UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Analyse probabiliste des systèmes temps réel

THÈSE

présentée et soutenue publiquement le 10 December 2013

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Dorin MAXIM

Composition du jury

Président :	Stephan MERZ, DR, Inria Nancy-Grand Est, France
Rapporteurs :	Marco Di NATALE, Professeur, École Supérieur Sant'Anna, Pisa, Italy
	Isabelle PUAUT, Professeur, Université Rennes I, France
Examinateurs :	Mihaela BARONI, Professeur, Université Dunarea de Jos, Romania
	Laurent GEORGE, MCF HDR, Université Marne la Valée, France
	Thomas NOLTE, Professeur, Université Mälardalen, Sweden
Encadrants :	Liliana CUCU-GROSJEAN, CR, Inria Rocquencourt, France
	Françoise SIMONOT-LION, Professeur, Loria, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Sommaire

To my brother Cristian,
my mother Aneta,
my father Ioan
and to all my family.

Remerciements

I am forever grateful to all those that I have met on my journey towards becoming a doctor of the University of Lorraine. Everyone has had an impact on me, sometimes without even realizing it, and for this I am thankful.

I want to thank Liliana Cucu-Grosjean for guiding me on this journey, for her support in all my endeavors, but also for letting me create my own path, experiencing first hand the world of research and learning the lessons that I needed to learn in order to become a doctor, even if sometimes I had to learn the hard way. I also thank Francoise Simonot-Lion for her support throughout the thesis and for making me feel welcomed in the team even since I was an intern.

I am grateful to Mihaela Baroni, the one that put everything in motion. Without her, none of this would have been possible. Likewise, a big thank you goes to Adriana Gogonel.

I will always remember the people I have met along the way, both inside and outside the laboratory of Inria Nancy Grand Est. Most notably I want to thank the members of the TRIO team for making it such a nice working environment, so that no mater how difficult it got at times, I was still happy to go to work. I recall here Aurelien Monot, Adrien Guenard, Pierre Caserta, Lionel Havet, Bilel Nefzi, Dawood Khan, Luca Santinelli, Patrick Yomsi, Maha Idrissi, Najet Boughanmi, Nicola Navet, Olivier Zendra, Laurence Benini, Marie-Francoise Loubresac and Sophie Drouot.

A special thanks goes to Rob Davis from the University of York in UK, for the many discussions we had and from whom I have learned so much. He is a true model.

I thank Isabelle Puaut and Marco di Natale for honoring me by being reviewers of the thesis, as well as Laurent George, Thomas Nolte and Stephan Merz for taking part in the jury. I thank them for the time they put in to evaluate my work and participating in the thesis defense, it honors me greatly.

Last, but not least, I am forever grateful to my brother, my parents and all my family, for their unconditional love, undying support and all the sacrifices they have made in order to offer me the best education, without which I wouldn't be here. Having my brother, Cristian, by my side has made my journey so much more enjoyable and easier.

General introduction

1 Context and motivation

Since 1973 when Liu and Layland published their seminal work on "Scheduling algorithms for multi-programming in a hard-real time environment" [Liu and Layland, 1973], research in the real-time domain has gone a long way, its applications being used in many branches of the industry. The domain of real-time systems has greatly evolved and gave birth to new directions of research so that nowadays it is not uncommon to find publications referring to hard real-time systems, soft real-time systems, critical real-time systems, mixed-criticality real-time systems, multi-mode real-time systems and more. Even the newly emerging field of cyber-physical systems owes its birth to the research in the real-time systems domain, to which it is closely related.

Amongst all branches of real-time systems, the most important one is, probably, the Critical Real-Time Embedded Systems ($CRTES$) domain. $CRTES$s are widely being used in fields like automotive, avionics, railway, health-care, etc. The performance of $CRTES$s are analysed not only from the point of view of their correctness, but also from the perspective of time.

The timing analysis of such systems has been extensively studied by considering deterministic approaches based on worst case scenarios. As the industries making use of $CRTES$s are demanding more functionalities which can only be provided by using advances high performance microprocessors, building real-time systems with reliable timing behavior represents a considerable challenge.

The wide variability of execution times due to aggressive hardware acceleration features like cache and deep memory hierarchies implies that deterministic approaches to Worst-Case Execution Time (WCET) analysis and Worst-Case Response Time (WCRT) analysis may indicate that timing constraints will not be met, when in practice the probability of a deadline miss actually occurring in the lifetime of the system is vanishingly small. In this way, deterministic analysis may lead to significant over-provision in the system architecture, effectively placing an unnecessarily low limit on the amount of new functionality that can be included in a given system.

Also, in order to apply worst-case analysis to a system, the systems parameters need to be considered in their respective worst case values or using a safe upper-bound/lower-bound of those values, which is not always possible. It is possible that some parameters may remain unknown until the instant when the system is running and tasks are instantiated. As well, the environment, in which the system is running, can change and so the system/application is forced to adapt by changing its parameters. Determining worst-case values for this kind of systems may be difficult or even impossible and using safe bounds when analysing the system introduces increased levels of pessimism.

Another case where deterministic analysis may not be efficient is the case of event triggered systems which interact with the real world. For this kind of systems it is not always necessary that an useful bound is placed on the arrival rate of jobs generated by interrupts from external sensors or network interfaces, which may not even have such a bound [Broster and Burns, 2004a]. The best known examples of systems with streams of jobs that arrive in a random fashion are controller area networks with faults and specifically faults generated by electro magnetic interference (EMI) [Broster and Burns, 2004a], [Navet et al., 2000].

Different industries may also build probabilistic real-time systems on purpose by introducing some parameters with random behaviour. For instance, some automotive manufacturers have randomised the sampling frequency for the reverse parking ultrasound sensor in order to avoid the situation when two vehicles reverse back-to-back as in Figure 1 and they both having the same sampling frequency reduces the efficiency of their parking sensors [Buttle, 2012]. By randomising the sampling frequency, the jobs that are generated by the sensor have a random arrival pattern. These jobs belong to a task that can be seen as a sporadic task with its period equal to the minimum inter-arrival time (MIT) amongst its jobs. Knowing that job arrivals are random, then we may describe the MITs by distributions, i.e., the arrival distribution of the generated jobs, which gives a more accurate description of the generated jobs.

The above are few examples of systems for which the worst case analysis is not only difficult to apply (or even impossible) but even when it is applied it is not

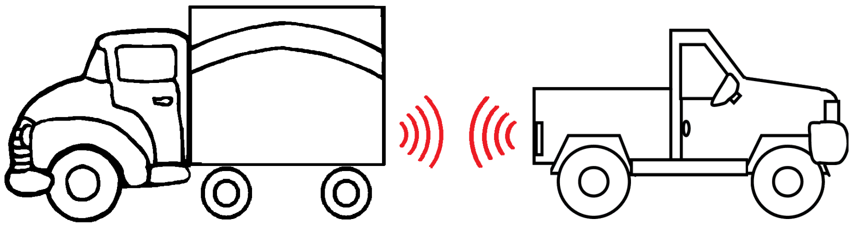


FIGURE 1 – When two vehicles reverse back-to-back, their respective parking sensors may sample at the same frequency, making them less efficient, or even blind to one another.

efficient. Here by efficient we understand that the worst case analysis, which only returns a "YES" or "NO" answer, can at most say about the system that there are no guarantees. While the analysis is correct, there are no guarantees in the sense that there exist scenarios in which deadlines might be missed, the probability of these scenarios actually happening is extremely small while the scenarios in which the system runs correctly are much more frequent. In this way, the worst case analysis fails to adequately describe the system.

From a practical point of view, system manufacturers are in the situation that they can not certify a system because it is deemed unfeasible by the worst-case analysis even though the system is functioning correctly and without (or very little) faults and all experimentation, simulation and human experience confirm that it is in fact a feasible system. In the best case, manufacturers end up limiting the functionality integrated in the system, severely over-provisioning it.

An alternative approach is the use of probabilistic analysis. System reliability is typically expressed in terms of probabilities for hardware failures, memory failures, software faults, etc. This approach also extends to the time domain. For example, the reliability requirements placed on the timing behaviour of a sub-system implemented on an advanced high performance microprocessor might indicate that the timing failure rate must be less than 10^{-9} per hour of operation. Probabilistic analysis techniques that seek to meet this requirement, rather than attempting to provide an absolute guarantee, have the potential to outperform deterministic techniques.

In recent years the probabilistic analysis of real-time systems gained ground, several research works addressing the problem from different points of view. Notable advances have been made in analysing systems that have a single source of variability, mostly at execution time level or at arrival frequency level (more details about the advances in the probabilistic analysis domain are given in Chapter 1). Even though advances have been made in analyzing systems with only one source of variability, there is little progress towards efficiently analyzing more complex models of probabilistic real time systems, i.e. models that allow for more than one sources of variability at the same time.

Ultimately, a complete probabilistic analysis needs to take into account all possible sources of variability present in the system. In order to express these variabilities in the system behaviour, the worst case assumptions need to be relaxed and the model generalized.

New models for real-time systems have been proposed in order to relax the restrictive assumptions and to better take into account the inherent variability of the system. The *implicit deadline task model*, also known as the Liu & Layland task model, is the first and simplest model, a task being described by two parameters, its execution time and its period (its deadline is considered equal to its period). The periodicity assumption was relaxed in the *sporadic task model* which allows tasks to have variable periods (formally called *inter-arrival times*). This model is, in the worst case, equivalent to the periodic model in the case of one processor, for instance. Another shortcoming of the sporadic task model is that it does not express in any way the variability at execution time level.

The next level of generalization was attained by the *multi-frame* (MF) task model, the *generalized multi-frame* (GMF) task model and the *recurrent real-time* (RRT) task model, which allow for different jobs of a task to have different execution times, different inter-arrival times and different deadlines, but only in a cyclic fashion. In the MF and GMF models if a task has n possible types of jobs, than, the parameters of job i are the same as the parameters of job $i + n, \forall i$. In the RRT model branching was allowed but the cyclicity was not eliminated.

The assumption of cyclicity of the GMF task model implies that instructions of the tasks' code are always executed in the same order and this is not necessarily the case since there are factors that can affect which instructions are performed, how many times and in which order. The input received by the system is an example of such a factor.

This cyclicity restriction was later removed by the non-cyclic GMF (NCGMF) task model, the non-cyclic RRT task model (which is a combination of the NCGMF and RRT models) and the digraph real-time task model.

A hierarchy of the afore mentioned models is presented in Figure 2. The higher it is the model in the hierarchy, the more general and more expressive the model is.

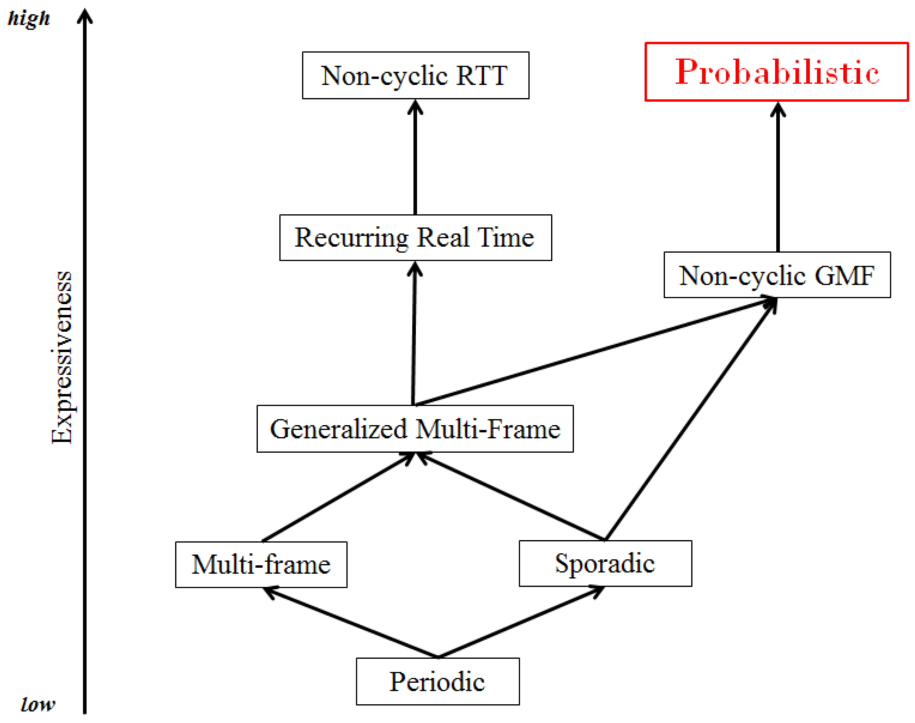


FIGURE 2 – Hierarchy of models used to represent real-time tasks. Arrows indicate generalization relationships. The higher it is the model, then the more general and more expressive the model is.

We have placed the probabilistic task model at a high level of expressiveness and we think that it has the potential to be one of the most expressive model up to date since it has one more degree of freedom compared to the other models. For instance, the execution time of a task in the probabilistic model can be completely decoupled from its inter-arrival time (the probabilistic task model is explained in detail in Chapter 2.2) so that the model can represent any combination of execution time, inter-arrival time and deadline that a job might have, whereas, in all the other models these combinations need to be explicitly described by tuples and precedence relationships. This can prove to be a considerable limitation in the models expressiveness for systems that allow for many such combinations of job parameters. We have placed the probabilistic model as a generalization of the NCGMF because we believe that the probabilistic model, that we are interested in, is the closest to the NCGMF model, i.e., jobs generated by a task are independent from one another. A probabilistic model that allows dependences amongst jobs of the same tasks would be closer to the RRT task model.

Even though the tendency is towards models that can better express the variability in the system, the most of available analyses are based on a worst case perspective, i.e., a system is declared feasible only if it can be guaranteed that no deadline is missed, regardless of how small is the occurrence probability of a scenario in which a deadline is actually missed.

The work presented in this thesis introduces new analysis techniques for probabilistic real-time systems as well as optimal scheduling algorithms for these systems and efficient techniques of reducing the analysis complexity.

2 Contributions

The main contributions of our work can be summarized as follows :

Optimal scheduling algorithms for probabilistic real-time systems : For tasks systems that have the execution time given as a random variable, we have identified three scheduling problems for which we have proposed optimal scheduling algorithms (optimal in a probabilistic sense). We note here the important contribution which is the observation that, in the case that jobs are allowed to execute past their respective deadlines, the relative order of tasks in the higher priority set does not influence the response time distribution of the task under analysis.

New models for probabilistic real-time systems : We have proposed a generalization of the implicit deadline task model that has all three task parameters described as probability distributions, namely the execution time, the inter-arrival time and implicitly the deadline which is given by the instant when a new job is realised and has the same distribution as the inter-arrival times.

New schedulability analysis techniques for probabilistic real-time systems : We have introduced new analyses techniques for probabilistic real-time systems in two directions :

- analysis for controller area networks for which the jitter is represented as a random variable

– analysis for real-time systems with multiple parameters given as random variables, namely any combination of probabilistic execution times, probabilistic inter-arrival times and probabilistic deadlines.

Techniques for reducing the complexity of the analysis : We have introduced and studied the effect of several re-sampling techniques that reduce the complexity of the analysis by reducing the sizes or random variables while introducing minimal pessimism.

3 Organization of the thesis

We first present the recent developments in the domain of probabilistic real-time systems analysis in Chapter 1. In Chapter 2 we present our system model, as well as the steps of validating a probabilistic analysis of real-time systems. We proceed afterwards with the presentation of optimal algorithms for real-time systems in Chapter 3 and the analysis of controller area networks with probabilistic jitter in Chapter 4. We then give a new model of real-time systems with multiple probabilistic parameters and its analysis in Chapter 5. In Chapter 6 we present re-sampling techniques for execution times. We conclude the thesis in Chapter 6.9.

Chapitre 1

State of the art

We present in this chapter the main work on probabilistic real-time systems that, at our best knowledge, is published until now. We organize the presentation of the results according to the parameters that are described by random variables. For instance, in Section 1.1 we present the results for systems with the (worst case) execution times described by the random variables in the case of one processor. We put together in the same section all results for the case of several processors (see Section 1.4). Some results do not follow this classification and we present them separately in Section 1.5. Even if this thesis does not contain results on the derivation of distributions for the parameters of a probabilistic real-time system, we present in Section1.6 a summary of such results for an interested reader.

1.1 Probabilistic execution times

In [Tia et al., 1995] the authors present an analysis for semi-periodic tasks, that is, tasks that have periodic releases but their execution times vary. The analysis is called Probabilistic Time Demand Analysis (PTDA) and computes the worst case probability that a task in the system misses its deadline by bounding the total amount of processor time demanded by all higher priority tasks. The proposed algorithm performs convolutions if there are at most 10 random variables to be combined, otherwise it uses the central limit theorem to approximate the response time with a normal distribution whose mean and variance are, respectively, the sums of the means and variances of all random variables in the sum. Since the algorithm is based on a bound of the processor time demand of higher priority tasks, it is highly pessimistic compared to more recent developments like [Gardner and Lui, 1999] and [Díaz et al., 2002].

In [Gardner and Lui, 1999], the authors present a stochastic analysis for real time systems that have probabilistic execution time. The analysis, called STDA (Stochastic Time Demand Analysis), computes a lower bound on the probability that jobs in each task will meet their deadlines. The proposed analysis is the basis for future developments, like the work of Diaz et all [Díaz et al., 2002] who refined the analysis into an exact one.

In [Díaz et al., 2002], [Kim et al., 2005], [Lopez et al., 2008] the authors present an exact analysis for real-time systems that have random execution times. The execution time is represented as a general random variable and the priorities may be job-level or task-level. The analysis is proven to be bounded in time and exact for both cases when the system utilization is lower or greater than one. The proposed analysis was compared to that of Gardner and Lui [Gardner and Lui, 1999] and has been shown to be more precise, Gardners' analysis providing results six times larger than the exact ones. Due to the cost of convolution, the proposed analysis can only be applied for small task systems - this problem was later studied in [Refaat and Hladik, 2010] and [Maxim et al., 2012b]. Also, the system model on which the analysis can be applied is restrictive in the sense that, except for the execution time, it does not allow for other sources of variability, such as variable minimum inter-arrival time, variable deadline, etc.

In [Diaz et al., 2004] the authors further refine their analysis by bringing into discussion the concepts of pessimism and optimism, relations between two random variables, as well as truncating the tails of execution time probability distribution and moving probabilities from small values to large values of the execution time distribution, which is a pessimistic way of reducing the analysis cost, later known as re-sampling. The paper also treats the application of the pessimism concept on the blocking in shared resources and on the priority assignment policy of the system under analysis. The authors also state that Audsley's' algorithm [Audsley, 1991,Audsley, 2001], which is optimal in the deterministic case, is also optimal in the stochastic case. The proof is left for the reader to construct based on the lemma that decreasing the priority of a task never increases its schedulability in the stochastic scenario (similar to the deterministic case). The work, while ample and rich in contributions, still leaves many questions unanswered, such as the quantification of the pessimism introduced in the analysis by the approximation methods used, i.e., truncation and re-sampling, the proof that Audsley's' algorithm is optimal should be fully stated and also, the types of problem on which the analysis and the scheduling algorithm can be applied are not explored. As well, the model of the system under analysis is not clearly described, in the sense that it is not mentioned if the jobs that fail to reach their deadline are kept in the system until they finish their execution or they are evacuated at the moment they breach their deadline. This specification is important for the analysis since, evicting jobs at deadline breach makes Audsley's' algorithm to be no longer be optimal.

In [Refaat and Hladik, 2010], the authors propose a refinement of the existing stochastic analysis by means of re-sampling of the values in the worst case execution time distribution. That is to say, the worst case execution time distribution is reduced in size, by selecting a subset of values to be kept from the original set of values, removing the unselected values and re-distributing their probability mass to the selected values. The proposed re-sampling techniques have two major shortcomings : *a)* the samples to be kept are randomly selected by assigning a selection probability to each values of the original distribution, and *b)* the probability mass of the unselected values goes entirely to the largest value of the distribution (the worst case execution time) thus greatly increasing the introduced pessimism, as opposed to re-distributing the probability mass to the values that were chosen to be kept.

In [Burns et al., 2003] and [Bernat et al., 2005], the authors provide a probabilistic analysis framework for systems with tasks that have stochastic execution times, given as random variables following some probability distribution derived from measurement. The particularity of this work is the fact that the basic blocks which are analysed can be correlated. The correlations between blocks is solved by use of copulas, a mathematical tool that investigates dependence structures between random variables. When dependences are not known, copulas produce upper and lower bounds of a joint distribution function of two correlated distribution functions such that it incorporates *any* possible dependences between the two distribution functions.

In [Manolache et al., 2004], the problem of uniprocessor scheduling of tasks with stochastic execution times is studied. The scheduling policies taken into consideration are non-preemptive, and tasks have precedence constraints between them. The tasks execution times are given as generalized probability density function and assumed independent from one another, but the periods of tasks are assumed to be harmonic, i.e., the period of one task is a common multiple of all periods of its predecessor tasks. Also, the system model assumed is one in which the maximum number of instances of a task-graph that can exist in the same time is specified by the system designer, and the case that is studied is the one in which only one instance of the task-graph is in the system at a given time, i.e., the system is reseated at the beginning of each hyper-period. This assumption does not diminish the correctness of the proposed solution for the case of non-preemptive scheduling, but the solution does not apply for the case of preemptive scheduling, or in the case of tasks that do not have precedence constraints.

In [Singh and Jeffay, 2007], the authors present a processor sharing scheme for systems comprised of both Real-Time and Non Real-Time tasks, with the particularity that for the real-time tasks the execution times are given by their worst case, deterministic values, whereas the non real-time tasks have more detailed execution time requirements, given as random variables. A theoretical comparison is done between Fixed Priority scheduling, Generalized Processor scheduling, Earliest Deadline at the Latest (EDL) and the authors propose a scheduling policy. The conclusion of the comparison is that the proposed algorithm exhibits the best behaviour, achieving better performances for both the real-time tasks and the non real-time tasks. The comparison is done only for the case when the system has two tasks, one real-time and one non real-time, a way of generalising to multiple tasks being presented afterward. Whilst the proposed solution is a step forward in applying probabilistic analysis to real-time systems, it is still pessimistic, sacrificing the functionality of the non real-time tasks in order to guarantee the constraints of the real-time ones, which are considered to have a deterministic worst case execution time. A great deal of improvement could be achieved by taking into consideration the probabilistic nature of the real-time tasks, which also can have variable execution times.

In [Hu et al., 2001], the authors are concerned about the feasibility of a system as a whole as opposed to the feasibility of separate tasks. The considered system if formed by periodic tasks that have probabilistic execution times given by random variables. Apart from these details, the description of the system under analysis may be improved, for example the algorithm for computing response time distributions is not presented, nor is it explained if jobs are discarded at deadline miss or if they are allowed to continue their execution - this specification has an impact on the nature of the analysis. Also, the motivating examples might not be valid, since the response time distributions presented are different than the ones that would be obtained by applying the analysis framework in [Díaz et al., 2002] and subsequent, which have been proved to be safe.

In [Axer and Ernst, 2013] the methodologies introduced by Diaz et al. [Díaz et al., 2002], [Kim et al., 2005], [Lopez et al., 2008] are applied to the analysis of Controller Area Networks with faults. The considered system is a deterministic one according to the CAN model, with the exception of the faults that can occur in the system which are considered to have a probabilistic arrival following the Poisson distribution. If an error is detected for frame $\tau_{i,j}$, then this frame needs to be retransmitted and so its execution time, which initially was C_i, now becomes $2(C_i)+E$ where E is the worst-case error signaling overhead. The same frame can be affected by errors several times, so its execution time after k errors can be computed as $C_i + k(C_i + E)$. Knowing the probability that $\tau_{i,j}$ suffers k errors, then one can also obtain the probability of requiring an execution time equal to $C_i + k(C_i + E)$. In this way a probability distribution function representing the execution requirement of τ_i has been obtained and by using the *convolution, split* and *merge* functions, the probability distribution function of the level i busy period as well as the probabilistic worst case response time of τ_i is further obtained.

In [Atlas and Bestavros, 1998], Rate Monotonic (RM) scheduling is extended in a statistical fashion in the sense that tasks have variable execution times and admission control of a job in the system is based on the likelihood of it finishing execution before its deadline. The conducted simulations show that Statistical Rate Monotonic Scheduling (SRMS) performs better than RM scheduling in overload conditions but not before overload occurs.

1.2 Probabilistic Inter-Arrival Times

In [Broster et al., 2002] the authors present an analysis framework dealing with random faults on CAN. The faults arrive according to a Poisson distribution, but the analysis can only cope with a single stream of faults, i.e., all faults are equivalent, so they are considered as instances of the same process. Also, the proposed method is based on analysing each scenario that could occur, i.e. the scenarios that there are $0, 1, 2, \cdots$ fault arrivals in the analysed time frame, the only way that the analysis can stop is when the considered time interval goes up to the deadline of the message under analysis, and the said message executes up to its deadline (or any other time limit imposed by the system designer). This characteristic of the analysis restricts the system model to system where the messages are aborted when they miss their deadline. In the case of systems where messages are kept even after they have executed past their deadlines the analysis would never stop, except for the case when an artificial time limit would be imposed, in which case the analysis would return an incomplete response time distribution.

In [Broster and Burns, 2004a] and [Broster and Burns, 2004b] the authors provide an intuition of how the random arrival model presented in previous work can be applied to fixed priority preemptive uni-processor scheduling. The main difference between faults arrivals in controller area networks and random job arrivals is that, regardless of their sources, the generated faults are all considered equivalent and so only one stream of faults applied in the analysis. In the case of one processor, the jobs with random behavior can be generated by multiple sources like external interrupts, network interfaces, etc, and have different characteristics. The previous analysis needs to be generalized in order to handle multiple streams of random arrivals.

In [Cucu and Tovar, 2006], a framework is presented for computing response time distributions in the case when in the system there are tasks that have random arrivals, given as independent discrete random variables. The rest of the parameters of the tasks are deterministic. The output of the analysis is the response time probability distribution of the first release of an analysed task, considering that all tasks are released synchronously. The analysis is bounded in time, being polynomial in the value of the deadline. That is, the analysis stops when the job under consideration reaches values of response time that are equal to its deadline. As it is the case of the analysis proposed by Broster et all [Broster and Burns, 2004b, Broster and Burns, 2004a], this assumption restricts the system model to only those systems where jobs are evicted at deadline, and excluding the systems where jobs are allowed to continue execution even after their deadline.

1.3 Multiple Probabilistic Parameters

In [Kaczynski et al., 2007] and [Kaczynski et al., 2006], the authors present an analysis for hybrid probabilistic systems which may include periodic, sporadic and aperiodic tasks. Tasks' execution times are given as execution time profiles (ETP)

given as random variables. The sporadic and aperiodic tasks are considered to have arrival profiles (AP) as well given by random variables, specifically representing the number of task activations during a fixed time interval. All random variables' distributions are considered known. The solution proposed is based on a Polling Server extending the work presented in [Díaz et al., 2002]. A method for obtaining ETPs for servers used to encapsulate hybrid task-sets is developed and presented. The method is simulated and shown to have a high level of accuracy both compared to the worst case analysis and to the probabilistic analysis presented in [Díaz et al., 2002] where tasks' periods are considered as deterministic.

In [Abeni et al., 2012] and [Manica et al., 2012], the authors present an analysis framework for tasks with probabilistic execution times and random arrivals. The task system is running on a preemptive uniprocessor according to a Constant Bandwidth Server based on Earliest Deadline First.

In [Lehoczky, 1996], Lehoczky presents an analysis for tasks that have arrivals according to a Poisson distribution process with rate λ and exponentially distributed execution times. These constraints are highly constrictive and not applicable for task-sets that have arrivals and/or execution requirements given by other distributions. Within these hypothesis, a lead-time profile is predicted which can then be compared with the empirical distribution seen during the simulation of the system. Even for these tight constraints there are no guarantees that the prediction will accurately reflect the state of the system, and, as seen from the presented simulations, there is an important difference between the two distributions and there is no metric that computes this difference.

1.4 Probabilistic Multiprocessors Analysis

In [Nissanke et al., 2002], the authors present a probabilistic framework for analysing global performance issues in multiprocessor scheduling environments. The tasks are considered to have probabilistic execution times and probabilistic arrival times, and also the number of processors available at a time instance can be variable. The possible number of tasks in the system at a moment in time is computed as a probabilistic quantity, and the analysis is performed based on the execution requirements of each task and its laxity. The laxity of each task is computed based on the execution time and deadline, hence it is also a probabilistic quantity. The authors provide formulas for the computation of failure rates and success rates on a wide system scale, and for particular tasks. The complexity of the analysis is not taken into discussion, which is a necessity since the analysis seems to be complex. Also, the paper could be improved by providing more details on a possible theoretical or experimental validation.

In [Manolache et al., 2002] the authors present a schedulability analysis for multiprocessor systems with tasks characterized by probabilistically distributed execution times. There is no assumption on the distributions that the analysis can cope with, i.e., arbitrary distributions are considered, but they are approximated with Coxian distributions (weighted sums of convoluted exponentials) in order to reduce the complexity of the analysis. The considered system is a task-level fixed priority, non-preemptive multiprocessor modeled as a task-graph and the analysis outputs the deadline miss probability of the task-graph.

1.5 Other relevant contributions in the probabilistic analysis of real-time systems

In [Abeni and Buttazzo, 1998] and [Abeni and Buttazzo, 1999], the authors present an analysis of real time systems which have probabilistic characteristics : either probabilistic execution times given as PDFs (probability density functions), either probabilistic inter arrival times given as PDFs. The problem of tasks that have both probabilistic execution times and probabilistic inter arrival times is left as future work. The priority assignment policy used is Earliest Deadline First, but the deadline is not a characteristic of the tasks, as it is chosen by the system designer, which makes the choice of deadlines an important problem, since it ultimately influences the priority of each task and the run time behavior of the system. The analysis is stable under the assumption that the average utilisation of the system is less than one.

In [Burns et al., 1999], the authors present an analysis framework for real time systems with faults. The system under consideration is a set of real time tasks with deterministic execution times, periods and constrained deadlines, that are scheduled preemptively on one processor. The particularity of the system is that faults can appear in a random manner, so the arrival times and the execution requirements of these errors are not known. Sensitivity analysis is used to answer questions regarding the maximal frequency and execution requirements of such faults that can be handled by a feasible system before it becomes unfeasible.

In [Navet et al., 2000], Navet et al. introduce the notion of Worst Case Deadline Failure Probability and present a probabilistic model for the errors that can occur in a CAN network, especially the errors caused by electro magnetic interference which have a probabilistic nature. Also, the authors present a numerical efficient analysis framework for computing the tolerable error threshold under which message deadlines are still met on the CAN network. All the parameters of the messages are taken with their worst case values, except for the errors that can arrive according to a probability distribution. The generalization of the analysis framework to the case when other parameters have probabilistic natures, for example probabilistic execution times, is left as future work.

In [Baruah et al., 1999], a model of tasks is proposed such that different jobs of the task can have different execution times (WCET), different periods (MIT) and different deadlines (D) and these values are cyclically taken from a set of possible values. The jobs WCET, MIT and D are not probabilistic but deterministic, and it is known exactly what these values will be for each job of each task. We consider that the model in [Baruah et al., 1999] is related to what we propose, in the sense that the probabilistic model is a further relaxation of the Liu and Layland model [Liu and Layland, 1973] in the same direction as the generalized multi-frame (GMF) task model. Later on, Moyo et al. [Moyo et al., 2010] generalized even further the multi-frame task model by relaxing the cyclicity constraint present in [Baruah et al., 1999]. In this way, Moyo et al. present the model of the Non-cyclic Multi-Frame (NCGMF) Task with the particularity that at the end of a frame there is a release of another frame but this frame can be any of the possible frames that the task can generate. This task model can in fact be viewed as a probabilistic/random task model, since the arrival of frames is random, but with the particularity that the arrival distributions are not known and there are correlations between execution times, periods and deadline. Indeed, if we consider a non-cyclic GMF task system such that $\forall i \in 1, n$, the periods of all the frames of task τ_i are equal amongst them selves and the deadlines of all frames of task τ_i are equal amongst themselves, then we obtain a a task system that fits the model presented in [Díaz et al., 2002] with the exception that the execution time distributions of the tasks are not known. Similar transformations can be made so that the non-cyclic GMF tasks fit a model with probabilistic arrivals or one with a probabilistic deadline. Also, the non-cyclic GMF task model can further be relaxed so that frames need not have fixed parameters, but when a frame arrives at the processor it chooses its execution time, period and deadline from certain intervals. This generalization would make the model equivalent to a probabilistic one in which there are no correlations between execution time, periods and deadlines. Moyo et al. study the feasibility problem of a non-cyclic GMF task set scheduled according to the Earliest Deadline First Policy providing a sufficient schedulability test based on the density of the system, concluding that a NCGMF task set is feasible under EDF if its worst case (maximum) density is less or equal to 1. They also provide an efficient approach for building an execution tree of the system in order to compute its worst case response time and perform an exact schedulability analysis of the task set. The performance of the method is compared to the performance of an exhaustive search of all possible execution paths and it is shown that the number of scenarios that need to be considered is greatly decreased in the efficient method, but this number is still exponential in the number of tasks, even though a lesser exponential then the exhaustive search. In [Stigge and Yi, 2012], the authors have shown that there can not be any pseudo-polynomial exact test for fixed-priority task-level scheduling of the non-cyclic GMF task model, or any more general model, such as the probabilistic task model that we are considering in this work.

In [Zeng et al., 2009b], [Zeng et al., 2009a] and [Zeng et al., 2010] the authors present different techniques, be it stochastic or statistic, to predict response times distributions of messages on a controller area network (CAN) based on simulation data on a reference CAN bus system. The statistical analysis is suitable for when there is missing information of the system parameters and it is best applied in the early stages of the system design when decisions need to be made without knowing all the details of the system. On the other hand, the stochastic analysis is applicable when the complete message set description is available, providing more accurate results but being significantly slower and more suitable for later stages of the design process.

In [Lu et al., 2012] and [Lu et al., 2011] the authors present a statistical response time analysis for real-time systems, based on a black-box approach, i.e., the analysis doesn't require estimates of tasks' parameters, such as worst case execution times, inter-arrival times, etc. The analysis provides a safe and tight upper-bound on the worst case response time (WCRT) estimate of the task under analysis, based upon, as the authors say, a "*calibrated* and *tight* predictive Probability Density Function (PDF) histogram of the task" meaning that the actual system that needs to be analysis is simulated and the response times of the task under analysis are recorded as a PDF which is then fitted to a Gumbel Max distribution by means of extreme values theory (EVT) and block maxima. The difference between this work and other results that use EVT and block maxima is that in this work the analysis is applied directly to the traces of tasks' response time in order to obtain an upper bound of the WCRT whereas other analysis use the EVT on the traces of tasks' execution times in order to obtain a distribution for the pWCET which is given as input to a probabilistic schedulability analysis in order to obtain an estimate of the WCRT distribution. The downside of this method is that the system needs to be built first in order to be analysed, at which point the estimates provided by the analysis are of little use, it can only confirm that the system is feasible but in order for the system to be build feasible decisions need to be made at the beginning of the design process otherwise manufacturers risk to end up in the situation of only building unfeasible systems and the analysis in [Lu et al., 2012] would only confirm that the systems are unfeasible.

1.6 Derivation of real-time distributions

In [Edgar and Burns, 2001] The authors introduce a technique for statistically estimating the worst case execution time (WCET) of real-time tasks based on the extreme values theory (EVT). The system under consideration is executed several times and data about the behavior of the task is gathered, namely by sampling of execution durations. The necessary parameters are computed so that the data is fitted to a Gumbel distribution and different values of the WCET are estimated, each with its own confidence level. The authors also argue that the three distributions suited for representing real-time parameters, out of which they employ the Gumbel distribution which gives satisfactory results. Under independence assumptions, the confidence of tasks also give the confidence of the entire system. It was later shown in [Edgar and Burns, 2001] that this technique is flowed in the sense that it incorrectly fits raw execution time data to the Gumbel distribution which is actually intended to model random variables that are the maximum or minimum of a large number of other random variables. This is not the case for execution time measurements.

In [Hansen et al., 2009] the authors improve on the results of [Edgar and Burns, 2001] by adding to the extreme value theory the method of block maxima. That is, rather than attempting to fit the Gumbel distribution directly to the sample execution times they are first grouped into blocks and the Gumbel theory is fitted only to the maximum values of each of the blocks, ensuring in this way that the samples used to estimate the Gumbel distribution are sampled from a maximum of random variables. The exceedance functions obtained with the analytical method are compared with the traces obtained through simulation and it can be seen that they are very close and the predicted distribution is a bound for almost all values of measured WCET with the exception of the tail of the distribution where the number of recorded samples has a great impact on the accuracy of the result. The authors gathered a trace of 2.2 million execution times but even so, capturing the worst case behavior of the system or task is not guaranteed.

In [Bernat et al., 2002] the concept of execution profile (EP) is introduced as a mechanism to capture the variability of the execution time of paths of a section of code and an analysis framework for obtaining the execution profiles is presented. Once the EPs of basic blocks of code are obtained they are further combined to obtain the EP of a path. The combination of EPs can be achieved in three ways according to the dependences that exist between them, i.e., the EPs are independent or they are dependent with known dependences or they are dependent but there is no information on the dependences.

In [David and Puaut, 2004] the authors present a framework for deriving probabilistic execution time profiles of single-task multi-application systems by statically analyzing its source code. The analysis first takes into account all the possible execution paths that a task can take, forming an execution tree via a tree based technique. The execution time of each path is computed together with the conditions that need to be met in order for the path to be accessed. The conditions are later evaluated in order to obtain the probability of meeting them. In this way, a probability distribution if derived, with its values given by the execution times of each path and the probabilities given by the probabilities of accessing the paths. The technique works for straight-line code as well as for code with loops (by probabilistically evaluating how many times each loop can be executed) and simulations show that it produces accurate results.

In [Cazorla et al., 2012], the authors introduces novel cache design which implements a random replacement policy instead of a deterministic replacement policy.

By introducing randomization in the timing behaviour of the hardware (while maintaining the functional behavior) the system gains independence from its execution history and thus the analysis of the system is greatly simplified by reducing the amount of knowledge required to perform the analysis as well as the costs of acquiring this knowledge. The authors also provide a static probabilistic timing analysis (SPTA) for programs running on this type of systems as well as intuition for a measurement based probabilistic timing analysis (MBPTA). The advantage of SPTA is that it reduces the amount of knowledge needed to achieve a tight pWCET estimation, requiring only the reuse distances for every access, which are much easier to obtain that the exact addresses of every memory operation, which is the case for conventional static timing analysis. It is also shown that the pWCET estimations obtained through SPTA don't have as much dependence on complete knowledge as conventional methods, and also that the lack of information in the analysis has less impact on the pWCET estimation.

In [Cucu-Grosjean et al., 2012] the authors extend the work presented in [Cazorla et al., 2012] with the presentation of a measurement-based probabilistic timing analysis (MBPTA) which requires little information about internal behavior of the platform on which the program is executed and it derives probabilities for execution times by collecting observations of end-to-end runs of the application. The proposed MBPTA technique makes use of the extreme values theory and it requires a relatively low number of observations in order to provide a tight upper-bound on the pWCET of the application. The obtained pWCET distribution is 15% more pessimistic than the result provided by a static probabilistic timing analysis. This is considered an acceptable overestimation while keeping in mind that the requirements to apply the technique are significantly reduced with respect to applying the static probabilistic timing analysis.

In [Davis et al., 2013a] the authors present an analysis for probabilistic cache-related preemption delays (pCRPD) integrated with static probabilistic timing analysis (SPTA), providing an upper bound on the $1 - CDF$ of the probabilistic worst case execution time distribution function (pWCET) of a task while also taking into account the effect of one or more preemptions at arbitrary points in the tasks execution. The presented analysis is the only existing one for probabilistic caches and it does have an important amount of pessimism introduced in the pWCET distribution due to the fact that the re-use distance of an instruction is given as a deterministic value increasing the probability of a preemption when it might not be the case. This pessimism could be eliminated by considering probabilistic re-use distance for instructions and, in this way, having a more accurate notion of the preemptions that might occur.

In [Davis et al., 2013b] the problem of static probabilistic timing analysis is taken further to the case of multi-core processors where the existing SPTA is no longer applicable due to the interactions that exist at cache level between cores. The authors formalize the problem and provide intuitions for several paths that could be employed to find a solution, but the problem is still open up to date.

In [Maxim et al., 2012a] a first attempt is made at deriving probabilistic minimum inter-arrival time (pMIT) distributions for sporadic tasks with a random arrival pattern. The mechanism through which this is achieved is similar to that of obtaining pWCET distributions, namely using extreme value theory together with block maxima but instead of considering the maximal values of each block (as it is done in the case of the pWCET) the minimal values are kept and fitted to a Gumbel distribution, obtaining in this way an exceedance function for the minimum inter-arrival time.

In [Hardy and Puaut, 2013] the authors introduce a static probabilistic timing analysis (SPTA) for systems with faulty caches. This analysis stems from the observation that the technological progress lead to system components that are more and more pron to failures. The precise component targeted by the mentioned work is the instruction cache, with future work directed towards other micro-architecture components with SRAM cells such as data caches and branch predictors. Unlike the work generated by the PROARTIS project [Cazorla et al., 2012] where the system is on purpose build probabilistic by randomizing the cache replacement policy, in the work of Hardy and Puaut [Hardy and Puaut, 2013] the cache uses the least recently used (LRU) replacement policy, and the only source of probability comes from the intrinsic probability of cache blocks failing. The probability of a permanent cache block failure is considered known and a failure actually happening in the systems implies the (permanent) disabling of the faulty block with an impact of the WCETs of the tasks in the system. The authors propose a technique to compute a probabilistic bound on the WCET in the presence of permanent faults. The technique is shown to be tight while remaining safe, due to the fact that it is based on static analysis, which is guaranteed to always find the longest execution path and hence the largest value of the execution time.

1.7 Conclusions

In this chapter we have presented the main developments concerning the probabilistic analysis of real-time systems in the recent years. These developments are in an incipient stage, the assumptions made over the analysed systems are sometimes restrictive, especially with respect to the number of properties of the systems that are considered to be probabilistic. Indeed, most existing research consider that systems have a single task parameter that can be probabilistic, be it the execution time as presented in Section 1.1 or the inter-arrival time as presented in Section 1.2. In the recent years some progress was made in considering two probabilistic quantities for a task (Section 1.3), but the state of the art has to provide solutions that can be applied to real case situations. The trend for probabilistic analysis for real-time systems is clear and not just concerning the single processor case, but also the multiprocessor case (Section 1.4) and controller area networks (Section 1.5). An important topic in the probabilistic analysis of real-time system is deriving the probability distributions of task' parameters. Research in this direction is promising (Section 1.6), with many important contributions in the past decade.

Even though the state of the art has progressed in the recent years, there is a long way left ahead of us, with many aspect of the system that still need to be taken into consideration, many assumptions that need to be relaxed as well as the need of tools that take advantage of these developments. For example, there is a lack of optimal algorithms of probabilistic real-time systems that can use the power of a probabilistic analysis in order to produce better scheduling policies and obtain better processor utilization. Indeed, although the system may be analysed in a probabilistic fashion, it will still be scheduled with a deterministic algorithm that might be incapable of obtaining the best processor utilization while ensuring that the constraints placed of the system are met.

In the present work we further advance the state of the art by providing solutions to some of the above mentioned problems, especially introducing optimal scheduling algorithms for probabilistic real-time systems and an efficient analysis of real-time system, analysis that can come with any and all combination of the following three parameters : probabilistic worst-case execution time, probabilistic minimum inter-arrival time and probabilistic deadline. We also provide a means of decreasing the complexity of the analysis via real-time re-sampling, technique that makes the analysis tractable and applicable to real-case systems.

Chapitre 2

Modeling a probabilistic real-time system

We present in this chapter our model of probabilistic real-time system. We start by giving the definition of a probabilistic real-time system and other notations and definitions that we use throughout the thesis.

2.1 Notations and definitions

Definition 1 *A* probabilistic real-time system *is a real-time system with at least one parameter defined by a random variable.*

A random variable $\mathcal{X}$ has a probability function (PF) $f_{\mathcal{X}}(\cdot)$ with $f_{\mathcal{X}}(x) = P(\mathcal{X} = x)$. The possible values of $\mathcal{X}$ belong to the interval $[x^{\min}, x^{\max}]$. In this work we associate the probabilities with the possible values of a random variable $\mathcal{X}$ using the following notation

$$\mathcal{X} = \begin{pmatrix} X^0 = X^{min} & X^1 & \cdots & X^k = X^{max} \\ f_{\mathcal{X}}(X^{min}) & f_{\mathcal{X}}(X^1) & \cdots & f_{\mathcal{X}}(X^{max}) \end{pmatrix} \tag{2.1}$$

where $\sum_{j=0}^{k_i} f_{\mathcal{X}}(X^j) = 1$. A random variable may also be specified using its cumulative distribution function (CDF) $F_{\mathcal{X}}(x) = \sum_{z=x^{min}}^{x} f_{\mathcal{X}}(z)$.

Throughout this work we will be using a calligraphic typeface to denote random variables.

Definition 2 *Two random variables $\mathcal{X}$ and $\mathcal{Y}$ are (probabilistically)* **independent** *if they describe two events such that the outcome of one event does not have any impact on the outcome of the other.*

Definition 3 *The sum $\mathcal{Z}$ of two (probabilistically)* **independent** *random variables $\mathcal{X}$ and $\mathcal{Y}$ is the* **convolution** *$\mathcal{X} \otimes \mathcal{Y}$ where $P\{\mathcal{Z} = z\} = \sum_{k=-\infty}^{k=+\infty} P\{\mathcal{X} = k\}P\{\mathcal{Y} = z - k\}$.*

For instance, the sum of two random variables $\mathcal{X} = \begin{pmatrix} 3 & 7 \\ 0.1 & 0.9 \end{pmatrix}$ and $\mathcal{Y} = \begin{pmatrix} 0 & 4 \\ 0.9 & 0.1 \end{pmatrix}$ is equal to

$$\mathcal{Z} = \begin{pmatrix} 3 & 7 \\ 0.1 & 0.9 \end{pmatrix} \otimes \begin{pmatrix} 0 & 4 \\ 0.9 & 0.1 \end{pmatrix} = \begin{pmatrix} 3 & 7 & 11 \\ 0.09 & 0.82 & 0.09 \end{pmatrix}$$

A complementary operator to the convolution is the operator $\ominus$, defined by $\mathcal{X} \ominus \mathcal{Y} = \mathcal{X} \otimes (-\mathcal{Y})$.

Applying this operator to the random variables introduced above we obtain the following result :

$$\mathcal{Z}' = \begin{pmatrix} 3 & 7 \\ 0.1 & 0.9 \end{pmatrix} \ominus \begin{pmatrix} 0 & 4 \\ 0.9 & 0.1 \end{pmatrix} = \begin{pmatrix} -1 & 3 & 7 \\ 0.01 & 0.18 & 0.81 \end{pmatrix}$$

Definition 4 *The* **coalescion** *of two partial random variables, denoted by the operator $\oplus$ represents the combination of the two partial random variables into a single (partial) random variable so that values that appear multiple times are kept only once gathering the summed probability mass of the respective values.*

For example, by coalescing two partial random variables $\mathcal{A}_1 = \begin{pmatrix} 5 & 8 \\ 0.18 & 0.02 \end{pmatrix}$ and $\mathcal{A}_2 = \begin{pmatrix} 5 & 6 \\ 0.72 & 0.08 \end{pmatrix}$ we obtain :

$$\begin{pmatrix} 5 & 8 \\ 0.18 & 0.02 \end{pmatrix} \oplus \begin{pmatrix} 5 & 6 \\ 0.72 & 0.08 \end{pmatrix} = \begin{pmatrix} 5 & 6 & 8 \\ 0.9 & 0.08 & 0.02 \end{pmatrix}$$

Definition 5 *[Lopez et al., 2008] Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two random variables. We say that $\mathcal{X}_1$ is* **greater than** *$\mathcal{X}_2$ if $F_{\mathcal{X}_1}(x) \leq F_{\mathcal{X}_2}(x)$, $\forall x$, and denote it by $\mathcal{X}_1 \succeq \mathcal{X}_2$.*

For example, in Figure 2.1 $F_{\mathcal{X}_1}(x)$ never goes below $F_{\mathcal{X}_2}(x)$, meaning that $\mathcal{X}_2 \succeq \mathcal{X}_1$. Note that $\mathcal{X}_2$ and $\mathcal{X}_3$ are not comparable.

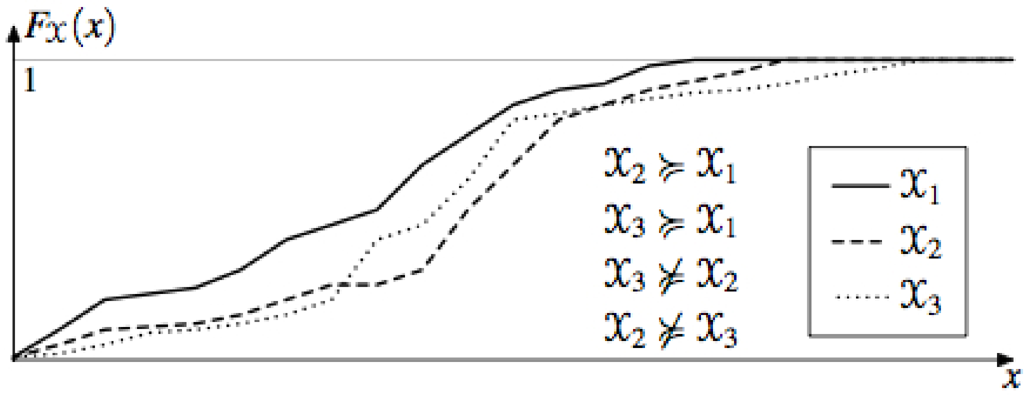


FIGURE 2.1 – Possible relations between the CDFs of various random variables. If two CDFs intersect, they are not comparable, as is the case of $\mathcal{X}_2$ and $\mathcal{X}_3$. Here, $\mathcal{X}_2$ and $\mathcal{X}_3$ are both (respectively) greater than $\mathcal{X}_1$, since their DCFs are always below that of $\mathcal{X}_1$.

2.2 Model of a probabilistic real-time system

We consider a system of n synchronous tasks $\{\tau_1, \tau_2, \dots, \tau_n\}$ to be scheduled on one processor according to a preemptive fixed-priority task-level scheduling policy. Without loss of generality, we consider that τ_i has a higher priority than τ_j for $i < j$. We denote by $hp(i)$ the set of tasks' indexes with higher priority than τ_i.

Each task τ_i generates an infinite number of successive jobs $\tau_{i,j}$, with $j = 1, \dots, \infty$. All jobs are assumed to be independent of other jobs of the same task and those of other tasks.

Definition 6 *The probabilistic execution time (pET) of a job of a task describes the probability that the execution time of the job is equal to a given value.*

Each task τ_i is a generalized sporadic task [Mok, 1983] and it is represented by a probabilistic worst-case execution time (pWCET) denoted by $\mathcal{C}_i$and by a probabilistic minimal inter-arrival time (pMIT). These notions are defined as follows.

Definition 7 *The probabilistic worst-case execution time (pWCET) $\mathcal{C}_i$ of a task τ_i is an upper bound on the pETs $\mathcal{C}_i^j$, $\forall j$ and it may be described by the relation $\succeq$ as $\mathcal{C}_i \succeq \mathcal{C}_i^j$, $\forall j$. Graphically this means that the CDF of $\mathcal{C}_i$ stays under the CDF of $\mathcal{C}_i^j$, $\forall j$.*

The worst-case execution time $\mathcal{C}_i$ can be written as follows :

$$\mathcal{C}_i = \begin{pmatrix} C_i^0 = C_i^{\min} & C_i^1 & \cdots & C_i^{k_i} = C_i^{\max} \\ f_{\mathcal{C}_i}(C_i^{\min}) & f_{\mathcal{C}_i}(C_i^1) & \cdots & f_{\mathcal{C}_i}(C_i^{\max}) \end{pmatrix}, \tag{2.2}$$

where $\sum_{j=0}^{k_i} f_{\mathcal{C}_i}(C_i^j) = 1$.

For example for a task τ_i we might have a worst-case execution time $\mathcal{C}_i = \begin{pmatrix} 2 & 3 & 25 \\ 0.5 & 0.45 & 0.05 \end{pmatrix}$; thus $f_{C_i}(2) = 0.5$, $f_{C_i}(3) = 0.45$ and $f_{C_i}(25) = 0.05$.

Following the same reasoning the probabilistic minimal inter-arrival time (pMIT) denoted by $\mathcal{T}_i$ describes the probabilistic minimal inter-arrival times of all jobs.

Definition 8 *The probabilistic inter-arrival time (pIT) of a job of a task describes the probability that the job's arrival time occurs at a given value.*

Definition 9 *The probabilistic minimal inter-arrival time (pMIT) $\mathcal{T}_i$ of a task τ_i is a lower bound on the pITs $\mathcal{T}_i^j$, $\forall j$ and it may be described by the relation $\succeq$ as $\mathcal{T}_i^j \succeq \mathcal{T}_i$, $\forall j$. Graphically this means that the CDF of $\mathcal{T}_i$ stays below the CDF of $\mathcal{T}_i^j$, $\forall j$.*

Hence, a task τ_i is represented by a tuple $(\mathcal{C}_i, \mathcal{T}_i)$. A job of a task must finish its execution before the arrival of the next job of the same task, i.e., the arrival of a new job represents the deadline of the current job [1]. Thus, the task's deadline may also be represented by a random variable $\mathcal{D}_i$ which has the same distribution as its pMIT, $\mathcal{T}_i$.

Since the seminal paper of Liu and Layland [Liu and Layland, 1973] the *independence* of tasks is defined such that the *requests for a certain task do not depend on the initiation or the completion of requests for other tasks.* Moreover the schedulability analysis of independent tasks may be studied under the hypothesis that the tasks do not share any resources except for the processor. In this work we assume this later hypothesis to be true, and since we consider probabilistic worst case parameters, then the hypothesis of Liu and Layland is the only hypothesis necessary for the analyis of the systems.

Since we consider probabilistic worst-case values for MIT and WCET, then the random variables are probabilistically independent as upper bounds of the pET and

1. In the analysis of GMF tasks this is known as the *frame separation* constraint.

pIT. On the other hand, using the pET and pIT distributions that describe distinct jobs of tasks does not guarantee the probabilistic independence of the random variables and so a probabilistic analysis that uses convolutions can not be applied (the convolution function requires probabilistic independence amongst the random variables on which it is applied). As an alternative, in the state of the art there are analysis frameworks [Burns et al., 2003], [Bernat et al., 2005] that make use of *copulas* instead of convolutions in order to be able to take into account random variables that are not probabilistically independent.

In this work we consider that the random variables representing the parameters of the real time system are *discrete*. This is a natural assumption, since a program can not terminate at any point in time, for example it can not terminate inside a loop, it must finish the computations it started and only after that it can terminate [Griffin and Burns, 2010]. Furthermore, the processor itself has some form of discrete time, hence a processor tick can only occur at precise moments in time meaning that the parameters of a program can not have values between two processor cycles.

2.3 Imposed properties on the system

We have presented in the previous section some implicit properties of the system, properties which the system exhibits independent of the priority levels at which tasks may be assigned. In order for the system to be certified as a real-time system, a series of requirements need to be imposed upon it to guarantee the correct functionality from the point of view of time. The most important metric that we use in our work to check the correctness of the system is the *response time*.

The response time of a job is the elapsed time between its release and its completion. Since we consider jobs with probabilistic worst case execution times, the response time of a job is also described by a random variable.

Requirements are placed on the system in the form of *probability of deadline miss*, either as a threshold, e.g. the probability of missing a deadline must be less or equal to 10^{-9}, or as an optimization function, e.g., minimizing the average deadline miss probability over all tasks in the system.

Definition 10 (Job deadline miss probability) *For a job $\tau_{i,j}$ the* deadline miss probability *$DMP_{i,j}$ is the probability that the j^{th} job of task τ_i misses its deadline* *and it is equal to :*

$$DMP_{i,j} = P(\mathcal{R}_{i,j} > D_i). \tag{2.3}$$

where $\mathcal{R}_{i,j}$ is the response time distribution of the j^{th} job of task τ_i.

If the task under analysis is periodic, i.e., its pMIT distribution only has one value, then a deadline miss ratio of the task may be computed by averaging the deadline miss probabilities of the jobs released in an feasibility interval $[a, b]$, e.g. the hyper-period. Alternatively, a conservative case deadline miss ratio can be assumed by considering it to be equal to the largest DMP amongst all jobs in the interval $[a, b]$.

Definition 11 (Task deadline miss ratio) *For a task τ_i and a time interval $[a, b]$, the tasks'* deadline miss ratio *is computed as follows*

$$DMR_i(a, b) = \frac{P(\mathcal{R}_i^{[a,b]} > D_i)}{n_{[a,b]}} = \frac{1}{n_{[a,b]}} \sum_{j=1}^{n_{[a,b]}} DMP_{i,j}, \tag{2.4}$$

where $n_{[a,b]} = \lceil \frac{b-a}{T_i} \rceil$ is the number of jobs of task τ_i released during the interval $[a, b]$.

2.4 Example of representing parameters as random variables

Example 1 *In order to explain the implications of representing task parameters as random variables we present here an example.*

Consider a task set $\gamma = \{\tau_1, \tau_2\}$ with $\tau_1 = \tau_2 = (\begin{pmatrix} 2 & 3 \\ 0.9 & 0.1 \end{pmatrix}, 5)$ with the interpretation that both tasks have the same deadline and pMIT $D_1 = D_2 = 5$ as well as the same worst case execution time distribution $\mathcal{C}_1 = \mathcal{C}_1 = \begin{pmatrix} 2 & 3 \\ 0.9 & 0.1 \end{pmatrix}$. Each job of task τ_1 and τ_2, at their release, can require either 2 units of processor time which happens with a probability of 0.9 (i.e. in 90% of the instances), or it can require its worst case execution time of 3 units of processor time and this can happen with a probability of 0.1 (i.e. in 10% of the instances). Let us presume that τ_1 has high priority and τ_2 low priority. The two tasks have synchronous releases and we presume that jobs are aborted if they do not finish their execution at the moment of the deadline.

For this task set, the worst case analysis only takes into consideration the largest values of execution time that the jobs of τ_1 *and* τ_2 *can take, namely* 3 *units of time. With such an analysis the system would be deemed unfeasible since there is a possibility that the lowest priority task misses its deadline. A worst case analysis disregards the possibilities and probabilities of jobs actually finishing with their deadlines. For this example there are four possible scenarios :*

1. *Both* τ_1 *and* τ_2 *have execution requirements of* 2 *time units in which case the response time of the lower priority task is equal to* 4*, finishing execution before the deadline which is at* $t = 5$*. The probability of this scenario if the combined probabilities of the two jobs have execution requirements of* 2 *units of time respectively, that is* $P(\mathcal{R}_{2,1} = 4) = 0.9 \times 0.9 = 0.81$*. This scenario is depicted in Figure 2.2.*

2. τ_1 *has an execution requirement of* 2 *time units and* τ_2 *has an execution requirement of* 3 *time units. In this case as well the instance of* τ_2 *finishes within its deadline, have a response time of* 5*. The probability of this scenario happening is* $0.9 \times 0.1 = 0.09$*. This scenario is depicted in Figure 2.3.*

3. τ_1 *has an execution requirement of* 3 *time units and* τ_2 *has an execution requirement of* 2 *time units. As in the previous scenario,* τ_2 *finishes execution with its deadline, at* $t = 5$ *and the probability of this scenario happening is also* $0.09 = 0.1 \times 0.9$*. This scenario combined with the previous one result in a combined probability of* τ_2 *having a response time of* 5 *time units equal to* $P(\mathcal{R}_{2,1} = 5) = 0.09 + 0.09 = 0.18$*. This scenario is depicted in Figure 2.4.*

4. *When both* τ_1 *and* τ_2 *have execution requirements of* 3 *time units* τ_2 *misses its deadline. The probability of this scenario happening is equal to* $P(\mathcal{R}_{2,1} > D_2) = 0.1 \times 0.1 = 0.01$*. This scenario is depicted in Figure 2.5.*

The same reasoning can be applied for the case of random arrivals of tasks to see that the worst case scenario and other scenarios in which deadlines are missed are in fact combinations of large execution times or small inter-arrival times. This scenarios are pathological and must not define the entire behavior of the system. A precise analysis of these scenarios can reveal that the system is actually feasible if the probabilities of occurrence of these pathological cases are sufficiently small.

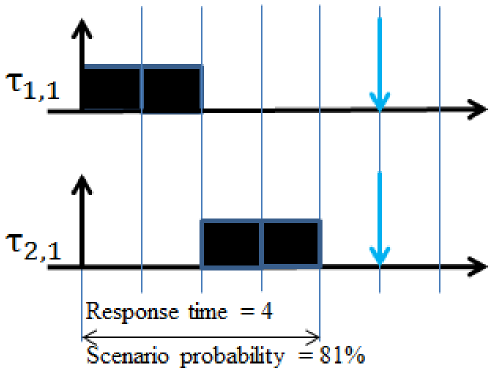


FIGURE 2.2 – Both τ_1 and τ_2 have execution requirements of 2 time units.

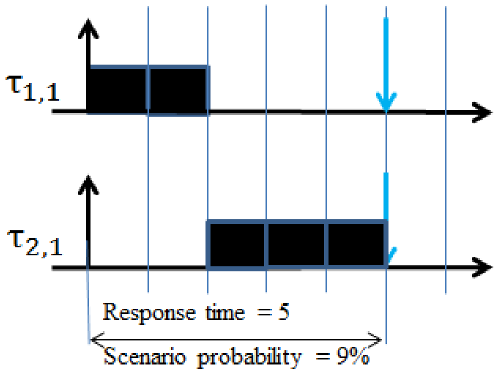


FIGURE 2.3 – τ_1 has an execution requirement of 2 time units and τ_2 has an execution requirement of 3 time units.

2.5 Validation of probabilistic analyses for real-time systems

When analysing real-time systems in a probabilistic way, one needs to ensure that the applied analysis is bounded in time and correct. In order to validate the probabilistic analyses presented in this work we will guide ourselves according to the following conditions that ensure not only that the proposed analyses are correct but that they are also bounded in time in order to make them usable in practice.

1. **Limit condition (compliance with the deterministic case) :** If all random variables describing tasks' parameters have only one value (which will implicitly have a probability of 1), then, the probabilistic analysis should re-

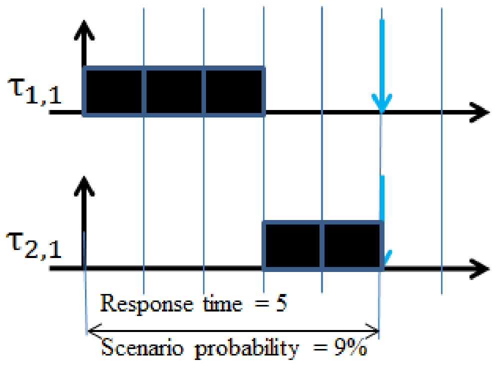


FIGURE 2.4 – τ_1 has an execution requirement of 3 time units and τ_2 has an execution requirement of 2 time units.

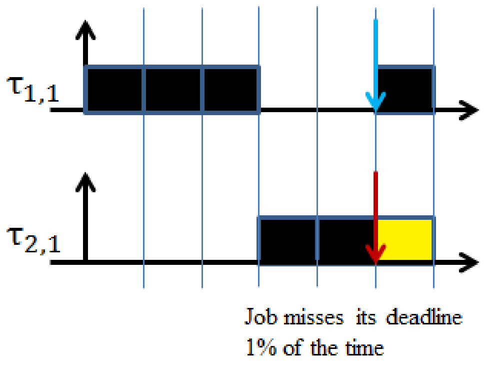


FIGURE 2.5 – Both τ_1 and τ_2 have execution requirements of 3 time units, in which case τ_2 misses its deadline.

turn the same result as the state of the art analysis (which will also have a probability of 1). For example, when applying a probabilistic analysis on a deterministic system represented modeled as a Liu and Layland [Liu and Layland, 1973] task set in order to compute tasks' response times, the returned results should be the same as those obtained by the analysis in [Joseph and Pandya, 1986].

2. **Convergence condition :** If at least one random variable contains at least two different values, than the probabilistic analysis should be proven bounded in time. The complexity of a probabilistic analysis is usually high due to the combinatorial nature of the operations used (for example the convolution ope-

rator) that there is a risk that, for systems with high utilization, the analysis will run indefinitely since there might be a scenario in which a job never finishes its execution, i.e. it has an infinite response time that has a vanishingly small probability of occurring. A solution to ensure the convergence of the analysis is to impose a probability threshold for the obtained response time values. For example, the analysis stops when it has computed all response times that have a probability of occurrence greater than 10^{-9}, i.e. it does not continue the computation of response times with occurrence probabilities of 10^{-9} or less.

3. **Worst case condition :** if at least one random variable contains at least two different values, then the analysis must be able to :
   - deal with systems of deterministic and random tasks. The probabilistic analysis should be applicable even on combinations of random and deterministic tasks.
   - ensure the worst-case behavior of the system. The analysis should detect the instance or the instances having the worst-case response times for tasks with hard deadlines.

2.6 Conclusions

We have presented in this chapter the baseline concepts needed to model a real-time system so that it can be analysed using a probabilistic analysis and to obtain probabilistic quantities such as response time distributions and deadline miss probabilities/ratios for tasks and jobs. This kind of modeling can be applied not just to real-time tasks but to messages in communication networks with real-time capabilities such as controller area networks (CAN).

In the following chapters we will be providing analysis solutions and scheduling algorithms for real-time systems that have at least one task parameter given as random variables. In Chapter 5 we also provide a comparison between two ways of modeling arrival distributions, both of them currently being used in the state of the art, but suited for different types of analyses.

Chapitre 3

Optimal priority assignment algorithms for probabilistic real-time systems

3.1 Introduction

In this chapter we consider the problem of finding a priority assignment at task level such that the associated schedule meets a requirement specifying the maximum acceptable probability of timing (i.e. deadline) failure within a given time interval. We refer to a priority ordering that meets such a requirement as a *feasible* priority assignment. We note that this use of the term feasible is an extension of its use in the deterministic case, where a feasible priority ordering is one in which the associated schedule has zero probability of timing failure.

In this chapter, we consider a set Γ of n synchronous constrained deadline periodic tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task τ_i is characterized by three parameters $(\mathcal{C}_i, T_i, D_i)$ where T_i is the period, D_i the relative deadline, and $\mathcal{C}_i$ the worst-case execution time described by a random variable. We assume a constrained-deadline task model such that $D_i \leq T_i$.

The set of tasks is scheduled according to a fixed-priority policy, i.e., all jobs of the same task have the same priority. Let $HP(i)$ to denote the set of tasks with priorities higher than that of task τ_i and $LP(i)$ denote the set of tasks with priorities lower than that of τ_i.

Further, we use Φ to denote a (partial or total) priority assignment defined by a list of tasks ordered from the *lowest* to the *highest* priority. $priority(i)$ is used to denote the priority level of task τ_i. All tasks are assumed to have unique priorities.

3.1.1 Computing the response time distribution

In the case of fixed-priority scheduling, the main result providing the calculation of $\mathcal{R}_{i,j}(\Phi), (\forall i, j, \Phi)$ is given by Diaz et al. [Díaz et al., 2002], where the PF of the response time of the j-th job of task τ_i is given by

$$f_{R_{i,j}(\Phi)} = f^{[0,\lambda_{i,j}]}_{R_{i,j}(\Phi)} + (f^{(\lambda_{i,j},\infty)}_{R_{i,j}(\Phi)} \otimes f_{C_i}), \tag{3.1}$$

where $\lambda_{i,j}$ is the release time of $\tau_{i,j}$. A solution for Equation (3.1) can be obtained recursively.

Equation (3.1) can be reformulated as follows :

$$\mathcal{R}_{i,j}(\Phi) = \mathcal{B}_i(\lambda_{i,j}, \Phi) \otimes \mathcal{C}_i \otimes \mathcal{I}_i(\lambda_{i,j}, \Phi), \tag{3.2}$$

where $\mathcal{B}_i(\lambda_{i,j}, \Phi)$ is the accumulated *backlog* of higher priority tasks released before $\lambda_{i,j}$ and still active (not completed yet) at $\lambda_{i,j}$. $\mathcal{I}_i(\lambda_{i,j}, \Phi)$ is the sum of the execution times of higher priority tasks arriving after $\lambda_{i,j}$.

For a task τ_i, a time interval $[a, b]$ and a priority assignment Φ, the response time $\mathcal{R}^{[a,b]}_i(\Phi)$ of τ_i is obtained from the calculation of its PF by averaging the job response times over the interval $[a, b]$:

$$f_{\mathcal{R}^{[a,b]}_i(\Phi)} = \frac{1}{n_{[a,b]}} \sum_{j=1}^{n_{[a,b]}} f_{\mathcal{R}_{i,j}(\Phi)}, \tag{3.3}$$

with $n_{[a,b]} = \lceil \frac{b-a}{T_i} \rceil$ giving the number of jobs belonging to τ_i released in the interval.

In critical real-time systems, the maximum allowed failure rate for a task is given as a failure rate per hour of operation [653, 2008, ARP4761, 2001]. This failure rate per hour of operation can be translated into the equivalent exceedence probability threshold, and vice-verse, by knowing the frequency at which jobs are released. For example, considering that a task τ_i is released with a frequency of 10^{-2} seconds $=$ 10 milliseconds, i.e. 10^2 activations per second, which is assumed to be the maximum rate at which a task can be released in current systems, then, in order for τ_i to have a failure rate of at most 10^{-9} per hour of operation (this is, for example, the highest integrity level for commercial airborne systems [ARP4761, 2001]) for a continuous 10^9 hours of operation, its exceedence probability threshold needs to be in the range of 10^{-13} (i.e. $10^2 \times 60 \times 60 \times 10^9 = 3.6 \times 10^{13}$).

3.1.2 Optimality in probabilistic real-time systems

We talk in this section about the concept of optimality of priority assignment when considering a probabilistic real-time system. There are many ways in which one could define optimality for probabilistic real-time systems, ways that will be detailed in Sections 3.3, 3.4 and 3.5. The simplest definition is the following :

Definition 12 *An algorithm is optimal for a probabilistic real-time system if, for any arbitrary task-set Γ, it always finds a priority assignment, whenever such an assignment exists, so that the deadline miss probabilities for each task does not exceed its respective threshold.*

In order to find optimal scheduling algorithms for probabilistic real-time systems, we will be generalising a known algorithm that is proved to be optimal in the deterministic case, namely the OPA (Optimal Priority Assignment) algorithm, also known as Audsleys' priority algorithm [Audsley, 1991, Audsley, 2001].

The pseudo code for the OPA algorithm for one processor, using some schedulability test S is as follows [Audsley, 1991, Audsley, 2001] :

Algorithm 1: Optimal Priority Assignment Algorithm

```
for each priority level k, lowest first do
    for each unassigned task $\tau$ do
        if $\tau$ is schedulable at priority k according to schedulability test S with
        all unassigned tasks assumed to have higher priorities then
            assign $\tau$ to priority $k$
    return unschedulable
return schedulable
```

The OPA algorithm is a greedy algorithm that assigns tasks to priority levels starting with the lowest priority first. For n tasks, the algorithm performs at most $n(n+1)/2$ schedulability tests and is guaranteed to find a priority assignment that is schedulable according to schedulability test S, if one exists.

In [Davis and Burns, 2011], the authors present three conditions that need to be met by a schedulability test S for fixed priority scheduling in order for the OPA algorithm to be applicable. We present here the three conditions, generalized for the probabilistic case, that a schedulability test S needs to meet when applying OPA to probabilistic real-time systems :

- **Condition 1** : The PF of the response time of the task τ_k may, according to test S, be dependent on the set of higher priority tasks, but not on the relative priority ordering of those tasks.
- **Condition 2** : The PF of the response time of the task τ_k may, according to test S, be dependent on the set of lower priority tasks, but not on the relative priority ordering of those tasks.
- **Condition 3** : When the priorities of two tasks of adjacent priorities are swapped, then the $1 - CDF$ of the response time of the task being assigned the higher priority cannot increase for any value of t. Similarly, the $1-CDF$ of the response time of the task being assigned the lower priority cannot decrease for any value of t.

In the rest of this chapter we only consider real-time systems that allow tasks to continue execution past their deadlines. In the following subsection (3.1.3) we motivate our choice and take a look at other possible alternatives.

In order to prove that the OPA algorithm is applicable to probabilistic real-time systems that allow tasks to continue past their deadlines, we introduce the following results, stating that the relative order of higher priority tasks does not affect the response time distribution of a task at priority level i and also that when a task is moved from priority level i to a lower priority level its response time distribution and its DMR can not increase.

Theorem 1 (Order of higher priority tasks) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of n constrained-deadline periodic tasks with probabilistic worst case execution times scheduled pre-emptively according to a fixed-priority algorithm on a single processor. If a task τ_i has its priority known and given, then the priority order of the higher priority tasks (belonging to $HP(i)$) does not impact the value of $DMP_{i,j}(\Phi)$ for any job of τ_i or the value of $DMR_i(a, b, \Phi), \forall a, b$. Stated otherwise, if membership of the sets $HP(i)$ and $LP(i)$ are unchanged, then the response time $\mathcal{R}_{i,j}$ of any job of τ_i is unchanged and the response time $\mathcal{R}_i^{[a,b]}$ of task τ_i is unchanged whatever the priority order of tasks within $HP(i)$ and within $LP(i)$.*

Proof 2 *Concerning the lower priority tasks, with pre-emptive scheduling, the tasks in $LP(i)$ do not influence the response time of task τ_i (or of that of any job $\tau_{i,j}$ of task τ_i), whatever their priority order.*

Let us consider now the case of higher priority tasks. Let $\tau_{i,j}$ be the job of the task τ_i that is released at time t. According to Equation (3.2) the response time $\mathcal{R}_{i,j}$

of $\tau_{i,j}$ is obtained by adding to the existing backlog $\mathcal{B}_i$ its own execution time and the sum of execution times of all higher priority jobs that arrive after its release.

Since the backlog $\mathcal{B}_i$ at time t represents the execution times of the higher priority tasks that have been released before time t (including those that have been released at time t) and that have not been completed yet, its value does not depend on the priority order of the higher priority tasks.

In a time interval of length t, any task τ_k can have only $\lceil t/T_k \rceil$ releases which are relevant to the calculation of the response time of task τ_i, whatever its priority. This is true for all the tasks that have higher priority than the task for which we are computing the response time. Furthermore, the same reasoning is valid for the number of pre-emptions, i.e., the number of pre-emptions from a task with priority higher than τ_i is the same, regardless of whether that task has the highest priority or any priority higher than that of τ_i. Moreover, since the summation of two random variables is commutative, it follows that the response time of task τ_i is the same whatever the priorities we assign to the rest of the tasks in $HP(i)$.

Theorem 2 (Monotonicity of the response time) *Let $\Gamma = \{\tau_1, \ldots, \tau_n\}$ be a set of constrained-deadline tasks with probabilistic execution times, scheduled pre-emptively according to a fixed priority algorithm. Recall that Φ is a priority assignment that may be represented by a list of tasks in sequence from lowest to highest priority. Let Φ_1 and Φ_2 be two priority assignments such that, to get from one assignment to the other, one only has to change the position of one unique task τ_i in the list, leaving the relative order of all other tasks unchanged. If the priority of τ_i is lower in Φ_1 than in Φ_2, then the response time of any of its jobs is such that $\mathcal{R}_{i,j}(\Phi_1) \succeq \mathcal{R}_{i,j}(\Phi_2)$. Consequently, the task response time $\mathcal{R}_i^{[a,b]}(\Phi_1) \succeq \mathcal{R}_i^{[a,b]}(\Phi_2)$.*

Proof 3 *Follows from the fact that the backlog and the interference increase with decreasing priority of the task, and consequently increasing priority of other tasks with respect to τ_i.*

More precisely for each job in the interval $[a, b)$, we compare the response time distributions and the way they are derived in the two cases. With priority assignment Φ_1 (where task τ_i is at lower priority) τ_i has backlog and interference that is no smaller than with priority assignment Φ_2. This is because all of the tasks that are at higher priority than task τ_i in Φ_2 are also at higher priority than τ_i in Φ_1. Hence, from Equation (3.2) the response time of any job of task τ_i cannot increase when the *priority assignment is changed from Φ_1 to Φ_2. Further, from Equation (3.3) neither can the response time of task τ_i*

The response time of a task increases with respect to the priority of the task. The lower the priority level, the more the interference from higher priority tasks, and so the larger the response time. The monotonicity of the task response time leads to a similar conclusion about the deadline miss probability (DMP).

Corollary 1 (Monotonicity of DMP and DMR**)** *Let $\Gamma = \{\tau_1, \dots, \tau_n\}$ be a set of constrained-deadline tasks with probabilistic worst case execution times, scheduled pre-emptively according to a fixed priority algorithm. Let Φ_1 and Φ_2 be two priority assignments such that, to get from one assignment to the other, one only has to change the position of one unique task τ_i in the list (priority sequence), leaving the relative order of all other tasks unchanged. If the priority of τ_i is lower in Φ_1 than in Φ_2, then $DMP_{i,j}(\Phi_1) \geq DMP_{i,j}(\Phi_2)$ and $DMR_i(a, b, \Phi_1) \geq DMR_i(a, b, \Phi_2)$.*

Proof 4 *Under the condition stated in Theorem 2 the response time of the generic task τ_i cannot decrease with a decrease in its relative priority. Under the same condition, since $DMP_{i,j}(\Phi) = P(\mathcal{R}_{i,j}(\Phi) > D_i)$ (Equation (6.3)) and $DMR_i(a, b, \Phi) = \frac{P(\mathcal{R}_i^{[a,b]}(\Phi) > D_i)}{n_{[a,b]}}$ (Equation (2.4)), the job deadline miss probability and the task deadline miss ratio cannot decrease with a decrease in the task's relative priority.*

Theorem 3 *The Optimal Priority Assignment (OPA) algorithm is an* optimal priority assignment policy *for any fixed priority scheduling test S compliant with Conditions 1-3.*

Proof 5 *In order to show that the OPA algorithm is applicable to probabilistic real-time systems we need to show that the three conditions listed above are met by a schedulability test* S *according to which the feasibility of the system is tested.*

From Theorem 1 we have that Condition 1 is met. From Theorem 2 and Corollary 1 we have that Condition 3 is met. Condition 2 is automatically met in preemptive system, which is the case of the present work.

3.1.3 What happens after a deadline is missed

At the moment when a deadline is missed there are multiple ways in which the system can proceed with the execution. Depending on the way the system continues execution, the (probabilistic) analysis applied is different, if there exists one. In this section we present three of these possibilities that we consider to be the most important ones. We then proceed to choose one (motivating the choice) for which we will construct our analysis and for which we propose scheduling solutions.

Job discarded at deadline

One way in which the system can proceed with the execution at the moment when a job reaches its deadlines and did not finish execution is by discarding that job. By proceeding in this way, the system looses the work done by the discarded job and the results that it might have produced and so, it needs to wait for a future job to produce these results. Another aspect that needs to be taken into consideration when designing the system to discard jobs at deadlines is the fact that the operating system needs to monitor these jobs which generates overhead.

For such a system it is hard to predict the way in which it will behave, the response time of jobs and, especially, what will happen if a different scheduling policy is to be used, since some properties of the system do not carry on from one policy to another. Notably, once a job $\tau_{i,j}$ is discarded at its deadline, this could leave enough processor time for a lower priority job $\tau_{m,n}$ to finish execution in time to reach its deadline. If, instead, $\tau_{i,j}$ receives a higher priority it might cause $\tau_{m,n}$ to become un-schedulable, since it might finish execution in time to reach its deadline, delaying another job that takes all the processor time required for $\tau_{m,n}$ to finish execution.

Lemma 1 *A system that discards jobs that did not finish execution at the moment of their respective deadlines is not OPA-compatible.*

Proof 6 *As a proof, we present an example in which the three conditions for OPA-compatibility stated in the previous section are not met.*

Let $\tau = \{\tau_1, \tau_2, \tau_3\}$ *be a task-set such that each task is characterized by the parameters* $(\mathcal{C}, T, D)$ *with* $\tau_1 = (\begin{pmatrix} 2 \\ 1 \end{pmatrix}, 10, 3)$, $\tau_2 = (\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \end{pmatrix}, 10, 4)$ *and* $\tau_3 = (\begin{pmatrix} 2 \\ 1 \end{pmatrix}, 10, 5)$.

Let us assume that task τ_3 *is assigned the lowest priority.*

If task τ_1 *is given the highest priority, then task* τ_3 *has a probability of* 0.5 *of missing its deadline, i.e. when all* 3 *tasks execute for 2 units of time.*

If task τ_2 is given the highest priority, then task τ_3 always *meets its deadline. This is because in the case that task τ_2 executes for* 2 *units, then task τ_1 misses its deadline at $t = 3$ and is aborted. This leaves* 2 *units of time, from $t = 3$ to $t = 5$ for τ_3 to finish its execution.*

This example shows that changing the order of higher priority tasks in a system that aborts jobs on a deadline miss may result in influencing the schedulability of a lower priority task, meaning that condition 1 *presented in the previous section is not guaranteed. This means that such a system is not OPA-compatible.*

Job carries on after its deadline has passed

Another way in which the system can proceed execution when a job misses its deadline is by letting it continue its execution. In this way, the results and work done by the job are not lost, but the deadline is only symbolic since no action is taken if it is missed. An important advantage of this type of system is that it is OPA compatible.

Among the drawbacks of this model is the fact that, if a job takes too long to finish execution and it is not aborted at deadline, then it might create backlog, potentially delaying other lower-priority jobs, forcing them to miss their respective deadlines.

Another drawback is the difficulty to analyze such a system since jobs from one hyper-period (or any study interval) can execute in the next hyper-period, creating and propagating backlog for further hyper-periods. In this way, response times differ from hyper-period to hyper-period, which makes it difficult to predict the behavior of a certain job in the future unless all jobs up to that point have been analyzed. This means that, in order to confirm the safeness of the system, all the response times of all the jobs of all tasks need to be calculated, one hyper-period not being enough. It is easy to see why this is not a practical model to analyze.

For this reason, we will use a simplified version of this model to analyse and provide optimal algorithms, leaving as future work the proof (or counter-proof) that these algorithms are optimal for this more complex model as well.

A subset of tasks are allowed to execute past their deadlines

In some systems, different actions can be taken depending on the importance of the tasks, i.e. some tasks may be allowed to execute past their deadlines while others are aborted. For example, in mixed criticality systems, the highest criticality level is privileged while tasks in lower criticality levels are sacrificed. That is, there is nothing to be gained from stopping the execution of a high priority task if it overruns, the best course of action is to let the task execute and recover over time, while limiting the impact of lower priority tasks which are aborted at deadline or even completely removed from the system if their criticality is sufficiently low.

System reset at synchronous release

In this chapter we consider that at the end of each hyper-period any incomplete job is aborted. This is a form of reset prior to synchronously releasing all of the tasks at the start of the next hyper-period. In future work we aim to relax this assumption and move towards a more general case.

The consequence of resetting the system after each hyper-period is that we can focus our analysis on the first hyper-period $[0, H]$ (with $H = \text{lcm}\{T_1, T_2, \cdots, T_n\}$) which becomes a feasibility interval.

3.2 Statement of the scheduling problems

Before defining the sub-problems we deal with, we first recall here the metrics that we employ in order to determine the feasibility of the system under analysis, namely the *job deadline miss probability* and the *task deadline miss ratio*.

Definition 13 (Job deadline miss) *For a job $\tau_{i,j}$ and a priority assignment Φ, the deadline miss probability $DMP_{i,j}$ is the probability that the j-th job of task τ_i misses its deadline :*

$$DMP_{i,j}(\Phi) = P(\mathcal{R}_{i,j}(\Phi) > D_i). \tag{3.4}$$

Definition 14 (Task deadline miss ratio) *For a task τ_i, a time interval $[a, b]$ and a priority assignment Φ, the task deadline miss ratio is computed as follows*

$$\begin{aligned} DMR_i(a, b, \Phi) &= \frac{P(\mathcal{R}_i^{[a,b]}(\Phi) > D_i)}{n_{[a,b]}} \\ &= \frac{1}{n_{[a,b]}} \sum_{j=1}^{n_{[a,b]}} DMP_{i,j}(\Phi), \end{aligned} \tag{3.5}$$

where $n_{[a,b]} = \lceil \frac{b-a}{T_i} \rceil$ is the number of jobs of task τ_i released during the interval $[a, b]$.

During the first hyperperiod of a schedule, i.e., $a = 0$ and $b = H$, the task deadline miss ratio is denoted by $DMR_i(\Phi)$ for a priority assignment Φ and a task τ_i. Unless specified otherwise, in the following we consider the interval $[a, b] = [0, H]$.

The DMP gives the probability that a job misses its deadline, whereas the DMR gives the average probability over the hyperperiod that a job of the task will miss its deadline, and hence the average probability over the entire lifetime of the system, that a job of the task will miss its deadline.

For each task τ_i we consider a specified parameter $p_i \in [0, 1]$, referred to as the maximum permitted deadline miss ratio.

We define three new problems :

1. Basic Priority Assignment Problem (**BPAP**). This problem involves finding a priority assignment such that the DMR of every task does not exceed the threshold specified, i.e. $DMR_i(\Phi) \leq p_i$. Hence, we search for a feasible priority assignment Φ^* such that $DMR_i(\Phi^*) \leq p_i, \forall i$.
2. Minimization of the Maximum Priority Assignment Problem (**MPAP**). This problem involves finding a priority assignment that minimizes the maximum deadline miss ratio of any task. Hence, we search for a priority assignment Φ^* such that $\max_i\{DMR_i(a, b, \Phi^*)\} = \min_\Phi\{\max_i DMR_i(a, b, \Phi)\}, \forall i$.
3. Average Priority Assignment Problem (**APAP**). This problem involves finding a priority assignment that minimizes the sum of the deadline miss ratios for all tasks. Hence, we search for a feasible priority assignment Φ^* such that $\sum_i DMR_i(a, b, \Phi^*) = \min_\Phi\{\sum_i DMR_i(a, b, \Phi)\}, \forall i$.

In the following we use the relationship $\succeq$ as it is defined in Section 2.1, summarized as follows. A random variable $\mathcal{X}$ is *greater than or equal to* another random variable $\mathcal{Y}$, noted $\mathcal{X} \succeq \mathcal{Y}$ if and only if $\forall V \ P(\mathcal{Y} \leq V) \leq P(\mathcal{X} \leq V)$. The relationship $\succeq$ between two probabilistic response times defines a probabilistic order among probabilistic variables and allows us to compare random variables.

3.3 Solution for the Basic Priority Assignment Problem

In this section we present an optimal algorithm that solves the Basic Priority Assignment Problem (BPAP) as defined in Section 3.1. First we provide a definition of what we mean by an optimal priority assignment in this case.

Definition 15 (Optimal algorithms for BPAP) *Let Γ be a set of constrained-deadline tasks with probabilistic execution times and each task characterized by parameters $(\mathcal{C}_i, T_i, D_i, p_i)$. An algorithm is optimal with respect to BPAP if, for any arbitrary task-set Γ, it always finds a feasible priority assignment whenever such an assignment exists.*

In the real-time systems domain, many optimal scheduling algorithms have been developed such as Rate Monotonic (RM), Deadline Monotonic (DM), Earliest Deadline First (EDF) and others. Unfortunately, these algorithms are not necessarily optimal when applied to probabilistic real-time systems and their respective optimality or non-optimality needs to be formally proved if one wants to employ such algorithms on a real-time system with probabilistic requirements. As an example of an algorithm that is optimal in the deterministic case but no longer optimal in the probabilistic case we present in Appendix A the case of Rate Monotonic algorithm. Since our purpose in this chapter is not that of proving non-optimality of existing algorithms, we leave this exploration for the curious reader and move on to present algorithms that ARE optimal for the mentioned problems.

3.3.1 The intuitive (but incorrect) solution for the Basic Priority Assignment Problem

At first glance, one might think that an optimal priority assignment for the Basic Priority Assignment Problem could be obtained by assigning priorities to tasks in increasing order of their maximum permitted deadline miss ratios, (i.e. the lower the maximum permitted deadline miss ratio, the higher the priority).

Below, we provide a counter example showing that this priority assignment policy is not optimal for BPAP.

Consider a task set $\Gamma = \{\tau_1, \tau_2\}$ such that τ_1 is defined by $(\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0.2 & 0.3 & 0.3 & 0.2 \end{pmatrix}, 5, 5, 0.4)$ and τ_2 by $(\begin{pmatrix} 4 \\ 1 \end{pmatrix}, 10, 10, 0.2)$.

Assuming that τ_2, which has the smaller permitted deadline miss ratio, is assigned the higher priority, then we have the following response times for the two jobs of task τ_1 :

$$\mathcal{R}_{1,1} = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 0.2 & 0.3 & 0.3 & 0.2 \end{pmatrix}$$

$\mathcal{R}_{1,2} = \begin{pmatrix} 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 0.04 & 0.12 & 0.21 & 0.26 & 0.21 & 0.12 & 0.04 \end{pmatrix}$.

The deadline miss ratio of task τ_1 is given by $DMR_1 = \frac{DMP_{1,1}+DMP1,2}{2} = \frac{0.8+0.16}{2} = \frac{0.96}{2} = 0.48$ which is higher than its maximum permitted deadline miss ratio of 0.4.

Alternatively, if τ_1 is assigned the higher priority, then τ_1 would have $DMR1 = 0$, and τ_2 would have a response time of $\mathcal{R}_2 = \begin{pmatrix} 5 & 7 & 8 & 9 & 10 & 11 & 12 \\ 0.2 & 0.06 & 0.15 & 0.22 & 0.21 & 0.12 & 0.04 \end{pmatrix}$ and $DMR2 = 0.16$ which is less then its maximum allowed DMR of 0.2.

This example shows that there exist task sets and constraints with their maximum permitted deadline miss ratios that are not feasible when priorities are assigned according to each task's maximum permitted deadline miss ratio, and yet are feasible with an alternative priority assignment. Hence assigning priorities based on the maximum permitted deadline miss ratio of each task is not an optimal solution for BPAP.

3.3.2 Optimal solution for the Basic Priority Assignment Problem

Theorem 1 indicates that the order of higher priority tasks does not impact the response time of a task. This result suggests that a greedy approach similar to Audsley's optimal priority assignment algorithm [Audsley, 1991, Audsley, 2001] can be used to solve BPAP.

Theorem 1 implies that we can assign priorities to tasks in a greedy fashion, starting at the lowest priority level and continuing up to the highest priority level. While assigning task τ_i to a particular priority level, we need not be concerned about the tasks that have already been assigned to lower priority levels (tasks in $LP(i)$) as their response times will be unchanged, and we do not need to know the priority assignment for the remaining tasks, since this does not impact the response time of task τ_i as all of those tasks will be in $HP(i)$ whatever their relative priority ordering.

Since the maximum permitted deadline miss ratio p_i for each task τ_i is specified as part of the problem description, we can use this information within the priority assignment algorithm to determine whether or not a task is feasible at a particular priority level or if it needs to occupy a higher one (see Algorithm 2).

To find a task to place at the next available priority level k, Algorithm 2 checks each unassigned task in turn to see if its maximum permitted deadline ratio would be exceeded if it were placed at priority level k, if not, then the task is assigned to that priority level, and the algorithm continues to the next higher priority. If it is exceeded, then that task needs to be placed at a higher priority level, and the algorithm continues to check other unassigned tasks to see if any of them can be assigned to priority level k.

Note that, when computing the deadline miss ratio for a task at priority level k we do not need to assign priorities to the unassigned tasks that will occupy the priority levels above priority k, any assignment can be assumed, since this will not affect the response time of the task placed at priority k.

Algorithm 2 is similar to Audsley's algorithm [Audsley, 1991] and does not require backtracking. Indeed, once we have assigned a subset of tasks to lower priority levels where they do not exceed their maximum allowed deadline miss ratio p, and we are now seeking a task to place at priority level k, if none of the tasks left unassigned can be assigned to this priority level, then it would be useless to backtrack and try and move any of the unassigned tasks to a lower priority level, where it would have an even higher DMR. Since none of the tasks left unassigned can be placed at priority level k without exceeding their maximum permitted deadline miss ratios the tasks set is infeasible, and no priority assignment exists such that all of the tasks can meet the requirements placed on their deadline miss ratios.

Theorem 4 *Algorithm 2 is optimal with respect to BPAP.*

Proof 7 *The proof follows directly from the description of Algorithm 2, Theorem 1, Definition 15 and the three conditions presented in section 3.1.2.*

3.4 Solution for the Minimization of the Maximum Priority Assignment Problem

3.4.1 The Lazy and Greedy Algorithm

For MPAP we propose a *Lazy and Greedy Algorithm* (LGA). It is inspired by Audsley's optimal priority assignment algorithm [Audsley, 1991]. Indeed, Algorithm 3 incrementally builds a solution Φ as a sequence of tasks, starting with the lowest priority first, and adding to Φ at each iteration an unassigned task. This algorithm is :

Algorithm 2: Solution to BPAP : the *feasibility* function verifies that for $\forall \tau_i$, $DMR_i < p_i$

```
Input: $\Gamma = \{\tau_i, i \in 1..n\}$ /* source set of tasks */
Output: $\Phi$ /* destination sequence of tasks */
$\Phi \leftarrow ()$
for $l \in n..1$ do
    $assignment \leftarrow FALSE$
    for $\tau_i \in \Gamma'$ do
        /* feasibility function such that the computed $DMR_i < p_i$ */
        if $feasible(\tau_i, \Phi)$ then
            $\Phi \leftarrow \Phi.\tau_i$
            $\Gamma' \leftarrow \Gamma' \backslash \{\tau_i\}$
            $assignment \leftarrow TRUE$
            break
    if $assignment = FALSE$ then
        /* could not find a task to put at this priority level */
        break
```

- *lazy* because, if there exists an unassigned task τ which could be used at the current priority level l without degrading the worst DMR, then τ is assigned to this priority level, and
- *greedy* because, if no such non-degrading task exists, then it assigns to the current priority level the task that would have the smallest deadline miss ratio, without reasoning about what will happen at higher priority levels.

Algorithm 3's main loop goes through the priority levels in order : $l \in n..1$. At each iteration, it performs a for loop over the unassigned tasks to search for the one (τ_{best}) that has the best DMR (DMR_{best}) at the current priority level l (this is the greedy part). This requires computing, for each unassigned task τ_i, its potential DMR δ (if τ_i is not in Φ, $DMR_i(\Phi)$ is the DMR of τ_i if appended at the end of Φ) and comparing it to the current most promising task. Further, each unassigned task's DMR is compared with the worst DMR over all of the tasks already in Φ (referred to as DMR_{worst}) so that, whenever a task is found with a deadline miss ratio better than or equal to DMR_{worst}, the search is cancelled and this task is assigned (this is

Algorithm 3: Lazy and Greedy Algorithm

```
Input: $\Gamma = \{\tau_i, i \in 1..n\}$ /* source set of tasks */
Output: $\Phi$ /* sequence of tasks */, $DMR_{worst}$ /* worst $DMR$ */
$\Phi \leftarrow ()$
$DMR_{worst} \leftarrow 0$
/* Loop over the priority levels (from lowest to highest) */
for $l \in n..1$ do
    /* Search among unassigned tasks */
    $(\tau_{best}, DMR_{best}) \leftarrow (0, +\infty)$
    for $\tau_i \in \Gamma$ do
        /* Compute DMR of current task $\tau_i$ */
        $\delta \leftarrow DMR_i(\Phi)$
        /* If this DMR is better than (or equal to) the current worst DMR in $\Phi$, be lazy : pick
        this task and stop the search. */
        if $\delta \leq DMR_{worst}$ then
            $(\tau_{best}, DMR_{best}) \leftarrow (\tau_i, \delta)$
            break
        /* If this DMR improves on other unassigned tasks, remember this task. */
        if $\delta < DMR_{best}$ then
            $(\tau_{best}, DMR_{best}) \leftarrow (\tau_i, \delta)$
    /* The search is done. The task in $\tau_{best}$ can be assigned at the current priority level. */
    $\Gamma \leftarrow \Gamma \backslash \{\tau_{best}\}$
    $\Phi \leftarrow \Phi.\tau_{best}$
    /* Update the value of the worst DMR in $\Phi$. */
    if $DMR_{worst} < DMR_{best}$ then
        $DMR_{worst} \leftarrow DMR_{best}$
return $(\Phi, DMR_{worst})$
```

the lazy part).

3.4.2 Optimality of the Lazy and Greedy algorithm

Before proceeding to show that the Lazy and Greedy algorithm provides an optimal solution to MPAP, we must first define what we mean by optimality in this case.

Definition 16 (Optimal algorithms for MPAP) *Let Γ be a set of constrained-deadline tasks with probabilistic execution times, where each task is characterized by the parameters $(\mathcal{C}_i, T_i, D_i)$. A priority assignment algorithm P is optimal with respect to MPAP if the priority ordering determined by algorithm P, for any arbitrary taskset Γ, has a maximum deadline miss ratio for any task, which is no larger than that obtained with any other priority ordering.*

Theorem 5 *The Lazy and Greedy Algorithm (LGA) is optimal with respect to MPAP.*

Proof 8 *Let us consider a problem with n tasks $\tau_1, \ldots, \tau_n$. Let Φ^g be the solution returned by the Lazy and Greedy Algorithm (Φ^g is a permutation over $1..n$, and $\Phi^g(i)$ is the index of the i-th task by order of priority).*

Let M be the set of positions in Φ^g corresponding to items with the maximum deadline miss ratio : $M = \arg\max_i DMR_{\Phi^g(i)}(\Phi^g) = \{m_1, \ldots, m_{|M|}\}$, as illustrated in Table 3.1. A first observation is that this set contains at least one item (but possibly more). Let us also assume that these indices are ordered as they appear in Φ^g, meaning that they verify : $m_1 > m_2 > \cdots > m_{|M|}$ (as the first tasks introduced have low priorities). Due to the algorithm, when the first of these tasks ($\tau_{\Phi^g(i)}$) is added to M ($m_1 = i$), each remaining task has a deadline miss ratio (DMR) at that priority level that is greater than or equal to that of $\tau_{\Phi^g(m_1)}$, but it will subsequently become equal or better, once the task is assigned a higher priority level. Note also that there is no a priori *reason for these "worse tasks" to be contiguous.*

Let Φ' be any other complete solution, and let us refer to :

- *$\tau_c = \tau_{\Phi^g(m_1)}$ as the* critical task *of Φ, i.e., the lowest priority task in Φ that has the maximum or worst DMR for any task in priority order Φ ;*
- *DMR_c^g as the DMR of τ_c in Φ^g ;*
- *$HP(\tau, \Phi)$ as the set of tasks with higher priorities than τ in Φ (Φ omitted when using the current permutation) ;*

	(low P.) ←			→ (high P.)
tasks	$\tau_{\Phi^g(4)}$	$\tau_{\Phi^g(3)}$	$\tau_{\Phi^g(2)}$	$\tau_{\Phi^g(1)}$
DMR	13	22	17	22
M	–	$m_1 = 3$	–	$m_2 = 1$

TABLE 3.1 – Example fixed priority assignment with associated DMRs and M set

– *$LP(\tau, \Phi)$ as the set of tasks with lower priorities than τ in Φ ;*
– *τ_b as the task in $HP(\tau_c, \Phi^g)$ which has the lowest priority in Φ' ; this will be our "bad" task in the proof;*
– *DMR'_b as the DMR of τ_b in Φ'.*

To show that Φ' has a "worst DMR" that is no smaller than that of Φ^g, we now describe a transform that turns Φ^g into Φ' while also demonstrating that DMR'_b is greater than or equal to DMR^g_c. This transform is illustrated in Table 3.2, with numbers indicating tasks, and with tasks that will be moved underlined. The transformation proceeds as follows (starting from Φ^g) :

1. *lower τ_b to the position just above τ_c without altering the relative priority order of the other tasks ; this only alters the response times, and thus the DMRs, of higher priority tasks, and can only improve these values ;*

2. *swap the priorities of tasks τ_c and τ_b ; task τ_b then has a DMR greater than or equal to DMR^g_c (otherwise the Lazy and Greedy Algorithm would have selected it—or another task—instead of τ_c) ;*

3. *permute the tasks in $LP(\tau_b)$ so that they are in the same relative priority order as in Φ' ; this does not alter the response time, and thus the DMR, of task τ_b ;*

4. *lower the priority of task τ_b to place it at its appropriate position among the tasks in $LP(\tau_b)$ as defined by the priority ordering Φ' ; this cannot make τ_b's DMR better ;*

5. *permute the tasks in $HP(\tau_b)$ to finally obtain Φ' ; this last operation does not alter τ_b's DMR.*

Following this process guarantees that, in Φ', at least one task (τ_b) has a DMR greater than or equal to DMR^g_c. This concludes the proof that the priority assignment found by the Lazy and Greedy Algorithm is optimal for MPAP

	(low P.) ←			→ (high P.)			
$\Phi^g =$	6	5	4	**3**	2	*1*	
	6	5	4	**3**	*1*	2	(1. lower τ_b)
	6	5	4	*1*	**3**	2	(2. swap τ_c and τ_b)
	4	5	6	*1*	**3**	2	(3. permute $LP(\tau_b)$)
	4	*1*	5	6	**3**	2	(4. lower τ_b)
$\Phi' =$	4	*1*	**3**	5	6	2	(5. permute $HP(\tau_b)$)

TABLE 3.2 – Example process leading from a permutation Φ^g to another one Φ' with τ_c=task **3** and τ_b=task *1*.

Complexity Let us consider the non-lazy version of LGA, i.e., not picking a task just because its DMR is better than the current worst DMR in Φ. Then, numbering iterations from n to 1 (reflecting the number of unassigned tasks), iteration i has to compute the DMRs of i tasks and choose the best of these i tasks to assign. Assuming that a DMR is computed with a fixed cost, this means that iteration i has complexity $O(i)$ and that the complete algorithm has a complexity of $O(n(n+1)/2)$ DMR computations.

In the normal (lazy) version of LGA, many searches for the best task to choose will be interrupted early, which may lead to significant speed-ups. In the best case, the algorithm has complexity $O(n)$ DMR computations.

3.5 Solutions for the Average Priority Assignment Problem

The *max* criterion used in MPAP can be seen as pessimistic as it optimizes the worst DMR among all tasks. A different objective is to optimize the *average* DMR over all tasks, which is equivalent to using a sum operator instead of max. The sum and max criteria can be written as :

$$g(\Phi) = \bigoplus_{\tau_i \in \Phi} DMR_i(\Phi), \tag{3.6}$$

where $\bigoplus$ represents either the *sum* ($\sum$) or the *max* operator.

This section investigates solutions to APAP (i.e. the optimization of the summation criterion). First we provide a definition of what we mean by an optimal solution in this case.

Definition 17 (Optimal algorithms for APAP) *Let Γ be a set of constrained-deadline tasks with probabilistic execution times, where each task is characterized by parameters $(\mathcal{C}_i, T_i, D_i)$. A priority assignment algorithm P is optimal with respect to APAP if the priority ordering determined by algorithm P, for any arbitrary taskset Γ, has a value for the metric $g(\Phi)$ (as defined by Equation (3.6)), which is no larger than that obtained with any other priority ordering.*

3.5.1 When Being Greedy is not the Best Choice

A first question is whether the Lazy and Greedy Algorithm produces an optimal solution for APAP. As we will now see, a simple example demonstrates that this is not the case.

Theorem 6 *The Lazy and Greedy Algorithm may return a solution with a non-optimal average DMR.*

The above theorem is proved by the following counter example, using a time interval $[a, b]$ corresponding to one hyperperiod.

Example 9 (counter-example for the *sum* criterion) *Let $\tau = \{\tau_1, \tau_2\}$ with each task characterized by parameters $(\mathcal{C}_i, T_i, D_i)$. We consider τ_1 defined by $(\begin{pmatrix} 1 & 3 \\ 0.5 & 0.5 \end{pmatrix}, 4, 2)$ and τ_2 defined by $(\begin{pmatrix} 1 & 2 & 4 \\ 0.3 & 0.2 & 0.5 \end{pmatrix}, 4, 4)$.*

For this example, we can compute the response times, the DMRs, and the values of the sum criterion for the two possible Priority orderings :

– $\tau_2 > \tau_1$:

$$\mathcal{R}_1 = \begin{pmatrix} 1 & D_1^+ \\ 0.5 & 0.5 \end{pmatrix}$$

$$\mathcal{R}_2 = \begin{pmatrix} 2 & 3 & 4 & D_2^+ \\ 0.15 & 0.1 & 0.15 & 0.6 \end{pmatrix}$$

	$\tau_2 > \tau_1$	$\tau_1 > \tau_2$
DMR_1	0.5	0.85
DMR_2	0.6	0
sum	1.1	0.85

TABLE 3.3 – Results for Example 9 depending on the task priorities.

– $\tau_1 > \tau_2$:

$$\mathcal{R}_2 = \begin{pmatrix} 1 & 2 & 4 \\ 0.3 & 0.2 & 0.5 \end{pmatrix}$$

$$\mathcal{R}_1 = \begin{pmatrix} 2 & D_1^+ \\ 0.15 & 0.85 \end{pmatrix}$$

Table 3.3 gives the DMR of each task and the value of the *sum* criterion depending on the priority assignment.

Algorithm 3 would pick task τ_2 to be at the lower priority level, since with that priority it would have a deadline miss ratio of $DMR_2 = 0.6$, which is less than the deadline miss ratio of $DMR_1 = 0.85$ that task τ_1 would have if it were assigned that priority.

This priority assignment would be a good solution if we tried to minimize the maximum DMR, but when we look at the sum of the two deadline miss ratios, we see that, with task τ_1 at the higher priority and task τ_2 at the lower priority, the sum of the deadline miss ratios is 1.1. However, if instead we reverse the priority order, we obtain a sum of the deadline miss ratios of 0.85, which is less than the solution provided by Algorithm 3, thus proving that Algorithm 3 is not an optimal algorithm for APAP.

3.5.2 Tree Search algorithm

As the Lazy and Greedy Algorithm is not optimal, we need to resort to a different approach.

To optimize $g(\Phi) = \sum_i DMR_i(\Phi)$, a simple approach is to use a tree search algorithm enumerating all solutions. Among various possible tree search algorithms, we choose here Depth-First Search (DFS), which explores each branch as far as possible before backtracking. As in previous algorithms, we start with the lowest priority, extending the partial priority ordering Φ progressively as we go down the tree. The partial assignment Φ obtained in any node of the tree gives a lower bound for any complete assignment Φ' containing it ($\Phi \subseteq \Phi'$ and $|\Phi'| = n$) :

$$\begin{aligned} g(\Phi) &= \sum_{\tau_i \in \Phi} DMR_i(\Phi) = \sum_{\tau_i \in \Phi} DMR_i(\Phi') \\ &\leq \sum_{\tau_i \in \Phi'} DMR_i(\Phi') = g(\Phi'). \end{aligned}$$

This allows for :

– incrementally refining the lower bound while going down the tree : $g(\Phi.\tau_i) = g(\Phi) + DMR_i(\Phi)$; and

– pruning branches when the current lower bound is higher than the best complete solution found so far (variable g^{best} initialized to $+\infty$).

The resulting process is detailed in Algorithm 4.

Because of the different criteria optimized in APAP, one cannot be as lazy as in LGA. Nevertheless, if a task is encountered with a DMR of zero, then the search loop can also be interrupted early.

Complexity As in any tree search algorithm, the worst-case scenario is when all candidate solutions have to be visited, which means a complexity of $O(n!)$ (again assuming a fixed cost for computing a task's response time and thus a DMR).

Improvement Other natural candidate algorithms are best-first search algorithms, e.g., A* or IDA* [Russell and Norvig, 1995]. In all cases an important point would be to improve the lower bounds by using an admissible heuristic which can be obtained :

– by solving all problems with, e.g., 2 or 3 tasks ; but the cost of these precomputations could be detrimental ;

– by using Algorithm 3, which will presumably return a first good candidate solution to complete the current partial priority assignment ; or

– by using their current DMRs to decide in which order to try candidate tasks.

Note that there is no point trying to refine this search algorithm as long as one does not know how problems are distributed. Indeed, as stated by the no-free-lunch theorem(s) for search [Wolpert and Macready, 1995], there is no "best" overall algorithm. Given two algorithms A and B, one will be better than the other on a subset of problems.

Algorithm 4: Depth-First Search

```
$f_{best} = +\infty$ /* best value so far (glob. var) */
$\Gamma = \{\tau_i, i \in 1..n\}$ /* source set of tasks */
$(\Phi, g) \leftarrow$ RECUR$(\Gamma, (), n, 0)$
return $(\Phi, g)$

/* Function recursively completing the current solution $\Phi$. */
RECUR$(\Gamma, \Phi, l, g)$ /* Note : $g = g(\Phi)$ */
/* If priority level 0 is attained, we have a complete solution. */
if $l = 0$ then
    /* Is this solution the new best solution ? */
    if $g < g^{best}$ then
        $g^{best} \leftarrow g$
    return $(\Phi, g)$
/* Otherwise, if the current partial solution is worse than the best solution so far, then backtrack. */
if $g \geq g^{best}$ then
    return $(\Phi, g)$
/* Try each unassigned task $\tau_i$ at the current priority level. */
$(\Phi^{min}, g^{min}) \leftarrow ((), +\infty)$
for $\tau_i \in \Gamma$ do
    $\delta \leftarrow DMR_i(\Phi)$
    /* Get the best solution completing $\Phi.\tau_i$. */
    $(\Phi', g') \leftarrow$RECUR$(\Gamma \backslash \{\tau_i\}, \Phi.\tau_i, l-1, g+\delta)$
    /* Memorize the best completed solution. */
    if $g' < g^{min}$ then
        $(\Phi^{min}, g^{min}) \leftarrow (\Phi', g')$
    /* If task $\tau_i$ has a null DMR, then backtrack. */
    if $\delta = 0$ then
        break
/* Return the best completed solution. */
return $(\Phi^{min}, g^{min})$
```

One could also exploit the fact that, if Φ and Φ' are partial and contain the same tasks, then one only needs to develop the subtree for the one with the lowest value.

Finally, a reversed search (setting highest priority tasks first) would give a simple admissible—but not necessarily very informative—heuristic by computing the deadline miss ratio of all remaining tasks as if each were the next task.

Tree-search algorithms can be used with other criteria, for example to optimize the mean squared DMR with $g(\Phi) = \sum_{\tau_i \in \Phi} DMR_i^2(\Phi)$, which provides an intermediate objective between the worst and the average criteria.

3.6 Conclusions

In this chapter we indicated that a key problem that needs to be addressed by probabilistic analysis is the calculation of the timing failure rate, the rate at which we can expect deadlines to be missed during the operation of a system. This failure rate can then be compared with reliability requirements to determine if the system is acceptable.

As a first step towards a solution to this problem, we proposed three sub-problems. These sub-problems relate to finding an optimal priority assignment for fixed priority pre-emptively scheduled systems with probabilistic execution times. The problems involve optimising three different metrics based on the probability of tasks missing their deadlines over some interval of time.

The three sub-problems are as follow :

1. Basic Priority Assignment Problem (**BPAP**). This problem involves finding a priority assignment such that the deadline miss ratio of every task does not exceed the threshold specified.
2. Minimization of the Maximum Priority Assignment Problem (**MPAP**). This problem involves finding a priority assignment that minimizes the maximum deadline miss ratio of any task.
3. Average Priority Assignment Problem (**APAP**). This problem involves finding a priority assignment that minimizes the sum of the deadline miss ratios for all tasks.

For each sub-problem we proposed an optimal algorithm. The first two algorithms were inspired by Audsley's algorithm which is a greedy, lowest priority first, approach that is optimal for the equivalent deterministic case. As shown in Section 3.5 such a greedy approach is not suitable for the third sub-problem (APAP). Therefore we proposed a tree search algorithm in this case.

As future work we intend to provide conditions that need to be met in order to obtain an optimal priority ordering for APAP using a greedy algorithm. Such general conditions exist for the deterministic case [Davis and Burns, 2011] and our first step will be to consider their extension to the probabilistic case.

The results presented in this chapter were published in the International Conference on Real Time and Network Systems (RTNS2011) that took place in Nantes in November 2011 [Maxim et al., 2011].

Chapitre 4

Controller Area Networks with probabilistic jitter

4.1 Introduction

Context of the study Controller Area Network (CAN) is a communications bus widely used in automotive applications. Originally developed in 1983 by Bosch and included in a car for the first time in 1991 (by Mercedes), nowadays every new car produced in Europe is equipped with at least one CAN bus [CAN, 1992].

The real-time performances of periodic activities on CAN bus (and distributed or networked systems in general) have been extensively studied. For these activities, the corresponding response times can be evaluated when all their parameters are known, but this is not the case for all real-time applications. Some parameters can be unknown until the time instant when the activity is released and/or the environment can change forcing the application to adapt by changing its parameters. For instance, the problem is quite different for activities with queuing jitter since, in many practical cases, it is not possible to have a precise knowledge of the activation pattern before the implementation of the system. This is the case of messages subject to soft real-time constraints exchanged among Electronic Control Units (ECUs) in the body network of a vehicle. Such messages are usually assigned a low priority and they do not delay the hard real-time periodic traffic. However it is difficult to estimate their own response times using worst-case techniques which provide an overly pessimistic evaluation, leading to severe over-provisioning of the system.

Problem definition In this chapter we discuss the problem of evaluating the real-time performances of periodic messages with queuing jitter (described by a random variable) that are scheduled on a CAN bus. We deal in this context with non-preemptive fixed-priority scheduling under the hypothesis that each message has its own priority different from the priorities of other messages. We are interested in obtaining distribution functions for the response times of messages, be they periodic or periodic with queuing jitter. We are here in the realm of soft real-time constraints, but their satisfaction is important since large response times may jeopardize the execution of a function and may even raise safety concerns in some cases. In addition, low responsiveness is negatively perceived by the user.

4.2 Model

We consider a set τ of n synchronous constrained deadline messages $\{\tau_1, \tau_2, \cdots, \tau_n\}$ scheduled on CAN. Each message has a unique priority and the set of messages is ordered in the decreasing order of their priorities such that τ_1 has the highest priority and τ_n the lowest priority. Each message τ_i is characterized by $(C_i, \mathcal{J}_i, T_i, D_i)$ where C_i is the worst-case transmission time, $\mathcal{J}_i$ the worst-case queuing jitter, T_i the exact period and D_i the relative deadline with $D_i \leq T_i$. The worst-case queuing jitter is the longest time between the event initiating the release of the message and the moment when the message is ready to be transmitted on the bus. Similar to the pWCET and pMIT presented in Chapter 2, the jitter $\mathcal{J}_i$ is described by a random variable representing the worst-case jitter with a known probability function (PF) $f_{J_i}(j) = P(\mathcal{J}_i \leq j)$ as the probability for τ_i to have the jitter smaller or equal to j. The values of $\mathcal{J}_i$ belong to $\{J_i^{min}, \cdots J_i^{max}\}$. The worst-case queuing jitter $\mathcal{J}_i$ could be written as follows
$\mathcal{J}_i = \begin{pmatrix} J_i^0 = J_i^{min} & J_i^1 & \dots & J_i^{k_i} = J_i^{max} \\ f_{\mathcal{J}_i}(J_i^{min}) & f_{\mathcal{J}_i}(J_i^1) & \dots & f_{\mathcal{J}_i}(J_i^{max}) \end{pmatrix}$, where $\sum_{k=0}^{k_i} f_{\mathcal{J}_i}(J_i^k) = 1$. If $J_i^{min} = J_i^{max}$ then the model presented here is the same as the scheduling model introduced in [Davis et al., 2007]. For instance for a message τ_i we have

$$\mathcal{J}_i = \begin{pmatrix} 2 & 3 & 5 \\ 0.50 & 0.45 & 0.05 \end{pmatrix}.$$

We assume that all random variables are independent and all considered values are multiples of τ_{bit} which is the transmission time of a single bit on CAN.

We denote by $hp(i)$ the set of indices of the messages with higher priority than τ_i and by $lp(i)$ the set of indices of messages with lower priority than message τ_i.

For any iterative algorithm, we denote by $W^{(k)}$ the value of a variable W obtained at the k^{th} iteration of the algorithm.

The response time of an instance of a message is the elapsed time between its release and the end of its execution (see Figure 4.1 for a graphical illustration). Since we consider messages with probabilistic jitters then the response time of an instance is a random variable. Let $\mathcal{R}_{i,j}$ be the response time of the j^{th} instance of message τ_i. Let $\mathcal{R}_i$ be the worst-case response time of a message τ_i.

We consider the problem of calculating the distribution of the worst-case response time of a message scheduled on CAN. Each node (connected to CAN) is assumed to be capable of ensuring that at any given time when arbitration starts, the highest priority message queued at that node is entered into arbitration.

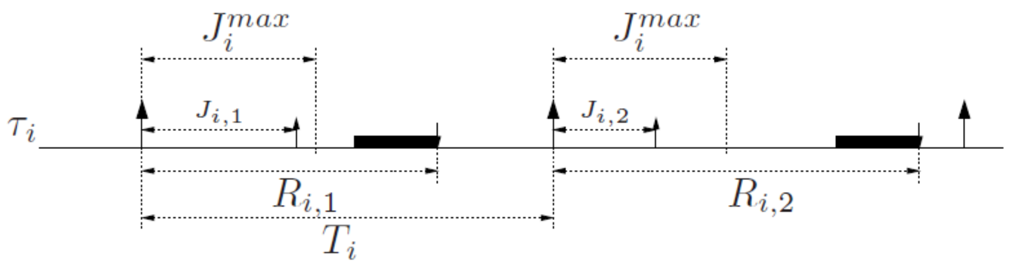


FIGURE 4.1 – An example of a message τ_i. Here, $\mathcal{J}_{i,1}$ and $\mathcal{J}_{i,2}$ are jitters of different instances of message i and are upper bounded by $\mathcal{J}_i^{max}$. Also, $\mathcal{R}_{i,1}$ and $\mathcal{R}_{i,2}$ are response times of different instances of message i and are not necessarily equal in value but they both follow the same probability distribution.

4.3 Deterministic analysis for Controller Area Networks

We present here existing results on the calculation of worst-case response time of a message with exact jitter scheduled on CAN [Davis et al., 2007]. Since we consider exact jitter (aka deterministic jitter), then each message τ_i is characterized by its worst-case queuing jitter J_i. We denote by R_i the worst-case response time of a message τ_i.

According to the schedulability analysis presented in [Davis et al., 2007], we compute first the length of the busy period t_i for a message τ_i by solving iteratively

Equation (4.1).

$$t_i = B_i + \sum_{k \in hp(i) \cup \{i\}} \left\lceil \frac{t_i + J_k}{T_k} \right\rceil C_k, \tag{4.1}$$

where the blocking term $B_i = \max_{k \in lp(i)} C_k$. The iterative algorithm solving Equation (4.1) starts with $t_i^{(0)} = C_i$ and it stops at step k if $t_i^{(k)} = t_i^{(k-1)}$ [Davis et al., 2007].

Then the worst-case response time of τ_i is obtained as the largest value among the first Q_i instances of τ_i as follows :

$$R_i = \max_{q=0 \ldots Q_i - 1} \{W_i(q) - qT_i + C_i + J_i\} \tag{4.2}$$

where $Q_i = \left\lceil \frac{t_i + J_i}{T_i} \right\rceil$ and $W_i(q)$ is the solution of the following equation :

$$W_i(q) = B_i + qC_i + \sum_{k \in hp(i)} \left\lceil \frac{W_i(q) + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{4.3}$$

Equation (4.3) is solved iteratively by considering at the first step $W_i^{(0)}(q) = B_i + qC_i$ and the algorithm stops at step l if $W_i^{(l)}(q) = W_i^{(l-1)}(q)$.

4.4 Probabilistic feasibility analysis : worst-case response time

We present now our contribution, which is *the worst-case response time analysis of a periodic message with jitter described by a random variable.* This work extends the results given in [Davis et al., 2007] from the deterministic case to the probabilistic case.

In the deterministic case the calculation of the worst-case response time is based on the calculation of t_i, Q_i and $W_i(q), \forall q = 0, \cdots, Q_{i-1}$. For each variable we provide below the corresponding probabilistic calculation.

Probabilistic t_i

For a message τ_i we compute t_i using Equation (4.1) with $J_k = J_k^{max}, \forall k \in hp(i) \cup \{i\}$ thus we have

$$t_i = B_i + \sum_{k \in hp(i) \cup \{i\}} \left\lceil \frac{t_i + J_k^{max}}{T_k} \right\rceil C_k \tag{4.4}$$

where the blocking term $B_i = \max_{k \in lp(i)} C_k$. As in the deterministic case the iterative algorithm starts with $t_i^{(0)} = C_i$ and it stops at step k if $t_i^{(k)} = t_i^{(k-1)}$.

Probabilistic Q_i

For a message τ_i we compute Q_i with $J_k = J_k^{max}, \forall k \in hp(i) \cup \{i\}$, thus in the probabilistic case we have

$$Q_i = \left\lceil \frac{t_i + J_i^{max}}{T_i} \right\rceil . \tag{4.5}$$

Probabilistic $\mathcal{W}_i(q)$

For a message τ_i and any $q = 0, \cdots, Q_i - 1$, the probabilistic $\mathcal{W}_i(q)$ is a random variable and it is obtained as the solution of the following equation :

$$\mathcal{W}_i(q) = (\otimes_{k \in hp(i)} \left\lceil \frac{\begin{pmatrix} \tau_{bit} \\ 1 \end{pmatrix} \otimes \mathcal{W}_i(q) \otimes \mathcal{J}_k}{T_k} \right\rceil C_k) \otimes \begin{pmatrix} B_i + qC_i \\ 1 \end{pmatrix} \tag{4.6}$$

.

Equation (4.6) is solved iteratively by Algorithm 1 and it starts with $\mathcal{W}_i^{(0)}(q) = \begin{pmatrix} B_i + qC_i \\ 1 \end{pmatrix}$. Algorithm 1 stops at step l if $\mathcal{W}_i^{(l)}(q)$ and $\mathcal{W}_i^{(l-1)}(q)$ have exactly the same set of values. Note that it it is not necessary for the two distributions to have the same probabilities, having the same set of values suffices. $\mathcal{W}_i^{(l)}(q)$ will be slightly more pessimistic than $\mathcal{W}_i^{(l-1)}(q)$.

The solution $\mathcal{W}_i$ is equal to $\mathcal{W}_i^{(l-1)}(q)$, thus we keep the probabilities obtained at step $l-1$. For instance Algorithm 1 stops at step l if $\mathcal{W}_i^{(l-1)}(q) = \begin{pmatrix} 6 & 8 & 15 \\ 0.5 & 0.3 & 0.2 \end{pmatrix}$ and $\mathcal{W}_i^{(l)}(q) = \begin{pmatrix} 6 & 8 & 15 \\ 0.4 & 0.3 & 0.3 \end{pmatrix}$.

Obtaining the distribution of the worst-case response time $\mathcal{R}_i$

For each message τ_i and $q = 0, \cdots, Q_i - 1$ we calculate the response time $\mathcal{R}_i(q)$ of the q^{th} instance of message τ_i as follows :

$$\mathcal{R}_i(q) = \mathcal{W}_i(q) \otimes \begin{pmatrix} -qT_i + C_i \\ 1 \end{pmatrix} \otimes \mathcal{J}_i \tag{4.7}$$

The worst-case response time $\mathcal{R}_i$ of message τ_i is obtained as the envelope of the response times $\mathcal{R}_i(q), \forall q = 0, \cdots, Q_i - 1$. The calculation of the envelope uses

Algorithm 1 Algorithm solving iteratively Equation(4.6)

```
Input: $\tau$ and $q$ ;
Output: $\mathcal{W}_i(q)$ ;
  $B_i = \max_{k \in lp(i)}\{C_k\}$ ;
  $\mathcal{W}_i^{(0)}(q) = \begin{pmatrix} B_i + qC_i \\ 1 \end{pmatrix}$ ;
  while $\mathcal{W}_i^{(l)}(q)$ and $\mathcal{W}_i^{(l-1)}(q)$ don't have exactly the same values do
    $\mathcal{W}_i^{(l+1)}(q) = (\otimes_{k \in hp(i)} \left\lceil \frac{\begin{pmatrix} \tau_{bit} \\ 1 \end{pmatrix} \otimes \mathcal{W}_i^{(l)}(q) \otimes \mathcal{J}_k}{T_k} \right\rceil C_k) \otimes \begin{pmatrix} B_i + qC_i \\ 1 \end{pmatrix}$ ;
  end while
```

the notion of *cumulative distribution function* (CDF) $F_{\mathcal{X}}$ associated to a random variable $\mathcal{X}$ with $F_{\mathcal{X}}(x) = P(\mathcal{X} \le x)$. We obtain

$$\mathcal{R}_i = \max_{q=0,\cdots,Q_i-1}\{1 - F_{\mathcal{R}_i(q)}\} \tag{4.8}$$

4.5 Validation, correctness and complexity

In this section we validate the proposed approach (Algorithm 1) according to the conditions provided in Section 5.3. Furthermore, we prove the correctness and the complexity of our approach. Possible solutions to reduce the complexity are also discussed.

Theorem 7 (Convergence) *Algorithm 1 converges (i.e. the number of iterations is finite) as long as any algorithm, solving the corresponding deterministic sub-problems obtained by considering all combinations of jitters, converges.*

Proof 10 *The theorem is proved by applying on each value of the response time a reasoning similar to that proposed in [Joseph and Pandya, 1986].*

From Theorem 7 we obtain directly the validation of the robustness condition presented in Section 5.3. Moreover, concerning the first part of the worst-case condition, Theorem 7 ensures that the periodic messages are taken into account by the analysis.

Theorem 8 (Worst-case) *Algorithm 1, which solves iteratively Equation (4.6), computes exactly the same results as the deterministic analysis given in [Davis et al., 2007] in the case where we consider only random variables with one unique value.*

Proof 11 *In Algorithm 1 only the loop calculates new values for the distribution function of the response time using Equation (4.6). In the case of all random variables having one unique value* $\mathcal{J}_i = \begin{pmatrix} J_i \\ 1 \end{pmatrix}, \forall i$*, then,* $W_i(q)$ *as well as* $\mathcal{R}_i$ *have single values.*

From Theorem 8 we obtain directly the validation of the initial condition presented in Section 5.3. Moreover, Theorem 8 also guarantees the worst-case condition of Section 5.3 as the worst-case response time provided by the deterministic analysis being included in the resulting probabilistic response time.

Theorem 9 (Correctness) *Let* τ *be a set of synchronous constrained deadline messages. Algorithm 1, which solves iteratively Equation (4.6), provides for each message* τ_i *a probabilistic worst-case response time* $\mathcal{R}_i^{our}$ *which is pessimistic with respect to the exact response distribution of* τ_i *obtained by exhaustively enumerating all possible combinations of jitter values that the messages in* τ *could experience.*

Proof 12 *To prove the correctness of our approach we analyze all the steps of the approach :*

I) ***Computation of*** t_i ***and*** Q_i

Since Equation (4.4) is equivalent to Equation (4.1), then the busy period corresponding to the response time is the same in case of the probabilistic and of deterministic analysis. The number of message instances $Q_i = \lceil \frac{t_i + J_i^{max}}{T_i} \rceil$ *is the same for all the analyses with* $J_i^{max} = J_i$ *and* $Q_i = \lceil \frac{t_i + J_i}{T_i} \rceil$*.*

II) ***Computation of*** $\mathcal{W}_i$*. The formulation we propose in Equation (4.6) considers all the combinations of message jitters, including* J_i^{max} *applied to the worst-case response time, as in Equation (4.3). Moreover, the convolution operator applied in Algorithm 1 ensures that the obtained response time distribution contains the worst-case queuing delay as well as all other possible queuing delays* $W_i(q)$ *(with* $W_i(q) < W_i^{max}$*) that might result from different jitter combinations.*

III) ***Computation of*** $\mathcal{R}_i^{our}$*. The response time* $\mathcal{R}_i^{our}(q)$ *computed by Equation (4.8) by applying* $\mathcal{W}_i$*, takes into account the worst-case queuing delay and other* W_i *values to compute the worst-case response time distribution. It includes the worst-case response time* R_i^{max}*.*

IV) ***Computation of the envelope****. According to the definition of the envelope for different* q_i *instances, (see Equation (4.8)), the worst-case response time* R_i^{max} *is among the values of* $R_{i,j}$ *(with* $R_{i,j} < R_i^{max}$*).*

Steps I) - IV) prove the correctness of Algorithm 1.

The computation of the worst-case response time in the deterministic case has been proven to be NP-hard [Eisenbrand and Rothvob, 2008]. In the probabilistic case, the response time computation inherits this complexity. Furthermore, the fact that convolutions of random variables are performed makes the computation of the response time more difficult, but still the number of iterations is finite as long as the number of iterations of any algorithm, solving the deterministic corresponding sub-problems, is finite.

Re-sampling techniques on random variables similar to those applied for execution times (Chapter 6) could be extended to the case of jitter distributions. The development of such techniques is left as future work.

4.6 Numerical example

Let τ be a set of three messages with, each message being described by $\tau_i = (C_i, \mathcal{J}_i, T_i, D_i)$:

$\tau_1 = (1, \begin{pmatrix} 0 & 1 \\ 0.7 & 0.3 \end{pmatrix}, 3, 3)$,

$\tau_2 = (2, \begin{pmatrix} 0 & 1 & 2 \\ 0.5 & 0.3 & 0.2 \end{pmatrix}, 10, 10)$ and

$\tau_3 = (1, \begin{pmatrix} 0 & 1 \\ 0.8 & 0.2 \end{pmatrix}, 4, 4)$.

The messages are ordered in decreasing order of their priorities such that τ_1 has the highest priority and τ_3 the lowest priority.

Deterministic analysis

The deterministic analyses considers all jitters equal to J_i^{max} and it provides the worst-case response times $R_1 = 4 > D_1 = 3$ (for $t_1 = 4$, $Q_1 = 2$), $R_2 = 6$ (for $t_2 = 5$, $Q_2 = 6$) and $R_3 = 6 > D_3 = 4$ (for $t_3 = 7$, $Q_3 = 2$). Figure 4.2 depicts a scenario in which τ_3 experiences its worst-case response time and misses its deadline. While it is possible that this scenario occurs in practice, the chances of its occurrence might be vanishingly small and the deterministic doe not provide its probability of occurrence, only that it can happen.

Combinatorial Analysis

By considering all combinations of jitters (2 values for τ_1, 3 values for τ_2 and

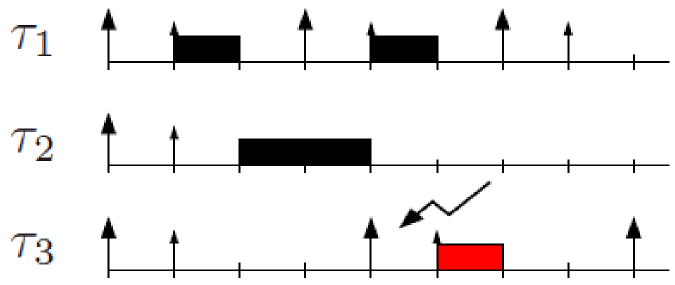


FIGURE 4.2 – Example of a scheduling scenario where τ_3 misses its deadline : $R_1 = 2 < D_1$, $R_2 = 4 < D_2$ and $R_3 = 6 > D_3$.

2 values for τ_3) we obtain two scenarios for τ_1, 6 scenarios for τ_2 and 12 scenarios for τ_3. Each scenario corresponds to a deterministic schedulability problem, thus to a possible value for the worst-case response time. By applying the deterministic schedulability analysis, presented in Section 4.3, to each of these scenarios we obtain the following combinatorial worst-case response time distributions :

$$\mathcal{R}_1^{combinatorial} = \begin{pmatrix} 3 & 4 \\ 0.7 & 0.3 \end{pmatrix},$$

$$\mathcal{R}_2^{combinatorial} = \begin{pmatrix} 4 & 5 & 6 \\ 0.5 & 0.3 & 0.2 \end{pmatrix} \text{ and}$$

$$\mathcal{R}_3^{combinatorial} = \begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 0.28 & 0.07 & 0.28 & 0.31 & 0.06 \end{pmatrix}.$$

We note that the combinatorial worst-case response time is not a complete probabilistic worst-case response time since it considers that all instances of the same message have the same jitter.

Probabilistic analysis

Our probabilistic analysis provides distributions of the worst-case response times by means of probabilistic operation such as convolution and thus does not need to exhaustively enumerate all the possible scenarios that can occur in the lifetime of the system. All these scenarios and their combinations are implicitly obtained by the analysis.

We obtain for message τ_1 : $t_1 = 4$ and $Q_1 = 2$. Thus we have :

$$\mathcal{W}_1(0) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathcal{R}_1(0) = \begin{pmatrix} 3 & 4 \\ 0.7 & 0.3 \end{pmatrix},$$

$\mathcal{W}_1(1) = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$ and $\mathcal{R}_1(1) = \begin{pmatrix} 1 & 2 \\ 0.7 & 0.3 \end{pmatrix}$

From these results we can conclude that τ_1 has a 30% probability of having the worst-case response time larger than the deadlines and a 70% probability of finishing execution before its deadline.

We obtain for message $\tau_2 : t_2 = 5$ and $Q_2 = 1$. The response time distribution of τ_2 is $\mathcal{R}_2(0) = \begin{pmatrix} 4 & 5 & 6 \\ 0.5 & 0.3 & 0.2 \end{pmatrix}$ with $\mathcal{W}_2(0) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

We obtain for message $\tau_3 : t_3 = 7$ and $Q_3 = 2$.

For the first instance of τ_3 we have

$\mathcal{W}_3(0) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 0.0007 & 0.0162 & 0.0339 & 0.7409 & 0.2083 \end{pmatrix}$,

and consequently $\mathcal{R}_3(0) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0.0005 & 0.0131 & 0.0304 & 0.5996 & 0.3148 & 0.0416 \end{pmatrix}$.

For its second instance we obtain $\mathcal{W}_3(1) = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$ and $\mathcal{R}_3(1) = \begin{pmatrix} 2 & 3 \\ 0.8 & 0.2 \end{pmatrix}$.

By taking the envelope of the two response time distributions, we obtain :

$\mathcal{R}_3 = \begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 0.0136 & 0.0304 & 0.5996 & 0.3148 & 0.0416 \end{pmatrix}$

which is the response time distribution in the worst case of message τ_3.

Since τ_3 has a deadline $D_3 = 4$, then there are $P(R_3 > 4) = 0.3564 = 35.64\%$ chances that τ_3 has the worst-case response time larger than its deadlines.

4.7 Conclusion and future work

In this chapter we extend an existing deterministic analysis for CAN to the case of messages with queuing jitters described by random variables. The proposed formulation is validated using a three steps method that covers three aspects of a probabilistic approach : ensuring that the probabilistic analysis generalizes the deterministic one, complexity issues and treating mix systems. These three aspects are inherited from the satisfaction of the real-time constraints. Moreover the proposed analysis is proved correct and its complexity is provided.

A direct implication of our results is the utilisation of such analysis to design CAN-based systems that are less pessimistic. For instance the possibles jitter values can be used to define the "good” values for the watchdog timers in communication networks.

Chapitre 5

Response-time analysis of systems with multiple probabilistic parameters

In this chapter we consider synchronous tasks and we provide a theorem (see Theorem 10 in Section 5.2.1) indicating that the response time distribution of the first job of any task is the greatest among all jobs of that task. Here greatest is defined with respect to the relation $\succeq$ and it indicates that the response time distribution of the first job upper bounds the response time distribution of any other job of that task. Thus calculating the response time distribution of the first job of a task provides the worst-case response time distribution of the task.

Problem : In this chapter we address the problem of computing the response time distributions and, implicitly, Deadline Miss Probabilities of tasks with pMIT and pWCET. The response time of a job is the elapsed time between its release and its completion. Since we consider jobs with probabilistic parameters, the response time of a job is also described by a random variable. The DMP of a job is obtained by comparing the response time distribution of said job and its deadline, be it a probabilistic deadline or a deterministic one. This is a novel problem, and the fact that the system under consideration has more than one task parameter given as a distribution makes it a complex one. The solution that we describe is exponential in the number of tasks and the size of the random variables representing the task parameters. We describe techniques to decrease the analysis duration and to make it tractable. Note that in [Stigge and Yi, 2012], the authors have shown that there can not be any pseudo-polynomial exact test for fixed-priority task-level scheduling of the non-cyclic GMF task model, or any more general model - that encompasses it - such as the probabilistic task model.

Before we proceed with the proposed analysis, we first present main the differences between two model used to represent arrivals of jobs generated by probabilistic tasks. Both of these models are used in the state of the art techniques for modeling teal0time systems but with different types of analysis. For the analysis that we are interested in, it is better suited to use pMIT distributions, as presented in Section 2.2 to describe job arrivals.

5.1 Different models for probabilistic arrivals of real-time tasks

In this section, we present the differences between two models of probabilistic arrivals that co-exist in the real-time literature : one associating probabilities to the possible values of the inter-arrival times and a second model associating probabilities to the number of arrivals within a time interval. We recall the first model in Section 5.1.1 and in Section 5.1.2 we present the second model. In Section 5.1.3 we provide a comparison of the two models.

5.1.1 Real-time systems with probabilistic MIT

Our paper uses the first model, that has been introduced in papers like [Lehoczky, 1996, Abeni and Buttazzo, 1998].

Within this model, for a task τ_i the pMIT $\mathcal{T}_i$ is defined by a distribution as follows :

$$\mathcal{T}_i = \begin{pmatrix} T^0 = T^{min} & T^1 & \cdots & T^k = T^{max} \\ f_{\mathcal{T}_i}(T^{min}) & f_{\mathcal{T}_i}(T^1) & \cdots & f_{\mathcal{T}_i}(T^{max}) \end{pmatrix}$$

For instance τ_1 has a pMIT $\mathcal{T}_1 = \begin{pmatrix} 5 & 10 \\ 0.3 & 0.7 \end{pmatrix}$ indicating that the MIT of τ_1 is equal to 5 with a probability of 0.3 and to 10 with a probability of 0.7 .

5.1.2 Real-time systems with probabilistic number of arrivals

This model has been used in papers like [Kaczynski et al., 2007, Broster and Burns, 2004a].

Within this model, for a task τ_i^* the number of possible arrivals $\mathcal{N}_i$ within a time interval of length t_Δ is defined by a distribution as follows :

$$\mathcal{N}_i = \begin{pmatrix} N^0 = N^{min} & N^1 & \cdots & N^k = N^{max} \\ f_{\mathcal{N}_i}(N^{min}) & f_{\mathcal{N}_i}(N^1) & \cdots & f_{\mathcal{N}_i}(N^{max}) \end{pmatrix}$$

For instance if $\mathcal{N}_1 = \begin{pmatrix} 1 & 2 & 4 \\ 0.4 & 0.3 & 0.3 \end{pmatrix}$ for $t_\Delta = 12$, then the task τ_1^* has at most 4 arrivals from $t = 0$ to $t = 12$.

5.1.3 Comparing the two models

We present here the main difference between the two models from Section 5.1.1 and Section 5.1.2.

We consider here the tasks defined in Sections 5.1.2 and 5.1.1. For those tasks only the parameters related to the arrival of the tasks are relevant to our discussion.

The first model provides information to a schedulability analysis, information that the second model does not provide

- **Probabilistic MIT** : The task τ_1 has at most two arrivals before $t = 7$ (with a probability 0.3).
- **Probabilistic number of arrivals** : It is not possible to estimate how many times τ_1^* was released from 0 to 7. Different situations are possible like those described in Figure 5.1.

The first model can also provide the information that the second model provides to a schedulability analysis

- **Probabilistic MIT** : From $t = 0$ to $t = 12$ there are three scenarios of arrivals for task τ_1 :
  - 3 arrivals at $t = 0$, $t = 5$ and $t = 10$ with a probability of 0.21 ;
  - 2 arrivals at $t = 0$ and $t = 5$ with a probability of 0.09 ;
  - 2 arrivals at $t = 0$ and $t = 10$ with a probability of 0.7.

  Thus, from $t = 0$ to $t = 12$ the possible number of arrivals of τ_1 is described by $\begin{pmatrix} 2 & 3 \\ 0.79 & 0.21 \end{pmatrix}$.

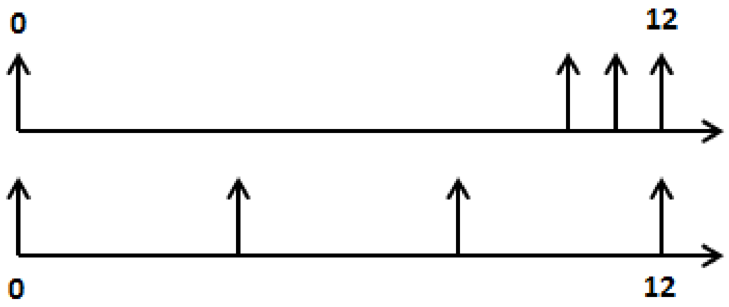


FIGURE 5.1 – The arrival distribution defined number of arrivals in a given interval may correspond to different situations.

- **Probabilistic number of arrivals** : from $t = 0$ to $t = 12$ the number of arrivals of τ_1^*
  $\mathcal{N}_1 = \begin{pmatrix} 1 & 2 & 4 \\ 0.4 & 0.3 & 0.3 \end{pmatrix}$ is provided by the model.

5.2 Response time analysis

In this section, we introduce an analysis computing the response time distribution of a given task. Since the system under consideration is a task-level fixed-priority preemptive one, then a given task is not influenced by tasks of lower priority but only by those of higher priority. Thus, we consider without loss of generality, the task of interest to be the lowest priority task, τ_n, in a set of n tasks.

Before we proceed to the response time analysis for tasks that have pWCET as well as pMIT, we first recall here the response time analysis for tasks that have only the WCET described by a random variable [Díaz et al., 2002]. The response time $\mathcal{R}_{i,j}$ of a job $\tau_{i,j}$ that is released at time instant $\lambda_{i,j}$ is computed using the following equation :

$$\mathcal{R}_{i,j} = \mathcal{B}_i(\lambda_{i,j}) \otimes \mathcal{C}_i \otimes \mathcal{I}_i(\lambda_{i,j}), \tag{5.1}$$

where $\mathcal{B}_i(\lambda_{i,j})$ is the accumulated *backlog* of higher priority tasks released before $\lambda_{i,j}$ and still active (not completed yet) at time instant $\lambda_{i,j}$. $\mathcal{I}_i(\lambda_{i,j})$ is the sum of the execution times of higher priority tasks arriving after $\lambda_{i,j}$ and that could preempt the job under analysis, $\tau_{i,j}$. The operator $\otimes$ is the convolution between two random variables. In the case of synchronous tasks the backlog of the first job of τ_n is equal to $\mathcal{B}_n = \bigotimes_{i \in hp(n)} \mathcal{C}_i$.

Equation (5.1) is solved iteratively, integrating new possible preemptions by modifying the tail of the response time distribution $\mathcal{R}_{i,j}$ at each iteration. The iterations stop once either there are no more preemptions to be integrated or all the newly obtained values in the tail of the response time distribution are larger than the tasks' deadline.

Intuitively : In the case when the MIT is also given as a random variable, we need to modify Equation (5.1) to take into account the fact that a preemption can occur at different time instants with different probabilities. We do so by making a copy of $\mathcal{R}_{i,j}$ for each value in the pMIT distribution of the preempting task and scaling each copy with the probability of its respective value. We then modify the tail of each copy in order to integrate, as in Equation (5.1), the execution requirement of the preempting task. The distributions obtained are then coalesced and the process is repeated until either there are no more preemptions to be integrated or the newly obtained values in the tails of each copy of the response time distribution are larger than the tasks' deadline. Note that, if the MIT of the preempting task is deterministic, then the analysis is the same as Equation (5.1). Furthermore, our analysis can handle any combination of probabilistic and deterministic parameters, and in the case that all parameters are deterministic the returned result is the same as the one provided by the worst case response time analysis in [Joseph and Pandya, 1986].

We present first a numerical example of the analysis before introducing it formally.

Example 13 *We introduce here an example of a task system and the response time computation for the lowest priority task of the system. In order to better understand the implication of the probabilistic parameters, let us start with a deterministic task set and slowly move our way to a probabilistic version of it.*

Given a task system $\tau = \{\tau_1, \tau_2\}$ *scheduled under task-level fixed-priority scheduling policy with* τ_1 *at higher priority and* τ_2 *at lower priority, and the tasks described by the following worst case values :* $\tau_1 = (C_1 = 2, T_1 = 5)$ *and* $\tau_2 = (C_2 = 4, T_2 = 7)$*, with* $D_i = T_i, \forall i$*. A deterministic analysis of this task system would conclude that it is unschedulable, since the response time of* $\tau_{2,1}$ *(the first job of* τ_2*) is greater than its deadline.*

First generalization (pMIT) : *Let us now assume that, after studying the system, we have extra information about its behaviour, namely that not all the jobs* *of τ_1 arrive with* 5 *units of time between them, but instead they follow a distribution equal to* $\mathcal{T}_1 = \begin{pmatrix} 5 & 6 \\ 0.2 & 0.8 \end{pmatrix}$ *meaning that an instance of τ_1 has a* 20% *probability of arriving* 5 *units of time after the previous instance, and a* 80% *probability of arriving* 6 *time units after the previous instance. All other parameters of the system keep their worst case representation.*

In this case, $\tau_{2,1}$ also has an 80% *probability of finishing execution before its deadline, i.e. the cases when $\tau_{1,2}$ arrives at $t = 6$.*

Second generalization (pWCET) : *Going back to the deterministic task system that we started with this time let us assume that we have extra information about the execution requirement of τ_2, namely that it follows the distribution* $\mathcal{C}_2 = \begin{pmatrix} 3 & 4 \\ 0.9 & 0.1 \end{pmatrix}$ *meaning that only* 10% *of the jobs generated by τ_2 have an execution requirement of* 4 *time units and the other* 90% *require only* 3 *time units. All other parameters of the system are considered with their worst case values.*

In this case as well $\tau_{2,1}$ has a high probability, 90%, *of finishing execution before its deadline, namely the cases when it requires* 3 *units of execution time.*

Combining the two cases : *If we now combine both cases presented above, taking into consideration the probabilistic nature of $\mathcal{T}_1$ and of $\mathcal{C}_2$, we note that $\tau_{2,1}$ misses its deadline only when the two worst case scenarios happen at the same time, i.e. $\tau_{1,2}$ arrives at $t = 5$ and $\tau_{2,1}$ needs to execute for* 4 *units of time. The probability of this happening is the combined probability of the two scenarios, namely $\tau_{2,1}$ has a probability of $DMP_2 = 0.2 \times 0.1 = 0.02 = 2\%$ of missing its deadline.*

For this example we have found the deadline miss probability of $\tau_{2,1}$ by (manually) exploring all the possible combinations of inter-arrival times and execution times of the two tasks. This is not always possible to do, considering that a system can have many tasks and each parameter distribution may have tens, hundreds or even thousands of values, leading to a large number of possible combinations.

The analysis that we introduce computes the worst case response time distribution of a task by means of convolution of random variables, ensuring in this way that all scenarios have been taken into account without needing to explicitly investigate all of them.

The analytical response time computation : *The probabilistic representation of the system under analysis is* $\tau = \{\tau_1 = (\mathcal{C}_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathcal{T}_1 = \begin{pmatrix} 5 & 6 \\ 0.2 & 0.8 \end{pmatrix}), \tau_2 =$

$(\mathcal{C}_2 = \begin{pmatrix} 3 & 4 \\ 0.9 & 0.1 \end{pmatrix}, \mathcal{T}_2 = \begin{pmatrix} 7 \\ 1 \end{pmatrix})\}$ *and we are interested in finding the response time distribution* $\mathcal{R}_{2,1}$ *of* $\tau_{2,1}$ *by applying our analysis.*

The computation starts by initialising the response time distribution $\mathcal{R}_{2,1}$ *with the combined execution time requirements of higher priority tasks, in this case* $\mathcal{C}_1$*, and adding to it the execution requirement of the task under analysis :*
$\mathcal{R}_{2,1} = \mathcal{C}_1 \otimes \mathcal{C}_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 3 & 4 \\ 0.9 & 0.1 \end{pmatrix} = \begin{pmatrix} 5 & 6 \\ 0.9 & 0.1 \end{pmatrix}$.

The possible preemption that can occur from $\tau_{1,2}$ *can be either at* $t = 5$ *with probability* 0.2 *or at* $t = 6$ *with probability* 0.8*. For each of these two cases we make a copy of* $\mathcal{R}_{2,1}$ *and proceed in the following way :*
$\mathcal{R}^1_{2,1} = \left(\begin{pmatrix} 5 \\ 0.9 \end{pmatrix} \oplus \begin{pmatrix} 6 \\ 0.1 \end{pmatrix} \otimes \begin{pmatrix} 2 \\ 1 \end{pmatrix}\right) \otimes \begin{pmatrix} 0 \\ 0.2 \end{pmatrix} = \left(\begin{pmatrix} 5 \\ 0.9 \end{pmatrix} \oplus \begin{pmatrix} 8 \\ 0.1 \end{pmatrix}\right) \otimes \begin{pmatrix} 0 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 5 & 8 \\ 0.9 & 0.1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 5 & 8 \\ 0.18 & 0.02 \end{pmatrix}$ *represents the case that* $\tau_{1,2}$ *arrives at* $t = 5$ *preempting* $\tau_{2,1}$*. In this case,* $\tau_{1,2}$ *can only affect the tail of the distribution, i.e.* $\tau_{2,1}$ *did not finish execution by* $t = 6$*. Two units of time are added to the tail of the distribution, and the entire resulting distribution is updated with the probability* 0.2 *of* $\tau_{1,2}$ *arriving at* $t = 5$*.*

$\mathcal{R}^2_{2,1} = \begin{pmatrix} 5 & 6 \\ 0.9 & 0.1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0.8 \end{pmatrix}$ *represents the case that* $\tau_{1,2}$ *arrives at* $t = 6$ *and so it does not preempt* $\tau_{2,1}$*. The tail of the distribution is not affected, but the entire distribution is updated with the probability* 0.8 *of* $\tau_{1,2}$ *arriving at* $t = 6$*.*

Once the two copies of $\mathcal{R}_{2,1}$ *have been obtained they are coalesced and the final result is obtained :*
$\mathcal{R}_{2,1} = \mathcal{R}^1_{2,1} \oplus \mathcal{R}^2_{2,1} = \begin{pmatrix} 5 & 8 \\ 0.18 & 0.02 \end{pmatrix} \oplus \begin{pmatrix} 5 & 6 \\ 0.72 & 0.08 \end{pmatrix} = \begin{pmatrix} 5 & 6 & 8 \\ 0.9 & 0.08 & 0.02 \end{pmatrix}$.

The value 8 *of the response time distribution is not possible since* $\tau_{2,1}$ *will not be allowed to continue its execution past* $t = 7$*. Any value strictly greater that the jobs deadline is replaced by "DMP" and their summed probability mass represents the Deadline Miss Probability of the job :* $\mathcal{R}_{2,1} = \begin{pmatrix} 5 & 6 & DMP \\ 0.9 & 0.08 & 0.02 \end{pmatrix}$.

Since the earliest arrival of $\tau_{1,3}$ *is at* $t = 10$ *then this job cannot preempt* $\tau_{2,1}$ *and the analysis stops here. We have obtained the worst case response time distribution of* τ_2 *and its Deadline Miss Probability,* 0.02*, which is exactly the same as the one obtained earlier by enumerating all possible scenarios.*

Third generalization (probabilistic deadline) : *In order to show the effect of a probabilistic deadline, we further generalize the task system by considering that the arrival distribution and hence the deadline distribution of* τ_2 *is equal to* $\mathcal{T}_2 = \begin{pmatrix} 7 & 8 \\ 0.3 & 0.7 \end{pmatrix}$. *In this case* $\tau_{2,1}$ *will miss its deadline if and only if the following worst case scenario happens :* $\tau_{1,2}$ *arrives at* $t = 5$ *(probability* 0.2*),* $\tau_{2,1}$ *executes for* 4 *units of time (probability* 0.1*) and* $\tau_{2,2}$ *arrives at* $t = 7$ *(probability* 0.3*). The probability of this scenario happening is* $DMP_2 = 0.2 \times 0.1 \times 0.3 = 0.006$.

This probability can be obtained directly by applying Equation (5.7) :

$$\mathcal{B}_2 = \mathcal{R}_{2,1} \ominus \mathcal{D}_2 = \begin{pmatrix} 5 & 6 & 8 \\ 0.9 & 0.08 & 0.02 \end{pmatrix} \ominus \begin{pmatrix} 7 & 8 \\ 0.3 & 0.7 \end{pmatrix} = \begin{pmatrix} -3 & -2 & -1 & 0 & 1 \\ 0.63 & 0.83 & 0.24 & 0.014 & 0.006 \end{pmatrix}.$$

The value that are less or equal to zero are discarded, since they represent the cases when the job finishes execution before or at the deadline. The strictly positive values are kept and their added probabilities represent the tasks' DMP, in this case the probability of the value 1 *which is* 0.006 *as found by the descriptive method above.*

We introduce now a formal description of our analysis.

Formally : The arrival time of the j^{th} job of a task τ_n is computed for $j \geq 1$ as follows

$$\mathcal{T}_{n,j} = \mathcal{T}_n \otimes ... \otimes \mathcal{T}_n, \quad j-1 \; times \tag{5.2}$$

and for $j = 0$ we have $\mathcal{T}_{n,0} = 0$.

The worst case response time of task τ_n is initialized as :

$$\mathcal{R}_n^0 = \mathcal{B}_n \otimes \mathcal{C}_n \tag{5.3}$$

where the backlog at the arrival of τ_n is equal to

$$\mathcal{B}_n = \bigotimes_{i \in hp(n)} \mathcal{C}_i \tag{5.4}$$

After adding the execution time of the task under analysis to the backlog accumulated at its arrival, its response time is updated iteratively with the possible preemptions as follows :

$$\mathcal{R}_n^i = \bigoplus_{j=1}^{k} \mathcal{R}_n^{i,j} \tag{5.5}$$

where i is the current iteration, k is the number of values in the random variable representing the pMIT distribution of the preempting task, j is the current value taken into consideration from the pMIT distribution of the preempting task, and $\mathcal{R}_n^{i,j}$ is the j^{th} copy of the response time distribution and it integrates the possible preemption in the following way :

$$\mathcal{R}_n^{i,j} = (\mathcal{R}_n^{i-1,head} \oplus (\mathcal{R}_n^{i-1,tail} \otimes \mathcal{C}_m^{pr})) \otimes \mathcal{P}_{pr} \tag{5.6}$$

where :

- n is the index of the task under analysis ;
- i is the current step of the iteration ;
- j represents the index of the current value taken into consideration from the pMIT distribution of the preempting task ;
- $\mathcal{R}_n^{i-1,head}$ is the part of the distribution that is not affected by the current preemption under consideration ;
- $\mathcal{R}_n^{i-1,tail}$ is the part of the distribution that may be affected by the current preemption under consideration ;
- m is the index of the higher priority task that is currently taken into account as a preempting task ;
- $\mathcal{C}_m^{pr}$ is the execution time distribution of the currently preempting task ;
- $\mathcal{P}_{pr}$ is a fake random variable used to scale the j^{th} copy of the response time with the probability of the current value i from the pMIT distribution of the preempting task. This variable has one unique value equal to 0 and its associated probability is equal to the i^{th} probability in the pMIT distribution of the preempting job.

For each value $v_{m,i}^j$ in $\mathcal{T}_{(m,j)}$ for which there exists at least one value $v_{n,i}$ in $\mathcal{R}_n^{i-1}$ so that $v_n^i > v_{m,i}^j$, the distribution $\mathcal{R}_n^{i-1}$ is split in two parts :

- $\mathcal{R}_n^{i-1,head}$ which contains all values $v_{n,i}^-$ of $\mathcal{R}_n^{i-1}$ that are less or equal than $v_{m,i}^j$, i.e., $v_{n,i}^- \leq v_{m,i}^j$, and
- $\mathcal{R}_n^{i-1,tail}$ which contains all values $v_{n,i}^+$ of $\mathcal{R}_n^{i-1}$ that are greater than $v_{m,i}^j$, i.e., $v_{n,i}^+ > v_{m,i}^j$.

The iterations end when there are no more arrival values $v_{m,i}^j$ of any job i of any higher priority task τ_m that is smaller than any value of the response time distribution at the current step. A stopping condition may be explicitly placed in order to stop the analysis after a desired response time accuracy has been reached. For example, the analysis can be terminated once an accuracy of 10^{-9} has been reached for the response time. In our case, the analysis stops when new arrivals of the preempting tasks are beyond the deadline of the task under analysis, i.e., the type of analysis required for systems where jobs are aborted once they reach their deadline.

Once the jobs' response time distribution can be computed, the Deadline Miss Probability can be obtained by comparing the response time distribution with that of the deadline, as follows :

$$\mathcal{B}_i = \mathcal{R}_i \ominus \mathcal{D}_i = \mathcal{R}_i \oplus (-\mathcal{D}_i), \tag{5.7}$$

where the $\ominus$ operator indicates that the values of the distribution are negated. We use the notation $\mathcal{B}_i$ even though the resulting distribution is not a backlog distribution in the strict sense for the model we consider, but it is still the formula for computing backlog for systems where jobs are allowed to execute past their deadline.

The DMP of the job under analysis is given by the probability mass corresponding to the values strictly greater than 0, i.e. the job would need more units of time to finish its execution. The probability mass corresponding to the values less or equal to 0 gives the probability that the job finishes execution before its deadline and the next release.

5.2.1 Critical instant of a task with multiple probabilistic parameters

Lemma 2 *We consider a task system of n tasks with τ_i described by deterministic C_i and probabilistic $\mathcal{T}_i, \forall i \in \{1, 2, \cdots, n\}$. The set is ordered according to the priorities of the tasks and the system is scheduled preemptively on a single processor. The response time distribution $\mathcal{R}_{i,1}$ of the first job of task τ_i is greater than the response time distribution $\mathcal{R}_{i,j}$ of any j^{th} job of task τ_i, $\forall i \in \{1, 2, \cdots, n\}$.*

Proof. The response time distribution $\mathcal{R}_{i,j}$ of a job within a probabilistic system may be obtained by composing response time values $R^k_{i,j}$ of jobs within all corresponding deterministic systems obtained by considering all values of the minimal inter-arrival times and the probability associated with the respective scenario

k and we have $\begin{pmatrix} R^k_{i,j} \\ p_{scenario_k} \end{pmatrix}$. For each of these deterministic systems we know from [Liu and Layland, 1973] that the critical instant of a task occurs whenever the task is released simultaneously with its higher priority tasks. Thus we have that $R^k_{i,1} \geq R^k_{i,j}, \forall k, j > 1$ and we obtain $\mathcal{R}_{i,1} \succeq \mathcal{R}_{i,j}$ as the associated probabilities of $R^k_{i,1}$ and$R^k_{i,j}, \forall k$ are the same. $\square$

Theorem 10 *We consider a task system of n tasks with τ_i described by probabilistic $\mathcal{C}_i$ and $\mathcal{T}_i, \forall i \in \{1, 2, \cdots, n\}$. The set is ordered according to the priorities of the tasks and the system is scheduled preemptively on a single processor. The response time distribution $\mathcal{R}_{i,1}$ of the first job of task τ_i is greater than the response time distribution $\mathcal{R}_{i,j}$ of any j^{th} job of task τ_i, $\forall i \in \{1, 2, \cdots, n\}$.*

Proof. The response time distribution $\mathcal{R}_{i,j}$ of a job within a probabilistic system is obtained by convolving response time distributions $\mathcal{R}^l_{i,j}$ of jobs within all corresponding probabilistic systems obtained by considering tasks described by $C_i, \mathcal{T}_i, \forall i$. Then within each scenario l we have from Lemma 2 that $\mathcal{R}^l_{i,1} \succeq \mathcal{R}^l_{i,j}$. We have then $\mathcal{R}_{i,1} = \otimes_{l=1}^{nb_{of}scenarios} \mathcal{R}^l_{i,1} \succeq \otimes_{l=1}^{nb_{of}scenarios} \mathcal{R}^l_{i,j} = \mathcal{R}_{i,j}$. $\square$

5.3 Validation of the method

Our response time analysis provides safe, but pessimistic results with respect to an exact analysis that would provide the response time of any job of a task. We leave the reduction of pessimism as future work, for now we just note that a pessimistic result is safe, and since this is the first analysis framework of its kind, we do not strive for absolute accuracy, only for safeness. We validate our method by ensuring the three conditions presented in Section 5.3 : limit condition, insurance of the worst case scenario and convergence.

Limit condition : To ensure the limit condition we need to prove that our method provides the same or larger than the worst case response time of a corresponding deterministic task system obtained by considering only the worst case values for all parameters of the tasks.

By applying the analysis to a deterministic task system, the convolution specific to random variables becomes the summation of deterministic values, the coalescing operator $\oplus$ that combines two random variables is no longer necessary since there is no *head* and *tail* sections of random variables, the splitting into *head* and *tail* is replaced with the corresponding inequality verification, checking if the next arrival of a higher priority task will preempt the job under execution. The analysis stops when there are no higher priority jobs left that can preempt the job under analysis.

Insurance of the worst case scenario : In order to obtain the worst case response time of the given probabilistic task system, the existing deterministic analysis [Joseph and Pandya, 1986] is applied to the deterministic task system obtained by considering from each task its minimum inter-arrival time value and its maximum worst case execution time value. Besides the worst case response time, the probabilistic analysis framework that we propose also provides the best case response time of the task, and all possible response time values between the best case and the worst case, each with its probability of occurring.

Convergence : In order to ensure the convergence condition we need to prove that the analysis of a task system does not run indefinitely without, i.e., sooner or later returning a result. This condition follows from these two assumptions over the system model :

a) the distribution representing the tasks' parameters are of finite length and so the analysis will complete a loop of integrating new preemptions in the tail of the response time distribution in a finite time.

b) the system under consideration makes use of an *a*bort on deadline policy, which means that the analysis stops when the returned response time values are larger than the largest possible deadline of the task under analysis, i.e. the largest value in the pMIT distribution.

5.4 Implementation and evaluation of the method

We implemented our response time analysis in MATLAB. The pseudo-code for the associated steps is presented in Algorithms 2 and 3 and the complete scripts are available [2].

Before we proceed with the description of the simulations performed we recall here the concept of re-sampling [3].

Definition 18 *[Maxim et al., 2012b]* ***[Re-sampling]*** *Let $\mathcal{X}_i$ be a distribution with n*

2. The scripts are available at http ://www.loria.fr/˜maxim or uppon request to the authors
3. Note that in statistics, re-sampling has a different meaning from that used in real-time systems. For an example of re-sampling in real-time systems see Chapter 6 or refer to [Maxim et al., 2012b].

values describing a parameter of a task τ_i. The process of ***re-sampling to*** *k* ***values*** *or k*-***re-sampling*** *consists of reducing the initial distribution $\mathcal{X}_i$ from n values to a distribution $\mathcal{X}_i^*$ with k values.*

The re-sampling is safe with respect to the response time analysis as the response time $\mathcal{R}_i$ of any task τ_i of the initial system is greater than the response time $\mathcal{R}_i^*$ of the considered task within the re-sampled task system.

The re-sampling of a real-time distribution is performed in the following two sequential steps : *1*) selection of the k samples to be kept in the reduced distribution and *2*) redistribution of the probabilities from the values that are not kept.

Re-sampling for pWCET differs from re-sampling for pMIT in the second step, namely, as larger values of pWCET produce greater probabilistic response times as well as smaller values of pMIT produce greater probabilistic response times.

5.4.1 Experiment 1 : Complexity

One may be concerned when mentioning probabilistic analyses by their complexity as operations like convolutions of random variables are involved. Our analysis is tested first with respect to the complexity.

In Figure 5.2, a 3D plot of the analysis duration is presented. On the z-axis the analysis duration is given in seconds, on the x-axis is the variation of the number of values per random variable, from 2 to 16 values, and on the y-axis is the number of tasks per task system, also from 2 to 16 tasks. Every point on the surface corresponds to the average analysis duration of 100 task sets. The worst case utilization of each considered task is between 1.5 and 2 and the expected utilization is between 0.5 and 1. The pWCETs are decreasing distributions while the pMITs are increasing distributions.

We note that the analysis duration of a task set with 16 tasks, each of its random variables having 16 values, takes in average 140 seconds, i.e. the highest point on the z-axis.

The analysis duration increases both with respect to the number of tasks per task system and with respect to the number of values per random variable, indicating the exponential complexity of the analysis. Nevertheless, solutions exist to make such analysis affordable even for large task systems with parameters described by large random variables.

Algorithm 2 Worst case response time distribution computation

```
Input: $\Gamma$ a task set and $target$ the index of the task we analyze
Output: $\mathcal{R}_{target}$ the worst case response time distribution of $\tau_{target}$
  $\mathcal{R}_{target} = \mathcal{C}_{target}$ ; //initialize the response time with the execution time of the task under analysis
  for $(i = 1; i < target; i++)$ do
    $\mathcal{R}_{target} = \mathcal{R}_{target} \otimes \mathcal{C}_i$ ; //add the execution times of all higher priority tasks
  end for
  for $(i = 1; i < target; i++)$ do
    $\mathcal{A}_i = \mathcal{T}_i$ ; //initialize the arrivals of each higher priority task with their inter-arrival times distribution
  end for
  for $(i = 1; i < max(\mathcal{T}_{target}); i++)$ do
    for $(j = 1; j < target; j++)$ do
      if $max(\mathcal{R}_{target}) > min(\mathcal{A}_j)$ and $min(\mathcal{A}_j) = i$ then
        $\mathcal{R}_{target} = doPreemption(\mathcal{R}_{target}, \mathcal{A}_j, \mathcal{C}_j)$ ; //update the response time with the current possible preemption
        $\mathcal{A}_j = \mathcal{A}_j \otimes \mathcal{T}_j$ ; //the next arrival of $\tau_j$
      end if
    end for
  end for
  $\mathcal{R}_{target} = sort(\mathcal{R}_{target})$
Output: $\mathcal{R}_{target}$
```

A solution to decreasing the probabilistic analysis duration is re-sampling, which reduces the analysis duration while introducing minimal pessimism [Maxim et al., 2012b].

In Figure 5.3, the diagonal of the surface from Figure 5.2 is represented by a solid line, having an exponential behaviour. The same analysis is performed with re-sampling of the pWCET to 50 values and of the pMIT to 5 values both done after each iteration. The improvement is shown in the same figure, represented by the dotted line ; this time the average analysis duration over 100 task sets each having 16 task with 16 values per random variable is only 1.29 seconds, compared to 140 seconds when no re-sampling was performed. We note the important gain in speed when the analysis is performed with re-sampling, even for systems that have 32 tasks

Algorithm 3 doPreemption function

```
Input: $\mathcal{R}$ the current response time,
  $\mathcal{A}$ the arrival distribution of the preempting job and
  $\mathcal{C}$ the execution time distribution of the preempting job
Output: $\mathcal{R}$ the response time distribution updated with the current preemption
  $\mathcal{R}_{intermediary} = empty$ ;
  $\mathcal{A}_{fake} = empty$ ;
  for $(i = 1; i < length(\mathcal{A}); i++)$ do
    //constructing the fake random variable giving the probability of the preemption occurring
    $\mathcal{A}_{fake}.value = 0$ ; //the value of the fake random variable
    $\mathcal{A}_{fake}.probability = \mathcal{A}(\mathrm{i}).probability$ ; //the probability of the fake random variable
    Split $\mathcal{R}$ into $head$ and $tail$ according to the preemption value ;
    if $tail \mathrel{!}= empty$ then
      $tail = tail \otimes \mathcal{C}$ ;
    end if
    $\mathcal{R}_{intermediary} = head \oplus tail$ ;
    $\mathcal{R}_{intermediary} = \mathcal{R}_{intermediary} \otimes \mathcal{A}_{fake}$ ;
    $\mathcal{R} = \mathcal{R} + \mathcal{R}_{intermediary}$ ;
    $\mathcal{R} = sort(\mathcal{R})$
  end for
  $\mathcal{R} = \mathrm{sort}(\mathcal{R}_{intermediary})$
Output: $\mathcal{R}$
```

and each random variable has 32 values it takes 11 seconds to perform the analysis, indicating that it is affordable even for considerable larger systems. We show in the next experiment that distributions with 5 values for pMIT bring significant increase in precision with respect to the worst case response time analysis.

5.4.2 Experiment 2 : Improvement with respect to existing analysis

The second set of experiments that we performed show the precision that is gained by having tasks' parameters given as random variables. In order to do so we randomly generated probabilistic task systems to which we applied our analysis with different levels of re-sampling applied either at pWCET level or at pMIT level.

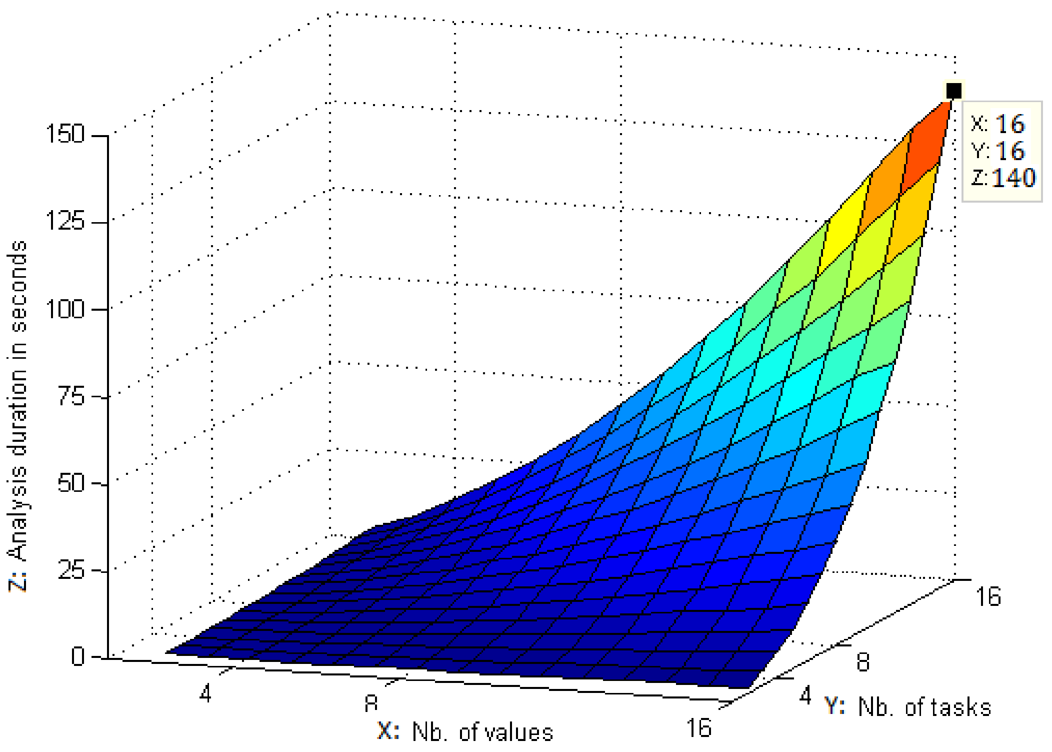


FIGURE 5.2 – Analysis duration of random task systems. On the X axis the number of values oer random varible is varied while in the Y axis the number of task per task system is varied. The Z axis gives the average response time analysis duration of a task system containing Y tasks and each task having X values in its pWCET and pMIT distribution (averaged over 100 task-systems). The analysis is exponential both with repect to the number of tasks per task-system and with respect to the number of values per random variable.

Precision gained by having a more detailed pMIT distribution

To show the increase in precision brought by a more detailed pMIT distribution, we repeated three times the analysis on the generated task system, each time varying the re-sampling level of the pMIT, using 10 values, 5 values and 1 value, respectively. A pMIT distribution with only one value is in fact a worst case MIT. The pWCET distribution was not re-sampled.

Figure 5.4 shows the task Deadline Miss Probability averaged over 100 task systems of 10 tasks. We note that the deterministic reasoning that considers one value for the MIT (and no re-sampling for pWCET) provides a DMP equal to 0.0167

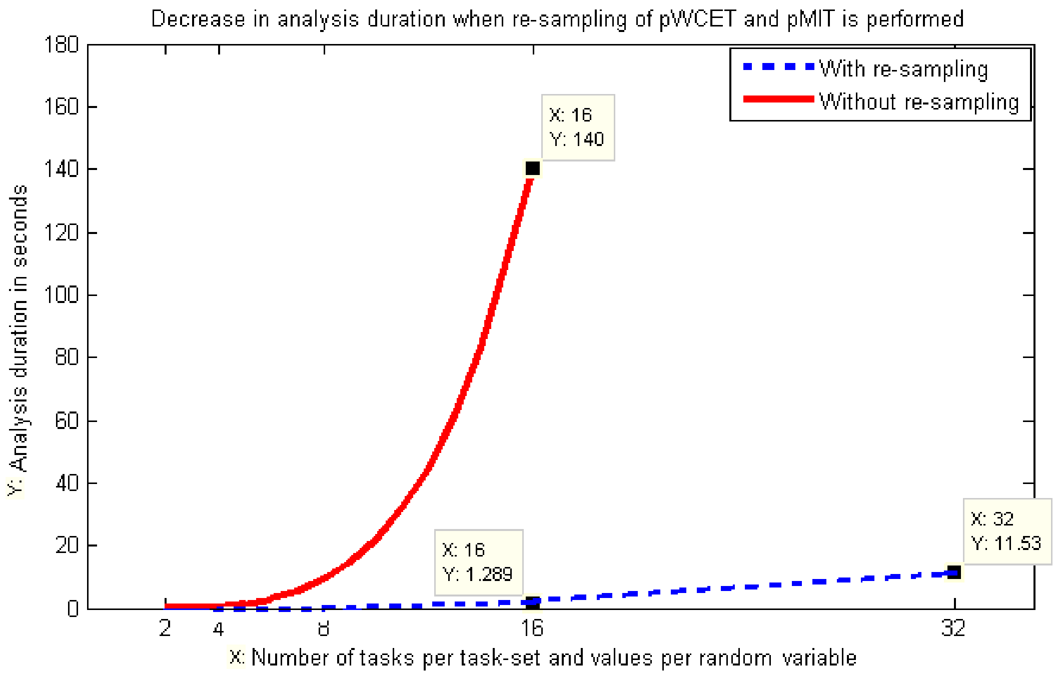


FIGURE 5.3 – Analysis duration of random task systems , increasing both the number of tasks per task system and the number of values per random variable in the same time.

(left bar in Figure 5.4) - this is the case of the existing analysis presented in [Díaz et al., 2002]. By considering a probabilistic reasoning with 10 values for the pMIT we decrease by a factor of 3 the DMP. Then further increasing the number of values within the pMIT only marginally decreases the DMP and this is shown by comparing the values of DMP for 1, 5 and 10.

Nevertheless, having a 10-value pMIT does not bring much increased precision over a 5-value pMIT, as can also be seen in Figure 5.4 where their respective DMPs are almost equal. We note that it is not necessary to have a large pMIT distribution to have a precise analysis, depending on the system under analysis just 5 values can be sufficient.

This increase in precision comes at a cost, namely an increase in the analysis duration : the larger the distribution, the more time it takes to perform the analysis. In Figure 5.5 we show the analysis duration of the three cases described above. The duration of the 10 values analysis is close to that of no-re-sampling, where the duration of 5 values analysis is decreased. In this case the 5 values analysis seems to

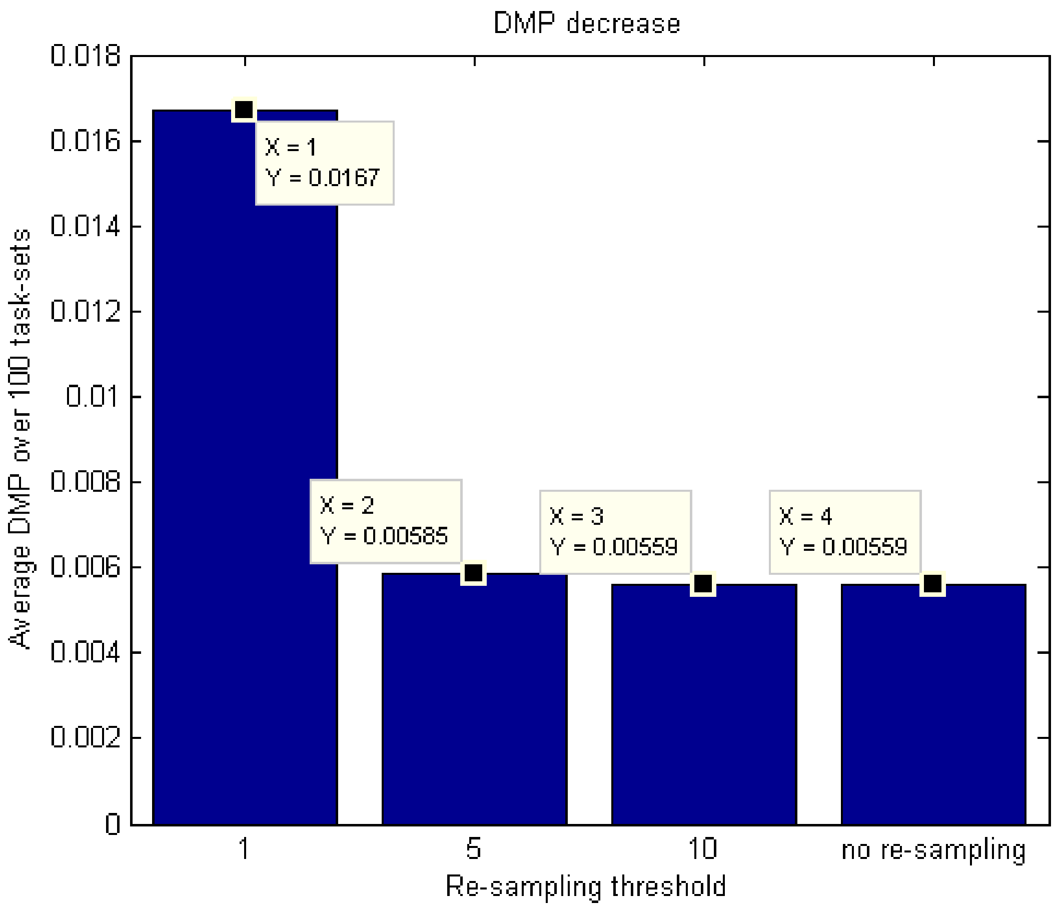


FIGURE 5.4 – The difference in DMP when the tasks' pMIT distribution has 1, 5 and respectively 10 values. Here 1 value indicates that only the worst case value of the pMIT distribution is considered.

be a comfortable compromise between the duration and the gained DMP.

Precision gained by having a more detailed pWCET distribution

We performed a set of experiments to show the difference in precision when the tasks' pWCET distribution has 1000 values, 100 values, 10 values or only 1 value. A pWCET distribution with only one value is a deterministic WCET. The analysis was performed on the randomly generated task systems on which there were applied, in turn, different levels of re-sampling to the pWCET distribution. The pMIT distribution was not re-sampled.

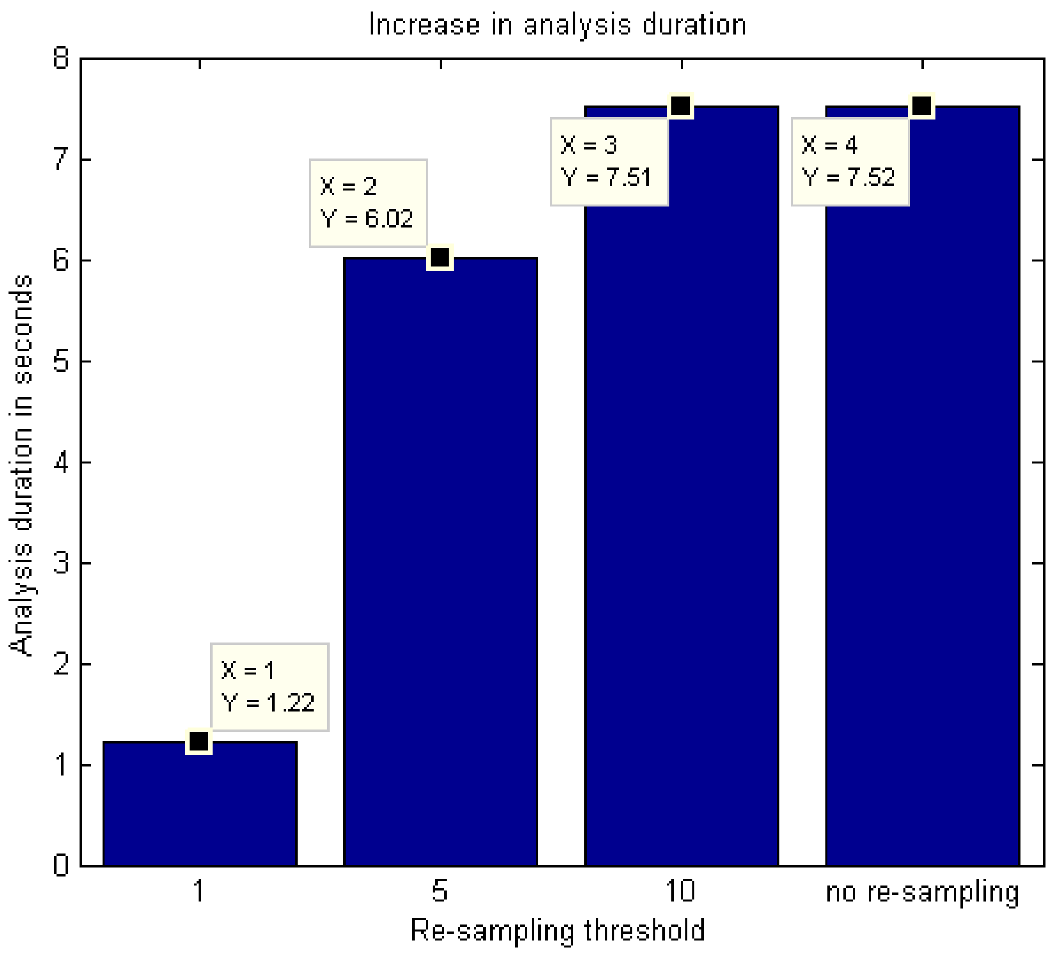


FIGURE 5.5 – The difference in seconds of the analysis duration when the tasks' pMIT distribution has 10, 5 and respectively 1 value, i.e. only the worst case value of the pMIT distribution.

In Figure 5.6 the difference in DMP between the four cases is depicted. Note that having 1 or 10 values in the pWCET distribution returns a task DMP equal to 1 which means that the system would be deemed unfeasible. This is not necessarily true, as can be seen from the bar representing the case when the pWCET distribution has 100 values. In this case the average DMP values of the analysed tasks does not surpass 0.02 which means that the systems could be feasible if they can afford a 0.02 Deadline Miss Probability for their lowest priority tasks. In conclusion we decreased the DMP fifty times by analysing the system with 100 values for the pWCET.

As in the case of re-sampling at pMIT level, the re-sampling at pWCET level

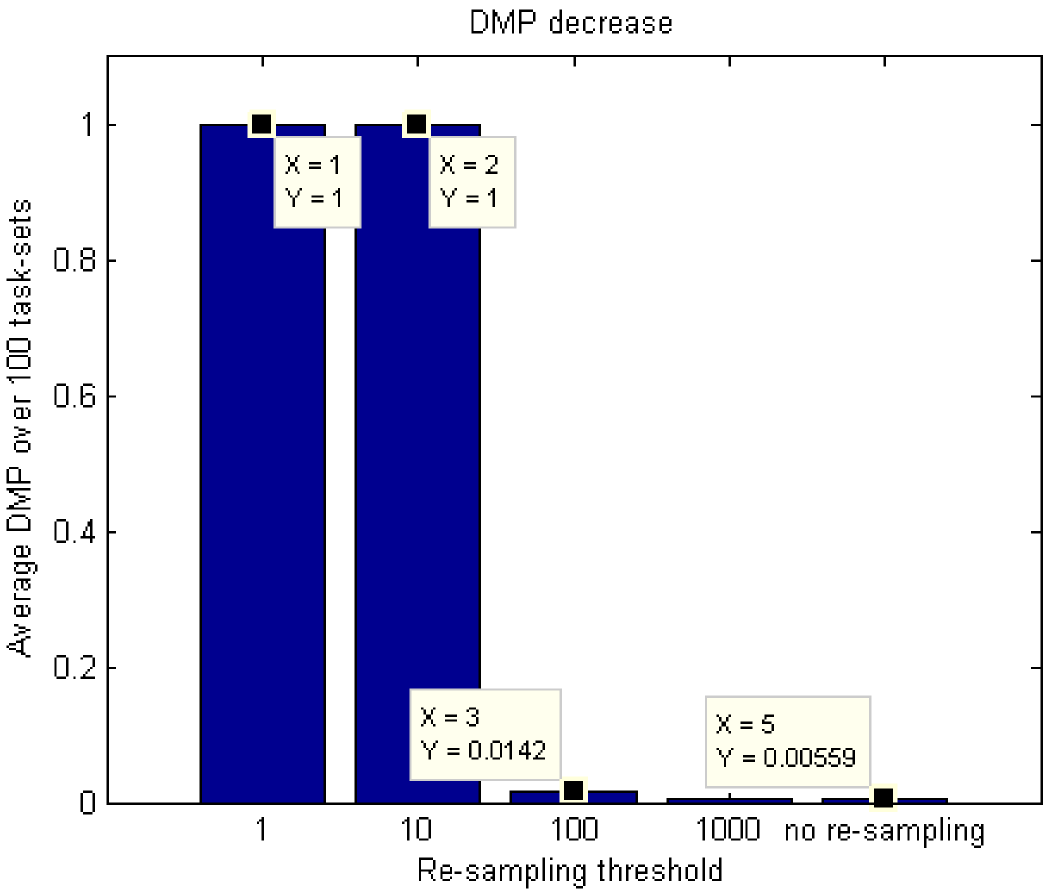


FIGURE 5.6 – The difference in DMP when the tasks' pWCET distribution has 1000, 100, 10 and respectively 1 value, i.e. only the worst case value of the pMIT distribution.

also comes with an increase in the analysis duration. Figure 5.7 depicts the analysis duration of the four cases described above, with the 1000 values for the pWCET having an exponential behavior but also being the distribution that has the most precision at DMP level. For pWCET re-sampling 100 is a compromise level that allows to obtain affordable duration and important increase in DMP.

In Chapter 6 a study is performed on different re-sampling strategies and novel re-sampling strategies are proposed that introduce very little pessimism. Also, by combining pWCET re-sampling and pMIT re-sampling, the analysis duration can be decreased considerably while retaining a high level of accuracy, regardless of the system under analysis.

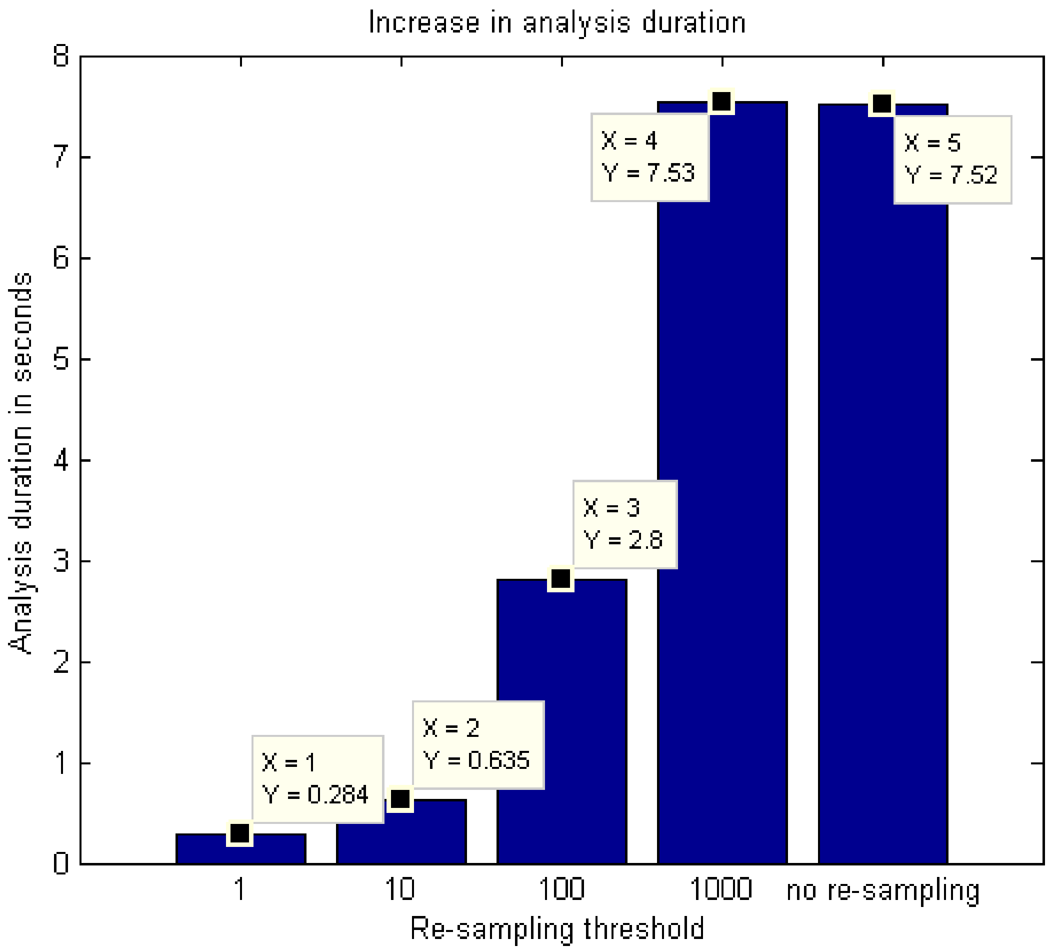


FIGURE 5.7 – The difference in seconds of the analysis duration when the tasks' pWCET distribution has 1000, 100, 10 and respectively 1 value, i.e. only the worst case value of the pMIT distribution.

5.5 Conclusions

In this chapter we presented a probabilistic worst-case response time analysis framework for synchronous tasks scheduled on one processor under a fixed-priority preemptive policy. This analysis is proved safe wince it is pessimistic. The complexity of experimentally studied and together with the possibility of using re-sampling techniques in order to make the analysis affordable even for large systems with detailed probability distributions. The improvement in schedulability terms with respect to existing (worst case) analysis has also been investigated, together with the effect of different levels of re-sampling, from the execution time point of view and from the inter-arrival time point of view, which can be applied separately or together. We leave as future work the study of more complex analysis techniques for the inter-arrival times distribution. Also, the research at hand allows opens the door for the search of optimal fixed-priority algorithms for system with probabilistic worst case execution times and probabilistic minimum inter-arrival times.

The result presented in this chapter is published in the 34^{th} IEEE Real Time System Symposium (RTSS2013) that took place in Vancouver, Canada, between 4 and 6 of December 2013 [Maxim and Cucu-Grosjean, 2013].

Chapitre 6

Re-sampling of pWCET distributions

6.1 Introduction

In this chapter we consider systems that have the execution time given as a random variable in order to make the analysis tractable by means of reducing the distributions and hence the amount of time it takes to perform convolutions to obtain the response time of a task. The analysis under consideration is that of computing response time distributions for jobs under a preemptive uni-processor fixed-priority scheduling policy as introduced in [Díaz et al., 2002, Lopez et al., 2008] and which we have also presented in Section 3.1 and is summarised as follows :

$$\mathcal{R}_{i,j} = \mathcal{W}_{i,j}(\lambda_{i,j}) \otimes \mathcal{C}_i \otimes \mathcal{I}_i, \tag{6.1}$$

where all the parameters are random variables and the release time $\lambda_{i,j}$ of the job $\tau_{i,j}$ is deterministic. Here $\mathcal{W}_{i,j}(\lambda_{i,j})$ is the backlog at time $\lambda_{i,j}$ obtained as the workload of higher priority jobs than $\tau_{i,j}$ that have not yet been executed immediately prior to $\lambda_{i,j}$. Equation (6.1) is solved iteratively in [Lopez et al., 2008]. $\mathcal{C}_i$ is the execution time of job $\tau_{i,j}$ and $\mathcal{I}_{i,j}$ is the interference in $\tau_{i,j}$ of all higher priority jobs than $\tau_{i,j}$, $hp(i)$, released at or after $\tau_{i,j}$, $\mathcal{I}_i = \sum_{\tau_{k,l} \in hp(i)} \mathcal{C}_k$.

The analysis takes as input the distributions $\mathcal{C}_i$ of all tasks and computes the response time (as a distribution of values and probabilities) by applying the convolution operator over PFs, here denoted as $\otimes$. The complexity of the response time computation is related to the number of possible values of $\mathcal{C}_i$. For instance, for two discrete random variables $\mathcal{C}_i$ and $\mathcal{C}_j$, during each iteration of the timing analysis we obtain, for any convolution, a new random variable that may have, in the worst-case, $k_i * k_j$ values, where k_i and k_j are the respective numbers of values of the two distributions. This implies that for any convolution $k_i * k_j$ operations are done and it makes the real-time analysis intractable except for simple scenarios, which is not always the case for real-time applications. Indeed, in complex applications the worst-case execution time distributions may have thousands of values [Cucu-Grosjean et al., 2012].

Example 6.1.1 *In this example we investigate the computational costs of the response time analysis provided by Equation (6.1).*

We used a task-set with 10 tasks, each task having a randomly generated PF characterizing its worst-case execution time. For this simple experiment, we assumed that the tasks were executed non-pre-emptively and that the periods were long in relation to the response time. Hence the problem simply involved the convolution of the PFs of the 10 tasks. The PFs contained just 100 *values spread over a range of approximately 10,000 possible values, intended to simulate execution times measured to an accuracy of 0.1 microseconds, with a largest value of around 1 millisecond (typical of embedded real-time systems).*

Figure 6.1 shows how the computation time required to compute the response time increases with each additional convolution. Even for this simple example the computational cost becomes prohibitive after about 10 convolutions.

The richness that the probabilistic real-time model offers to the analysis [Santinelli et al., 2007] is limited by the complexity of the analysis itself. The complexity problem has a possible solution via a reduction in the number k_i of possible values of $\mathcal{C}_i$ (samples) as long as the real-time constraints are met.

Hence, in this chapter we investigate the sampling problems in real-time scenarios, in particular the *simplification* of distributions of worst-case execution time and we give an answer to the question : *given a distribution with n values, how do we select k significant values out of n so that the analysis is still 'accurate'?*

From the original WCET distribution, we derive another distribution of worst-case execution times which offers the same real-time guarantees although with a reduced amount of information (simplified distribution). Furthermore, we build the

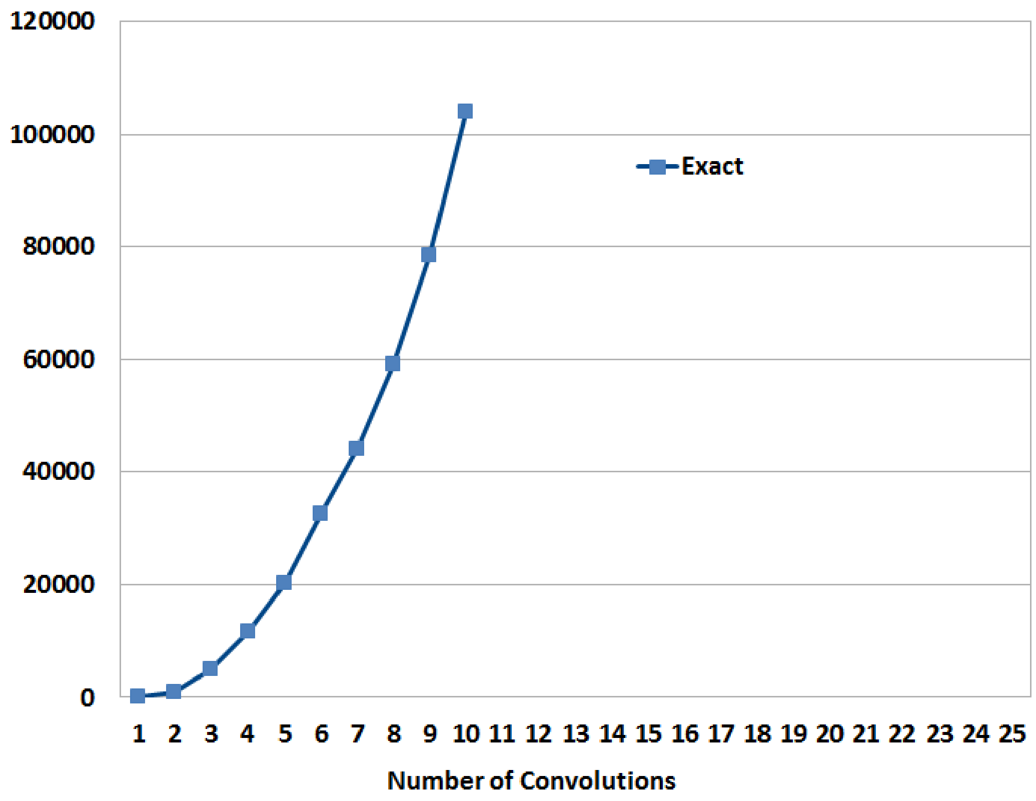


FIGURE 6.1 – Computation time relative to the number of convolutions

basis for an analysis of simplification mechanisms in order to provide a way of judging them in terms of their accuracy.

6.2 Re-sampling : Simplifying Distributions

Definition 6.2.1 (Re-sampling) *Let $\mathcal{C}_i$ be a distribution with n values representing the probabilistic execution times of a task τ_i. The process of **re-sampling to** k **values** or k**-re-sampling** consists of reducing the initial distribution $\mathcal{C}_i$ from n values to k values, while not being optimistic.*

In [Lopez et al., 2008], the authors identify the increased time complexity that a probabilistic analysis implies and mention the possibility of applying techniques to reduce the size of the distributions input in the analysis, without specifying any details of how such a technique might function, nor its effects on the real-time analysis. Later, in [Refaat and Hladik, 2010] the authors introduce the first formal re-sampling techniques which were meant to reduce the complexity of the real-time analysis by randomly eliminating samples from the original distribution until the resulting distribution has at most k samples, where k is given as a parameter of the re-sampling technique. This re-sampling technique was highly inefficient due to the fact that the discarded samples had their probability mass moved to the worst case value which meant that after each performed convolution the resulting distribution was less and less accurate going very close to the worst case result, loosing all the advantage provided by a probabilistic approach. Also, the fact that samples to be removed were randomly chosen made the technique even less effective.

These drawbacks were later fixed in [Maxim and Cucu-Grosjean, 2010] by providing a strategy for choosing the samples to be kept and also by redistributing the probability mass amongst the samples that are kept, not just the worst case value sample. The differences between these two re-sampling techniques can be seen in Example 6.2.2.

The re-sampling techniques we propose simplify the original distribution of WCETs but also take into account the pessimism that the operation introduces to the analysis.

In this section we improve previous results presenting three new re-sampling techniques and studying their effect on the worst-case execution time distribution. In the next section we study the effects of the re-sampling on the response time distribution.

A re-sampling technique has two sequential steps, namely :

- *Selecting the k samples to be kept in the reduced distribution.*
- *Re-distribution of the probabilities from the values that are not kept.*

Both steps differentiate and characterize a re-sampling method, although in our opinion it is the selection of the remaining values that differentiates them most. Therefore the selection step is the place where the re-sampled distribution may be improved.

a) Selecting the k Samples In this phase, the question to answer is *how do we select k samples out of n ?* For an input distribution of worst-case execution times, we know that the largest value has to remain in the re-sampled distribution in order to ensure the real-time analysis. Thus the worst-case value is guaranteed as well as the worst-case analysis. Deciding which are the $k-1$ samples to stay is the decision that impacts the most on the response time analysis. For instance in [Refaat and Hladik, 2010] the $k-1$ samples are chosen randomly, while in [Maxim and Cucu-Grosjean, 2010] they are chosen among those with the largest probabilities. We present in the next section three different selection techniques and study them in detail.

b) Probability re-distribution Once the k values to be kept have been selected, the probabilities of the un-drafted samples have to be re-distributed among these k values. This is imposed by the fact that the final result of re-sampling has to be a distribution. This means that the cumulative probability of the re-sampled distribution $\mathcal{C}'_i$ has to be equal to 1, $\sum_j \mathbb{P}\{\mathcal{C}'_i = C^j_i\} = 1$. Therefore we need to answer the question *How are the probabilities of the non selected samples re-distributed to the k selected samples ?* In order to have a pessimistic re-sampled distribution, and hence a safe analysis[4], there are no alternatives other than accumulating 'from-left-to-right'. This means that the probabilities of a subset of samples have to be added to the probability of a larger value. We note that in [Refaat and Hladik, 2010] the probabilities of deleted values are accumulated and added to the probability of the highest value. In [Maxim and Cucu-Grosjean, 2010] the remaining probabilities are added to the probabilities of the k selected values. The later approach increases the accuracy of the analysis and at the same time decreases the pessimism. Therefore we use in our work the probability re-distribution methods presented in [Maxim and Cucu-Grosjean, 2010].

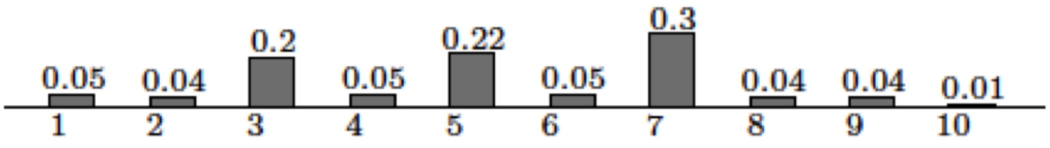
Another issue that a re-sampling mechanism has to take into account is how large k should be. It is intuitive that k plays a key role in the accuracy of re-sampling methods. A large value of k means less probability mass to re-distribute, so more accuracy in the resulting distribution. On the other hand, a large computation time to get results from the analysis. So, k comes from a trade-off of complexity versus accuracy we want to achieve. In our work we consider k as given.

Example 6.2.2 *To better understand the concept of re-sampling we present here an example. We consider a task with execution time given by a random variable with 10 values. Without loss of generality we consider these ten values as the integer numbers from* 1 *to* 10. *The probabilities of these values are as follows :*

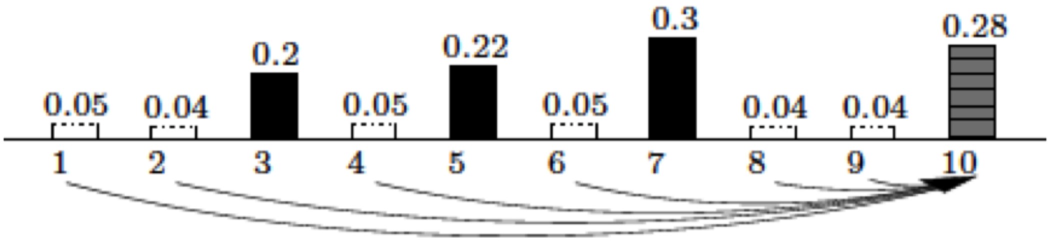
$$\mathcal{C} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0.05 & 0.04 & 0.2 & 0.05 & 0.22 & 0.05 & 0.3 & 0.04 & 0.04 & 0.01 \end{pmatrix},$$

and are represented in Figure 6.2a.

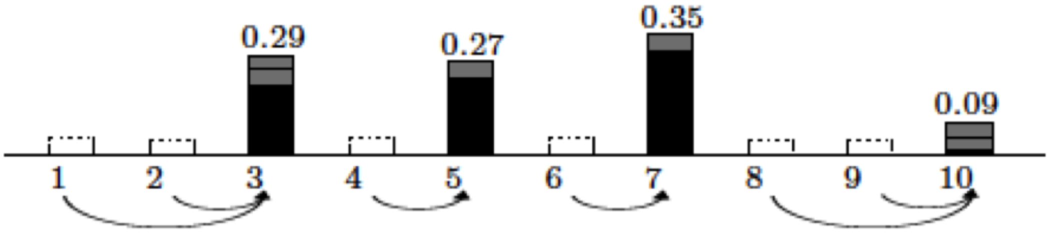
4. Safety and pessimism are clarified in sections 6.3 and 6.4 respectively



(a) Original distribution



(b) Re-sampled distribution obtained by moving the probability mass of the unkept samples to the worst case (largest) value as proposed [Refaat and Hladik, 2010]. The values kept from the original distribution are those with the most respective probability mass.



(c) Re-sampled distribution where the probability mass of the unkept samples is shared amongst the samples that are kept. In this way, the re-sampled distribution is closer to the original distribution, providing more accuracy to the analysis.

FIGURE 6.2 – The example distribution with two different re-samplings.

The objective is to reduce this random variable to only $k = 4$ *values instead of* 10 *in order to ease the convolution and hence the real-time analysis.*

According to [Maxim and Cucu-Grosjean, 2010], these 4 *resulting values will be the last value of the original distribution (*10*-th value) and the* 3 *values with the highest probabilities, i.e., the* 7*-th value which has a probability of* 0.3*, the* 5*-th value which has a probability of* 0.22 *and the* 3*-rd value which has a probability of* 0.2*.*

The samples according to the method in [Refaat and Hladik, 2010] should result from a random selection. In order to compare the results of these two methods, we keep the same values that we have indicated earlier. It is notable that the largest value of the distribution gets all the mass-probabilities of the values that were not *selected to be kept in the new random variable. Because of that, the largest value has a higher probability associated with it compared to the original distribution, the resulting distribution being equal to* $\mathcal{C}^{I} = \begin{pmatrix} 3 & 5 & 7 & 10 \\ 0.2 & 0.22 & 0.30 & 0.28 \end{pmatrix}$*, represented in Figure 6.2b.*

Applying the mechanism presented in [Maxim and Cucu-Grosjean, 2010], the re-sampled distribution becomes $\mathcal{C}^{II} = \begin{pmatrix} 3 & 5 & 7 & 10 \\ 0.29 & 0.27 & 0.35 & 0.09 \end{pmatrix}$*, Figure 6.2c. With this method the probability of the unused samples are distributed to the closest larger sample in the new distribution. For instance here the probabilities associated with the values* 1 *and* 2 *go to value* 3 *which results in a probability of* $0.29 = 0.2 + 0.05 + 0.04$ *in the new random variable. The last sample ends up with a probability of* 0.09*, much lower than the one obtained using the mechanisms presented in [Refaat and Hladik, 2010], but still larger than in the original distribution. Since that value represents the largest execution time, then assigning a lower probability to it makes the second re-sampling method less pessimistic in terms of response time and the resulting deadline miss ratio (see Section 6.6). Nonetheless, this is not sufficient to judge the quality of re-sampling methods in real-time scenarios.*

In the rest of this chapter we will propose new re-sampling techniques as well as a study of their impact on the real-time analysis.

6.3 Safeness of re-sampling

A safe real-time analysis has to guarantee the results in any possible conditions including those assured by the worst-case values in the deterministic scenario. Since the re-sampling has an important impact on the real-time analysis, it is mandatory that re-sampling does not introduce optimism since the safety of real-time analysis has to be guaranteed. Such a requirement affects the way of doing re-sampling.

In order to ensure the safeness of the real-time analysis, the re-sampling technique used in the process needs to be validated using the apropriate criterions from Section 5.3 :

- If all random variables have a single value, then the re-sampling has to keep that value so that the analysis could find the same result as existing deterministic analyses ; this is the *limit condition* for re-sampling algorithms as well as probabilistic real-time analysis.

- A first constraint when re-sampling a worst-case execution time distribution is that the largest value of the original distribution has to be included in the final distribution being critical for the safety of the analysis. This is the *worst-case condition* for both the analysis and the re-sampling (that the analysis requires and the re-sampling has to provide). Indeed, removing that value would provide optimistic results with respect to the exact analysis. The selection of the remaining values for the re-sampled distribution does not affect the safety because the selected samples are a subset of the original ones.
- The probability re-distribution among the new samples has an effect on the safety of the new distribution. In order to be safe, a distribution has to be pessimistic with respect to the original one which is considered exact. This is again the *worst-case condition.*

The input distribution of worst-case execution times (the original one) is the one that results in the precise analysis, so we consider it as the reference. We then say that a re-sampled distribution is safe, from the real-time analysis perspective if it is pessimistic with respect to the original distribution. In order to compare two distributions (original and re-sampled or two re-sampled distributions), we recall here the ordering relationship defined in [Lopez et al., 2008].

Definition 6.3.1 (Order amongst random variables [Lopez et al., 2008]) *Let $\mathcal{X}$ and $\mathcal{X}'$ be two random variables. $\mathcal{X}'$ is greater than or equal to $\mathcal{X}$ (or alternatively, $\mathcal{X}$ is less than or equal to $\mathcal{X}'$), and denote it by $\mathcal{X}' \succeq \mathcal{X}$ (alternatively, $\mathcal{X} \preceq \mathcal{X}'$) if $\mathbb{P}\{\mathcal{X}' \leq D\} \geq \mathbb{P}\{\mathcal{X} \leq D\}$ (alternatively $\mathbb{P}\{\mathcal{X}' \leq D\} \leq \mathbb{P}\{\mathcal{X} \leq D\}$) for any D, and the two random variables are not identically distributed.*

The same ordering holds with the Cumulative Distribution Function (CDF) of PFs, $F(x) = \mathbb{P}\{\mathcal{X} \leq x\}$.

Definition 6.3.2 *[Lopez et al., 2008] The random variable $\mathcal{X}'$ is* greater than or equal to *the random variable $\mathcal{X}$ ($\mathcal{X}' \succeq \mathcal{X}$), if $F_{\mathcal{X}'}(x) \leq F_{\mathcal{X}}(x)$, for any x, and the two random variables are not identically distributed.*

Corollary 6.3.3 (Safe re-sampling) *By moving probability mass from smaller execution times to larger execution times, we obtain a distribution that is greater than the original one. This new distribution is safe from the perspective of the execution time.*

Indeed, the method of accumulating the probabilities from discarded values up to the nearest larger values selected, so rightward with increasing ordered values, results in a distribution $\mathcal{X}^{re-sampled}$ 'greater than or equal to' the exact one $\mathcal{X}^{original}$. This ends up in a pessimistic distribution which is safe with respect to the original one.

From Corollary 6.3.3, the definition of ordering among distributions and the results in [Refaat and Hladik, 2010] we can conclude that any probability re-distribution 'from-left-to-right' is safe because the unselected probabilities are moved to larger execution time values. On the other hand, the re-distribution right-to-left produces optimistic results in terms of response time.

6.4 Quantifying the Pessimism introduced by re-sampling with respect to the original distribution

Once re-sampling has been performed on a distribution, we know that pessimism (inaccuracy) has been introduced due to the loss of information (with respect to the original distribution), but the question is *how much?* In order to have an answer to this question we need to define a metric that computes the pessimism introduced by the re-sampling, and maybe a metric or a method to compare two re-sampling techniques to see which one is better.

An effective approach is to multiply the probability and execution time values at each point together to obtain the overall 'weight' (linear combination) of the distribution and then compare the re-sampled distributions to the original. In the probability theory, this is the *expectation* $E(\mathcal{X})$ of a distribution $\mathcal{X}$, and is defined as $\sum_{x \,\in\, val(\mathcal{X})} x \cdot \mathbb{P}\{\mathcal{X} = x\}$, where $val(\mathcal{X})$ is the set of values of $\mathcal{X}$. The weight W_i of a distribution i with k_i values is given by

$$W_i = \sum_{j=1}^{k_i} v_j \cdot p_j \tag{6.2}$$

where v_j is the value C_i^j at the j-th position in the distribution and p_j is it's corresponding probability, $\mathbb{P}\{\mathcal{C}_i = C_i^j\}$.

We note that the closer the weight of a re-sampled distribution is to the weight of the original distribution, the better the re-sample is at representing the distribution as a whole.

6.5 Re-Sampling : Techniques

In this section we present three of the re-sampling techniques that we have implemented and we currently use in our analysis tool which has been developed at Rapita Systems [RapitaSystems, Ltd]. We compare these three methods in order to underline the impact that re-sampling has on the probabilistic response time analysis. The first one is a 'uniform spacing' re-sampling technique that is frequently used in practice and provides good results. The second one, which we introduce in this work, is a more complex technique that provides better results, i.e., introduces less pessimism, compared with other techniques that we have studied. The third technique is 'domain quantisation' which reduces the number of distinct values produced by convolution and so increases performance.

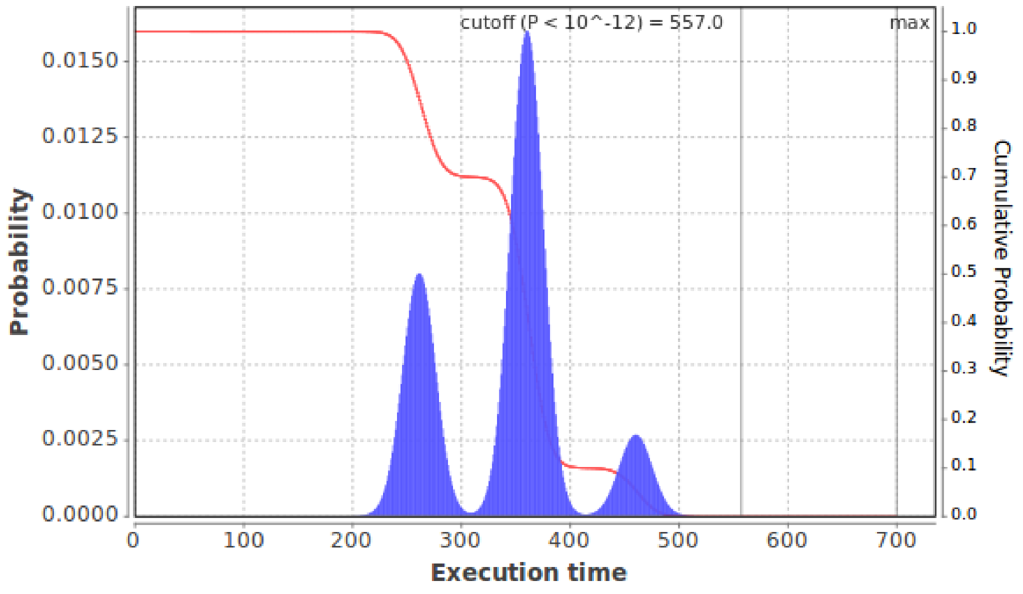


FIGURE 6.3 – Original distribution

6.5.1 Uniform Spacing Re-sampling

We present here a first re-sampling method that we use and we have analyzed in order to see its effects on the response time analysis of probabilistic real time tasks. The technique is called *uniform spacing* and it is frequently used in the probabilistic real-time domain as it is easy to implement.

The re-sampling is done by choosing equally distanced values out of the initial distribution. Algorithm 4 describes the steps of this re-sampling technique. Here $\mathcal{C}$ is the initial distribution, $\mathcal{C}^{new}$ is the distribution obtaining after the re-sampling process. We denote by $\mathcal{C}.size$ the number of values of $\mathcal{C}$, and by $\mathcal{C}_j$ the j-th value of $\mathcal{C}$. The probability associated with the j-th value of $\mathcal{C}$ is denoted by $\mathcal{C}_j.prob$.

For example, by applying the *uniform re-sampling technique* (with the objective of obtaining 50 values) to the distribution presented in Figure 6.3, that has 650 values, then the new distribution contains every 13-th value of the original. This value gathers the probability mass of the 12 values that precedes it. The resulting distribution can be seen in Figure 6.4. Although it keeps the general form of the original distribution, it gathers high peeks, the largest probability being close to 0.2 compared to 0.016 in the initial distribution.

Algorithm 4 Uniform Spacing Re-sampling Algorithm

```
Input: $\mathcal{C}$ a distribution and $k$ the number of values to be selected
Output: $\mathcal{C}^{new}$
  $m = 1$;
  $p = 0$;
  $q = ceil(\mathcal{C}.size/k)$
  for $i = 0; i \leq \mathcal{C}.size; i++$ do
    $p = p + \mathcal{C}_j.prob$;
    if $i \ mod \ q = 0$ or $i = \mathcal{C}.size$ then
      $\mathcal{C}^{new}_m.value = \mathcal{C}_j.value$
      $\mathcal{C}^{new}_m.prob = p$
      $p = 0$;
      $m = m + 1$;
    end if
  end for
```

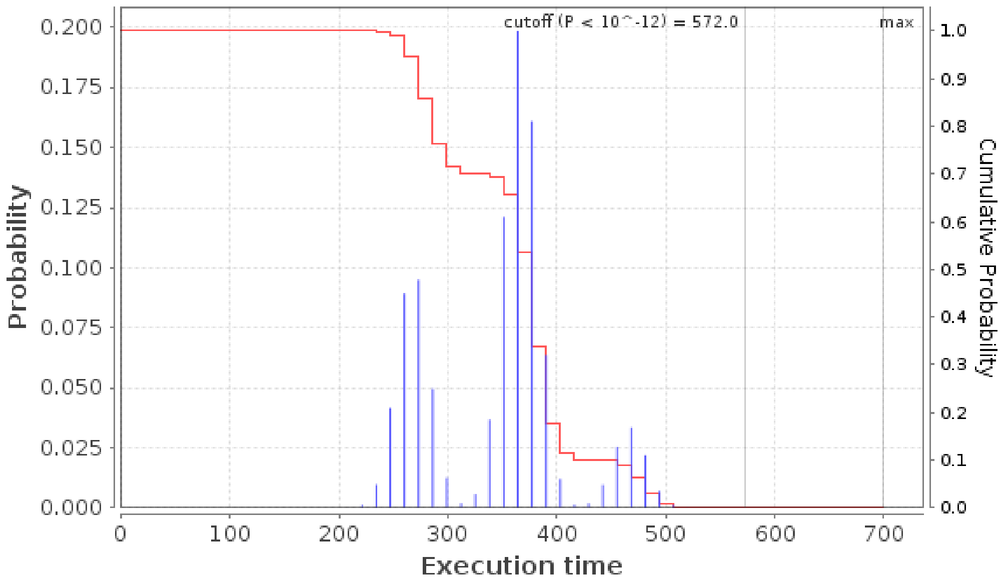


FIGURE 6.4 – Re-sampling obtained by Uniform Spacing

6.5.2 Domain Quantization

We now introduce an enhanced re-sampling technique, which we refer to as Domain Quantisation. The rationale for this technique is as follows. When convolving two distribution that have m and respectively n values each, the resulting distribution can have up to $m \times n$ values. This is true when the two distributions that are convolved are very different from one another. In the best case the resulting distribution can have as little as $m + n - 1$ values. For large distributions, having only $m + n - 1$ values instead of $m \times n$ values can make a big difference, reducing the total amount of re-sampling required since the number of values per distribution does not increase so fast.

For example, by convolving two distributions that have respectively the values (1 4 7) and (2 6 19) we obtain a distribution that has the values(3 6 7 9 10 13 16 19 22), i.e. by convolving two distributions that have three values each we obtain a distribution with nine values. If, on the other hand, the second distribution has the values (9 12 15), so the distance between values is the same as for the first input distribution, then we obtain a resulting distribution that has only five values, namely

(10 13 16 19 22). By virtue of having less values, the resulting distribution does not need to be re-sampled and so less pessimism is introduced.

This suggests that a way of decreasing the number of values in the resulting distribution is to quantize the values of the input distributions using a re-sampling strategy so that they have the same spacing between them. In this way, a first re-sampling is necessary in order to obtain the same quantization for the input distributions. Further re-sampling of the results with the same quanta will then be unnecessary.

Choosing the quanta to be used is an important problem, since it determines the number of samples to be kept per distribution, scaling a large distribution to a large quanta means that few values are kept out of its initial number of values, and so the loss in precision is potentially large ; on the other hand, scaling a large distribution to a small quanta results in too many values of the distribution to be kept, which makes the re-sampling inefficient.

This problem can be solved by taking advantage of the fact that the convolution is commutative, so, when there are multiple distributions to be convolved with each other, which is often the case in probabilistic response time analysis, first the small distributions (representing tasks with relatively short execution times) are convolved amongst themselves until they become bigger and they can be convolved with larger distributions. To facilitate this, we set the quanta for each distribution to the smallest power of 2 (e.g. 1,2,4,8...) that results in at most k samples. We note that this form of re-sampling has much in common with the Piece-wise Constant Approximation (PCA) method used in re-sampling time-series data [Keogh et al., 2001].

6.5.3 Reduced-Pessimism Re-sampling

A uniform selection of values for re-sampling will, most of the time, not result in a satisfactory reallocation of probability mass. Some of the selected values will accumulate a disproportionate share of the overall probability. This can lead to large amounts of pessimism in the re-sampled profile, and this pessimism is compounded by later applications of re-sampling which may occur during response time computation.

We have implemented an improved algorithm for the selection of re-sampling values which we call *Reduced-Pessimism Re-sampling.* It works by considering ranges of values and calculating the pessimism that would be introduced if the range of

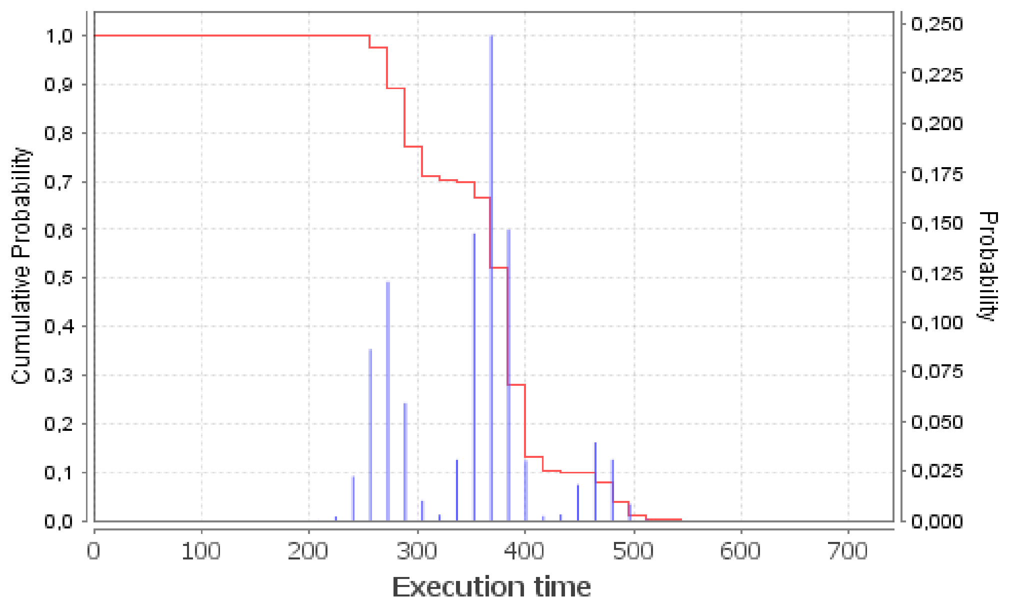


FIGURE 6.5 – Re-sampling obtained by Domain Quantization

values were to be aggregated into a single entry with the highest value in the range taking all of the probability mass.

Algorithm 5 describes the method we use for computing the pessimism, which is based on the relative probability 'weight' as described in Section 6.4 : the execution time multiplied by the probability.

Algorithm 6 provides the method by which re-sampling values are selected. Starting with the entire distribution, ranges are examined to identify the range with the highest pessimism. The selected range is then split into two sub-ranges. When sufficient sub-ranges have been generated, the upper bound of each range is used to perform re-sampling using the method described in Section 6.5.1.

6.5.4 Comparison

To compare the three methods at execution time level we make use of the metric presented in section 6.4. The original distribution depicted in Figure 6.3 has a weight of 340, while the distribution obtained using Uniform Spacing and Reduced-

Algorithm 5 Algorithm to compute the pessimism associated with replacing a range with a single value

```
Input: $\mathcal{C}$ and $(x, y)$ a range of values;
Output: $p$ pessimism;
  $shifted = (\sum_{i=x}^{y} \mathcal{C}_i.probability) \times \mathcal{C}_y.value$;
  $original = \sum_{i=x}^{y} (\mathcal{C}_i.probability \times \mathcal{C}_i.value)$;
  $p = shifted - original$;
```

Algorithm 6 Algorithm for selecting values which create the least pessimism when re-sampling

```
Input: $\mathcal{C}$ a distribution and $k$ the number of values to be selected
Output: $\mathcal{C}^{new}$
  $\mathcal{Q} = \varnothing$; // Priority queue
  $r = (1, \mathcal{C}.size)$; // Full range of $\mathcal{C}$
  $p = pessimism(\mathcal{C}, r)$;
  $\mathcal{Q}.add(r, p)$; // Add $r$ to $\mathcal{Q}$ with priority $p$
  while $\mathcal{Q}.size < n$ do
    $r = \mathcal{Q}.remove\_first$;
    $(a, b) = split(r)$; // Split into two equal sub-ranges
    $\mathcal{Q}.add(a, pessimism(\mathcal{C}, a))$;
    $\mathcal{Q}.add(b, pessimism(\mathcal{C}, b))$;
  end while
  $\mathcal{C}^{new} = resample(\mathcal{C}, \mathcal{Q}.upper\_bounds)$;
```

Pessimism Re-sampling have weights of 346 and 342 respectively. The distribution obtained using Domain Quantization has the highest weight, equal to 348.

At execution time level the Reduced-Pessimism Re-sampling technique performs best, having a weight that is closer to that of the original distribution.

We will further analyze the three re-sampling methods in the next section, where they are applied in the response time analysis and compared via the pessimism introduced in the exceedence functions, i.e. at the level of deadline miss probabilities.

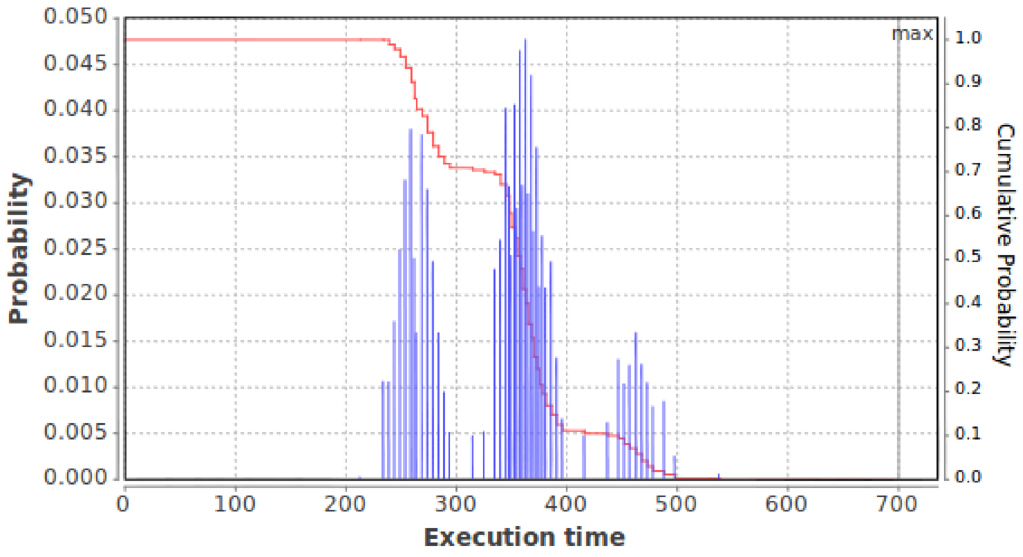


FIGURE 6.6 – Re-sampling obtained with Reduced-Pessimism method

6.6 Impact of Re-Sampling on the Response Time Analysis

In probabilistic analysis, where the response times are random variables, the real-time constraints are expressed in terms of probabilities of deadline misses. The response time $\mathcal{R}$ is one of the metrics applied to evaluate the probabilistic real-time analysis. Based on that, Diaz et al. in [Lopez et al., 2008] have defined relationships in order to tackle the pessimism in probabilistic analyses.

Definition 6.6.1 (Exact probabilistic analysis) *An exact random variable defines the exact results of a probabilistic analysis.*

A probabilistic response time $\mathcal{R}'$ is pessimistic when it is greater than or equal to the exact one $\mathcal{R}$, $\mathcal{R}' \succeq \mathcal{R}$. The challenge when using re-sampling techniques is how to perform approximations with probabilistic analysis parameters, while guaranteeing that the resultant response time is pessimistic or identically distributed, thus making the approximated analysis safe.

The pessimism introduced when re-sampling execution times is propagated through the analysis to the response time distribution. A good re-sampling technique should not introduce so much pessimism that the system is deemed unfeasible when in reality it is feasible, i.e., a more precise analysis done on the system without any re-sampling would deem the system feasible. In other words, the response time distribution obtained when re-sampling is performed should be as close as possible to that obtained when re-sampling is not performed.

A way of computing the relative pessimism introduced at the response time level is to use the weight presented in the previous section, Equation (6.2). A simpler way is to compare the exceedence functions and deadline miss probability of the jobs [Maxim et al., 2011] for which we have computed the response time.

Definition 6.6.2 (Job deadline miss probability) *For a job $\tau_{i,j}$ and a priority assignment Φ, the deadline miss probability $DMP_{i,j}$ is the probability that $\tau_{i,j}$ misses its deadline :*

$$DMP_{i,j}(\Phi) = \mathbb{P}\{\mathcal{R}_{i,j}(\Phi) > D_i\}. \tag{6.3}$$

When re-sampling is done, the pessimism introduced translates in an increased DMP. A decreased DMP in the re-sampled results with respect to the original results means that optimism has been introduced and the result is no longer safe.

6.7 Experimental Investigation

In this section, we provide details of three experiments used to investigate the performance and precision of the three re-sampling techniques described in section 6.5.

As the basis of these experiments, we used randomly generated sets of tasks with a PF characterizing the worst-case execution time of each task. We assumed that the tasks were executed non-pre-emptively and that their periods were long in relation to the response time. Hence the problem simply involved the convolution of the PFs of the tasks. (Thus each convolution effectively represents the addition of a further task ; or equivalently, the computation of the response time of a further lower priority task). This assumption does not change the nature of the problem, it simply means that less convolutions need to be performed to compute the response time of a job. We have opted for a simplified problem since the task-sets that we analyze are randomly generated and taking into account preemptions would have meant a more strict control on the generated task-sets, in order to avoid cases when a job suffers many preemptions, making the problem highly intractable, or even that the mentioned job never finishes execution. Nevertheless, the effects of the re-sampling strategies proposed and the results presented are valid for any number of convolutions and tasks in a task-set, regardless on whether preemptions are present or not.

The PFs contained 100 values spread over a range of approximately 10,000 possible values, intended to simulate execution times measured to an accuracy of 0.1 microseconds, with a largest value of around 1 millisecond, typical of embedded real-time systems.

All the data was obtained from a prototype java implementation running on an i7-2720QM CPU @ 2.20GHz computer.

In the first experiment, we examined the computational cost of the three re-sampling techniques, compared to an exact approach. For this experiment, 10 task-sets were randomly generated with 25 tasks each. When re-sampling was enabled, once the PF of each task was convolved, the resulting distribution was re-sampled to a maximum of 1000 values (the re-sampling threshold). Figure 6.7 plots the total computation time required in milliseconds (averaged over 10 task-sets) versus the number of convolutions from 1-25. Note the log scale on the graph. From Figure 6.7 it is clear that the exact approach initially exhibits exponential growth, with the computation time required approaching 100,000ms for 10 convolutions. All of the re-sampling techniques exhibit significantly improved performance with respect to the exact analysis ; with an improvement of around a factor of 100 for 10 convolutions for the Domain Quantisation technique. We observe that the more complex reduced pessimism re-sampling takes significantly longer than the uniform spacing and domain quantisation techniques.

In the second experiment, we examined the precision of the re-sampling techniques by comparing the exceedence functions $(1 - CDF)$ of the resulting distribution after 25 convolutions, i.e., the response time of the 25th highest priority task. This experiment used a similar configuration to the first experiment, i.e., a randomly generated task-set with 25 tasks, each task having an execution time given by a random variable with 100 values, but a lower re-sampling threshold of 100 to highlight the differences between the re-sampling techniques. Figure 6.8 shows the exceedence functions for the resultant distribution. We observe that the reduced pessimism technique results in a distribution that remains very close to the exact distribution at

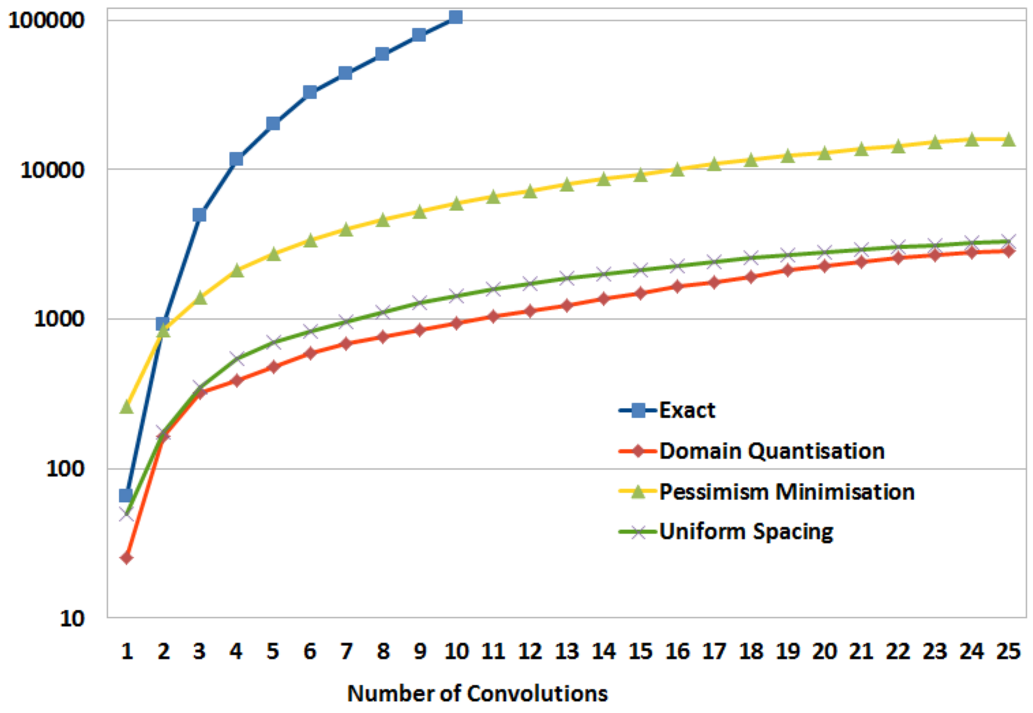


FIGURE 6.7 – Computation time in the exact case and with different re-samplings

high probabilities, even when the re-sampling threshold is just 100 values. This is because the technique is designed to minimise the increase in the weight of the distribution. However, Figure 6.9 shows the same exceedence functions on a log scale, thus illustrating the tails of the distributions. This shows that by selecting values to minimise the overall pessimism in the distribution, the shape of the tail is heavily compromised. (The reduced pessimism technique does not provide enough values in this region where the probabilities are small). With the Uniform Spacing and Domain Quantisation techniques although correspondence to the exact distribution is worse at high probabilities, it is much better in the tail of the distribution. This is reflected by the fact that the response time with a 10^{-9} probability of being exceeded is 89,600 for the exact distribution, 94,300 for the Domain Quantisation technique, and 141,500 for Reduced Pessimism re-sampling.

In the third experiment, we examined the trade-off between performance (computation time) and pessimism introduced in terms of the weight of the resulting distributions as a function of varying the re-sampling threshold. The experimental configuration was similar to that of the second experiment, but averaged over 10 task-

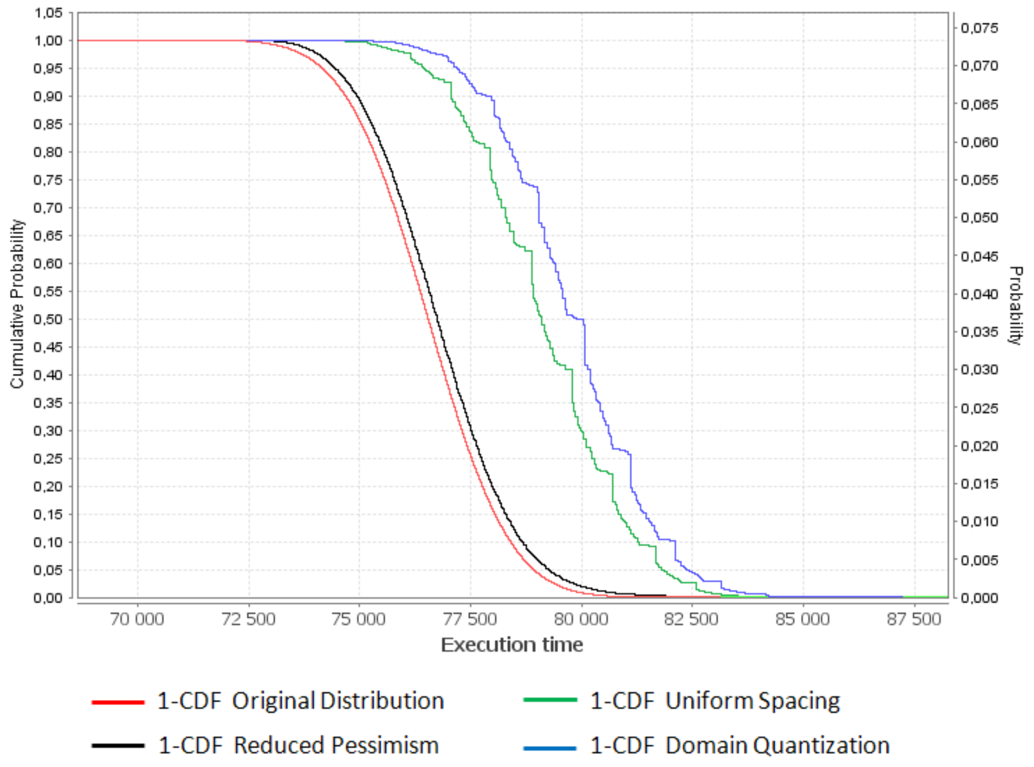


FIGURE 6.8 – $1 - CDF$ of re-sampled distributions and exact one

sets, i.e. 10 runs, 25 tasks per task-set, 100 values per random variable. Figure 6.10 gives corresponding results showing how the weight of the resultant distributions changes in relation to the exact analysis. The values plotted in Figure 6.10 are the ratio of the weight of the distribution produced via the re-sampling technique to the weight of the distribution from the exact analysis. Here the Reduced Pessimism technique shows a significant advantage albeit at the expense of precision in the tail of the distribution.

6.8 Future Improvements

While re-sampling is a good way of reducing the computation time of convolutions, it is not the only one. There are further improvements that can be made at the implementation level so that the time it takes to perform convolutions and response time analysis is reduced even more.

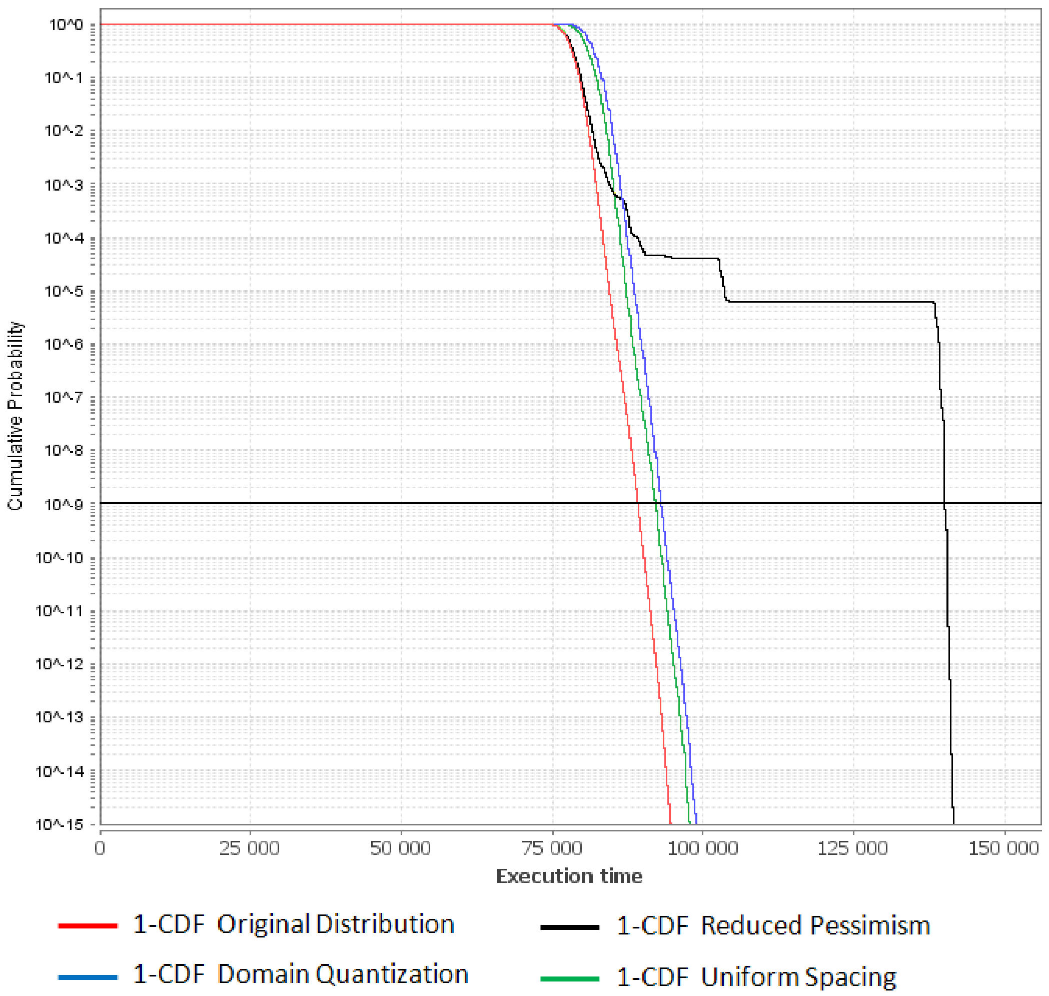


FIGURE 6.9 – Zoom-in in logarithmic scale on $1 - CDF$ of re-sampled distributions and exact one

6.8.1 Range quantization

One reason why performing convolutions is costly, besides the fact that there are many operations (multiplications, additions) to be performed, is how these operations are performed and, implicitly, the way data is stored and handled by the computer.

In order to represent and to perform operations on very small numbers, of the order of 10^{-k} where k can be tens or even hundreds, one needs to use arbitrary precision arithmetic, which is very expensive.

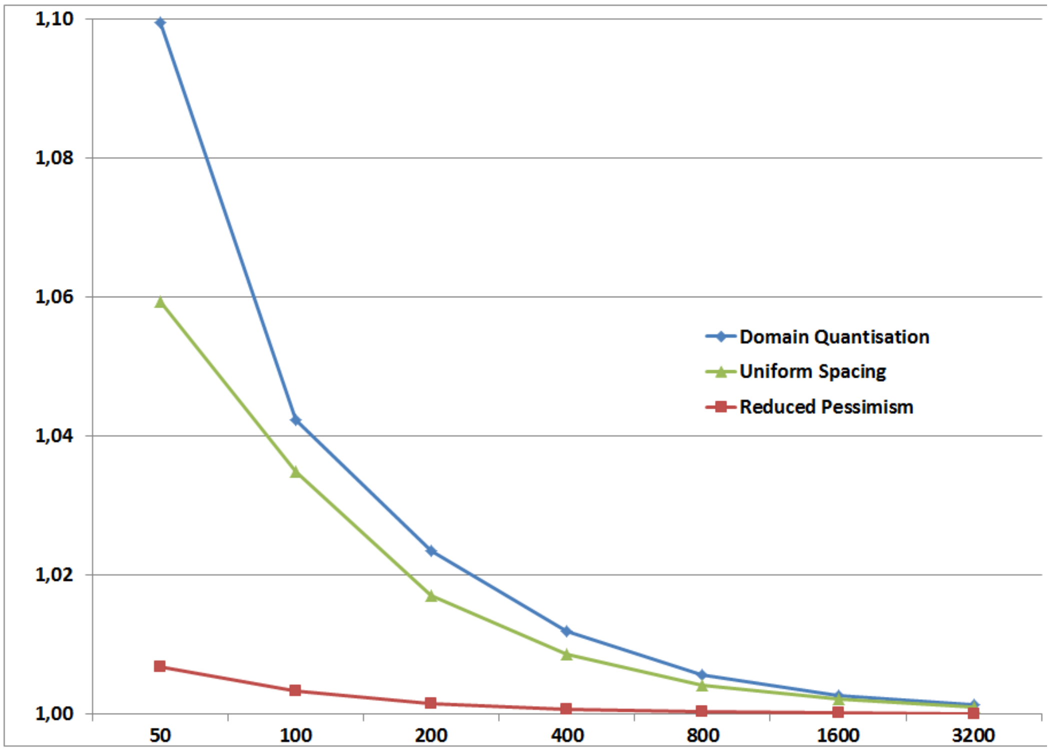


FIGURE 6.10 – Relative weight of re-sampled distributions with different re-sampling thresholds after 10 convolutions

One solution is to change the representation to something faster and simpler, like floats in the manner of one or few significant figures in the mantissa and a variable exponent. For example, the number $x = 0.0000000000000007$ can be written as 7×10^{-16}, so, one only needs to store the mantissa, which in this case is 7, and the exponent, which is 16.

Using this form of representation, performing multiplication is as simple as multiplying the two mantissas and adding the two exponents.

The addition of two such numbers can be more complicated if they are of different sizes, for example adding 7×10^{-16} and 2×10^{-4} results in $2.00000000007 \times 10^{-4}$ which takes us back to the problem of requiring a large amount of precision. This can be solved by truncating the probability value to the largest exponent, keeping a few values in the mantissa, and moving the probability mass that is cut off to the smallest larger value of the probability distribution, computing it and potentially decreasing its exponent. The truncation is done to all values of the distribution, starting from the smallest one to the largest one.

6.9 Conclusions

In this chapter we addressed the problem of re-sampling complex distributions of worst-case execution times, with a large number of values. We compared three re-sampling techniques, uniform spacing which is commonly used, and two techniques which we derived ; reduced pessimism re-sampling and domain quantisation.

We showed via a series of experiments that all three techniques are effective in addressing the complexity problem of exact analysis, which exhibits an exponential increase in computation times with the number of convolutions.

Our experiments showed that reduced pessimism re-sampling is highly effective in terms of obtaining a close, but safe approximation to the distribution resulting from exact analysis at high levels of probability, even when the re-sampling threshold is set as low as 100 values. However, selecting points in this way can compromise the shape of the tail of the distribution leading to significant and undesirable pessimism in that region, which is an issue for probabilistic real-time analysis. The domain quantisation and uniform spacing techniques did not suffer from this problem to the same extent.

In the future we intend to investigate the performance of re-sampling techniques aimed at preserving the shape of the tail of the distribution after a variable number of convolutions. We also intend to investigate the use of the complementary technique referred to as range quantisation (described in section 6.8). This technique is orthogonal to the re-sampling techniques described in this chapter and so can be applied in conjunction with any of them. Range quantisation removes the requirement to use arbitrary precision arithmetic and so can significantly increase the speed of the basic multiplication and addition operations used in convolution. Initial experiments in this area are promising, with the pessimism introduced by range quantisation found to be insignificant.

The results presented in this chapter were published in the International Conference on Real Time and Network Systems (RTNS2012) that took place in Pont-à-Mousson in November 2012 [Maxim et al., 2012b].

General conclusions and perspective

The main objective of this work was to propose new analysis techniques for probabilistic real-time systems. This objective was a difficult one to achieve keeping in mind the fact that the research branch of probabilistic analyses for real-time system is a young one with many pieces still missing from the puzzle. The advances in the analysis of probabilistic real-time systems mainly consist of generalizations of deterministic real-time systems where only a single task parameter is considered to be probabilistic. Little progress has been made in further generalizing the system model, to allow multiple task parameters to be probabilistic. Furthermore, in this later case, when the models allow multiple probabilistic parameters, then the considered type of distributions are usually too restrictive to be applicable to realistic systems. This is the main motivation of our work on systems with multiple probabilistic parameters as an important step forward, generalizing not just the system model, but also being able to accept any type of probability distribution as input. We have succeeded in providing such an analysis, and we consider it to be the most important contribution of this work. We have shown that taking into account multiple probabilistic parameters for real-time tasks can greatly decrease the pessimism introduced in the analysis with respect to the worst case analysis as well as with respect to the probabilistic analysis that uses only one probabilistic parameter. In this way, the chances that a real-time system is unjustly deemed infeasible are greatly decreased at design time, allowing for a better utilisation to be achieved by the system. As future work, we would like to further expand the analysis in order to take into account more probabilistic task parameters (where appropriate), such as offsets, jitters, etc. The mentioned technique also opens the door for numerous other advances in the state of the art, most notably the introduction of optimal scheduling algorithms for probabilistic real-time systems with multiple probabilistic parameters.

The proposition of such optimal algorithms is another direction on which our research has evolved. In this sense, we have identified three important optimization problems concerning probabilistic real-time systems and we have proposed the first optimal scheduling algorithms to solve these problems. Most notable, we have observed that for certain system configurations, a task response time distribution is not influenced by the relative priority ordering amongst tasks in the higher priority set (nor by those in the lower priority set). This is an important observation since it means that for these types of systems, we may apply algorithms like Audsley's to obtain an optimal scheduling algorithms, that is, a scheduling policy leading to minimized task failure rates. So far, these optimization problems are solved just for tasks with a single parameter given as a random variable, namely the worst case execution time and a generalization is in progress to check the aforementioned property and solve the three problems for systems containing tasks with multiple probabilistic parameters. Our preliminary results show that, like in the case of a single probabilistic parameters, in the case of multiple probabilistic parameters the relative ordering of tasks in the set of higher priority does not influence the response time distribution of a task at a given priority and so the solutions proposed for the simpler case of a single probabilistic parameter can be generalized to solve the problems for multiple probabilistic parameters. Nevertheless, proving the validity of the aforementioned property is a much more difficult endeavor in the case of multiple probabilistic parameters than in the case of a single probabilistic parameter given the fact that the study interval is not the same in the two cases and hyper-period can not be defined when inter-arrival times are given as distributions.

Our work on re-sampling techniques for probabilistic analysis for real-time systems is also an important addition to the state of the art, making possible for the existing analysis techniques to be applied to realistic cases, whereas until now they were intractable and could only be applied to toy examples. As we have shown, the complexity of a probabilistic analysis is highly exponential both in the number of tasks per task-set and in the size of the distributions representing task' parameters. The fact that convolutions are used in the analysis means that, even when starting with very simple toy examples, the analysis duration can increase drastically. Re-sampling resulting distributions throughout different steps in the analysis proved to be an efficient way of keeping under control the complexity of the analysis while sacrificing very little accuracy in the obtained results. As future work we plan on improving the proposed re-sampling techniques and also extending them for inter-arrival time distributions with a thorough analysis of their effect on the response time analysis, much like the case of re-sampling for execution times distributions.

Amongst other possible directions in which our work could be extended, we mention the analysis of probabilistic systems that exhibit statistical correlations and dependencies amongst the random variables representing different task parameters of functional dependencies amongst different tasks. In this sense, some results exist for the case when there are dependencies between distributions. For such problems, convolutions are not suited to use and instead they are replaced by copulas, but the research in this area is still very young. For example there are no complexity reduction methods and a study of the applicability of re-sampling techniques and their effects is an interesting problem to pursue. The problem of systems that exhibit functional dependencies amongst tasks is also an interesting one, but also necessary to be solved since realistic systems usually have some dependencies amongst tasks. Such a system is usually represented as a directed acyclic graph (DAG) and a feasible scheduling needs to take into account the precedence constraints that exist between tasks. A probabilistic layer could be introduced at task level to represent task parameters as distributions and obtain failure probabilities, but also at task interaction level, to probabilistically decide which task will execute next in the case when there is not a strict precedence constraint from one task to another.

Another problem on which our work can have a good applicability is that of scheduling mixed-criticality tasks. So far, most solutions to this problem only address the simple case of two criticality levels. The case of multiple criticality levels is a very difficult problem to solve, but a probabilistic solutions could turn out to be a fruitful one. For example, setting tasks maximum allowed failure probabilities as a function of their criticality one can ensure that, the higher a tasks criticality is, the higher its chances of success, without unnecessarily sacrificing lower criticality tasks. Another way in which a probabilistic approach can be used on mixed criticality systems is that of quantifying the probability that the system goes from a lower criticality state to a higher criticality state, and offering guarantees that this probability will not exceed a given threshold or if it does exceed the threshold enduring that higher criticality tasks will still finish execution within their respective deadlines.

We have listed here only some of the many directions in which our work could evolve. We plan on continuing on this path and further improve the probabilistic real-time analysis and scheduling, with the conviction that the present work is an important step forward, not just for us, but for the entire community that is interested in further exploring the potential of probabilistic approaches for real-time systems.

Appendix

1 Non-optimality of Rate Monotonic priority ordering for probabilistic real-time systems

Rate Monotonic priority assignment [Liu and Layland, 1973] is optimal for the deterministic problem of scheduling implicit deadline tasksets according to a fixed-priority policy. We therefore analyzed the applicability and optimality of RM to BPAP. Below, we present a counter example, a case where there exists a feasible priority assignment for BPAP, but RM priority assignment is not feasible.

		$\tau_{1,1}$		$\tau_{1,2}$			
		2	3	2		3	
		0.5	0.5	0.5		0.5	
	2	4	5	7		8	
	0.5	0.25	0.25	0.125		0.125	
$\tau_{2,1}$	3	5	6	7	8	8	D_2^+
	0.5	0.25	0.25	0.125	0.125	0.125	0.125

TABLE 1 – The response time of τ_2 when τ_2 is the lowest priority task

Let $\Gamma = \{\tau_1, \tau_2\}$ be a task-set such that each task is characterized by the parameters $(\mathcal{C}, T, D, p)$. We have τ_1 defined by $(\begin{pmatrix} 2 & 3 \\ 0.5 & 0.5 \end{pmatrix}, 4, 4, 0.5)$ and τ_2 defined by $(\begin{pmatrix} 2 & 3 \\ 0.5 & 0.5 \end{pmatrix}, 8, 8, 0.1)$.

According to RM priority assignment, τ_1 has the higher priority and τ_2 the lower priority. In this case the response time of task τ_1 is $\mathcal{R}_1 = \begin{pmatrix} 2 & 3 \\ 0.5 & 0.5 \end{pmatrix}$ and the response time of task τ_2 is $\mathcal{R}_2 = \begin{pmatrix} 4 & 7 & 8 & D_2^+ \\ 0.25 & 0.25 & 0.375 & 0.125 \end{pmatrix}$

Table 1 summarizes the computation of the response time of task τ_2. The higher priority task, τ_1, has two jobs released in the hyperperiod, while the lower priority task, τ_2, has only one job released in the hyperperiod.

If jobs $\tau_{2,1}$ and $\tau_{1,1}$ have execution times equal to 2, then job $\tau_{2,1}$ is not pre-empted by the second job of τ_1 and it has a response time equal to 4. In the other three cases job $\tau_{2,1}$ is pre-empted and its response time is further increased. In these cases, only if all three jobs have execution times equal to 3 does job $\tau_{2,1}$ miss its deadline, with a probability of 0.125. We indicate by D_2^+ that the response time values are larger than the deadline.

Given that the deadline miss ratio of task τ_2 is $DMR_2 = 0.125$, which is greater than the maximum permitted value, $p_2 = 0.1$, then RM priority assignment is not feasible for this taskset with respect to the specified constraints.

Next we consider the alternative priority ordering where task τ_2 has a higher priority than task τ_1. For this priority ordering, the response time of task τ_2 is given by $\mathcal{R}_2 = \begin{pmatrix} 2 & 3 \\ 0.5 & 0.5 \end{pmatrix}$.

Task τ_1 has two jobs released in the hyperperiod $[0, 8)$. The response time of its first job is $\mathcal{R}_{1,1} = \begin{pmatrix} 4 & 5 & 6 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$, giving a deadline miss probability of $DMP_{1,1} = 0.75$.

Its second job has a response time of $\mathcal{R}_{1,2} = \begin{pmatrix} 2 & 3 & 4 & D_1^+ \\ 0.125 & 0.375 & 0.375 & 0.125 \end{pmatrix}$, giving a deadline miss probability of 0.125.

Combining the response times of the two jobs we get a response time for task τ_1 of $\mathcal{R}_1 = \begin{pmatrix} 2 & 3 & 4 & D_1^+ \\ 0.0625 & 0.1875 & 0.3125 & 0.4375 \end{pmatrix}$ and hence a deadline miss ratio of $DMR_1 = 0.4375$ which is less than the maximum permitted value of $p_1 = 0.5$. Thus the priority assignment is a feasible one, with all tasks having deadline miss ratios that are less than their maximum permitted values.

As RM priority assignment is not feasible for this task-set with respect to the specified constraints, and there exists an alternative priority ordering that is feasible, then RM is not an optimal priority assignment policy for the Basic Priority Assignment Problem (BPAP).

As a general rule, for a fixed priority scheduling algorithm to be optimal with respect to any of the three problems presented, it needs to take into account the probabilistic nature of the system and of the constraints, i.e. optimization criteria. With this intuition in mind, it would be easy to provide counterexamples to prove the non-optimality of other deterministic fixed priority algorithms, such as Deadline Monotonic, similar to the counterexample for Rate Monotonic.

2 A step by step example of applying the analysis for systems with multiple probabilistic parameters

Let $\tau = \{\tau_1, \tau_2\}$ be a task set with $\tau_1 = \left(\mathcal{C}_1 = \begin{pmatrix} 1 & 2 \\ 0.7 & 0.3 \end{pmatrix}, \mathcal{T}_1 = \begin{pmatrix} 4 & 3 \\ 0.9 & 0.1 \end{pmatrix}\right)$, and $\tau_2 = \left(\mathcal{C}_2 = \begin{pmatrix} 2 & 4 \\ 0.6 & 0.4 \end{pmatrix}, \mathcal{T}_2 = \begin{pmatrix} 8 & 6 \\ 0.8 & 0.2 \end{pmatrix}\right)$. the first random variable of the task representing its probabilistic execution time and the second one, its inter-arrival times distribution. We consider that τ_1 has the highest priority and τ_2 the lowest priority and they are scheduled on one processor, preemptively and synchronously released.

Since τ_1 is the highest priority task and we are in a preemptive context, its response time distribution is equal to that of its worst case execution time.

For τ_2 we calculate the pWCRT using the method described in Section5. This method stands for any number of tasks and we consider here the example with two tasks to ease the presentation.

Since the two tasks have the first release at the same moment $t = 0$, then the backlog of $\tau_{2,1}$, the first job of τ_2, is equal to the execution time of the higher priority task, τ_1, i.e. $\mathcal{B}_2 = C_1 = \begin{pmatrix} 1 & 2 \\ 0.7 & 0.3 \end{pmatrix}$.

We execute now the first step of the computation of the response time of a job, which is obtained by convolving its execution time with the backlog at the moment of its release. We have $\mathcal{R}_2 = \mathcal{B}_2 \otimes C_2 = \begin{pmatrix} 1 & 2 \\ 0.7 & 0.3 \end{pmatrix} \otimes \begin{pmatrix} 2 & 4 \\ 0.6 & 0.4 \end{pmatrix} = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 0.42 & 0.18 & 0.28 & 0.12 \end{pmatrix}$. To this partial result we need to add the possible preemptions.

The next release of τ_1 can be either at $t = 3$ or at $t = 4$. If it is the first case, then the preemption affects the tail of the response time distribution starting from the time instant $t = 4$. If it is the later, then the preemption affects the tale of the response time distribution from the time instant $t = 5$. For each of the two cases we split the calculation in a part corresponding to the head and a second part that corresponds to the tail. Following this reasoning the part corresponding to the tail it is convolved with the execution time of the preempting task (in this case τ_1). These two parts need to be associated to their probability of appearance which corresponds to the probability of appearance of the scenarios that produce them.

We denote by $\mathcal{R}_2^{1,1}$ the partial response time of the first job of task τ_2, in preemption 1, case 1, the lower indexes meaning the task number and job number respectively, and the upper indexes representing the preemption number and the case number respectively.

Since we are integrating the impact of the first preemption, then we obtain in the first case $\mathcal{R}_{2,1}^{1,1} = \begin{pmatrix} 3 \\ 0.42 \end{pmatrix} \oplus \begin{pmatrix} 4 & 5 & 6 \\ 0.18 & 0.28 & 0.12 \end{pmatrix} \otimes \begin{pmatrix} 1 & 2 \\ 0.7 & 0.3 \end{pmatrix} =$ $\begin{pmatrix} 3 \\ 0.42 \end{pmatrix} \oplus \begin{pmatrix} 5 & 6 & 7 & 8 \\ 0.126 & 0.25 & 0.168 & 0.036 \end{pmatrix} = \begin{pmatrix} 3 & 5 & 6 & 7 & 8 \\ 0.42 & 0.126 & 0.25 & 0.168 & 0.036 \end{pmatrix}$.

In order to calculate the probability of the appearance of this specific scenario, we multiply all probabilities associated with the release of $\tau_{1,2}$ at $t = 3$, which in this case it is equal to 0.1. We do this calculation by convolving the partial response time distribution with a fake random variable equal to $\begin{pmatrix} 0 \\ 0.1 \end{pmatrix}$.

We obtain then a partial result $\mathcal{R}_{2,1}^{1,1} = \begin{pmatrix} 3 & 5 & 6 & 7 & 8 \\ 0.42 & 0.126 & 0.25 & 0.168 & 0.036 \end{pmatrix} \otimes$ $\begin{pmatrix} 0 \\ 0.1 \end{pmatrix} = \begin{pmatrix} 3 & 5 & 6 & 7 & 8 \\ 0.042 & 0.0126 & 0.025 & 0.0168 & 0.0036 \end{pmatrix}$.

Since we convolve a random variable which has the sum of its probabilities equal to one, with a fake random variable which has only one probability, equal to 0.1, we obtain a (partial) random variable which has a sum of probabilities equal to 0.1.

Similarly we compute the partial response time of the first job of task τ_2, in preemption 1, case 2 as being $\mathcal{R}_2^{1,2} =$
$\left(\begin{pmatrix} 3 & 4 \\ 0.42 & 0.18 \end{pmatrix} \otimes \begin{pmatrix} 5 & 6 \\ 0.28 & 0.12 \end{pmatrix} \otimes \begin{pmatrix} 1 & 2 \\ 0.7 & 0.3 \end{pmatrix} \right) \otimes \begin{pmatrix} 0 \\ 0.9 \end{pmatrix} =$
$\left(\begin{pmatrix} 3 & 4 \\ 0.42 & 0.18 \end{pmatrix} \oplus \begin{pmatrix} 6 & 7 & 8 \\ 0.196 & 0.168 & 0.036 \end{pmatrix} \right) \otimes \begin{pmatrix} 0 \\ 0.9 \end{pmatrix} =$
$\begin{pmatrix} 3 & 4 & 6 & 7 & 8 \\ 0.378 & 0.162 & 0.1764 & 0.1512 & 0.0324 \end{pmatrix}$. The probabilities of this (partial) random variable sum up to 0.9.

The two (partial) distributions are combined into a single one, giving the partial response time of job τ_2 after the first preemption. We have
$\mathcal{R}_2 = \begin{pmatrix} 3 & 5 & 6 & 7 & 8 \\ 0.042 & 0.0126 & 0.025 & 0.0168 & 0.0036 \end{pmatrix} \oplus$

$$\begin{pmatrix} 3 & 4 & 6 & 7 & 8 \\ 0.378 & 0.162 & 0.1764 & 0.1512 & 0.0324 \end{pmatrix} =$$
$$\begin{pmatrix} 3 & 4 & 5 & 6 & 7 & 8 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.168 & 0.036 \end{pmatrix}.$$

The sum of the probabilities of this distribution is once more equal to 1.

In order to integrate the next (possible) preemption, we need to know the arrival time distribution of the next job that might preempt $\tau_{2,1}$. This next preempting job is in this case $\tau_{1,2}$ which has an arrival time distribution equal to

$$\mathcal{T}_{1,2} = \mathcal{T}_1 \otimes \mathcal{T}_1 = \begin{pmatrix} 4 & 3 \\ 0.9 & 0.1 \end{pmatrix} \otimes \begin{pmatrix} 4 & 3 \\ 0.9 & 0.1 \end{pmatrix} = \begin{pmatrix} 8 & 7 & 6 \\ 0.81 & 0.18 & 0.01 \end{pmatrix}.$$

Comparing this arrival distribution with the response time obtained before, we see that there are three possible scenarios :

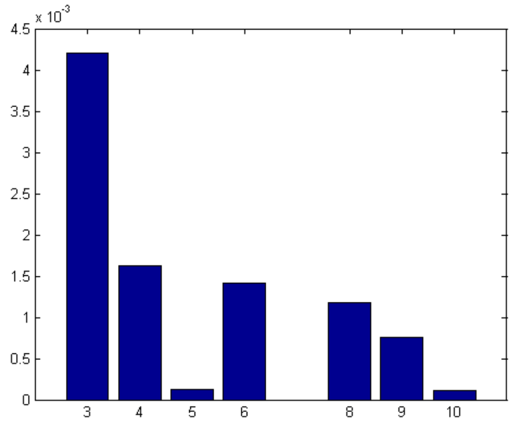
- $\tau_{1,2}$ arrives at $t = 6$ with a probability of 0.01 ; in this case its arrival affects the last two values of the response time distribution.
- $\tau_{1,2}$ arrives at $t = 7$ with a probability of 0.18 ; in this case its arrival affects only the last value of the response time distribution.
- $\tau_{1,2}$ arrives at $t = 8$ with a probability of 0.81 ; in this case its arrival does not affect the response time distribution, i.e., it does not preempt τ_2.

Using the same reasoning of integrating the preemption and updating with the occurrence probability, we compute the partial responses for the three cases. These computations are presented in Equation (1) and depicted in Figure 1. The first (partial) random variable, $\mathcal{R}_2^{2,1}$, has a sum of probabilities equal to 0.01. The second (partial) random variable, $\mathcal{R}_2^{2,2}$, has a sum of probabilities equal to 0.18. Finally, the last (partial) random variable, $\mathcal{R}_2^{2,3}$, has a probability sum equal to 0.81.

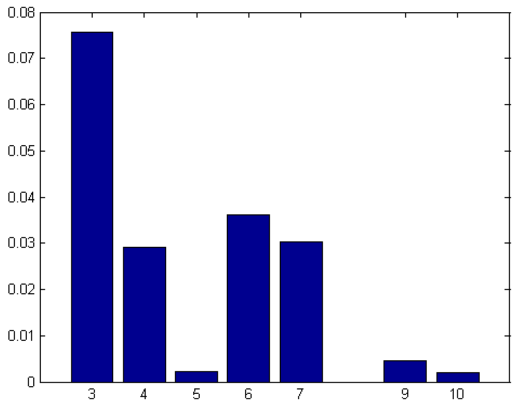
By combining the distributions obtained in the three cases, we obtain a (complete) random variable which has a sum of probabilities equal to 1. This distribution is presented in Equation (2) and depicted in Figure 2.

The next job of τ_1 has an arrival distribution as given in Equation (3). Since the least arrival value of $\tau_{1,3}$ is not less than the larger values of $\mathcal{R}_2$, it means that $\tau_{1,3}$ can not influence the Deadline Miss Probability of τ_2 so the analysis can be stopped here. For the sake of completeness we continue the analysis until the entire response time distribution of τ_2 is obtained.

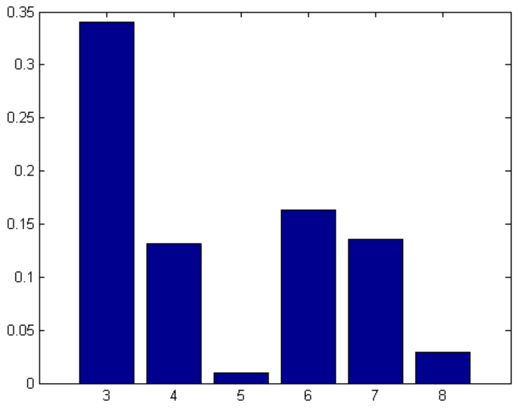
Comparing this arrival distribution with the partial response time distribution obtained at the previous step we conclude that there is one possible scenario in which τ_2 may be preempted, and that is when $\tau_{1,3}$ arrives at $t = 9$ and τ_2 has not finished



(a) $R_2^{2,1}$



(b) $R_2^{2,2}$



(c) $R_2^{2,3}$

FIGURE 1 – Partial results for R_2^2

its execution. In this case, only the last value of the response time distribution needs to integrate the preemption effect. We obtain in this scenario a partial response time that is presented in Equation (4) and depicted in Figure 3a.

The sum of the probabilities of this (partial) random variable is 0.001.

Also, the scenario that τ_2 is not preempted by $\tau_{1,3}$ needs to be updated with its respective probability, that is the sum of the probabilities of the three scenarios in which the arrival time is larger than any value of the distribution time, i.e., $0.729 + 0.243 + 0.027 = 0.999$.

The obtained distribution is presented in Equation 5 and depicted in Figure 3b.

By combining the two scenarios, we obtain *the response time* of τ_2, which is

$$
\boxed{
\begin{aligned}
\mathcal{R}_2^{2,1} = {} & \left(\left(\begin{array}{cccc} 3 & 4 & 5 & 6 \\ 0.42 & 0.162 & 0.0126 & 0.2014 \end{array}\right) \oplus \left(\begin{array}{cc} 7 & 8 \\ 0.168 & 0.036 \end{array}\right) \otimes \left(\begin{array}{cc} 1 & 2 \\ 0.7 & 0.3 \end{array}\right)\right) \otimes \left(\begin{array}{c} 0 \\ 0.01 \end{array}\right) = \\
& \left(\left(\begin{array}{cccc} 3 & 4 & 5 & 6 \\ 0.42 & 0.162 & 0.0126 & 0.2014 \end{array}\right) \oplus \left(\begin{array}{ccc} 8 & 9 & 10 \\ 0.1176 & 0.0756 & 0.0108 \end{array}\right)\right) \otimes \left(\begin{array}{c} 0 \\ 0.01 \end{array}\right) = \\
& \left(\begin{array}{ccccccc} 3 & 4 & 5 & 6 & 8 & 9 & 10 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.1176 & 0.0756 & 0.0108 \end{array}\right) \otimes \left(\begin{array}{c} 0 \\ 0.01 \end{array}\right) = \\
& \left(\begin{array}{ccccccc} 3 & 4 & 5 & 6 & 8 & 9 & 10 \\ 0.0042 & 0.00162 & 0.000126 & 0.0014098 & 0.001176 & 0.000756 & 0.000108 \end{array}\right). \\[2ex]
\mathcal{R}_2^{2,2} = {} & \left(\left(\begin{array}{ccccc} 3 & 4 & 5 & 6 & 7 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.168 \end{array}\right) \oplus \left(\begin{array}{c} 8 \\ 0.036 \end{array}\right) \otimes \left(\begin{array}{cc} 1 & 2 \\ 0.7 & 0.3 \end{array}\right)\right) \otimes \left(\begin{array}{c} 0 \\ 0.18 \end{array}\right) = \\
& \left(\left(\begin{array}{ccccc} 3 & 4 & 5 & 6 & 7 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.168 \end{array}\right) \oplus \left(\begin{array}{cc} 9 & 10 \\ 0.0252 & 0.0108 \end{array}\right)\right) \otimes \left(\begin{array}{c} 0 \\ 0.18 \end{array}\right) = \\
& \left(\begin{array}{ccccccc} 3 & 4 & 5 & 6 & 7 & 9 & 10 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.168 & 0.0252 & 0.0108 \end{array}\right) \otimes \left(\begin{array}{c} 0 \\ 0.18 \end{array}\right) = \\
& \left(\begin{array}{ccccccc} 3 & 4 & 5 & 6 & 7 & 9 & 10 \\ 0.0756 & 0.02916 & 0.002268 & 0.036252 & 0.03024 & 0.004536 & 0.001944 \end{array}\right). \\[2ex]
\mathcal{R}_2^{2,3} = {} & \left(\begin{array}{cccccc} 3 & 4 & 5 & 6 & 7 & 8 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.168 & 0.036 \end{array}\right) \otimes \left(\begin{array}{c} 0 \\ 0.81 \end{array}\right) = \\
& \left(\begin{array}{cccccc} 3 & 4 & 5 & 6 & 7 & 8 \\ 0.3402 & 0.13122 & 0.010206 & 0.163134 & 0.13608 & 0.02916 \end{array}\right).
\end{aligned}
}
\tag{1}
$$

depicted in Equation (6) and Figure 4.

The next arrival of τ_1 can not be earlier than $t = 12$, which indicates that this arrival does not preempt τ_2. We conclude that this is the final step of the computation of $\mathcal{R}_2$.

If jobs would be allowed to continue execution paster their deadline, then, at the next arrival of τ_2 there would be backlog that has a distribution given by Equation (7) and depicted in Figure 5.

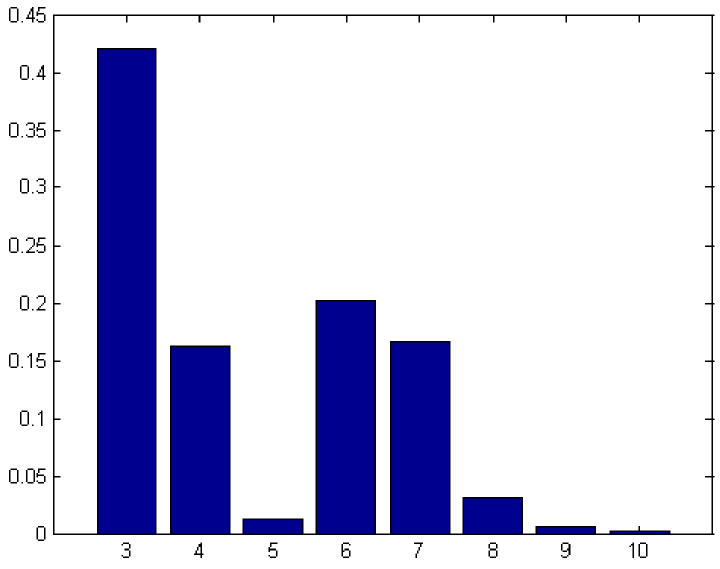
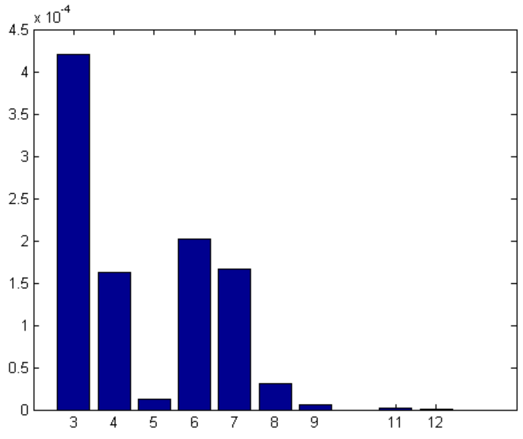


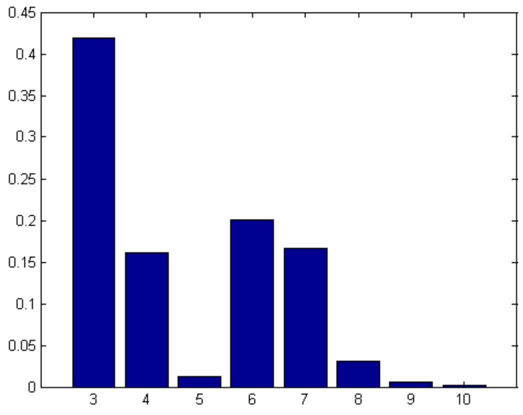
FIGURE 2 – R_2^2

$$\mathcal{R}_2^2 = \begin{pmatrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.16632 & 0.030336 & 0.005292 & 0.002052 \end{pmatrix}. \tag{2}$$

$$\mathcal{T}_{1,3} = \mathcal{T}_1 \otimes \mathcal{T}_1 \otimes \mathcal{T}_1 = \begin{pmatrix} 4 & 3 \\ 0.9 & 0.1 \end{pmatrix} \otimes \begin{pmatrix} 4 & 3 \\ 0.9 & 0.1 \end{pmatrix} \otimes \begin{pmatrix} 4 & 3 \\ 0.9 & 0.1 \end{pmatrix} = \begin{pmatrix} 12 & 11 & 10 & 9 \\ 0.729 & 0.243 & 0.027 & 0.001 \end{pmatrix} \tag{3}$$



(a) $R_2^{3,1}$



(b) $R_2^{3,0}$

FIGURE 3 – Partial results for R_2^3

$$\mathcal{R}_2^{3,1} = \left(\left(\begin{matrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.16632 & 0.030336 & 0.005292 \end{matrix}\right) \oplus \left(\begin{matrix} 10 \\ 0.002052 \end{matrix}\right) \otimes \left(\begin{matrix} 1 & 2 \\ 0.7 & 0.3 \end{matrix}\right)\right) \otimes \left(\begin{matrix} 0 \\ 0.001 \end{matrix}\right) = \left(\begin{matrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 11 & 12 \\ 42e5 & 162e6 & 126e7 & 2014e7 & 16632e8 & 30336e9 & 5292e9 & 14364e10 & 6156e10 \end{matrix}\right) \tag{4}$$

$$\mathcal{R}_2^{3,0} = \left(\begin{matrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.16632 & 0.030336 & 0.005292 & 0.002052 \end{matrix}\right) \otimes \left(\begin{matrix} 0 \\ 0.999 \end{matrix}\right) = \left(\begin{matrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0.41958 & 0.161838 & 0.0125874 & 0.2011986 & 0.16615368 & 0.03030566 & 0.00528671 & 0.00204995 \end{matrix}\right). \tag{5}$$

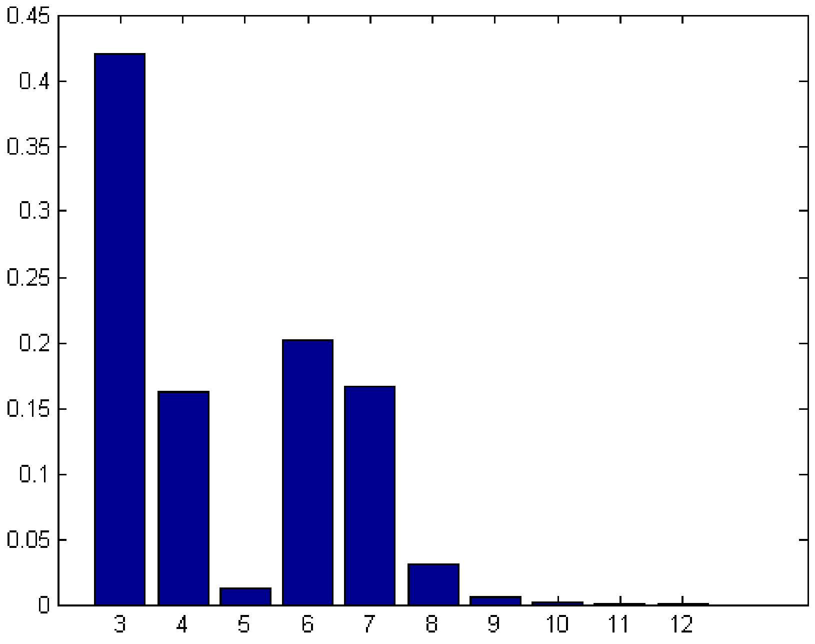


FIGURE 4 – R_2

$$\mathcal{R}_2 =$$

$$\begin{pmatrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 11 & 12 \\ 42e5 & 162e6 & 126e7 & 2014e7 & 16632e8 & 30336e9 & 5292e9 & 14364e10 & 6156e10 \end{pmatrix} \oplus$$

$$\begin{pmatrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0.41958 & 0.161838 & 0.0125874 & 0.2011986 & 0.16615368 & 0.03030566 & 0.00528671 & 0.00204995 \end{pmatrix} =$$

$$\begin{pmatrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 0.42 & 0.162 & 0.0126 & 0.2014 & 0.16632 & 0.030336 & 0.005292 & 0.002049948 & 0.0000014364 & 0.0000006156 \end{pmatrix} \tag{6}$$

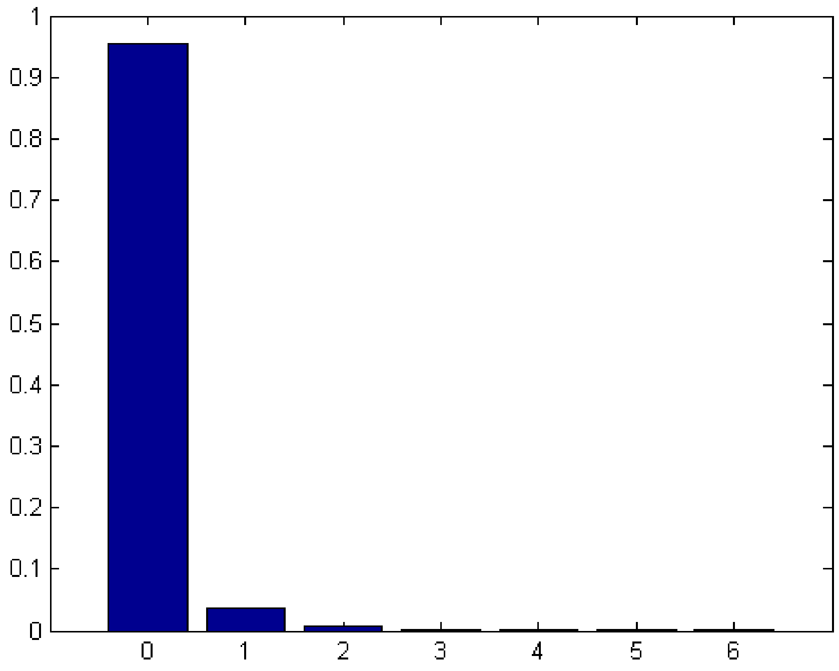


FIGURE 5 – B_2

$$\mathcal{B}_2 = \mathcal{D}_2 \ominus \mathcal{R}_{2,0} =$$
$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0.9533248 & 0.0374976 & 0.0077071584 & 0.00105954912 & 0.00041048208 & 0.0000028728 & 0.00000012312 \end{pmatrix} \quad (7)$$

Table des figures

Liste des tableaux

List of acronyms and notations

AP	Arrival Profile
$BPAP$	Basic Priority Assignment Problem
CAN	Controller Area Network
CDF	Cumulative Distribution Function
$CRTES$	Critical Real-Time Embedded Systems
$CRTS$	Critical Real-Time Systems
DM	Deadline Monotonic
DMP	Deadline Miss Probability
DMR	Deadline Miss Ratio
EDF	Earliest Deadline First
EP	Execution Profile
ET	Execution Time
ETP	Execution Time Profile
GMF	Generalized Multi-Frame
$MPAP$	Minimization of the Maximum Priority Assignment Problem
MIT	Minimum Inter-arrival Time
$NCGMF$	Non-Cyclic Generalized Multi-Frame
PDF	Probability Density Function
PMF	Probability Mass Function
$pCRPD$	probabilistic Cache-Related Preemption Delays
pET	probabilistic Execution Time
$pMIT$	probabilistic Minimum Inter-arrival Time

$pWCET$	probabilistic Worst Case Execution Time
RM	Rate Monotonic
RTS	Real Time System
$SPTA$	Static Probabilistic Timing Analysis
$WCET$	Worst Case Execution Time
$WCRT$	Worst Case Reponse Time

Abstract

Résumé

Mots-clés: Systèmes Temps Réel Probabilistes, Algorithmes d'Ordonnancement Optimaux, Re-échantillonnage Temps Réel, Monoprocesseur, Réseau CAN.

Les systèmes embarqués temps réel critiques intègrent des architectures complexes qui évoluent constamment afin d'intégrer des nouvelles fonctionnalités requises par les utilisateurs finaux des systèmes (automobile, avionique, ferroviaire, etc.). Ces nouvelles architectures ont un impact direct sur la variabilité du comportement temporel des systèmes temps réel. Cette variabilité entraîne un sur-approvisionnement important si la conception du système est uniquement basée sur le raisonnement pire cas. Approches probabilistes proposent des solutions basées sur la probabilité d'occurrence des valeurs les plus défavorables afin d'éviter le sur-approvisionnement, tout en satisfaisant les contraintes temps réel.

Les principaux objectifs de ce travail sont de proposer des nouvelles techniques d'analyse des systèmes temps réel probabilistes et des moyens de diminuer la complexité de ces analyses, ainsi que de proposer des algorithmes optimaux d'ordonnancement á priorité fixe pour les systèmes avec des temps d'exécution décrits par des variables aléatoires.

Les résultats que nous présentons dans ce travail ont été prouvés surs et á utiliser pour les systèmes temps réel durs, qui sont l'objet principal de notre travail. Notre analyse des systèmes avec plusieurs paramètres probabilistes a été démontrée considérablement moins pessimiste que d'autres types d'analyses. Cet analyse combinée avec des algorithmes d'ordonnancement optimaux appropriées pour les systèmes temps réel probabilistes peut aider les concepteurs de systèmes á mieux apprécier la faisabilité d'un systéme, en particulier de ceux qui sont jugé irréalisable par des analyses/algorithmes d'ordonnancement déterministes.

Abstract

Keywords: Probabilistic Real-Time Systems, Optimal Scheduling Algorithms, Real-Time Re-sampling, Uniprocessor, Controller Area Networks.

Critical real-time embedded systems integrate complex architectures that evolve constantly in order to provide new functionality required by the end users of the systems (automotive, avionics, railway, etc). These new architectures have a direct impact on the variability of the timing behavior of the real-time system. This variability leads to important over-provisioning if the design of the system is based only on worst case reasoning. Probabilistic approaches propose solutions are based on the probability of occurrence of the worst case values in order to avoid over provisioning while satisfying real-time constraints.

The main objectives of this work are new analysis techniques for probabilistic real-time systems and ways of decreasing the complexity of these analyses, as well as to propose optimal fixed priority scheduling algorithms for systems that have variability at the level of execution times.

The results that we provide in this work have been proved applicable to hard real-time systems, which are the main focus of our work. Our proposed analysis for systems with multiple probabilistic parameters has been shown to greatly decrease the pessimism introduced by other types of analyses. This type of analysis combined with the proper optimal scheduling algorithms for probabilistic real-time system help the system designers to better appreciate the feasibility of a system, especially of those that are deemed unfeasible by deterministic analyses/scheduling algorithms.

Bibliographie

[653, 2008] 653, A. (2008). An avionics standard for safe, partitioned systems. In *Wind River Systems/IEEE Seminar.*

[Abeni and Buttazzo, 1998] Abeni, L. and Buttazzo, G. (1998). Integrating multimedia applications in hard real-time systems. In *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, pages 4 –13.

[Abeni and Buttazzo, 1999] Abeni, L. and Buttazzo, G. (1999). QoS guarantee using probabilistic deadlines. In *IEEE Euromicro Conference on Real-Time Systems (ECRTS99)*, pages 242–249.

[Abeni et al., 2012] Abeni, L., Manica, N., and Palopoli, L. (2012). Efficient and robust probabilistic guarantees for real-time tasks. *Journal of Systems and Software*, 85(5) :1147–1156.

[ARP4761, 2001] ARP4761 (2001). Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment.

[Atlas and Bestavros, 1998] Atlas, A. and Bestavros, A. (1998). Statistical rate monotonic scheduling. In *19th IEEE Real-Time Systems Symposium (RTSS 1998).*

[Audsley, 1991] Audsley, N. (1991). Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical Report YCS 164, University of York.

[Audsley, 2001] Audsley, N. (2001). On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1) :39–44.

[Axer and Ernst, 2013] Axer, P. and Ernst, R. (2013). Stochastic response-time guarantee for non-preemptive, fixed-priority scheduling under errors. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 172 :1–172 :7, New York, NY, USA. ACM.

[Baruah et al., 1999] Baruah, S., Chen, D., Gorinsky, S., and Mok, A. (1999). Generalized multiframe tasks. *Real-Time Systems*, 17 :5–22.

[Bernat et al., 2005] Bernat, G., Burns, A., and Newby, M. (2005). Probabilistic timing analysis : An approach using copulas. *Journal of Embedded Computing*, 1(2) :179–194.

[Bernat et al., 2002] Bernat, G., Colin, A., and Petters, S. (2002). Wcet analysis of probabilistic hard real-time systems. In *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, pages 279 – 288.

[Broster and Burns, 2004a] Broster, I. and Burns, A. (2004a). Applying random arrival models to fixed priority analysis. In *the Proceedings of the Work-In-Progress of the 25th IEEE Real-Time Systems Symposium (RTSS04).*

[Broster and Burns, 2004b] Broster, I. and Burns, A. (2004b). Random arrivals in fixed priority analysis. In *1st International Workshop on Probabilistic Analysis Techniques for Real-time and Embedded Systems (PARTES2004).*

[Broster et al., 2002] Broster, I., Burns, A., and Rodriguez-Navas, G. (2002). Probabilistic analysis of can with faults. In *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, pages 269 – 278.

[Burns et al., 2003] Burns, A., Bernat, G., and Broster, I. (2003). A probabilistic framework for schedulability analysis. In *Third International Embedded Software Conference (EMSOFT 2003)*, pages 1–15.

[Burns et al., 1999] Burns, A., Punnekkat, S., Strigini, L., and Wright, D. (1999). Probabilistic scheduling guarantees for fault-tolerant real-time systems. In *Dependable Computing for Critical Applications 7, 1999*, pages 361 –378.

[Buttle, 2012] Buttle, D. (2012). Real-time in the prime-time. In *Keynote talk at the 24th Euromicro Conference on Real-Time Systems (ECRTS).*

[CAN, 1992] CAN (1992). `http://www.can-cia.org`.

[Cazorla et al., 2012] Cazorla, F., Quinones, E., Vardanega, T., Cucu-Grosjean, L., Triquet, B., Bernat, G., Berger, E., Abella, J., Wartel, F., Houston, M., Santinelli, L., Maxim, D., Kosmidis, L., and Lo, C. (2012). Proartis : Probabilistically analyzable real-time system. *ACM Transactions on Embedded Computing Systems.*

[Cucu and Tovar, 2006] Cucu, L. and Tovar, E. (2006). A framework for response time analysis of fixed-priority tasks with stochastic inter-arrival times. *ACM SIGBED Review*, 3(1).

[Cucu-Grosjean et al., 2012] Cucu-Grosjean, L., Santinelli, L., Houston, M., Lo, C., Vardanega, T., Kosmidis, L., Abella, J., Mezzeti, E., E., Q., and Cazorla, F. (2012). Measurement-based probabilistic timing analysis for multi-path programs. In *the 24th Euromicro Conference on Real-Time Systems (ECRTS12).*

[David and Puaut, 2004] David, L. and Puaut, I. (2004). Static determination of probabilistic execution times. In *Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004 (ECRTS 2004)*, pages 223–230. IEEE.

[Davis and Burns, 2011] Davis, R. and Burns, A. (2011). Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems Journal*, 47(1) :1–40.

[Davis et al., 2007] Davis, R., Burns, A., Bril, R., and Lukkien, J. (2007). Controller are network (CAN) schedulability analysis : refuted revisited and revised. *Real-Time Systems*, 35 :239–272.

[Davis et al., 2013a] Davis, R. I., Santinelli, L., Altmeyer, S., Maiza, C., and Cucu-Grosjean, L. (2013a). Analysis of probabilistic cache related pre-emption delays. In *IEEE Euromicro Conference on Real-Time Systems (ECRTS13).*

[Davis et al., 2013b] Davis, R. I., Whitham, J., and Maxim, D. (2013b). Static probabilistic timing analysis for multicore processors with shared cache. *the 4th International Real-Time Scheduling Open Problems Seminar (RTSOPs2013), in conjunction with ECRTS2013.*

[Diaz et al., 2004] Diaz, J., Lopez, J., M., G., Campos, A., Kim, K., and Lo Bello, L. (2004). Pessimism in the stochastic analysis of real-time systems : Concept and applications. In *25th IEEE International Real-Time Systems Symposium (RTSS 2004)*, pages 197–207.

[Díaz et al., 2002] Díaz, J. L., García, D. F., Kim, K., Lee, C.-G., Lo Bello, L., López, J. M., Min, S. L., and Mirabella, O. (2002). Stochastic analysis of periodic real-time systems. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS 2002)*, page 289.

[Edgar and Burns, 2001] Edgar, S. and Burns, A. (2001). Statistical analysis of WCET for scheduling. In *22nd IEEE International Real-Time Systems Symposium (RTSS 2001)*, pages 215–224.

[Eisenbrand and Rothvob, 2008] Eisenbrand, F. and Rothvob, T. (2008). Static-priority real-time scheduling : Response time computation is np-hard. In *Proceedings of the 2008 Real-Time Systems Symposium.*

[Gardner and Lui, 1999] Gardner, M. and Lui, J. (1999). Analyzing stochastic fixed-priority real-time systems. In *5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems.*

[Griffin and Burns, 2010] Griffin, D. and Burns, A. (2010). Realism in statistical analysis of worst case execution times. In *10th Intl. Workshop on Worst-Case Execution Time Analysis*, pages 49–57.

[Hansen et al., 2009] Hansen, J., Hissam, S., and Moreno, G. (2009). Statistical-based WCET estimation and validation. In *9th International Workshop on Worst-Case Execution Time (WCET) Analysis.*

[Hardy and Puaut, 2013] Hardy, D. and Puaut, I. (2013). Static probabilistic worst case execution time estimation for architectures with faulty instruction caches. In *Proceedings of the 21st International conference on Real-Time Networks and Systems (RTNS2013)*, pages 35–44. ACM.

[Hu et al., 2001] Hu, X., Zhou, T., and Sha, E.-M. (2001). Estimating probabilistic timing performance for real-time embedded systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 9(6) :833 –844.

[Joseph and Pandya, 1986] Joseph, M. and Pandya, P. (1986). Finding response times in real-time system. *The Computer Journal.*

[Kaczynski et al., 2007] Kaczynski, G., Lo Bello, L., and Nolte, T. (2007). Deriving exact stochastic response times of periodic tasks in hybrid priority-driven soft real-time systems. In *the 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'07), Greece.*

[Kaczynski et al., 2006] Kaczynski, G. A., Bello, L. L., and Nolte, T. (2006). Towards stochastic response-time of hierarchically scheduled real-time tasks. In *Proceedings of 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2006)*, pages 453–456.

[Keogh et al., 2001] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3 :263–286. 10.1007/PL00011669.

[Kim et al., 2005] Kim, K., Diaz, J. L., Lo Bello, L., Lopez, J. M., Lee, C.-G., and Min, S. L. (2005). An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Trans. Comput.*, 54(11) :1460–1466.

[Lehoczky, 1996] Lehoczky, J. (1996). Real-time queueing theory. In *10th of the IEEE Real-Time Systems Symposium (RTSS96)*, pages 186–195.

[Liu and Layland, 1973] Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1) :46–61.

[Lopez et al., 2008] Lopez, J., Diaz, J. L., E., J., and Garcia, D. (2008). Stochastic analysis of real-time systems under preemptive priority-driven scheduling. *Real-time Systems*, 40(2) :180–207.

[Lu et al., 2011] Lu, Y., Nolte, T., Bate, I., and Cucu, L. (2011). A new way about using statistical analysis of worst-case execution times. In *in the WiP sesion of the Euromicro Conference on Real-Time Systems.*

[Lu et al., 2012] Lu, Y., Nolte, T., Bate, I., and Cucu-Grosjean, L. (2012). A statistical response-time analysis of real-time embedded systems. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, pages 351–362. IEEE.

[Manica et al., 2012] Manica, N., Palopoli, L., and Abeni, L. (2012). Numerically efficient probabilistic guarantees for resource reservations. *Emerging Technologies and Factory Automation (ETFA), 2012 17th IEEE International Conference on.*

[Manolache et al., 2002] Manolache, S., Eles, P., and Peng, Z. (2002). Schedulability analysis of multiprocessor real-time applications with stochastic task execution times. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, ICCAD '02, pages 699–706, New York, NY, USA. ACM.

[Manolache et al., 2004] Manolache, S., Eles, P., and Peng, Z. (2004). Schedulability analysis of applications with stochastic task execution times. *ACM Trans. Embed. Comput. Syst.*, 3(4) :706–735.

[Maxim et al., 2012a] Maxim, C., Gogonel, A., Maxim, D., Cucu, L., et al. (2012a). Estimation of probabilistic minimum inter-arrival times using extreme value theory. In *the 6th Junior Researcher Workshop on Real-Time Computing (JRWRTC2012).*

[Maxim et al., 2011] Maxim, D., Buffet, O., Santinelli, L., Cucu-Grosjean, L., and Davis, R. (2011). Optimal priority assignments for probabilistic real-time systems. In *the 19th International Conference on Real-Time and Network Systems (RTNS2011).*

[Maxim and Cucu-Grosjean, 2010] Maxim, D. and Cucu-Grosjean, L. (2010). Towards optimal priority assignment for probabilistic CAN-based systems. In *WIP session of the 8th IEEE International Workshop on Factory Communication Systems (WFCS'2010).*

[Maxim and Cucu-Grosjean, 2013] Maxim, D. and Cucu-Grosjean, L. (2013). Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *proceeding of the* 34^{th} *IEEE Real Time Systems Symposium, 2013. RTSS'2013.*

[Maxim et al., 2012b] Maxim, D., Houston, M., Santinelli, L., Bernat, G., Davis, R. I., and Cucu-Grosjean, L. (2012b). Re-sampling for statistical timing analysis of real-time systems. In *Proceedings of the 20th International Conference on Real-Time and Network Systems*, pages 111–120. ACM.

[Mok, 1983] Mok, A. (1983). *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment.* PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology.

[Moyo et al., 2010] Moyo, N. T., Nicollet, E., Lafaye, F., and Moy, C. (2010). On schedulability analysis of non-cyclic generalized multiframe tasks. In *Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on*, pages 271–278. IEEE.

[Navet et al., 2000] Navet, N., Song, Y.-Q., and Simonot, F. (2000). Worst-case deadline failure probability in real-time applications distributed over controller area network. *J. Syst. Archit.*, 46(7) :607–617.

[Nissanke et al., 2002] Nissanke, N., Leulseged, A., and Chillara, S. (2002). Probabilistic performance analysis in multiprocessor scheduling. *Computing Control Engineering Journal*, 13(4) :171 – 179.

[RapitaSystems, Ltd] RapitaSystems (Ltd.). `http://www.rapitasystems.com`.

[Refaat and Hladik, 2010] Refaat, K. S. and Hladik, P.-E. (2010). Efficient stochastic analysis of real-time systems via random sampling. In *IEEE Euromicro Conference on Real-Time Systems (ECRTS 2010)*, pages 175–183.

[Russell and Norvig, 1995] Russell, S. and Norvig, P. (1995). *Artificial Intelligence : A Modern Approach.* Englewood Cliffs, NJ : prentice Hall.

[Santinelli et al., 2007] Santinelli, L., Yomsi, P. M., Maxim, D., and Cucu-Grosjean, L. (2007). A component-based framework for modeling and analyzing probabilistic real-time systems. *16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'11).*

[Singh and Jeffay, 2007] Singh, A. and Jeffay, K. (2007). Co-scheduling variable execution time requirement real-time tasks and non real-time tasks. In *the 19th Euromicro Conference on Real-Time Systems, 2007. ECRTS '07*, pages 191 –200.

[Stigge and Yi, 2012] Stigge, M. and Yi, W. (2012). Hardness results for static priority real-time scheduling. In *the 24th Euromicro Conference on Real-Time Systems (ECRTS).*

[Tia et al., 1995] Tia, T., Deng, Z., Shankar, M., Storch, M., Sun, J., Wu, L., and Liu, J. (1995). Probabilistic performance guarantee for real-time tasks with varying computation times. In *IEEE Real-Time and Embedded Technology and Applications Symposium (ETFA 1995).*

[Wolpert and Macready, 1995] Wolpert, D. and Macready, W. (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

[Zeng et al., 2009a] Zeng, H., Di Natale, M., Giusto, P., and Sangiovanni-Vincentelli, A. (2009a). Statistical analysis of controller area network message response times. In *Industrial Embedded Systems, 2009. SIES '09. IEEE International Symposium on*, pages 1 –10.

[Zeng et al., 2009b] Zeng, H., Di Natale, M., Giusto, P., and Sangiovanni-Vincentelli, A. (2009b). Stochastic analysis of can-based real-time automotive systems. *Industrial Informatics, IEEE Transactions on*, 5(4) :388 –401.

[Zeng et al., 2010] Zeng, H., Di Natale, M., Giusto, P., and Sangiovanni-Vincentelli, A. (2010). Using statistical methods to compute the probability distribution of message response time in controller area network. *Industrial Informatics, IEEE Transactions on*, 6(4) :678 –691.