ISSN 2007-9737

Deterministic and Probabilistic Schedulability Analysis for Real-Time Systems

Rebai Rim[1,*], Hedi Tmar[1,2], Adel Mahfoudhi[1,2]

[1] University of Sfax, Enis, Computer Embedded System, Tunisia

[2] University of Sfax, FSS, Tunisia

rebai_rima@yahoo.fr

Abstract. Traditional deterministic schedulability analysis provides a binary guarantee—a task set is either schedulable or not—based on fixed worst-case execution times (WCET). While safe, this approach often leads to pessimistic and resource-inefficient system designs, as it cannot account for the runtime variations inherent in complex computing platforms. This paper introduces a deterministic schedulability analysis and a new paradigm for schedulability analysis founded on probabilistic principles. Our approach generalizes the traditional task model by explicitly incorporating execution time variations. Rather than relying on static WCETs, the proposed model captures this variability using probabilistic distributions, clustering, and adaptive estimation techniques. This shift enables a more realistic and flexible assessment of system timing.

Keywords. Real time systems, schedulability analysis, probabilistic systems, execution time, dependence.

1 Introduction

Schedulability analysis represents a cornerstone of real-time systems design, providing critical verification methodologies to ensure that computational tasks complete within their specified deadlines. Traditional approaches have predominantly relied on deterministic models, which operate under worst-case execution time (WCET) assumptions to guarantee absolute temporal correctness [10]. While these methods offer stringent safety assurances, their inherent pessimism often leads to significant underutilization of system resources, rendering them impractical for modern applications with mixed criticality and complex execution constraints [2, 1].

In response to these limitations, probabilistic schedulability analysis has emerged as a transformative paradigm. This approach models key temporal parameters—particularly execution times—as random variables, thereby capturing the inherent variability observed in real-world systems [4, 14]. By leveraging statistical distributions instead of deterministic bounds, probabilistic analysis enables more efficient resource allocation while providing high-confidence guarantees on meeting deadlines [7, 20].

The evolution toward probabilistic frameworks has unveiled new research challenges, particularly concerning the accurate characterization of dependencies between task executions. History-dependent factors such as cache behavior, pipeline states, and data-dependent execution paths introduce complex correlations that violate the common assumption of statistical independence [2, 1]. Recent advances have begun addressing these challenges through sophisticated statistical tools, including copula theory for modeling dependency structures [2, 13].

This paper contributes to this evolving landscape by presenting a comprehensive probabilistic schedulability analysis framework. Our work makes empirical validation demonstrating significant reduction in analysis pessimism compared to both deterministic and independent probabilistic approaches.

The remainder of this paper is organized as follows: Section 2 reviews fundamental concepts and related work in schedulability analysis. Section 3 introduces our system model and theoretical foundation. Section 4 details the deterministic Real-Time Task Schedulability analysis, while Section 5 presents the System Cost Implications of Probabilistic Schedulability Analysis. Section 6 concludes with directions for future research.

2 Related Work

Deterministic schedulability analysis is a cornerstone of real-time systems engineering, ensuring that all tasks in a system can meet their deadlines under worst-case conditions. This analysis is crucial for safety-critical applications, such as industrial automation, avionics, and automotive systems, where predictability and timing guarantees are non-negotiable.

Deterministic schedulability analysis involves mathematically verifying whether a set of real-time tasks can be scheduled to meet all deadlines, given specific scheduling algorithms and system constraints. Approaches include fixed-priority and dynamic-priority scheduling, with analysis methods such as response-time analysis and demand bound functions providing necessary and sufficient conditions for schedulability [21, 6, 11]. These analyses are essential for both uniprocessor and multiprocessor systems, and often require modeling task behaviors, resource contention, and system constraints in formal or algorithmic terms [9].

Traditional real-time schedulability analysis often relies on worst-case execution time (WCET) assumptions, which can lead to overly pessimistic results and unnecessary resource over-provisioning. Probabilistic schedulability analysis addresses this limitation by modeling execution times and other parameters as random variables, enabling a more realistic assessment of deadline miss probabilities and system performance.

Extensive research has focused on the real-time execution of periodic tasks on uniprocessor systems, typically using deterministic WCETs to guarantee deadlines [14]. While this approach ensures safety, it can be excessively conservative for many applications. Probabilistic real-time analysis offers a promising alternative by incorporating stochastic models for task execution times and soft real-time constraints, thus providing a more nuanced evaluation of system behavior [20, 15, 3, 14, 4]. When execution times are treated as random variables, methods such as response time distribution analysis and deadline miss probability estimation become feasible, though early work often assumed independence between tasks [13, 8]. More recent studies have begun to address dependencies and complex interactions, improving the accuracy of probabilistic models [18, 20].

3 Application and Architecture Model

3.1 Application model

The application model represents the system as a task graph $GT(V,E)$, where $V = \tau_1,\tau_2,...,\tau_n$ denotes the set of n tasks within the system (the graph's nodes) and $E = \{e_{ij}\}$ denotes the set of arcs indicating the dependencies between tasks. ${}^{\circ}\tau_i = \{\tau_i / i \in [1,..,n]\}$ is the set of predecessors of V and $\tau_i^{\circ} = \{\tau_i / i \in [1,..,n]\}$ is the set of successors of V.

A task τ_i can be an input task if ${}^{\circ}\tau_i = \varnothing$, and output task if $\tau_i^{\circ} = \varnothing$ or an internal task if ${}^{\circ}\tau_i \neq \varnothing$ and $\tau_i^{\circ} \neq \varnothing$.

e_{ij}=(τ_i, τ_j), τ_i and τ_j are respectively the predecessor task and the successor task.

e_{ij} represents an input channel for the task τ_j and an output channel for the task τ_i.

3.2 Architecture Model

If the application model enables us to describe the system, the architecture model is designed to support this description and show how the system is actually implemented. The architecture, represented as $Arch = U \cup C$, consists of a set of components U connected by communication channels C.

U = $U_{SW} \cup U_{HW}$ defines a set of hardware and software components. $U_{SW} = U_{SW1}, U_{SW2}, ..., U_{SWp}$ represents the set of p flexible components comprising the software part, which can include a general-purpose processor

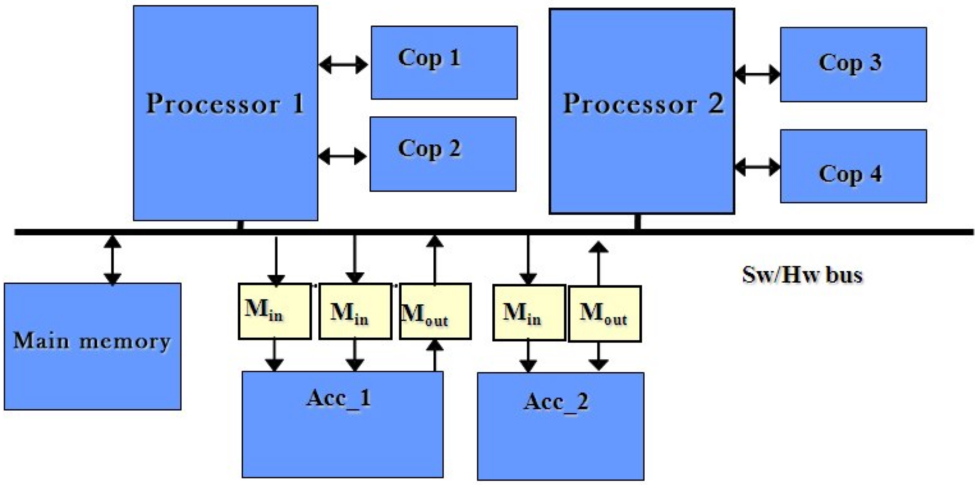

Fig. 1. Generic architecture model

(GPP) or a specialized processor (such as a DSP for signal processing applications). When the cardinality of U_{SW} is 1, $\mid U_{SW} \mid= 1$, the architecture is referred to as uniprocessor; otherwise, the architecture is multiprocessor. Additionally, $U_{HW} = U_{HW1}, U_{HW2}, ..., U_{HWj}$ defines the set of j components representing the hardware part. U_{HW} can include accelerators or IP.

$U_i = \langle \text{param}1, \text{param}2, \ldots, \text{param}\alpha \rangle$ is a vector of characteristics for component Ui. These characteristics include the name, type, and operation mode.

$C = \text{com}1, \text{com}2, \ldots, \text{com}_l$ is the set of communication components. These can include communication buses, memory, or registers.

Figure 1 represents a generic architecture. In our work, we focus on a single processor, so $\mid U_{\text{SW}} \mid= 1$.

3.3 Task Temporal Parameters and Probabilistic Execution Time

A real-time task is characterized by several temporal parameters: the release time (r_i), period (T_i), execution time (C_i), and deadline (D_i). Task τ_i is released at time r_i. The execution time C_i indicates the CPU time required for each instance of τ_i. The period T_i defines the interval between the activations of two consecutive instances, and each instance must finish within D_i time units from its activation.

This study considers the variability in job execution times. Execution time is influenced by many factors, each subject to change, making it more accurately modeled as a random variable. Thus, the execution time of a task is inherently probabilistic, reflecting the combined effect of several fluctuating characteristics [17].

The execution time C_i can be described as:

$$C_i = \begin{pmatrix} C_j \\ P(C_i = C_j) \end{pmatrix}_{j \in [1, j_i]}. \quad (1)$$

In equation (1), $C_j \in [C_i^{min}, C_i^{max}]$ and $j_i \in N^*$ denotes the number of distinct values that the random variable C_i can assume.

If $C_i^{min} = C_i^{max} = C_i$ then task τ_i is deterministic and we have: $C_i = \begin{pmatrix} C_i \\ 1 \end{pmatrix}$.

If $C_i^{min} \neq C_i^{max} \neq C_i$ then the task $\tau_i$is a random task.

Task execution time in real-time systems is best modeled as a random variable due to its dependence on multiple, variable factors. This probabilistic modeling is essential for accurate schedulability and timing analysis in both soft and hard real-time systems.

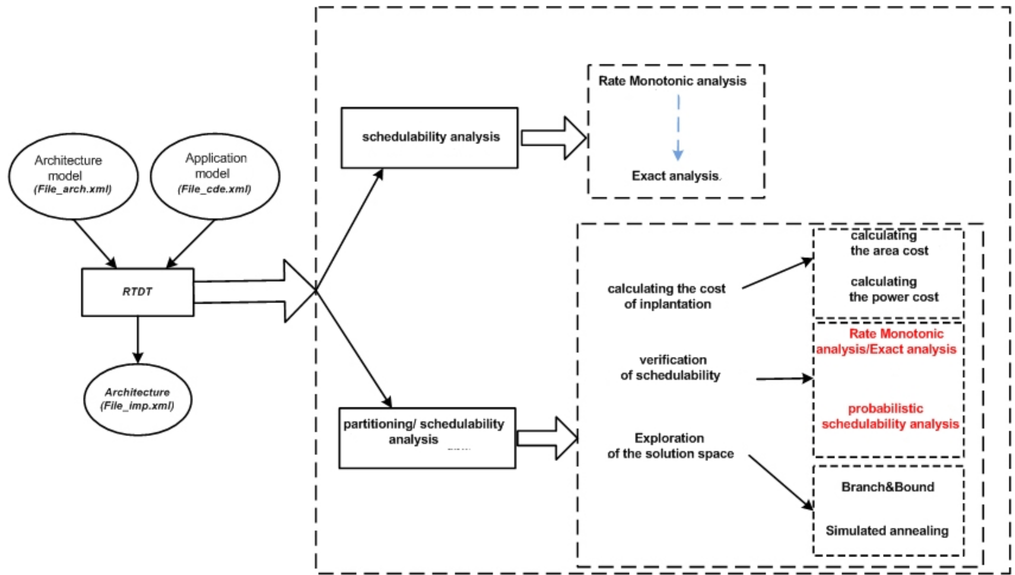

Fig. 2. RTDT codesign flow

```
Boolean Schedulable (S) {
U= ProcUseRate (S)
Step1: if (U >1)
        Return false
Step2: else if U ≤ n(2^(1/n) − 1)   return true
Step3 : else {
        For all τ_i by increasing Priority order
        Ri =ExactResponseTimeAnalysis ( τ_i )
        if Ri > Ti return false
        else return true
            }
}
```

Fig. 3. Test of schedulability [3]

4 Deterministic Real-Time Task Schedulability Analysis

One of the key challenges in real-time systems is determining the execution time of a task, which can vary depending on the input data. Relying on worst-case scenarios for timing estimates can result in overly expensive implementations.

The work in the field of schedulability analysis has led to the development of schedulability tests. Given a scheduling algorithm, such a test checks whether the temporal constraints of the tasks are respected.

The RTDT (Real-Time Design Trotter) framework is a tool designed to assist in the development and analysis of real-time systems ([3]). This framework provides an interactive interface for designers, allowing them to easily test various configurations of task versions and architectural implementations. It enables users to quickly evaluate the impact of probabilistic schedulability analysis on overall system costs (Figure 2).

The inputs to the RTDT tool are represented as two XML files. The input data is provided in the architecture file ($file\_arch.xml$) and the specification file ($file\_cde.xml$). The $file\_arch.xml$ file defines the architectural parameters, such as the Vdd/clock modes for both hardware and software components, the bus protocol, and other relevant settings. The $file\_cde.xml$ file includes the selected task constraints and a description of all task implementations.

ISSN 2007-9737

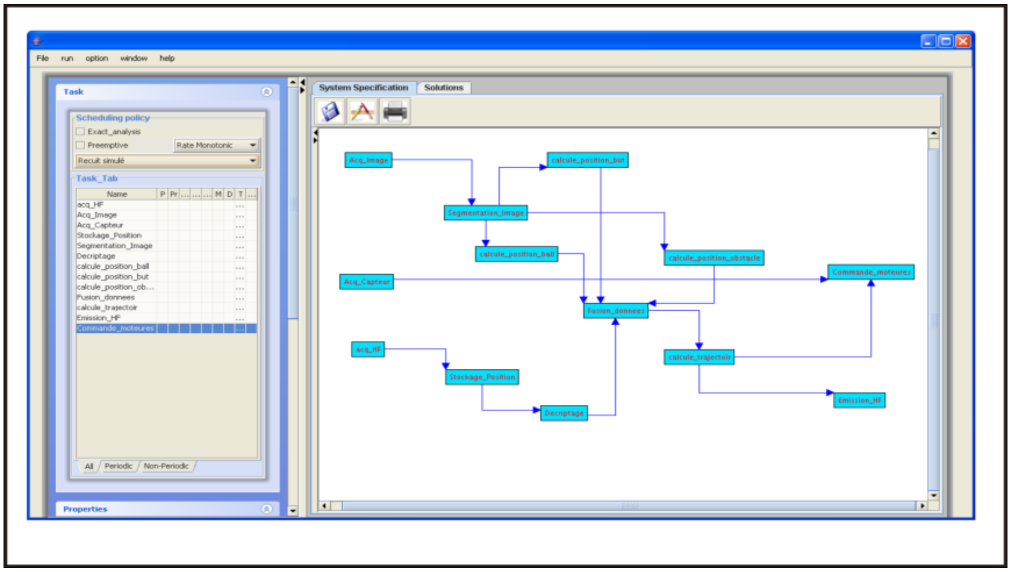

Fig. 4. Graph application task

Among the steps in RTDT, we focus on verifying schedulability during the partitioning and schedulability analysis phase. Schedulability is verified using a deterministic approach in the first case and a probabilistic approach in the second. This partitioning and schedulability analysis step generates optimized solutions that meet the timing constraints of tasks while minimizing area and power consumption. The various files $File\_imp.xml$ describe the chosen solutions and represent the tool's output.

In our work, schedulability verification begins by calculating the processor utilization (U) and determining whether it exceeds 1 ($U > 1$). If this is the case, the system is deemed not schedulable. Otherwise, if $U \leq 1$ the Rate Monotonic (RM) condition is checked, and if $U < n * (2^{1/n} - 1)$ the system is considered schedulable.

Finally, if $U < n*(2^{1/n}-1) < 1$, the schedulability of the software tasks is verified by calculating the Worst-Case Response Time (WCRT) of each task [3]. If any task fails to meet its deadline, the system is considered not schedulable (Figure 3).

We demonstrate the deterministic schedulability analysis using a case study of the ”football player robot” application [3]. The application's task graph is shown in Figure 4, with task characterization (e.g., timing and I/O data) provided in the architecture and estimation files.

We implemented two partitioning algorithms, Simulated Annealing and Branch & Bound, within the RTDT tool. The tool supports both Rate Monotonic and probabilistic analysis for schedulability analysis.

In this example, we show the exploration results generated by RTDT. These results were produced using the simulated annealing algorithm and deterministic analysis in Figure 5.

5 System Cost Implications of Probabilistic Schedulability Analysis

We demonstrate the impact of probabilistic analysis on final costs. The partitioning step, using probabilistic analysis, produces schedulable solutions as shown in Figure 6.

This figure demonstrates that both area and power costs are effectively reduced when probabilistic schedulability analysis is applied.

Two solutions were selected for each approach—deterministic and probabilistic—as illustrated in Figure 7.

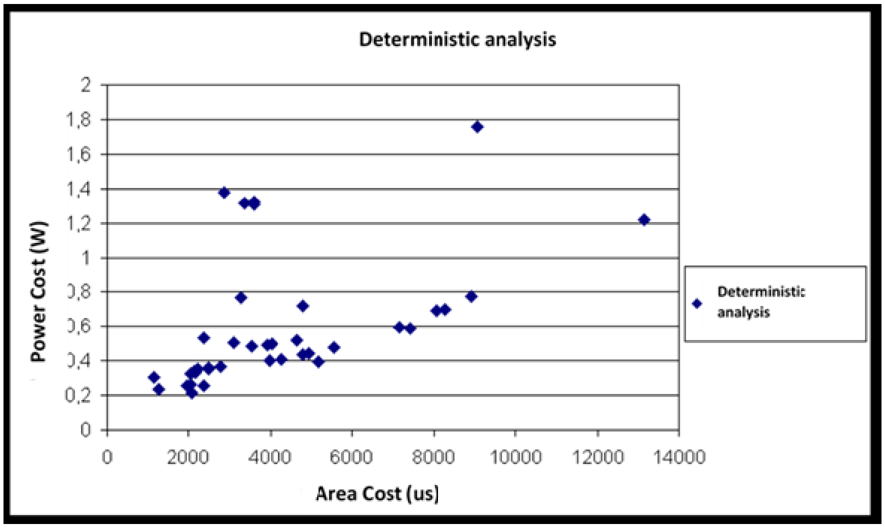

Fig. 5. Trade-off curve between area cost and power consumption cost for possible system implementations (Deterministic Analysis)

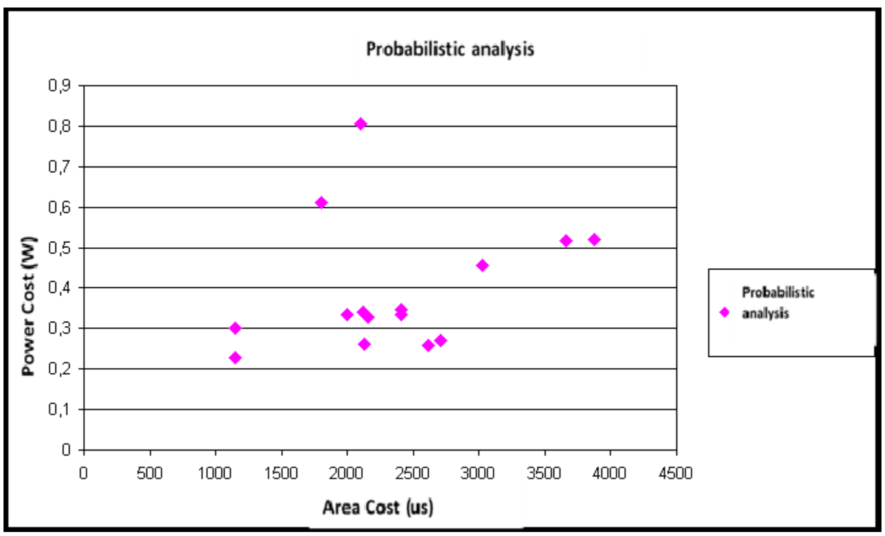

Fig. 6. Deterministic analysis/ Probabilistic analysis

In the second solution, we highlight the impact of probabilistic analysis.

As shown in the figure, the area cost and power consumption cost are reduced from 100% under deterministic analysis to 41% and 63%, respectively, when applying probabilistic schedulability analysis.

6 Conclusion

This paper has demonstrated that probabilistic schedulability analysis offers a practical path to reducing area and power consumption in real-time systems. By moving beyond worst-case guarantees, our approach unlocks optimizations that are impossible with purely deterministic

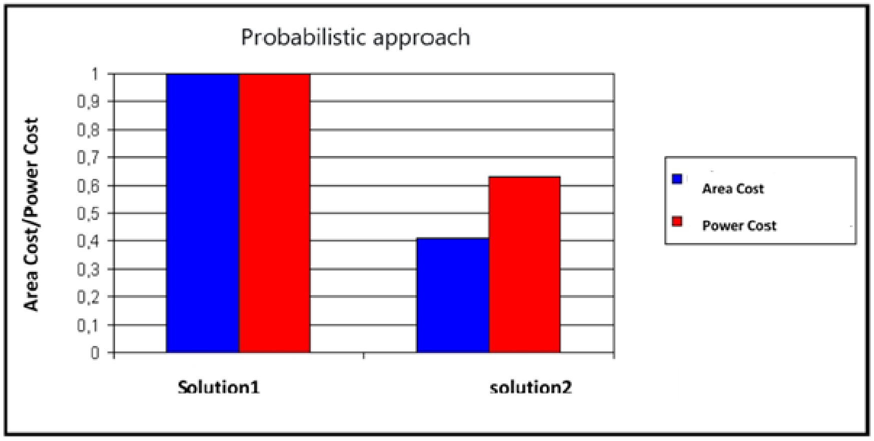

Fig. 7. Deterministic analysis/ Probabilistic analysis

methods. These findings not only contribute to more efficient system design but also pave the way for future research into refining probabilistic models and applying them to a broader range of complex, real-world applications.

References

1. **Burns, A., Bernat, G., Broster, I. (2023).** A probabilistic framework for schedulability analysis. Third International Embedded Software Conferences.

2. **Bernat, G., Colin, A., Petters, S.M. (2002).** WCET Analysis of Probabilistic Hard Real-Time Systems. 23rd IEEE Real-Time Systems Symposium.

3. **Tmar, H., Diguet, J.Ph., Azzedine, A., Abid, M., Philippe,J.L. (2006).** RTDT: A static QoS manager, RT scheduling, HW/SW partitioning CAD tool. Microelectronic Journal.

4. **Diaz, J.L., Garcia, D.F., Kim, K., Lee, C.G., Bello, L.L., Lopez, J.M., Min, S.L., Mirabella, O. (2002).** Stochastic analysis of periodic real-time systems. 22nd IEEE Real-Time Systems Symposium.

5. **Chen, K.H., Chen, J.J. (2017).** Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. 12th IEEE International Symposium on Industrial Embedded Systems (SIES).

6. **Guasque, A., Aceituno, J., Balbastre, P., Simó, J., Crespo, A. (2022).** Schedulability analysis of dynamic priority real-time systems with contention. The Journal of Supercomputing.

7. **Cucu-Grosjean, L. (2009).** Probabilistic real-time schedulability analysis: from uniprocessor to multiprocessor when the execution times are uncertain. INRIA Nancy Grand-Est.

8. **Cucu-Grosjean, L. (2013).** Independence – A misunderstood property of and for probabilistic real-time systems. Real-Time Systems: the past, the present and the future.

9. **Lee, H., Choi, J. (2020).** Constraint-Based Schedulability Analysis in Multiprocessor Real-Time Systems. IEEE Access.

10. **Lehoczky, J.P. (1996).** Real-time queueing theory. 17th IEEE Real-Time Systems Symposium, Washington, DC, USA, pp. 186–195.

11. **Ortiz, L., Guasque, A., Balbastre, P., Simó, J., Crespo, A. (2024).** Schedulability Analysis in Fixed-Priority Real-Time Multicore Systems with Contention. Applied Sciences.

12. **Rim, R., Tmar, H., Mahfoudhi, A. (2024).** Probabilistic Schedulability Analysis with De-

pendent Execution Times. Intelligent Systems Design and Applications, ISDA, Lecture Notes in Networks and Systems, vol 1050, Springer, Cham.

13. **Ben Amor, S., Maxim, D., Cucu-Grosjean, L. (2016).** Schedulability Analysis of Dependent Probabilistic Real-Time Tasks. 24th International Conference on Real-Time Networks and Systems.

14. **Manolache, S., Eles, P. Peng, Z. (2004).** Schedulability Analysis of Applications with Stochastic Task Execution Times. ACM Transactions on Embedded Computing Systems.

15. **Maxim, D., Cucu-Grosjean, L., Davis, R. (2022).** Probabilistic Analysis.

16. **Pailler, S., Choquet-Geniet, A. (2004).** Analyse hors ligne d'applications temps réel fortement couplées comportant des tâches à durées variables.

17. **Prabhakara, A., Steinwender, B., Elmenreich, W. (2021).** Statistical analysis of execution time profile for temporal validation of a distributed hard real-time system. 22nd IEEE International Conference on Industrial Technology (ICIT).

18. **Santinelli, L., Guo, Z., George, L. (2016).** Fault-aware sensitivity analysis for probabilistic real-time systems. 2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT).

19. **Cherubini, U., Luciano, E., Vecchiato, W. (2004).** Copula Methods in Finance.

20. **Abdeddaïm, Y., Maxim, D. (2017).** Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems. Design, Automation and Test in Europe Conference and Exhibition (DATE).

21. **Zhang, F., Burns, A. (2009).** Schedulability Analysis for Real-Time Systems with EDF Scheduling. IEEE Transactions on Computers.

Article received on 29/09/2024; accepted on 10/04/2025.
**Corresponding author is Rebai Rim.*