0.5%. It can be seen that KL DIV stops sooner than *ALARP* while EMD stops later, thus, it meets the ALARP criteria. However, as it was mentioned earlier the next issue would be the choice of been size such that less testing effort is associated with EMD, i.e., *StopPointEMD* is preferred to stop sooner after ALARP than later.

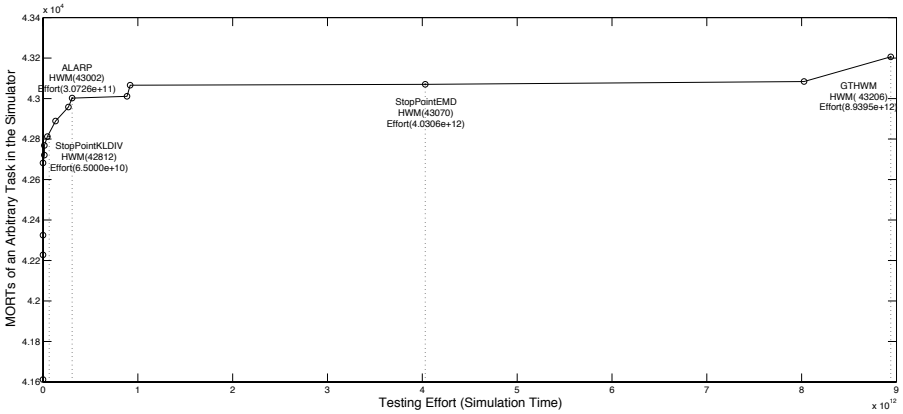

Fig. 10: EMD outperforming KLDIV with $\lambda = 1400$

From this section results, it can be concluded that EMD is a more sensitive test to subtle differences between histograms when dealing with lots of data compared to KL DIV and can improve the performance of the algorithm in situations where KLDIV has shown to be weak.

7 Conclusions and Future Work

In order to make a stop-test decision in the context of the worst-case timing characteristics, we propose and develop a convergence algorithm. However, we observe experiments in which the algorithm could not make a proper stop-test decision according to the ALARP principle. In order to deal with such situations, we propose that the KL DIV test to be replaced by the EMD test. We experimentally evaluate the performance of the algorithm using EMD against KL DIV to determine whether EMD does make an improvement.

The immediate extension of the work will focus on making the algorithm *robust*. For the convergence algorithm to become *robust*, it is important that it holds when task set characteristics vary across some required ranges, e.g., number of tasks ranging from 10 to 50, task period ranging from [200, 400] μs to [200, 1000] μs. In order to achieve robustness, we have taken the following steps so far: firstly, in [3], we derived a set of task set characteristics and their values that adversely affect the algorithm performance. Secondly, in the current work, it is examined whether the algorithm itself can be improved with a focus on the statistical test it uses. Thirdly, in the next piece of work, the algorithm would be stress tested using the influential task set parameters identified earlier and tuned so that the algorithm could hold across the required ranges of task set characteristics.

Further future work will be around applying the convergence algorithm in systems with more complex timing behaviour. This may include using advanced

bench marks, real system considering hardware dependency or the support of *mixed criticality*. Criticality is a designation of the level of assurance against failure needed for a system component and a *mixed criticality system (MCS)* is the one that has two or more distinct levels, e.g., safety critical, mission critical and low critical [15].

Acknowledgement

We acknowledge the Swedish Foundation for Strategic Research (SSF) SYNOPSIS Project for supporting this work.

References

1. M. Malekzadeh and I. Bate, "Making an ALARP Decision of Sufficient Testing," in *IEEE International Symposium on High-Assurance Systems Engineering*, 2014, pp. 57–64.
2. M. Malekzadeh, I. Bate, and S. Punnekkat, "Using design of experiments to optimise a decision of sufficient testing," in *Euromicro Conference series on Software Engineering and Advanced Applications*, 2015.
3. M. Malekzadeh and I. Bate, "Influential nuisance factors on a decision of sufficient testing," in *The 15th International Conference on Algorithms and Architectures for Parallel Processing*, November 2015. [Online]. Available: http://www.es.mdh.se/publications/4203-
4. "Health and safety at work act 1974," The National Archives, Tech. Rep., 1974. [Online]. Available: http://www.legislation.gov.uk/ukpga/1974/37/contents
5. S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics (AMS)*, vol. 22, no. 1, pp. 79–86, 1951.
6. S. Kullback, *Information Theory and Statistics*. Wiley, 1959.
7. F. J. Cazorla, E. Quinones, T. Vardanega, L. Cucu-Grosjean, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim, "PROARTIS: Probabilistically Analysable Real-Time Systems," INRIA, Research Report RR-7869, Jan. 2012. [Online]. Available: https://hal.inria.fr/hal-00663329
8. E. Gumbel, *Statistics of Extremes*, ser. Dover books on mathematics. Dover Publications, 2004.
9. L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs." in *Proceedings of the Euromicro Conference on Real-Time Systems*, 2012, pp. 91–101.
10. S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, Dec 2001, pp. 215–224.
11. J. Hansen, S. Hissam, and G. A. Moreno, "Statistical-Based WCET Estimation and Validation," in *9th International Workshop on Worst-Case Execution Time Analysis (WCET'09)*, ser. OpenAccess Series in Informatics (OASIcs), N. Holsti, Ed., vol. 10. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2009, pp. 1–11, also published in print by Austrian Computer Society (OCG) with ISBN 978-3-85403-252-6. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2009/2291

12. I. Bate, P. Nightingale, and J. McDermid, "Establishing timing requirements for control loops in real-time systems," *Microprocessors and Microsystems*, vol. 27, no. 4, pp. 159–169, 2003.
13. Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vision*, vol. 40, no. 2, pp. 99–121, Nov. 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1026543900054
14. E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, no. 1-2, pp. 129–154, 2005.
15. A. Burns and R. I. Davis, "Mixed criticality systems - a review," University of York, 2015.