# A New Way about using Statistical Analysis of Worst-Case Execution Times

Yue Lu[1], Thomas Nolte[1], Iain Bate[2], and Liliana Cucu-Grosjean[3]
[1]Mälardalen Real-Time Research Centre (MRTC), Västerås, Sweden
[2]Department of Computer Science, University of York, York, United Kingdom
[3]INRIA Nancy-Grand Est, Nancy, France
yue.lu@mdh.se

***Abstract*—In this paper, we revisit the problem of using Extreme Value Theory (EVT) in the Worst-Case Execution Time (WCET) analysis of the programs running on a single processor. Our proposed statistical WCET analysis method consists of a novel sampling mechanism tackling with some problems that hindered the application of using EVT in the context, and a statistical inference about computation of a WCET estimate of the target program. To be specific, the presented sampling mechanism takes analysis samples from the target program based around end-to-end measurements. Next, the statistical inference using EVT together with other statistical techniques, analyzes such timing traces which contain the execution time data of the program, to compute a WCET estimate with a certain predictable probability of being exceeded.**

## I. Introduction

One very important part of real-time analysis is Worst-Case Execution Time (WCET) analysis, that determines the longest time a piece of software will execute. Accurate WCET estimates are fundamental to most of the research conducted in the real-time research community. They are essential in real-time systems development in the substantial step of creating schedulers and performing Response-Time Analysis (RTA) and schedulability analysis [1]. A lot of research has been done in the realm of WCET analysis, and a good overview can be found in [2]. The result of WCET analysis is one WCET estimate, i.e., the longest possible execution time of a program that is running without any interrupts, on a specific hardware platform. Because of high complexity, WCET analysis tools are not able to find the exact WCET in general.

Techniques to perform WCET analysis can broadly be categorized as follows [3]:

1) Static WCET Analysis (SA) computes a WCET estimate by statically analyzing the program source code and all its input value combinations together with a model of the hardware, i.e., a *processor model* (which synthesizes the functional and temporal behavior of the hardware). A static WCET analysis has to make pessimistic assumptions in uncertain cases[1], in order to produce a safe upper bound, i.e., a WCET estimate that is guaranteed to never be smaller than the actual, real WCET, as a safe overestimation.
2) Measurement-Based WCET Analysis (MBA) is to perform *end-to-end* measurements of running the target program on the hardware (or simulator) for the subset of all input value combinations. The underlying premise is that the testing regime is representative of real system operation. Furthermore, with enough testing, the *High Water-Mark Time* (HWMT) obtained by measurement-based WCET analysis lies in close proximity to the actual WCET. However, using extensive measurements to ensure enough test case coverage or alternatively attempting to enforce the program to execute its worst-case path may be very difficult. Therefore, the selection of test cases to reach the best path coverage (covers the worst-case path) is crucial. Measurement-based WCET analysis may not guarantee to find the actual WCET in the general case, and may consequently underestimate the WCET. To remedy the situation, a *safe margin* or an ad hoc *safety factor* is usually to be added to the WCET estimate given by measurement-based WCET analysis, which for instance is from engineering wisdom from previous projects. Nonetheless, there is no systematic way to determine the appropriate safety margin or predict how good the WCET estimate will be.
3) Hybrid Measurement-Based WCET Analysis (HMBA) combines MBA and SA, with the intention of reducing the potential for underestimation and overestimation raised by both approaches. To this end, HMBA gleans the execution time of program segments (i.e., instruction blocks) via *instrumentation points* (ipoints) as the software runs on target. Such observed execution times are used in the subsequent stage of WCET calculation. However, the key assumption in HMBA is that the WCET of each instruction block has been exercised during testing and that measurements are sufficient to provide upper loop bounds; otherwise, the safety of the final estimate is compromised, i.e., it will either underestimate or overestimate the actual WCET. An example of HMBA approaches is *pWCET* [4], behind which the technique is based on probabilistically combining the worst-case effects seen in individual blocks to build

[1]For example, a loop bound given by *flow analysis* is larger than the actual value, or *low-level analysis* clarifies a memory access as a cache miss even though it always may result in a cache hit.

the execution time model of the worst-case path of the program. The primary weakness of this approach is that the execution time is modeled directly by an empirical distribution function, which requires a very large number of samples to obtain an accurate model to represent the worst-case (i.e., the tail) behavior of the Execution Time (ET) distribution. There are also some interesting work in [5], [6], [7] on using Genetic Algorithms (GA) [8] to generate test vectors to produce a WCET estimate which lies in close proximity to the actual WCET.

4) Parametric (or symbolic) WCET analysis expresses the WCET estimate as a formula consisting of parameters of the program, rather than just a single numerical value. The parameters can be either external, or internal like a symbolic upper bound on a loop. A parametric WCET formula contains much more information about the program, and it can be used for applications like on-line scheduling of tasks where the value of parameters are unknown until runtime, or to find which parts of a code that has the strongest influence on the WCET. Furthermore, parametric WCET analysis is naturally more complex than classical static WCET analysis and should not be used on large systems with millions of lines of code; rather, the parametric estimation is most efficiently used on smaller program parts such as smaller tasks or functions which have input data dependent execution times [9].
5) There is another set of WCET analysis methods, which use statistical methods to provide a WCET that is guaranteed never to be exceeded, under a certain predictable probability. The ones based around Extreme Value Theory [10] (EVT hereafter) are introduced in the following section.

## II. Problem Setting

### A. The State of the Art on using EVT in WCET Analysis

Extreme Value Theory (EVT) [15] was first codified in 1958 and is a separate branch of statistics for dealing with the tail behavior of a distribution. EVT is used to model the risk of the extreme, rare events, without the vast amount of sample data required by a brute-force approach. Example applications of EVT include risk management, insurance, hydrology, material sciences, telecommunications and so on.

There are two recent approaches to using EVT for WCET estimation [11], [12]. In [11], the execution time measurements are first fit to the Gumbel Max distribution using an unbiased estimator. A WCET estimate is then obtained by using a pertaining excess distribution function. However, the problem with this approach is that it incorrectly fits (raw) ET data to the Gumbel Max distribution, as there may be some dependencies caused by caches in the ET data. In addition, the Gumbel Max distribution and other EVT distributions are intended to model random variables that are the maximum or minimum of a large number of other random variables. Nonetheless, there is no such case for execution time measurements. Further, there is no evidence that any *Goodness-Of-Fit (GOF) test* is used to ensure if the estimated parameters of the Gumbel Max distribution can actually fit the measured data in [11].

Hansen [12] furthers the study by presenting the work on predicting how likely a WCET estimate generated by EVT will be exceeded in the future, for a single trace of the target program. In the context of using EVT for WCET analysis, they made some changes including using the GOF hypothesis test when fitting block maxima to the Gumbel Max distribution, proposing a simple search algorithm by doubling the block size for the determination of the best-fit Gumbel Max parameters, validating the method based around an intensive evaluation, etc. However, none of these two work actually touched the key points which are lately discussed in [13] by Griffin (to be introduced in Section II-B).

### B. Problems with using EVT in WCET Analysis

The problems argued in [13] are mainly concerned from the following two perspectives:

1) **Continuous vs. Discrete Distributions:**
It is not realistic to use a *continuous random variable* i.e., the Gumbel Max distribution, to model the execution time of the program, since the program cannot terminate at any point in its control flow graph.
2) **I.I.D. Assumption in EVT:**
EVT also makes the assumption required by statistics and probability theory, i.e., the observations (or analysis samples) have to be *independent and identically distributed* (i.i.d.). Unfortunately, the execution times of programs usually are not independent or identically distributed, for instance, due to the dependencies between different states of the data structure in the program.

In summary, the analysis samples used by EVT in WCET analysis cannot be from the ET sampling distribution of the program, which is collected in the traditional way of running a series of sample executions of the program. A new way of collecting *qualified* analysis samples (or *individuals*) which could be used by EVT is thereof necessary.

### C. Our Contributions

The contributions of this paper concern both aspects:

1) In Section III-A, we propose a novel sampling mechanism which employs the Simple Random Sampling technique to collect qualified analysis samples (i.e., timing traces) which could be used by EVT without raising any problems mentioned previously.
2) We propose a statistical WCET analysis method namely *RapidET* which computes a WCET estimate of the program under analysis based around analyzing timing traces, by using EVT.

## III. A Statistical WCET Analysis RapidET

### A. The Sampling Mechanism for Collecting Timing Traces Taken from Programs

First, when any type of statistical techniques is applied to analyze the observations (or samples), there is a key issue of selecting such samples from the population of all individuals concerning the desired information, i.e., any bias on the sampling has to be avoided. In this work, we therefore employ the technique of Simple Random Sample (SRS), which gives every possible sample of a given size the same chance to be chosen. In practice, when such samples are taken from target programs, the SRS can be done in terms of randomizing program inputs by using the *uniform distribution*. Next, due to the existence of dependencies between different states of the data structure in the program, an upcoming ET data may not be independent with the ET data previously measured at program executions, when the SRS technique is used. This violates another important assumption required by any statistical methods, i.e., each sample used in the statistical analysis has to be i.i.d.. This is also the second problem introduced in Section II-B. In order to tackle with this problem, we propose the sampling mechanism which first executes the target program for $N$ times by using the SRS technique which results in $N$ *sub-timing traces*, and each of sub-timing traces contains $m$ ET data. Next, per sub-timing trace, the highest value of $m$ ET data of the program measured, will be chosen as a sample to construct the new sampling distribution of the ET data of the program. Since there are no dependencies between any maximum of the ET data of the program from two independent sub-timing traces, as a result, all the individuals in the new reconstructed sampling distribution are mutually independent. Hence, the underline i.i.d. assumption is realistic and satisfied. We call such new constructed ET data sampling distribution of the program as the *qualified* ET data sampling distribution, hereafter. The implementation of SRS is the function *SRS* as shown in line 2 in Algorithm 1 (i.e., pseudo-code for our proposed statistical WCET analysis which is to be introduced in Section III-B), where the parameters $P$ and $m$ represent the program under analysis and a certain number of timing traces taken from the target program respectively. It is interesting to note that the SRS technique also gives us the confidence that no matter how big the population is, the statistical inference based on the sampling distribution collected by using SRS can successfully estimate the parameters of the underline population [14], such as the tail behavior of the underline population, i.e., the WCET of the programs under analysis in our case.

### B. RapidET

Our proposed statistical WCET analysis method *RapidET* is based on EVT and end-to-end measurements. Further, RapidET is a recursive procedure which, as the first two arguments, takes $n$ reference data sets each of which contains $m$ simulation traces containing programs' execution times. For each reference data set, the algorithm returns the WCET estimate of the program under analysis with a probability of being exceeded, e.g., $10^{-9}$, which is the third algorithm argument. For instance, Airbus uses such the value $10^{-9}$ which is at the highest development assurance level in the safety-critical system domain. Next, RapidET will verify if the sampling distribution consisting of $n$ WCET estimates given by EVT for all $n$ reference data sets (we refer to such a sampling distribution as the EVT distribution hereafter) conforms to a normal distribution or not, according to the result given by the non-parametric Kolmogorov-Smirnov test (the KS test hereafter). If it is, then RapidET will calculate the Confidence Interval (i.e., CI hereafter) of the EVT distribution, at a given confidence level, and choose the upper bound on the CI as the final WCET estimate. Otherwise, if the EVT distribution cannot be fitted to a normal distribution, some other statistical methods will be adopted, e.g., a *resampling* statistic *bootstrap* [14]. The detailed implementation of RapidET is described by Algorithm 1.

RapidET consists of the following three steps: 1) construction of the referenced data sets, 2) WCET estimation of each referenced data set by using EVT, and 3) derivation of a final WCET estimate that is given by the algorithm. To be specific, the outline of the algorithm is as follows:

1) Construct $n$ reference data sets for the WCET estimates by using our sampling mechanism proposed in Section III-A.
2) Calculate the WCET estimates of the program under analysis per each reference data set, i.e., $est_i$ where $1 \leq i \leq n$.
   a) Set the initial block size $b$ to 1, for each reference data set.
   b) If the number of blocks $k = \left\lfloor \frac{m}{b} \right\rfloor$ is less than 30, the algorithm stops as there are not enough samples to generate an estimate [12].
   c) Segment $m$ execution times into blocks of size $b$, and for each of the $\left\lfloor \frac{m}{b} \right\rfloor$ blocks find the maximum values.
   d) Estimate the best-fit Gumbel parameters $\mu$ and $\beta$ to the block maximum values by using a proposed exhaustive search algorithm introduced as shown in lines 6 to 17 in Algorithm 1.
   e) Calculate a WCET estimate based on the best-fit Gumbel Max parameters estimated through Step $d$), i.e., $\mu$, $\beta$, and a target acceptance probability $P_e$.
3) After verifying if the EVT distribution (i.e., $EST \leftarrow est_1, ..., est_i, ..., est_n$) can successfully be fitted to a normal distribution by using the KS test, RapidET will return a result, according to different confidence level applied in the algorithm, e.g., $\overline{EST} + 3\sigma_{EST}$ (the sum

of the *mean* value and three *standard deviation* of the EVT distribution at the confidence level 99.7%).

**Algorithm 1** $RapidET(n, m, P_e, cl)$

**for all** $est_i$ such that $1 \leq i \leq n$ **do**
  $X_i \leftarrow et_{i,1}, ..., et_{i,m} \leftarrow SRS(P, m)$
  $b \leftarrow 1$
  $k \leftarrow \left\lfloor \frac{m}{b} \right\rfloor$
  $success \leftarrow false$
  **while** $k \geq 30$ **and** $success = false$ **do**
    $S_i \leftarrow s_{i,1}, ..., s_{i,k} \leftarrow segment(m, b)$
    $Y_i \leftarrow y_{i,1}, ..., y_{i,k} \leftarrow maxima(S_i)$
    **if** $passChiSquareTest(Y_i, GumbelMax) > 0$ **then**
      $success \leftarrow true$
      $l, s \leftarrow ChiSquareTest(Y_i)$
      $est_i \leftarrow evtgumbelmax(b, l, s, P_e)$
    **else**
      $b \leftarrow b + 1$
      $k \leftarrow \left\lfloor \frac{m}{b} \right\rfloor$
    **end if**
  **end while**
**end for**
$EST \leftarrow est_1, ..., est_i, ..., est_n$
**if** $passKS(EST, Normal)$ **then**
  $\overline{EST} \leftarrow \frac{1}{n} \times \sum_{i=1}^{n} est_i$
  $\sigma_{EST} \leftarrow \sqrt{\frac{1}{n} \sum_{i=1}^{n} (est_i - \overline{EST})^2}$
  **if** $cl = 0.997$ **then**
    $et_{est} \leftarrow \overline{EST} + 3\sigma_{EST}$
  **else**
    **if** $cl = 0.95$ **then**
      $et_{est} \leftarrow \overline{EST} + 2\sigma_{EST}$
    **else**
      **if** $cl = 0.682$ **then**
        $et_{est} \leftarrow \overline{EST} + 1\sigma_{EST}$
      **else**
        $et_{est} \leftarrow \overline{EST}$
      **end if**
    **end if**
  **end if**
**else**
  $et_{est} \leftarrow bootstraptest(EST)$
**end if**
**return** $et_{est}$

## IV. Summary

In this paper we present the ongoing work towards using our proposed statistical Worst-Case Execution Time (WCET) Analysis method *RapidET*, to compute a WCET estimate of programs running on a single processor. Specifically, RapidET consists of a novel sampling mechanism and a statistical inference based around Extreme Value Theory and other statistical techniques. Future work will focus on the method evaluation and the selection of good value of algorithm parameters from the perspective of accuracy of analysis results, as well as how to use the obtained WCET estimates and pertaining statistical constraints (i.e., certain predictable probabilities) in response time analysis and schedulability test which consider a system of tasks.

## Acknowledgment

This work was partially supported by the Swedish Foundation for Strategic Research (SSF) and the Swedish Research Council (VR).

## References

[1] *Handbook of Real-Time and Embedded Systems*. Chapman and Hall/CRC (July 23, 2007), 2007.

[2] R. Wilhelm et al., "The worst-case execution-time problem—overview of methods and survey of tools," *Trans. on Embedded Computing Sys.*, vol. 7, no. 3, pp. 1–53, 2008.

[3] Y. Lu, "Approximation Techniques for Timing Analysis of Complex Real-Time Embedded Systems," Lic. dissertation, School of Innovation, Design and Engineering, October 2010.

[4] G. Bernat, C. A., and S. Petters, "pWCET: A Tool for Probabilistic Worst-Case Execution Time Analysis of Real-Time Systems," in *Proc. of LCTES' 03*, 2003.

[5] U. Khan and I. Bate, "WCET Analysis of Modern Processors Using Multi-Criteria Optimisation," in *Proc. of SSBSE' 09*. IEEE Computer Society, 2009, pp. 103–112.

[6] J. Wegener and M. Grochtmann, "Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing," *Real-Time Syst.*, vol. 15, no. 3, pp. 275–298, 1998.

[7] J. Wegener and F. Mueller, "A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints," *Real-Time Syst.*, vol. 21, no. 3, pp. 241–268, 2001.

[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[9] S. Bygde, "Static WCET Analysis Based on Abstract Interpretation and Counting of Elements," Lic. dissertation, School of Innovation, Design and Engineering, March 2010.

[10] J. Beirlant, Y. Goegebeur, J. Segers, and J. Teugels, *Statistics of Extremes: Theory and Applications*. Wiley Press, 2004.

[11] S. Edgar and B. A., "Statistical Analysis of WCET for Scheduling," in *Proc. of RTSS' 01*, 2001, pp. 215–224.

[12] J. Hansen, S. Hissam, and G. Moreno, "Statistical-Based WCET Estimation and Validation," in *Proc. of WCET' 09*, 2009, pp. 123–133.

[13] D. Griffin and A. Burns, "Realism in Statistical Analysis of Worst Case Execution Times," in *Proc. of WCET' 10*, 2010.

[14] D. S. Moore, G. P. Mccabe, and B. A. Craig, *Introduction to the practice of statistics*, 6th ed. New York, NY 10010: W. H. Freeman and Company, 2009.

[15] E. Gumbel, *Statistics of Extremes*. Columbia University Press, 1958.