Work-in-Progress Abstract: WKS, a local unsupervised statistical algorithm for the detection of transitions in timing analysis

Marwan Wehaiba El Khazen mwek@statinf.fr
Liliana Cucu-Grosjean liliana.cucu@inria.fr
Adriana Gogonel ag@statinf.fr
Hadrien Clarke hadrien.clarke@inria.fr
Yves Sorel yves.sorel@inria.fr

***Abstract*—The increased complexity of programs and processors is an important challenge that the embedded real-time systems community faces today, as it implies substancial timing variability. Processor features like pipelines or communication buses are not always completely described, while black-box programs integrated by third parties are hidden for IP reasons. This situation explains the use of statistical approaches to study the timing variability of programs. Most existing work is concentrated on the guarantees provided by positive answers to statistical tests, while our current work concerns potential algorithms based on the negative answers to these tests and their impact on the timing analysis. We introduce here one such algorithm, the Walking Kolmogorov-Smirnov test (WKS).**

***Index Terms*—statistical estimation, worst-case execution time, KS-test**

I. Motivation

An embedded system is a computing system with a dedicated function, embedded within a larger device. Beside constraints like power consumption, size and weight, some embedded systems may have time constraints to meet and those systems are called embedded real-time systems, whose design is mainly based on commercial processors with an excellent average time behavior (in terms of the time taken to execute code). Unfortunately, these processors may have an important variation between the worst-case execution time and the average execution time of a program. This may justify the increased interest that statistical and probabilistic approaches have received from the real-time community [1].

Our contribution concerns the results of the required statistical tests before any statistical estimation of the worst-case execution time (WCET) of a program can be made. Indeed, in order to understand what type of statistical estimator one may use, or if one may be used at all, different statistical tests are applied to the sequences of the execution times. The existing results [1] consider the statistical tests as go or no-go means to use the statistical estimators, without necessarily underlining the lessons learnt from the negative results of such tests and how a designer should modify their measurement protocol to improve the WCET statistical estimation. Within this paper, we provide preliminary results filling this existing gap, by proposing an algorithm considering those negative results and their impact on the timing analysis.

This research is partially funded by the FR PSPC STARTREC and CIFRE agreement between StatInf and Inria

II. Our problem and the associated definitions

We consider a set τ of n periodic implicit deadline programs $\{\tau_1, \tau_2, ..., \tau_n\}$ scheduled according to a fixed-priority preemptive scheduling policy on a single core processor. A program τ_i is described by $(\mathcal{C}_i, T_i, D_i)$, $\forall 1 \leq i \leq n$, with its activation period T_i, an implicit deadline D_i and its probabilistic worst-case execution time (pWCET) $\mathcal{C}_i$. We mean by $\mathcal{C}_i$ an upperbound on any distribution of execution times obtained for any execution scenario of a program $\tau_i, \forall i$. To estimate a pWCET one may use statistical estimators [1]. Typically, the pWCET statistical estimators are applied once the identically distributed property is verified by appropriate statistical tests. In this paper, we discuss the possibility of learning from negative results to said tests. Proposing pWCET estimators is beyond the purpose of this paper.

III. Preliminary results

We consider the PX4 autopilot of a drone, an open source flight control software [2] designed by KTH Zurich. The autopilot controls the position and altitude of the drone while following a given trajectory. Following a given trajectory becomes an execution scenario for all programs of the autopilot.

Our dataset is obtained by executing the programs on an ARM Cortex-M4 unicore microcontroller included within a Pixhawk board. The PX4 programs run on top of NuttX, a Unix-like OS. The autopilot programs, called *sensors*, *EKF2*, *mc_position_cont*, *control_altitude*, or *output driver* [1], are executed according to a preemptive fixed priority scheduler. In this paper, we monitor the execution times of the *sensors* program within one flight and the evolution of the execution times is illustrated in the blue scatterplot in Figure 1. In this figure, the horizontal axis describes the order of appearence of the 9161 execution times, while the vertical axis describes the values of the execution times. We have normalized values (divided by the maximum value of 112) in order to show on the same figure the evolution of the p-values of our statistical tests. Only the p-values smaller than 0.2 are shown, in orange in Figure 1. Indeed, only small enough p-values can be considered negative answers to the test, which are of interest for our current paper.

[1] See more details at https://px4.io for a complete list of programs.

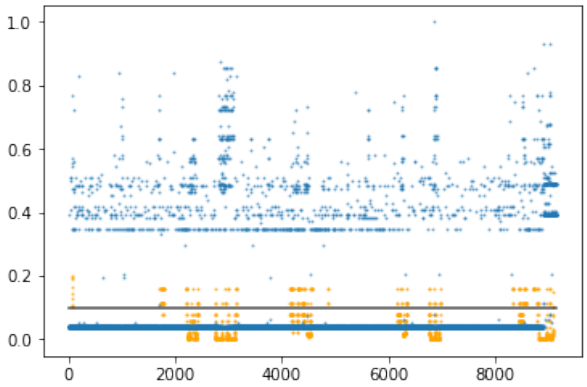

Fig. 1. Blue: evolution of the normalized execution times of the program. Orange: p-values of WKS that are less than 0.2. Black: threshold of 0.1

The program goes through multiple paths, loops in some, and skips some others, all according to its code, current conditions and the scenario it is accomplishing (here, the flight of the drone). Thus, the execution times tend not to be identically distributed in any real-life setting. There are *mode changes* in the distribution within the execution time sequences, and any attempt of estimating anything, like the distribution of the pWCET, needs to either account for that, or be interrupted. Here, we address the detection of those mode changes, in the hope that some subsets of the dataset are identically distributed, even if the whole dataset is not. That can make the estimation possible, not just despite negative results to statistical tests, but precisely because of them. On one hand, the execution times are unlabeled in general (one usually does not have access to the specific path each execution takes), which highlights the necessity of unsupervised approaches [3]. On the other hand, applying global statistical tests naively (i.e. on the dataset as a whole) may result in uninteresting results, that fail to reject the identically distributed null hypothesis H_0, because some mode changes are minute with respect to the number of observations, and a more local approach is then called for. In order to meet the needs for unsupervised and local approaches, we propose a novel method for detecting mode changes, by applying, on two windows around each datapoint, the two-sample Kolmogorov-Smirnov test (see Definition 1) and check for failure (the Anderson-Darling test can be used too). We name this method the Walking Kolmogorov-Smirnov test, or **WKS**. The idea is that a datapoint presenting a low enough p-value, rejecting H_0, would correspond to a mode change. Our approach belongs to the more general field of change point detection (CPD) in statistical analysis [4], where a change point can be associated to a mode change, as defined within the real-time literature [5]. Although sliding windows and statistical tests have been used in the CPD literature (likelihood ratio methods in [6], Kullback-Leibler divergence in [7], Kernel methods in [8], etc.), to the best of our knowledge, no existing sliding window method is based on such test statistics. Two hyperparameters stand out here: the width of each window w and the threshold α for low enough p-values representing a failure, i.e. a mode change. In order to automate the process, multiple widths can be considered and α can be set at 0.05, but that is still a work in progress.

Definition 1: Given two samples S_n^X and S_n^Y, respectively $X_1, ..., X_n$ and $Y_1, ..., Y_n$, the two-sample Kolmogorov-Smirnov statistic is defined by $D_n = \|F_n^X - F_n^Y\|_\infty = \sup_x |F_n^X(x) - F_n^Y(x)|$, where F_n^X and F_n^Y are the empirical cumulative distribution functions of S_n^X and S_n^Y. If the true distribution functions F^X and F^Y are continuous, assuming H_0 (identical distributions), $\sqrt{n}D_n$ has a known limit distribution K, the Kolmogorov distribution, for any F^X and F^Y. H_0 can be rejected at level α if $\sqrt{n}D_n$ is larger than K_α, the associated $1-\alpha$ quantile. The p-value p is the probability that K will take values at least as extreme as the observed statistic, $p = \mathbb{P}_{H_0}(K > \sqrt{n}D_n)$.

We show in Figure 1 a blue scatterplot of $n = 9161$ normalized exection times (we divided by the maximum: 112 CPU cycles) of one of the autopilot's programs. For every blue datapoint, two windows of length $w = 100$ each, are formed: one containing the w previous datapoints and the other containing the w next datapoints. Then we calculate the p-value of the KS test for these two windows, to check if their distributions are far enough from each other. For the sake of clarity, the p-value is only represented (in orange) if it is smaller than 0.2. Indeed, only small enough p-values can be considered negative answers to the test and can therefore suggest the possibility of a mode change, so these small p-values are those we are interested in, and if the threshold is set at the commonly used threshold of 0.05, only those smaller than that threshold are taken into account, but we show more of them to get a visual sense of the kind of information the WKS algorithm can offer. We also add a black horizontal line at 0.1. Now we can see the dips in orange p-values and the corresponding behavior of the blue scatterplot. In this case, the WKS algorithm is sensitive to the mode changes, specifically when values are concentrated on a higher level than what seems to be the baseline of the program, which is promising. Future work would of course, entail setting an automated process for the width and threshold hyperparameters, as well as considering other statistical tests, whether present in the literature or not. Furthermore, satisfying mode change detection could serve to hint at which distributions are well enough explored, and thus which branches in the program are well-visited and represented in our dataset, and which others might not be. This is paramount for any claim to representativity of the dataset and scalability, while improving the pWCET estimation.

REFERENCES

[1] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for real-time systems," *LITES*, vol. 6, no. 1, pp. 03:1–03:60, 2019.

[2] L. Meier, *PX4 Development Guide*, https://dev.px4.io/en/.

[3] K. Sindhu Meena and S. Suriya, "A survey on supervised and unsupervised learning techniques," in *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*, L. A.

Kumar, L. S. Jayashree, and R. Manimegalai, Eds. Cham: Springer International Publishing, 2020, pp. 627–644.

[4] S. Aminikhanghahi and D. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, 05 2017.

[5] A. Burns, "System mode changes - general and criticality-based," in *the 2nd International Workshop on Mixed-Criticality Systems*, 2014.

[6] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*, 2008, pp. 312–321.

[7] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 2, pp. 114–127, 2012.

[8] K. D. Feuz, D. J. Cook, C. Rosasco, K. Robertson, and M. Schmitter-Edgecombe, "Automated detection of activity transitions for prompting," *IEEE transactions on human-machine systems*, vol. 45, no. 5, pp. 575–585, 2014.