Stochastic analysis of stationary real-time systems

Kevin Zagalo

THÈSE DE DOCTORAT

École Doctorale Informatique, Télécommunications
et Électronique (ED 130)

Stochastic analysis of stationary real-time systems

Analyse stochastique des systèmes temps-réel stationnaires

présentée et soutenue par

Kevin ZAGALO

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

le 28 Septembre 2023

Jury:

Anne Bouillard	HDR	Huawei Technologies France	Rapportrice
Ye-Qiong Song	Prof.	Loria, France	Rapporteur
Alix Munier-Kordon	Prof.	Lip6, France	Examinatrice
Alain Girault	DR	Inria, France	Président
Avner Bar-Hen	Prof.	Cnam, France	Co-directeur
Liliana Cucu-Grosjean	DR	Inria, France	Co-directrice

Em memória do meo avô

Acknowledgements

I am deeply grateful to my supervisors, Liliana Cucu-Grosjean and Avner Bar-Hen, for their guidance and support throughout my research journey. Their expertise, patience, and encouragement have been crucial in shaping this thesis. I am grateful to the reviewers of this thesis, Ye-Qiong Song and Anne Bouillard, for their valuable feedback and constructive comments, which have greatly improved the quality and clarity of this work.

I would like to thank my colleagues and friends at the Kopernic team of Inria; Chiara Daini and Amine Khelassi, with whom I shared my office. Their support and collaborative spirit contributed to a positive, productive and very funny work environment; and, Slim Ben Amor and Ismail Hawila for their helpful discussions and support. I would also like to acknowledge the contributions of the students I supervised: Olena Verbytska, Margarita Tomina and Marc-Antoine Auvray, and wish them the best in their future endeavors.

I would like to express my appreciation to Yasmina Abdeddaïm for her guidance in understanding the complexities of real-time systems and scheduling theory. Her insights have been invaluable in the development of the results presented in this thesis. I also thank Anne Mathurin, Thomas Watteyne, Fawzi Nashashibi, Raul de Charette, Martina Balbi, Razanne Abu-Aisheh, Trifun Savić, Said Alvarado, Mališa Vučinić, Filip Maksimovic, Alexandre Abadie and the many other Ph.D. students, post-doctoral researchers, engineers and team assistants at Inria Paris, with whom I shared (way too much) coffee, chocolates and beers. Those moments contribute a lot in the positive feeling I have about those four years.

Finally, I thank my parents, sister and friends for their unwavering support and encouragement. In particular, Julie Loisel, who had to bear with me throughout this journey and suffered through the same challenges as I did, especially during the unprecedented circumstances of the COVID-19 pandemic. I am grateful to have had her by my side, as she has been a constant source of love, support, and understanding throughout this journey.

Glossary

DMP	Deadline Miss Probability
EDF	Earliest Deadline First
EM	Expectation-Maximization
EVT	Extreme Value Theory
FIFO	First-In-First-Out
MC	Mixed-Criticality
MLE	Maximum Likelihood Estimate
RAA	Reasonable Allocation Algorithm
RM	Rate Monotonic
TDA	Time Demand Analysis
WCET	Worst-Case Execution Time
WCRT	Worst-Case Response Time

List of symbols

$\mathbf{P}$	Probability operator
$\mathbf{E}$	Expectation operator
Γ	Task set
Π	Multiprocessor
τ_i	The i-th task
κ	A processor
$\Gamma_i^+(\kappa)$	Set of higher priority tasks than τ_i on a processor κ
$\Gamma_i^-(\kappa)$	Set of lower priority tasks than τ_i on a processor κ
$s(\kappa)$	the speed of processor κ
C_i	Execution time of the task τ_i
F_i	Distribution of the execution time of the task τ_i
m_i	Mean execution time of τ_i
v_i	Second order moment of the execution time of τ_i
T_i	Inter-arrival time of the task τ_i
D_i	Relative deadline of the task τ_i
G_i	Distribution of the inter-arrival times of the task τ_i
$p_i(x)$	Deadline miss probability of τ_i conditioned to a blocking time x
p_i^{max}	Worst-case deadline miss probability of τ_i
$\tilde{p}_i$	Steady-state deadline miss probability of τ_i
λ_i	Rate of arrivals of τ_i
u_i	Mean utilization of τ_i
$\bar{u}_i$	Mean utilization of level i
$\bar{u}_i(\kappa)$	Mean utilization of level i on κ
$\bar{u}_i^{max}$	Maximum utilization of level i
$\bar{u}_i^{max}(\kappa)$	Maximum utilization of level i on κ
$\bar{v}_i$	Demand deviation of level i
v_i^{max}	Maximum deviation of level i
$v_i^{max}(\kappa)$	Maximum deviation of level i on κ
$\tau_{i,j}$	j-th job of τ_i
$C_{i,j}$	Execution time of $\tau_{i,j}$
$A_{i,j}$	Arrival time of $\tau_{i,j}$
$T_{i,j}$	Inter-arrival time between $\tau_{i,j-1}$ and $\tau_{i,j}$
$R_{i,j}$	Response time of $\tau_{i,j}$
$R_{i,j}^{(\infty)}$	Heavy-traffic response time of $\tau_{i,j}$
$N_i(t)$	Number of jobs of τ_i released before $t > 0$

$\bar{W}_i(t)$	Demand of level i at $t > 0$
$\beta_i(t)$	Backlog of level i at $t > 0$
$B_i(t)$	Blocking time of level k at $t > 0$
$\bar{A}_{k,l}$	The arrival time of the l-th job of level k
$\bar{I}_{k,l}$	The mark (task) of the l-th job of level k
$\bar{C}_{k,l}$	The execution time of the l-th job of level k
$\bar{W}_i^{(\infty)}$	Heavy-traffic demand of level i
$\beta_i^{(\infty)}$	Heavy-traffic backlog of level i
$\mathcal{I}(x)$	First idle time with backlog x
$\mathcal{I}_k(x)$	First idle time of level k with backlog x
$\tilde{R}_i$	Steady-state response time of τ_i
$\tilde{H}_i$	Exceedence function of $\tilde{R}_k$
R_i^{max}	Heavy-traffic worst-case response time
H_i^{max}	Exceedence function of R_i^{max}
$\tilde{\beta}_k$	Steady-state backlog of level k
π_k	Steady-state backlog of level k distribution
η_k^{-1}	Mean of $\tilde{\beta}_k$
Φ	Standard normal distribution
ψ	Inverse Gaussian probability function
$\psi_k(\cdot, x)$	Probability function of $\mathcal{I}_k(x)$
$\Psi_k(\cdot, x)$	Exceedence function of $\mathcal{I}_k(x)$
$t_{idle}^{max}(\varepsilon)$	Maximum ε-idle time
$B_i^{max}(\kappa)$	Worst-case blocking time of level i on processor κ
$\mu_i(\kappa)$	Worst-case blocking time distribution of level i on processor κ
$\mathcal{A}_i$	Set of reasonable allocations of τ_i

Some notations rules

(i) Variables with a bar refer to priority level variables : $\bar{x}$, which does not mean that all priority level variables are written with a bar,

(ii) F_i, G_i and Φ are distribution functions,

(iii) H_i^{max}, $\tilde{H}_i$ and Ψ_k are exceedence functions,

(iv) All variables written in capital letters are random variables, except $\mathbf{P}, \mathbf{E}, \Gamma, \Pi$ and the ones listed on **(ii)** and **(iii)**.

List of Figures

List of Tables

Contents

1 | Introduction

Contents

Real-time systems birth dates back to the early days of computing, when the first computers have been used to perform a variety of functionalities that required immediate processing and response. One of the earliest examples of a real-time system has been the SAGE (Semi-Automatic Ground Environment) air defense system developed by the U.S. military in the 1950s. This system used a network of computers to track and identify incoming aircrafts, and was able to provide a rapid response to potential threats. In the 1960s and 1970s, the development of minicomputers and microprocessors led to the creation of a wide range of real-time systems, including process control systems in factories, medical monitoring systems, and traffic control systems. In the 1980s and 1990s, the widespread adoption of computers in a variety of industries and applications led to the development of

more advanced real-time systems. These systems often used distributed computing architectures and advanced networking technologies to enable the timely processing and exchange of data. In recent years, the increasing use of sensors, Internet of Things (IoT) devices, and other types of connected systems has led to the development of even more sophisticated real-time systems, including those used in autonomous vehicles, smart cities, and industrial automation. These systems rely on advanced technologies such as machine learning and artificial intelligence to enable real-time decision making and response.

In the 2010s, multiprocessor systems have made their entry in Commercial Off The Shelf (COTS) processors, and have been widely used since, which led to a bigger capacity to compute faster. Moreover, in what we define as *critical systems*[1] (avionics, automotive, space), a safe utilization of multiprocessor systems remains an open problem [Reilly, 2020]. Furthermore, the use of autonomous driving based on machine learning algorithms is challenging. For example, showing that computer vision is safe at any circumstance when controlling a car is still an open problem [Dixit et al., 2021]. Indeed, due to the pioneer utilization of multiprocessor systems by the smartphone market as well as the impressive expansion of this later market, the microprocessor industry has evolved towards general purpose processors with complex architectures that are not *time predictable*. Their lack of time predictability is due to features like several cores, multiple levels of caches and pipelines, speculative branching, communicating through shared memory or/and through a network on chip, *etc.*. Smartphone users are willing to charge their phones or to restart regularly their applications to compensate a bad design of the phones on multiprocessor systems. However, the rest of the real-time system industry is facing the open problem of time predictability of programs on multiprocessor systems in order to provide stable and low-energy consumption applications. Bounding execution times of a program on multiprocessor systems is known to be an open problem [Wilhelm et al., 2008,Maiza et al., 2019,Davis and Cucu-Grosjean, 2019]. In this thesis, the complexity of such system is considered as impossible to describe in

[1]Throughout the thesis, concepts formally defined after being mentioned are written in italic.

a deterministic manner, such that statistical methods provide a suitable estimation of time behaviors of the studied system.

1.1 Context

A real-time system is a computer system in interaction with an environment and users [Harel and Pnueli, 1984]. Sensors and programs are typically triggered periodically in order for the system to constantly interact with its environment. In order to make this interaction possible, it must satisfy temporal constraints. These temporal constraints can be more or less strict, depending on what functionality is asked from the system. We say that programs of a real-time system must meet their *deadlines*. We can classify deadlines according to the importance of functionalities associated to their associated programs:

(i) hard deadlines for programs such that a deadline miss consequence is not acceptable for either economic, human or ecological reasons, *e.g.*, the braking system of a car.

(ii) soft deadlines for programs such that a deadline miss consequence is a delayed output of a program also known as a task, *e.g.*, a sensor adjusting the temperature of a room.

Real-time systems consist in both hard and soft time constrained programs, and the analysis of their interference due to shared resources is a challenging open problem. Designing such system ensures that all programs meet their deadlines at a given rate. In practice, certification processes expect failure rates, *e.g.*, 10^{-12}-frequency associated to failures within one hour of functioning-, for hard deadlines and less restrictive rates for soft deadlines [Gumzej and Halang, 2010, Fault Forecasting, p. 64]. For this reason, in this thesis we use the concept of deadlines with a *permitted failure rate* and make no formal difference between hard and soft.

There is a wide range of research areas related to real-time systems and we present below those that we consider the most relevant:

(i) scheduling: a scheduling algorithm is responsible for ensuring that each program is allocated sufficient processing resources to meet its timing constraints. It determines how to allocate resources among the programs, and may use a variety of factors, such as task priority, task execution time, order of arrival, and processor utilization, to make this decision. Scheduling plays a critical role in the performance and reliability of a real-time system, and must be carefully designed and implemented to ensure that the system meets its timing requirements.

(ii) timing analysis: it is used to determine the Worst-Case Execution Time (WCET) of a program. The WCET is the maximum amount of time a task is expected to take to complete its execution. There are several techniques to estimate the WCET of a program. The *static analysis* is based on the code inspection, and modeling. The *Measurement-based analysis* involves collecting data on the program's time behavior. This can be done by instrumenting the system's code to measure the execution time of each program, or by using hardware performance counters to monitor the system's performance. Measurement-based analysis may provide less perssimistic results than static analysis, but it requires that the system is operational and may not be feasible for systems difficult to test or measure.

(iii) real-time architectures: it is the study of both hardware and software components that are optimized for speed and accuracy, such as specialized processors, algorithms, and communication protocols *e.g.*,Graphical Processing Units (GPU) or Field Programmable Gate Arrays (FPGA).

(iv) real-time operating systems (RTOSs): a RTOS is an operating system that is designed to ensure a certain level of processing capacity for a deterministic response to events occurring within a fixed time interval. It is typically used for time-critical applications that require a more predictable response than a non-real-time operating system can provide.

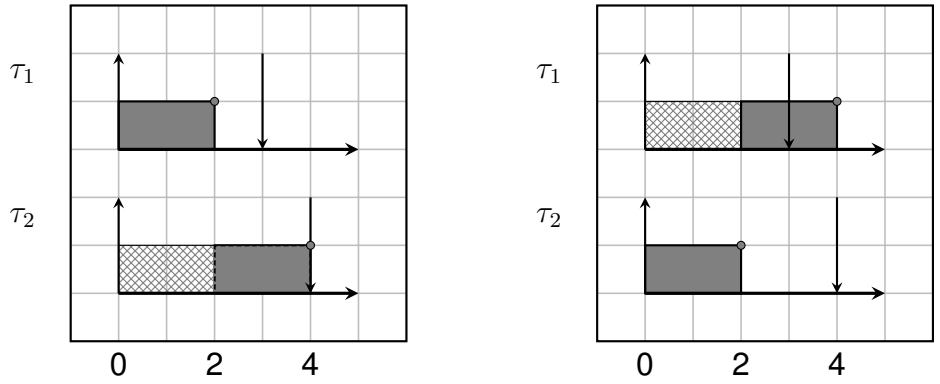
(v) fault-tolerance of real-time systems: it is a measure of the system's ability

to continue operating, even when one or more of its components fail. Fault-tolerance is achieved by creating redundancy in the system's components, programs, and data, so that if one component fails, the system can still operate. Fault-tolerance also includes the ability to detect and respond quickly and effectively to a failure. This ensures that the system can continue to meet its real-time requirements, even when a failure occurs.

(vi) real-time networks: they are networks designed to transmit data with strict timing requirements. These types of networks are used in a variety of applications where the timely delivery of data is critical, such as in industrial automation, avionics, and automotive systems. In Time Sensitive Network (TSN) for example, the transmission of data is synchronized to a common clock, and the network is designed to minimize the amount of delay and other types of variability in the transmission of data. TSNs may use a variety of technologies and protocols to ensure the timely delivery of data, including deterministic Ethernet.

(vii) real-time oriented machine learning: recently, there has been an increasing use of machine learning methods, *e.g.*, in the automotive industry, particularly for tasks such as autonomous driving, predictive maintenance, and traffic prediction. These methods often involve solving optimization problems with timing constraints, which are requirements that specify how long a task or operation is allowed to take. By incorporating these constraints into the optimization problem, it is possible to ensure that the system meets its timing requirements and performs reliably under all operating conditions.

In order to be safe, programs must meet their timing constraints. Real-time scheduling theory provides methods that orders the execution of programs over a finite or periodic number of time instances. A set of programs is considered schedulable for a given processing unit if it can be proven that each instance of a program has enough time to execute before its deadline. Scheduling is therefore mainly an optimization problem involving the allocation of limited resources,



(a) Deadlines are met. **(b)** Deadline of τ_1 is missed.

Figure 1.1: Example of a scheduling algorithm decision on a single processor. The up-arrow represents the activation of a task, the down-arrow a deadline. The black circle means that the task has finished its execution. The gray-squared areas indicates that a task awaits for processing resources

including hardware and time, to programs. Since the 2000s, several studies have been focused on probabilistic analyses to compute the probability of missing a deadline.

There are sources of uncertainties in every aspect of a real-time system, as it is by definition an object in continuous interaction with an unpredictable environment and users. One of the main goal of a real-time design is to minimize the impact of those uncertainties by construction and to ensure that the system is resilient to its environment. The purpose of using probability theory in real-time systems is to quantify those uncertainties. Solving this problem involves being able to adapt the system's decisions and, quantify and predict deadline misses.

1.2 Motivation

In a real-time system, it is common to deliberately over-dimension the system to ensure its ability to be reactive. However, this may lead to a large amount of wasted computing resource. The variability of the execution times is the degree to which the execution time of a task varies over time. There are several factors that can contribute to the variability of execution times in a real-time system, including:

(i) resource contention: if multiple programs are competing for the same resource

(*e.g.*, memory, pipelines, I/O devices), the execution time of each task may be affected.

(ii) unpredictable environment: sensors provide the input data for the system's programs. The execution time of these programs depends on values of input data that may trigger different branches of a program.

This variability can have serious consequences in a real-time system, as it may impact the ability of the system to meet its deadlines and fulfill its real-time requirements.

For embedded systems with low energy and computing resources, designing real-time systems means associating a suited micro-controller architecture to a set of programs. A key part of this design is to chose the appropriate processing unit (such as a CPU) for a given task set. To ensure that every task is executed within their specified timing constraints, the CPU and other computing resources are allocated to different programs, for example, according to their priority. During the run-time, each instance of programs competes for processing time. Timing correctness of real-time systems is traditionally guaranteed by a separate schedulability analysis and a WCET analysis. Classical techniques for WCET analyses aim at finding an upper-bound on execution times. The time taken by a program to respond to an input and provide the output or display the updated information is known as the *response time*. After determining the WCET, the Worst-Case Response Time (WCRT) is calculated by aggregating the WCET of programs in the worst-case scenario, *i.e.*, the scenario that produces the longest response times. However, this method [the worst-case reasoning] is sufficient to make schedulable task sets and it forces designers to over-estimate the quantity of processing unit necessary to run a task set.

In order to decrease the pessimism associated to this over-estimation, a significant amount of research in real-time systems focuses on statistical analyses (as discussed in Section 1.3.2).

1.3 State-of-the-art

In this section, we present existing work relevant for this thesis, to the best of our knowledge. That is to say, *applied probabilities on real-time scheduling algorithms, timing analysis and shared resources modeling*, and results on real-time multiprocessor scheduling. There are multiple sources of inter-core interference for a given architecture: shared caches, shared buses, memory and scheduling policies. Existing results mainly address issues related to the system stability, computational algorithm designs, optimal scheduling, allocation, or performance analysis. This first Section 1.3.1 contains a brief presentation of the two most known *preemptive* scheduling algorithms. We provide then a review of existing results on the use of stochastic processes in real-time systems and a brief review of *static-priority* preemptive scheduling on multiprocessor systems.

Definition 1.1. *Static priority algorithms are such that for any couple of tasks, whenever both are activated simultaneously, the same task always have priority.*

Finally, we conclude this section by discussing the link between Mixed-Criticality (MC) systems and probabilistic approaches.

1.3.1 Deterministic analyses

We understand by testing the schedulability of a system that we assert that by construction the system cannot fail. Several schedulability tests are based on the *utilization* of the system, which is the sum of the ratios between the units of time required by the programs to execute and the time separating two instances of this same program. Other schedulability tests are based on the worst-case reasoning, *i.e.*, finding the scenario producing the largest response times for every program. In the remainder of this thesis, a program is called a *task* and a *job* is an instance of a task.

We say that a scheduling policy is *work-conserving*, *i.e.*, does not idle when there is work to do. Moreover, we say that a scheduling policy is preemptive,

i.e., a job may be preempted before the end of its complete execution, and its execution can be resumed with no cost.

Single processor scheduling: Rate Monotonic and Earliest Deadline First

The Rate Monotonic (RM) scheduling policy is a static-priority scheduling algorithm, *i.e.*, it assigns priorities by task. RM assigns to each task a priority relatively to its rate of occurrence - the larger the rate, the higher the priority.

Theorem 1.1 (Theorem 2 [Liu and Layland, 1973])**.** *RM is optimal for single processor systems with implicit deadlines in the domain of static-priority preemptive scheduling policies.*

The Earliest Deadline First (EDF) algorithm is a deadline-driven scheduling policy, giving a priority to each job relatively to its absolute deadline - the smaller the absolute deadline, the higher the priority. We call dynamic any scheduling policy giving priorities at job boundaries. Once the priority is assigned to one job it does not change until the completion of this job.

Both RM and EDF are the most used scheduling algorithms of their own (static and dynamic) class because of their efficiency. Hence, we chose to restrict the state-of-the-art section to those two algorithms. In [Liu and Layland, 1973], authors prove that in single processor preemptive scheduling, the worst-case scenario occurs in a critical instant, which is the simultaneous activation of all tasks. This reasoning leads to the results in [Joseph and Pandya, 1986], where the response times are proven to satisfy a fixed-point equation. However, those two reasoning hold only for single processor preemptive scheduling.

Other schedulability tests focus on Time Demand Analysis (TDA) of the *critical instant*, which is the analysis of the workload sequence of jobs that produces the largest response time, *i.e.*, the time between the activation and the end of a job. This worst-case reasoning is based on the fact that a system is schedulable if the worst-case is schedulable. This property is called *sustainability* [Baruah and Burns, 2006] (or *predictability* [Ha and Liu, 1994]).

Definition 1.2 (Sustainability [Baruah and Burns, 2006])**.** *A schedulability test for a scheduling policy is sustainable if any system deemed schedulable by the schedulability test remains schedulable when the parameters of one or more individual jobs are changed in any, some, or all of the following ways: decreased execution requirements; later arrival times; and larger relative deadlines.*

Multiprocessor scheduling

The problem of priority-driven scheduling on multiprocessor systems consists in two decision problems: the allocation and the priority assignment of tasks.

(i) Allocation: Jobs are allocated to processors. The allocation decision is a load-balancing problem in which jobs are distributed among multiple processors.

(ii) Priority assignment: Priorities are assigned to tasks or jobs.

To the best of our knowledge, there are few results using probabilistic methods in multiprocessor scheduling. The main results using stochastic analysis are focused on shared resources quantification in multiprocessor systems, which is not what we focus on in this thesis. We present these results in the following, as well as deterministic results that we either extend or use as a baseline for a stochastic analysis.

Homogeneous and heterogeneous multiprocessor systems We divide multiprocessor systems into three different categories based on the speeds of the individual processors.

- homogeneous multiprocessor systems: all processor speeds are the same across all processors. Usually the speed is set to one cycle per unit of time.
- uniform heterogeneous multiprocessor systems: the processing speed depends on the processor. Each processor has its own constant speed. Each processor in an uniform multiprocessor system is characterized by a speed or computing capacity, with the interpretation that a job executing on a processor with speed s for t units of time completes $s \times t$ cycles.

- unrelated heterogeneous multiprocessor systems: processing units have dedicated speeds according to the tasks to be scheduled. We do not cover this case in this thesis.

We cover in this thesis uniform heterogeneous multiprocessor systems that we call for more readability uniform multiprocessor systems.

Multiprocessor static-priority online scheduling *Offline* scheduling algorithms determine decisions before the system starts its execution. These scheduling algorithms select jobs to execute according to predetermined priorities. Usually, schedules are repeated after a specific time period [Cucu and Goossens, 2007], and the specificity of offline scheduling is that one should be able to tell *a priori* from what time this repetition begins.

Online scheduling algorithms select jobs by examining properties of active jobs. They can be more flexible than offline algorithms since they can adapt their decision depending on the state at instant t of the system. For example, the system must use an online scheduling algorithm if the set of tasks includes tasks with unpredictable inter-arrival times that join and leave the system at undetermined times.

For a given class of multiprocessor scheduling algorithms, some are optimal in the sense that if they do not satisfy all deadlines, then no algorithm from this class can. However, satisfying deadlines in theory is not always the main goal in practice. For instance, preemptions have a cost and one may want to optimize this cost.

An important result of online multiprocessor scheduling algorithm is provided in the seminal paper of [Hong and Leung, 1992].

Theorem 1.2 (Theorem 1 in [Hong and Leung, 1992])**.** *There is no optimal online multiprocessor scheduling algorithm for real-time systems with two or more distinct deadlines.*

While optimal algorithms can be built for the multiprocessor scheduling problem under certain conditions, there cannot be one optimal scheduling algorithm for

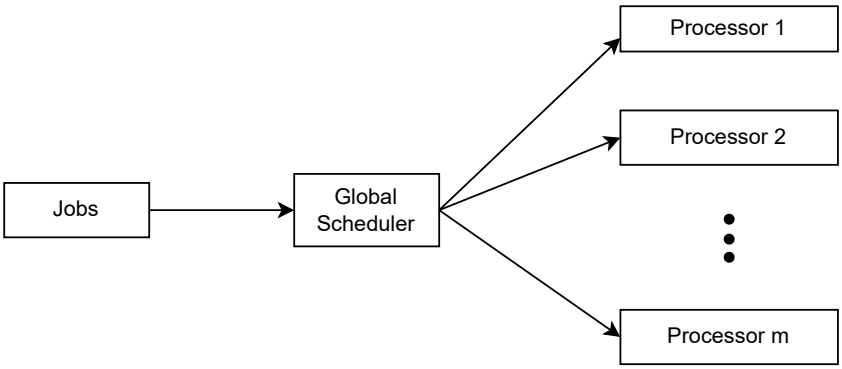


Figure 1.2: Global scheduling policy

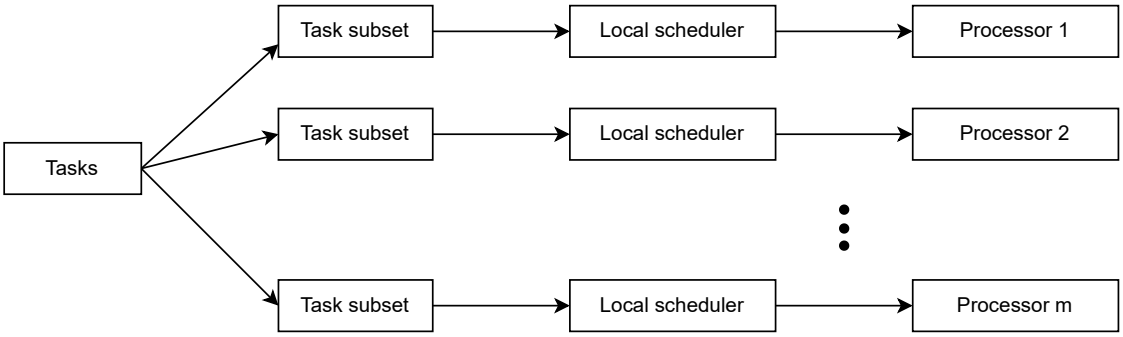


Figure 1.3: Partitioned scheduling policy

different instances of the multiprocessor scheduling problem as strong as RM and EDF are for the single processor case. However, there are some important results that we discuss below. Several multiprocessor scheduling algorithms have been proposed [Funk and Meka, 2009, Funk and Nanadur, 2009, Baruah et al., 1993, Levin et al., 2010, Srinivasan and Anderson, 2005, Anderson et al., 2008, Baruah and Fisher, 2005, Brandenburg and Gül, 2016, Fan and Quan, 2012, Hobbs et al., 2019, Anderson et al., 2016, Kato and Yamasaki, 2009, Bastoni et al., 2011, Anderson et al., 2008].

An important layer of complexity are the migrations and preemptions of jobs between processors. With different levels of migrations (full, restricted or none), we define three main classes of multiprocessor scheduling algorithms [Carpenter et al., 2004]:

- full migration: the scheduling algorithm is global. In global scheduling each task can be executed on any processor in the system. This type of process migration can maximize system utilization and provide an effective load

balance. The main problem that global scheduling encounters is the cost of migrations and preemptions. For instance, in [Andersson and Tovar, 2006] the authors focus on global scheduling with the goal of minimizing preemptions, see Figure 1.2 and [Bertogna et al., 2008, Bertogna and Cirinei, 2007].

- no migrations: the system is partitioned, *i.e.*, each task is assigned to a processor. Partitioned scheduling divides the available computing resources into distinct partitions and assigns tasks to the appropriate partition. While this approach can be useful for organizing workloads, it does not always use the computing resources of a system in an efficient manner. For example, if the workloads assigned to each partition are not evenly balanced, then one partition may become overloaded while other partitions remain idle. Additionally, if a task requires more resources than what is available in its assigned partition, then the task may not be able to complete in a timely manner. Partitioned scheduling also does not allow dynamic load balancing, meaning that the system cannot adjust to variable workloads or resource availability, see Figure 1.3 and [Andersson and Jonsson, 2000, Andersson et al., 2003].

- restricted migrations: tasks are dynamically assigned to processor during run-time. With restricted scheduling each task can be executed on any processor, but each of its jobs has to finish executing its workload on this same processor. Restricted migration can provide a more efficient load balancing while still allowing for some degree of task migration, see Figure 1.4 and [Baruah and Carpenter, 2005, Anderson et al., 2008, Goossens et al., 2012, Brandenburg and Gül, 2016].

1.3.2 Stochastic analyses

As discussed in the previous section, the sustainability [Baruah and Burns, 2006] permits to extend directly deterministic single processor results to the probabilistic approach, because what is valid for the worst-case scenario is proven valid for all

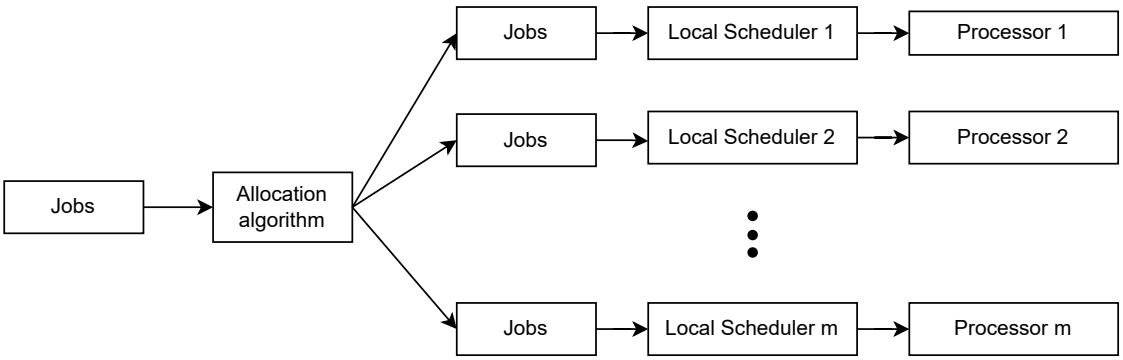


Figure 1.4: Restricted scheduling policy

scenarios. We list and discuss in this section some of the results using statistics and probabilities for the analysis of single and multiprocessor systems. Some of the challenges of the stochastic analysis of real-time systems are detailed in [Quinton et al., 2012].

Single processor scheduling

Statistical timing analysis For a task and a given processor, we call *timing analyses*, the methods determining the largest workload required for the execution of this task on that processor. In [Davis and Cucu-Grosjean, 2019], one can find a survey that details *probabilistic timing analyses* developed until 2018.

In [Edgar and Burns, 2001], the authors present Extreme Value Theory (EVT) as a candidate for estimating WCET and WCRT with random variables (also known as probabilistic WCET and probabilistic WCRT), followed by [Lu et al., 2011, Cucu-Grosjean et al., 2012, Lima et al., 2016, Cazorla et al., 2013, Santinelli et al., 2014, Wartel et al., 2013, Lu et al., 2012], finding the best-fitted parameters for extreme value distributions [Basrak, 2011, Hansen et al., 2009], computing maximum likelihood estimators and non-parametric tests to do so, and finally compute the Deadline Miss Probability (DMP). The EVT method is called a *measurement-based method*, as it infers knowledge on execution times. One may find a complete survey of these results in [Davis and Cucu-Grosjean, 2019], while open problems are underlined in [Gil et al., 2017]. According to these authors, deterministic WCET estimation of a task is often pessimistic, *i.e.*, unlikely to occur and often larger than most

execution times of that task. In fact, EVT has been used in several papers [Lu et al., 2012, Wartel et al., 2013, Liu et al., 2013, Lima et al., 2016, Lima and Bate, 2017]. In [Maxim et al., 2017a] the authors discuss the *reproducibility* of a measurement protocol and the *representativity* of the data used to estimate WCETs. Nevertheless, EVT has some limitations as it is too sensitive to outliers of the provided data. Some other limitations are pointed in [de Barros Vasconcelos and Lima, 2022]. A comparison with the deterministic approach is provided in [Abella et al., 2014].

Probabilistic schedulability tests of single processor systems One of the important part of a processor is the order in which it runs the tasks. There are several ways to schedule a *task set*. Some scheduling algorithms can be developed in order to schedule large sets of tasks, others can be energy optimized.

One may look at [Davis and Burns, 2011] for an exhaustive survey on the scheduling problem on multiprocessor processors. It reviews the first notions to know when we speak about task scheduling as *allocation* or *priority* problems. It also explains all the different notions needed to understand real-time scheduling, as *schedulability*, *comparability*, *predictability* and *sustainability*, and presents the main results from the late 1960's until 2009.

In [Díaz et al., 2002, Kim et al., 2005], the authors calculate the exact distribution of response times of periodic tasks with probabilistic execution times. They also prove that the backlog, *i.e.*, the remaining workload at each beginning of each hyper-period, can be modeled with a Markov chain, and they are the first to consider the computation of the exact deadline miss probabilities from a set of execution time distributions. Finally they prove that the backlog of periodic systems converge to some steady-state distribution and they approximate this distribution. This method can be used to quantify deadline miss probabilities for a given task set in periodic systems. However, the complexity of this proposed solution is exponential, even if some studies solve partially the issue [Marković et al., 2021, Maxim et al., 2012] by subsampling the execution times distributions. This is where analytical approximations make their entry.

Analytical methods for measuring deadline miss probabilities have been studied recently. In [von der Brüggen et al., 2018], the authors are interested in the DMP and overload probability introduced in [Chen and Chen, 2017], and provide both analytical and combinatorial descriptions of how to compute them. Their contribution relies on what they call the *multinomial approach* to compute deadline miss probabilities more efficiently. This work is then extended for EDF in [von der Brüggen et al., 2021]. In [Palopoli et al., 2012, Palopoli et al., 2015, Abeni et al., 2017], authors bring the idea of what they call *probabilistic deadlines* (also known as DMP). Around this concept, they build the backlog analysis introduced in [Díaz et al., 2002, Kim et al., 2005] with a Birth-Death process, and quantify the deadline miss probabilities for a reservation based schedule algorithm for periodic systems. In [Villalba Frias, 2018], the author implemented in his thesis the PROSIT tool, a simulation framework computing deadline miss probabilities for a given task set, with independent and non independent execution times, using the backlog as a Birth-Death process, and an Hidden Markov Model to model the dependencies between execution times.

Scheduling theory is mathematically not that far from queuing theory. In [Lehoczky, 1996, Lehoczky, 1997a], the authors add to queuing theory timing constraints. They use a Markov process to model the *lead-time* profile of all the jobs in the queue. The main result provided in [Lehoczky, 1997b] is that the multivariate queue length process converges to a Brownian network, assuming the *heavy-traffic condition*. In [Doytchinov et al., 2001], the authors apply real-time queueing networks to EDF.

Multiprocessor scheduling

A current trend in real-time systems is the application of statistical learning methods in order to find optimal schedules [Plassart, 2020, Mao et al., 2019, ul Islam and Lin, 2015, Lee et al., 1997, Kadaba et al., 1991]. In scheduling theory, the inference of *scheduling knowledge* [Shaw et al., 1992] is a natural step into the application of the parametric method provided in this thesis. The parametric estimation of

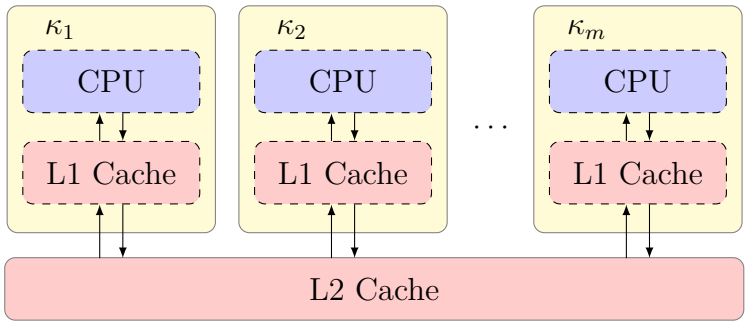


Figure 1.5: Level 2 shared cache

new parameters such as deadlines can improve scheduling decisions, see [Aytug et al., 1994] for a review in this field.

In [Manolache et al., 2002], the authors speak about the heavy-traffic assumption, saying that *it fails yet to handle systems where a task has more than one immediate successor task* and hence leads to high deadline miss ratio, which is unacceptable for critical real-time systems. They use Coxian distributions to model execution times, and use *Stochastic Petri Networks* to describe the dynamic of multiprocessor scheduling. They prove that under the Coxian distribution assumption, there exists a system steady-state.

In [Kim and Shin, 1996], the authors provide a model for execution time distribution including communication between sub-tasks with a queuing model and Markov chains, taking account of blocking time between sub-tasks induced by the *parallelism* of multiprocessor systems using the First-In-First-Out (FIFO) policy. It provides a closed Markov chain representing step-by-step the number of sub-tasks being computed at the same time, with $m + 1$ states, state 0 being the communicating/synchronization state, m being the number of cores available.

Shared cache interference Shared cache interference (see Figure 1.5) is one of the reasons why migrations and preemptions are important metrics in scheduling algorithms performances, as they quickly add overheads into the process. Those overheads are usually ignored, but in multiprocessor scheduling they are crucial.

In [Davis et al., 2013], the authors list different problems encountered in

multiprocessor systems with shared caches. They provide a *static probabilistic timing analysis* method on evict-on-miss random replacement cache policy single-core systems, based on geometric distribution with a missing rate, and give some clues on how to model cache miss latency and memory use in multiprocessor architectures, but with no actual results. It is actually based on the analysis provided in [Altmeyer et al., 2015a], where the authors provide an analysis of how cache misses occur with a random replacement policy in a cache dedicated to one CPU, and express execution times as a sequence of cache accesses, missed or not. WCET are then computed by adding all the delays induced by cache hit/misses.

In [Jalle et al., 2014, Fernandez et al., 2014], the authors go further in the analysis of cache misses and focus on the bus contention. The goal of both papers is to provide a probability of cache miss due to shared hardware. In contrast to [Altmeyer et al., 2015a], where the authors provide an analysis of how cache misses occur with a random replacement policy in a cache dedicated to one processing unit. The authors points have already been partially answered in [Yan and Zhang, 2008] where a non-preemptive model is provided to describe access interference between instructions in a *second-level* shared cache but deterministic.

However, we do not model the shared cache interference in this thesis. Nevertheless, the number of preemptions have an important cost when resources are shared [Phavorin and Richard, 2015]. This cost is a metric used in the simulations presented in Chapter 6, and migrations are restricted within the heuristics proposed for the same reason.

Machine learning in multiprocessor scheduling Techniques that address the multiprocessor scheduling problem are developed in papers like [Nakasuka and Yoshida, 1992, Kadaba et al., 1991, Lee et al., 1997, Ahmad and Dhodhi, 1996, Gupta et al., 2010, Padmajothi et al., 2022]. There are not many studies from real-time researchers using stochastic analysis, nor analytical approximations of the DMP in multiprocessor scheduling allowing inference, and, as we discuss in the last chapter, machine learning.

In Chapter 6 we focus on restricted multiprocessor scheduling algorithms.

1.3.3 Mixed-Criticality

Different values of execution times may correspond to various modes of a real-time system. Detecting modes changes in critical systems can be crucial and contribute to soften timing constraints. Such detection serves a higher-level objective: characterizing a functional mode that may be a normal, exceptional or degraded, in order to increase the reactivity of these systems and to predict mode transitions [Real and Crespo, 2004]. Indeed, by adapting the reaction of the system with respect to a given mode, an optimized utilization of resources is possible, which becomes another commercial trend within the time critical systems industry. Sometime the mode is obvious, such as a drone in a take-off mode for example, but tasks often depend of unobserved latent variables such as environmental variables. The MC model considers worst-case scenarios per mode [Vestal, 2007]. MC models have been widely studied recently [Altmeyer et al., 2015b, Guo et al., 2017, Baruah et al., 2011, Gettings et al., 2015, Baruah et al., 2012, Guo et al., 2015, Burns, 2014]. One may see a review of MC systems in [Burns and Davis, 2017]. We do not use the MC model in this thesis. However, we emphasize the link between probabilistic approaches and MC models as discussed in [Maxim et al., 2017b, von der Brüggen et al., 2022, Abdeddaïm and Maxim, 2017], and how probabilistic approaches can contribute to better define and understand how MC models can be used in practice. The introduction of MC systems in [Vestal, 2007] is concluded in those terms:

> An interesting theoretical question we encountered was: What is a good multi-criticality utilization metric? We considered computing a vector of utilizations (one per design assurance level), computing a utilization using for each task the compute time associated with its own criticality, and computing a utilization using the compute time associated with the highest criticality of any task of equal or lower priority. A vaguely troublesome property of all these metrics is that some workloads may be feasibly scheduled at higher than 100% utilization.

This is exactly the problem probabilistic approaches solve: what can we say when the maximal utilization is higher than 100% ? The MC model is a theoretical

model that is based on several levels of criticality (*e.g.*, high or low), which are different modes of the system that can provide different execution times of tasks. According to those modes, different quantities such as response times or deadline miss probabilities are computed. In [Draskovic et al., 2021] for example, authors provide the deadline miss probabilities for the MC model with the probabilistic approach and show the link there can be between the MC model [Vestal, 2007] and the stochastic analysis of real-time systems. However, as discussed in [Esper et al., 2015], certifications of real-time systems look for levels of confidence, that is to say a quantification of the failure rate of a given function (*i.e.*, the real-time task, the sensors it uses and the actuators it may communicate with) of a system. Furthermore, computing deadline miss probabilities is necessary, but to the best of our knowledge, the state-of-the-art results do not provide the appropriate granularity, as the certification requirements are placed at functionalities level. Sensors and actuators (the hardware part) is often not taken into account. In this perspective, a model allowing the inference of data from sensors and actuator to consider a level of confidence of the functionality of a system would be an improvement. In that regard, probabilities may play a crucial role but the lack of probabilistic models allowing such inference makes it not yet possible. Numerous papers try to generalize deterministic results to the probabilistic approach without succeeding to redefine the domain of application of such methods, which prevents the industry to actually trust such analyses. We use the MC model as our main motivation to define properly the domain of applicability of stochastic analyses.

1.4 Reader guidelines

In this thesis, we aim to approximate response times of real-time systems and provide statistical analyses for adaptive scheduling decisions. We propose a method for estimating the response time distribution and its *a priori* distribution. Finally, we apply this method by introducing a new class of allocation algorithms that we refer to as DMP-driven. The manuscript is composed of three main parts:

(i) Approximation: chapters 2, 3 and 4,

a. in chapter 2, the mathematical model used to describe real-time systems is introduced,

b. in chapter 3, the necessary conditions for the statistical analysis are built,

c. in chapter 4, real-time systems are described with queueing theory results,

(ii) Estimation: chapter 5, the statistical analysis itself: a parametric estimation adapted for response times,

(iii) Optimization: chapters 6 and 7,

a. in chapter 6, an online deadline miss probability-driven processor allocation,

b. in chapter 7, future potential work for more optimization.

The dependencies between chapters are illustrated in Figure 1.6.

1.5 Publications

The contributions presented in this thesis are published or under submission within the following four papers:

(i) *Identification of Execution Modes using Cluster Analysis.* Kevin Zagalo, Liliana Cucu-Grosjean and Avner Bar-Hen, *ETFA 2020*, [Zagalo et al., 2020],

(ii) *Response Time Stochastic Analysis for Stable Fixed-Priority Real-Time Systems.* Kevin Zagalo, Yasmina Abdeddaïm, Avner Bar-Hen and Liliana Cucu-Grosjean, *IEEE Transactions on Computers, Special issue on Real-time Systems, 2023*, [Zagalo et al., 2022a],

(iii) *Response Time Parametric Estimation of Real-Time Systems.* Kevin Zagalo, Olena Verbytska and Avner-Bar-Hen, *Arxiv Preprint, 2022*, [Zagalo et al., 2022b],

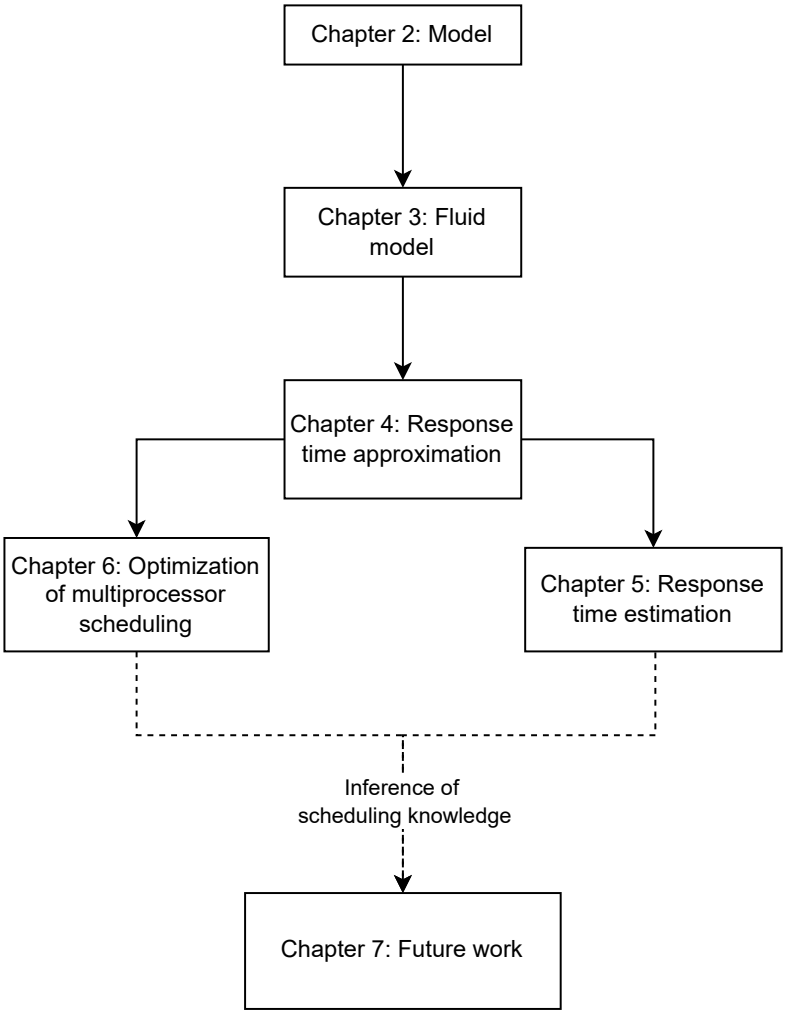


Figure 1.6: Manuscript structure and reading dependencies

1.6 Software

Numerical evaluations are implemented within three frameworks:

(i) Probabilistic SimSo: an adapted version of SimSo using probabilistic execution times [Zagalo and Auvray, 2022].

(ii) *rInverseGaussian*: a Python library for the re-parameterized inverse Gaussian distribution [Zagalo and Verbytska, 2022].

(iii) *Hypoexponential*: a Python library for the hypoexponential distribution [Zagalo, 2022].

2 | Stationary Real-Time Systems

Contents

Probabilistic approaches for real-time systems are based on the fact that the worst-case execution times and minimal inter-arrival times can be weighted by their probability of occurrence. The analysis induced by these approaches consists in

quantifying the probability that timing constraints may not be satisfied: the goal being to soften the description of *timing variables* as considering worst-case values may lead to an over-dimensioning of the processor. This over-dimensioning brings pessimism that may be measured as defined in [Díaz et al., 2004].

Utilization-based conditions of schedulability are well known. Indeed, the seminal work of Liu and Layland [Liu and Layland, 1973] introduces a sufficient condition for the feasibility of a real-time system using its *maximal utilization*. However, in probabilistic real-time systems, each value of execution times and inter-arrival times are weighted by a probability. Furthermore, the deterministic critical instant defined in [Liu and Layland, 1973], *i.e.*, the worst-case scenario in single processor, cannot be extended to the probabilistic case directly [Chen et al., 2022].

In this chapter, we introduce a new type of real-time tasks that we call *stationary*.

2.1 Probabilistic Real-Time Systems

In this first section we formalize the use of the probability theory in real-time systems and define the theoretical notions used by many studies.

2.1.1 Environment and probability space

Let Q be the set of possible states of the hardware, I the set of all possible input values of tasks. As explained in [Axer et al., 2014], Q and I are usually unknown and/or too large, thus

> Timing predictability [...] is incomputable for most systems. So, it is not possible to construct a general procedure that, given a system, computes its predictability exactly. [...] However, it is possible to develop procedures that compute approximations, that is, upper and/or lower bounds on a system's predictability.

We call the product space $\Omega_0 = Q \times I$ the *environment space*, representing all possible values of all inputs of the tasks, the exact state of the processor, or any information that varies during the execution of tasks, the physical properties in

which the system interacts. We suppose Ω_0 finite, *i.e.*, $|\Omega_0| < \infty$. For example, let $C(q, i)$ be the execution time of a task computed in an environment $(q, i) \in \Omega_0$. In a perfect world, by knowing all possible states of a system, the system would be time predictable, in the sense that we can in fact predict the worst-case environment in Ω_0 for a given task. In [Axer et al., 2014], authors define the *timing predictability* of a task with

$$\Pr(\Omega_0) = \min_{q_1,q_2 \in Q} \min_{i_1,i_2 \in I} \frac{C(q_1, i_1)}{C(q_2, i_2)} \tag{2.1}$$

which measures the ratio between the minimum and the maximum execution time of this task. A concept similar to the timing predictability can be measured by enumerating the subsets of Ω_0 that satisfy the occurrence of specific *events*. Let us call any subset $A \subset \Omega_0$ an event, and $\mathcal{A}$ the set of all possible events, such that the empty space $\varnothing \in \mathcal{A}$ and $\Omega_0 \in \mathcal{A}$.

Finally, let $\mathbf{P}$ be a *probability measure* on the environment space Ω_0, *i.e.*, a function mapping $\mathcal{A}$ to $[0, 1]$ such that

(i) $\mathbf{P}(\Omega_0) = 1$,

(ii) $\mathbf{P}$ is σ-additive, *i.e.*, for any collection $(A_n)_n$ of pairwise disjoint events in $\mathcal{A}$, $\mathbf{P}\left(\bigcup_n A_n\right) = \sum_n \mathbf{P}(A_n)$.

The *uniform probability* $\mathbf{P}$ of an event $A \in \mathcal{A}$ is the frequency of occurrence of this events in Ω defined by

$$\mathbf{P}(A) = \frac{|A|}{|\Omega_0|} \tag{2.2}$$

For example, let $c \in \mathbb{N}$. If $A = \{(q, i) \in \Omega_0 : C(q, i) = c\}$ its associated probability according to $\mathbf{P}$ is the frequency of occurrence of the equality $C(q, i) = c$ in Ω_0, *i.e.*, $\mathbf{P}(A) = \frac{\sum_{q \in Q} \sum_{i \in I} \mathbf{1}_{\{c\}}(C(q,i))}{|Q| \times |I|}$ where

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2.1 (Probability space)**.** *Let Ω_0 be an environment space, $\mathcal{A}$ the sets of all events of Ω_0 and $\mathbf{P}$ the uniform probability measure on $\mathcal{A}$ defined in Eq.* (2.2)*. We call $(\Omega_0, \mathcal{A}, \mathbf{P})$ a probability space.*

In the remainder of this thesis, Q and I are considered unknown.

Definition 2.2. *Let $(\Omega_0, \mathcal{A}, \mathbf{P})$ be a probability space and $A \in \mathcal{A}$. We say A holds if $\mathbf{P}(A) = 1$.*

The application of this definition is as follow: every equality $X = Y$ (resp. inequality $X \leqslant Y$) is defined by $P(X = Y) = 1$ (resp. $\mathbf{P}(X \leqslant Y) = 1$).

2.1.2 Timing variables

A task is instantiated at a given time, for a given sequence of environments. In this section we provide a formal definition of the instance of a task, and its associated execution time, inter-arrival time and deadline.

Tasks and jobs

We call *task* a tuple of timing variables $\tau = (X_1, \dots, X_n)$ mapping the environment set Ω_0 to the product space $\mathcal{J} \subset \mathbb{R}_+^n$. We call *job* an instance of a task τassociated to an environment ω_0 denoted $\tau(\omega_0)$, such that a task τ is the function

$$\tau : \omega_0 \in \Omega_0 \to \tau(\omega_0) \in \mathcal{J}$$

where $\mathcal{J}$ is the set of all jobs. We say that a job is *released* when its associated task is *activated*. When a job is released, several variables are released with it. Let $\Gamma = \{\tau_1, \tau_2, \dots\}$ be a *task set*.

Let X be a mapping from Ω_0 to $\mathbb{R}_+$, and X^{-1} be the inverse image of X in Ω_0.

Let $\mathcal{B}_X = \{X^{-1}(A) : A \in \mathcal{A}\}$. We define the *probability distribution* of X

$$\begin{aligned}\mathbf{P}_X(A) &= \mathbf{P} \circ X^{-1}(A) \\ &= \mathbf{P}\left(X^{-1}(A)\right) \\ &= \mathbf{P}(\omega_0 \in \Omega_0 : X(\omega_0) \in A)\end{aligned}$$

for $A \in \mathcal{B}_X$.

Lemma 2.1. $(\mathbb{R}_+, \mathcal{B}_X, \mathbf{P}_X)$ *is a probability space.*

Proof. First, $X^{-1}(\mathbb{R}_+) \in \mathcal{B}_X$, and $\mathbf{P}_X(\mathbb{R}_+) = 1$. Let $(B_p)_p$ be a sequence of pairwise distinct elements of $\mathcal{B}_X$. Let $p \neq q$ and $\omega_0 \in X^{-1}(B_p) \cap X^{-1}(B_q)$. Then $X(\omega_0) \in B_p \cap B_q = \varnothing$. Hence the $(X^{-1}(B_p))_p$ are pairwise disjoints events of $\mathbb{R}_+$, and thus $\mathbf{P}_X(\cup_p B_p) = \mathbf{P}(\cup_p X^{-1}(B_p)) = \sum_p \mathbf{P}_X(B_p)$. □

We call *timing variables* the variables characterizing tasks.

Definition 2.3 (Timing variable)**.** *A timing variable on* $(\Omega_0, \mathcal{A})$ *is a mapping* $X : (\Omega_0, \mathcal{A}) \to (\mathbb{R}_+, \mathcal{B})$ *such that for each* $B \in \mathcal{B}$ *there exist* $A \in \mathcal{A}$ *such that* $B = X(A)$.

Let X be a timing variable. We define the *distribution function* of X as

$$\begin{aligned}F_X(x) &= \mathbf{P}_X((-\infty, x]) \\ &= \mathbf{P}(\omega_0 \in \Omega_0 : X(\omega_0) \leqslant x)\end{aligned}$$

also called *cumulative distribution function* and is such that $F_X(+\infty) = 1$ and $F_X(-\infty) = 0$. The function

$$1 - F_X$$

is called the *exceedence function* of X.

Execution times

The execution time of a job is the amount of workload (or processor cycles) it needs to finish. For a given environment $\omega_0 \in \Omega_0$, and a task $\tau_i \in \Gamma$. Let C_i be a timing variable such that the job $\tau_i(\omega_0)$ has an execution time $C_i(\omega_0)$. We consider through this work that execution times are bounded, *i.e.*, for all $\tau_i \in \Gamma$, there exist c_i^{min} and c_i^{max} such that

$$\forall \omega_0 \in \Omega_0, C_i(\omega_0) \in \left[c_i^{min}, c_i^{max}\right]$$

We denote by F_i the distribution function of C_i, and its mean value by $\mathbf{E}[C_i]$ and its second order moment $\mathbf{E}\left[C_i^2\right]$.

Inter-arrival times

For a given environment $\omega_0 \in \Omega_0$, let T_i be a timing variable such that the inter-arrival time $T_i(\omega_0)$ is the elapsed time between the *release* (or *activation*) of the job $\tau_i(\omega_0)$ and the release of the previous one. We denote by G_i the distribution function of T_i. We call $\lambda_i = \mathbf{E}[T_i]^{-1}$ the *rate* of τ_i.

Deadlines

For a given environment $\omega_0 \in \Omega_0$, Let D_i be a timing variable such that the relative deadline $D_i(\omega_0)$ is the time given to the job $\tau_i(\omega_0)$ to execute. We say that the relative deadline D_i is

- *constrained* if any job of τ_i must complete its execution before the release of the next job of the same task in a deterministic fashion, *i.e.*, there exists $0 < d_i$ such that $\forall \omega_0 \in \Omega_0, D_i(\omega_0) = d_i \leqslant T_i(\omega_0)$,
- *implicit* if it is equal to the inter-arrival time between a job and its next job of the same task,
- *arbitrary* if there is no relation with inter-arrival times.

2.1.3 Properties of timing variables

The expectation operator $\mathbf{E}$ of a probability measure $\mathbf{P}$ is defined by

$$\mathbf{E}[g(X)] = \int_{\Omega_0} g(X(\omega_0))\mathbf{P}(d\omega_0) \tag{2.3}$$

for any function $g : \mathbb{R}_+ \to \mathbb{R}$. In order to estimate the distribution of a timing variable, we need to express (2.3) in terms of its possible values instead of the possible values in the hardware states in Q and inputs in I.

Lemma 2.2 (Transfer theorem)**.** *Let X be a timing variable and $g : \mathbb{R}_+ \to \mathbb{R}$. Then*

$$\mathbf{E}[g(X)] = \int_{\mathbb{R}_+} g(x)\mathbf{P}_X(dx)$$

Proof. Take $x = X(\omega_0)$ in Eq. (2.3), thus $d\omega_0 = X^{-1}(dx)$. Furthermore, by definition $X(\Omega_0) \in \mathcal{B}_X$ because $\Omega_0 \in \mathcal{A}$. □

We use the notation $\mathbf{P}_X(dx) = dF_X(x)$. A consequence of this last lemma is that for any $A \in \mathcal{B}_X$,

$$\mathbf{P}_X(A) = \int_{\mathbb{R}_+} \mathbf{1}_A(x)\mathbf{P}_X(dx)$$

Definition 2.4 (Independence)**.** *Let X and Y be two timing variables. We say that X and Y are independent if and only if $\forall B_1 \in \mathcal{B}_X, \forall B_2 \in \mathcal{B}_Y$,*

$$\mathbf{P}(\omega_0 \in \Omega_0 : X(\omega_0) \in B_1, Y(\omega_0) \in B_2) = \mathbf{P}_X(B_1)\mathbf{P}_Y(B_2)$$

Definition 2.5 (Identically distributed)**.** *Let X and Y be two timing variables. We say that X and Y are identically distributed if and only if $\forall B \in \mathcal{B}_X \cup \mathcal{B}_Y$,*

$$\mathbf{P}_X(B) = \mathbf{P}_Y(B)$$

and equivalently, $\forall x \in \mathbb{R}_+$,

$$F_X(x) = F_Y(x)$$

and we write $X \overset{(d)}{=} Y$ *or equivalently* $X \sim dF_Y$. *Particularly* $X \sim dF_X$.

Definition 2.6 (Conditional probability)**.** *Let* X *and* Y *be two timing variables. The conditional timing variable* X *given* $Y = y$ *is written* $X|Y = y$ *and is such that*

$$\mathbf{P}_X(A) = \int \mathbf{P}(X \in A|Y = y)dF_Y(y)$$

for all $x > 0$.

Definition 2.7 (Convolutions)**.** *Let* X *and* Y *be independent timing variables and* $Z = X + Y$. *Then the distribution function of* Z *is the convolution* $F_Z = F_X * F_Y$ *defined by*

$$F_Z(z) = \int F_X(z - y)dF_Y(y) = \int F_Y(z - x)dF_X(x)$$

2.1.4 Common distributions

We define in this section the distributions used in the remainder of this thesis.

Definition 2.8 (Uniform distribution)**.** *The uniform distribution function on the interval* $[0, t]$ *is defined by*

$$G(x) = \frac{\min(x, t)}{t}, x \geqslant 0$$

Definition 2.9 (Exponential distribution)**.** *The exponential distribution of mean* $1/\lambda$ *is characterized by the distribution function*

$$G(x) = 1 - e^{-\lambda x}, x \geqslant 0$$

Definition 2.10 (Standard Gaussian distribution)**.** *The standard Gaussian distri-*

bution is characterized by the distribution function Φ *defined by*

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-y^2/2\right) dy, x \in \mathbb{R} \tag{2.4}$$

Definition 2.11 (Gaussian distribution)**.** *The Gaussian distribution of mean* $m \in \mathbb{R}$ *and variance* $v^2 > 0$ *is characterized by the distribution function* $\Phi_{m,v^2}(x) = \Phi(\frac{x-m}{v})$.

Definition 2.12 (Inverse Gaussian distribution)**.** *Let* $\mu > 0$ *and* $\lambda > 0$. *The inverse Gaussian distribution of mean* μ *and shape* λ *has a probability function* ψ *defined by*

$$\psi\left(t; \mu, \lambda\right) = \sqrt{\frac{\lambda}{2\pi t^3}} \exp\left(-\frac{\lambda(t-\mu)^2}{2\mu^2 t}\right) \tag{2.5}$$

for $t > 0$. *Its mean is* μ *and variance* μ^3/λ. *Its cumulative distribution function is given by*

$$\Psi(t; \mu, \lambda) = \Phi\left(\sqrt{\frac{\lambda}{t}}\left(\frac{t}{\mu} - 1\right)\right) + \exp\left(\frac{2\lambda}{\mu}\right) \Phi\left(-\sqrt{\frac{\lambda}{t}}\left(\frac{t}{\mu} + 1\right)\right)$$

where Φ *is defined in* (2.4).

The inverse Gaussian family is a natural choice for a statistical modelling of positive and right-skewed distributions, see [Folks and Chhikara, 1978, Tweedie, 1957]. It is used in many fields, such as industrial degradation modelling [Ye and Chen, 2014], psychology [Schwarz, 2001, Palmer et al., 2011], and many others like hydrology, market research, biology, ecology, and so on *c.f.*, [Seshadri, 2012].

The tuple (C_i, T_i, D_i) varies depending on which environments τ_i is activated. This means that when a job $\tau_i(\omega_0)$ is released, it is after $T_i(\omega_0)$ units of time after the previous job of τ_i. Its execution time is $C_i(\omega_0)$ and it should be over before $D_i(\omega_0)$ units of time.

In the remainder of this thesis, we consider implicit deadlines.

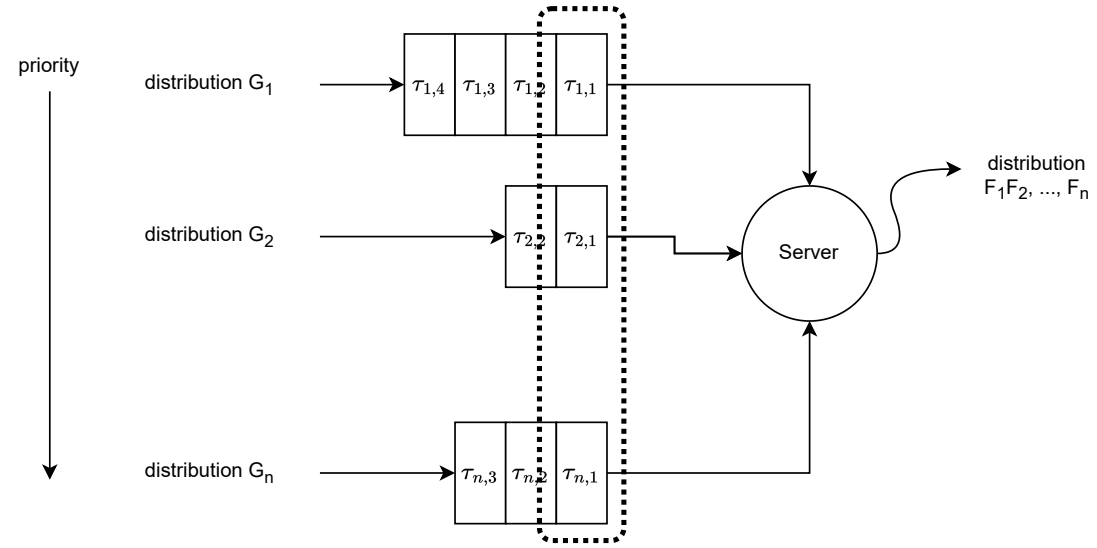


Figure 2.1: Scheme of a $\sum_i G_i / \sum_i G_i / 1 / SP$ system. The sorted queue (by priority) is the queue wrapped with dots

2.1.5 Kendall's notation

In queueing theory, Kendall's notation [Kendall, 1953] is the standard notation to describe a queueing system, *i.e.*, arrival and service times, processing units and scheduling policies. In our case, jobs arrive at a certain rate, and leave the system after the processor treats their execution time. So the general standard model for single processor scheduling is written $G/G/1$ and when written like this, the first G stands for *general* inter-arrival time distributions, the second G for general execution time distribution, the 1 refers to the single processor and implicitly we suppose that FIFO is used to schedule jobs. In the remainder of this thesis,

- D stands for deterministic,
- G for general,
- M for memoryless.

For example, a $D/D/1$ queue is a periodic system with only one value of execution time. It corresponds to the periodic worst-case analysis widely studied in real-time systems.

Each task generates its own queue. As we work under the assumption of memoryless inter-arrival times, each task is associated to a $M/G/1$ queue. Furthermore, as we work under a static-priority scheduling policy, and exponential inter-arrival distributions with several tasks (or priority classes), we write the model $\sum_i M_i/\sum_i G_i/1/\mathrm{SP}$, which is described on Figure 2.1.

2.2 Stationarity

Let $T > 0$ and $0 = t_0 < t_1 < t_2 < \cdots < t_k = T$ and the finite sequence of environments $\omega = (\omega_{t_0}, \ldots, \omega_{t_k})$. More generally, we define the space $\Omega_T = (\Omega_0)^{[0,T]}$ as the space of all functions mapping the interval of time $[0, T]$ to the environment space Ω_0, such that $\omega \in \Omega_T$ can be written $\omega = (\omega_t, t \in [0, T])$. When T goes to infinity, we denote $\Omega = (\Omega_0)^{\mathbb{R}_+}$.

We call *canonical process* the operator $\theta_t : \Omega \to \Omega_0$ such that

$$\theta_t(\omega) = \omega_t$$

is the projection of a sequence of environments to the associated environment at time $t > 0$. It makes the link between processes evaluated on sequences of environments and timing variables evaluated on given environments.

2.2.1 Task model

We consider a finite *task set* $\Gamma = \{\tau_1, \ldots, \tau_n\}$ of n tasks. A task τ_i is characterized by:

- $C_i > 0$ its *execution time*,
- $O_i \geqslant 0$ its *offset*, the time of its first activation,
- $T_i > 0$ its *inter-arrival time*,
- $D_i > 0$, its *relative deadline*,

- $\alpha_i \in [0, 1)$, its *permitted failure rate.*

C_i, O_i, T_i and D_i have their distributions, as α_i is deterministic. The task set Γ is ordered by decreasing priority order, *i.e.*, τ_i has priority over τ_{i+1}.

Arrival times Let $\tau_i \in \Gamma$ and $\omega^{(i)} = (\omega_0^{(i)}, \omega_{t_1}^{(i)}, \dots) \in \Omega$ be a sequence of environments. We call *arrival time* of the j-th activation of τ_i the variable defined by

$$A_{i,j}(\omega^{(i)}) = O_i(\omega_0^{(i)}) + \sum_{k=1}^{j-1} T_i(\omega_{t_k}^{(i)}) \tag{2.6}$$

such that $A_{i,1}(\omega^{(i)}) = O_i(\omega_0^{(i)})$ is the first activation of τ_i. Arrival times have the property of satisfying the relation $A_{i,j}(\omega^{(i)}) - A_{i,j-1}(\omega^{(i)}) = T_i(\omega_{t_j}^{(i)})$.

Job sequences The j-th job of the task τ_i is denoted $\tau_{i,j}$ and defined by

$$\tau_{i,j} = \tau_i \circ \theta_{A_{i,j}}$$

such that the job $\tau_{i,j}$ is mapping Ω to $\mathcal{J}$. Its execution time is

$$C_{i,j} = C_i \circ \theta_{A_{i,j}}$$

and the inter-arrival time between the jobs $\tau_{i,j-1}$ and $\tau_{i,j}$ is

$$T_{i,j} = T_i \circ \theta_{A_{i,j-1}}$$

There are two types of inter-arrival times:

(i) stationary: A task $\tau_i \in \Gamma$ is said *stationary* if the sequence $T_{i,j}, j \in \mathbb{N}$ is identically distributed with distribution function G_i. There are three subclasses widely studied of stationary inter-arrival times:

- periodic: A task $\tau_i \in \Gamma$ is said *periodic* if T_i is deterministic, *i.e.*, there exists $t_i > 0$ such that $G_i(x) = \mathbf{1}_{[t_i \infty)}$. In that case, we call T_i the period

of the task τ_i. A periodic task is stationary. In real-time systems, it models time-triggered programs, *i.e.*, each t_i units of time, the task τ_i is activated.

- memoryless: Let T be an inter-arrival time. T is said memoryless if $\mathbf{P}(T > t + s \mid T > s) = \mathbf{P}(T > t)$, which can be rewritten as the differential equation $f(t + s) = f(t)f(s), \forall t, s > 0$. The solution to this equation belongs to the family of exponential functions. In this case, $G_i(t) = 1 - e^{-\lambda_i t}$.
- sporadic: A stationary task τ_i is said *sporadic* if there exists a bound t_i^{min} such the inter-arrival time $T_{i,j}, j \in \mathbb{N}$ is greater than t_i^{min}. We call

$$\lambda_i^{max} = 1/t_i^{min}$$

the *maximum rate* of τ_i.

(ii) non-stationary: A task $\tau_i \in \Gamma$ is said *non-stationary* if there exists a positive function Λ such that a job released at the instant $t > 0$ is released with a rate $\Lambda(t)$. Thus, the sequence $T_{i,j}, j \in \mathbb{N}$ is not identically distributed and their distributions depend on the release time of their associated jobs. There are two sub-classes of non-stationary inter-arrival times:

- sporadic: A non-stationary task τ_i is said *sporadic* if there exists a bound t_i^{min} such the inter-arrival time $T_{i,j}, j \in \mathbb{N}$ are greater than t_i^{min}. Without more information on the individual distributions of $T_{i,j}, j \in \mathbb{N}$, the analyses of sporadic non-stationary tasks can be done only in the *worst-case* where they are bounded by periodic inter-arrival times of rate λ_i^{max}. This leads to suppose periodic inter-arrival times with $G_i^{max}(t) = \mathbf{1}_{[t_i^{min},\infty)}(t)$.
- aperiodic: If inter-arrival times of a given task are non-stationary and not sporadic.

Equivalently, we may write Eq. (2.6) as $A_{i,j} = O_i \circ \theta_0 + \sum_{k=1}^{j-1} T_{i,k}$.

Finally, the relative deadline of the job $\tau_{i,j}$ is the inter-arrival time between $\tau_{i,j}$ and $\tau_{i,j+1}$ *i.e.*,

$$D_{i,j} = T_{i,j+1}$$

because we consider that deadlines are implicit. A consequence of this is that the deadline D_i is with distribution function G_i.

The canonical process $\{\theta_t\}_t$ permits to write timing variables for any sequence of environments, without specifying the environment they are evaluated on.

2.2.2 Renewal theory

Let $A_{i,j}, i = 1, \ldots, n, j \in \mathbb{N}$ be a sequence of arrival times of the tasks in Γ as previously defined, *i.e.*, with $O_i = A_{i,1}$ called its *offset* and $A_{i,j+1} - A_{i,j} \sim dG_i$ for all $j \geqslant 1$. We define the *renewal process* N_i

$$N_i(t) = \sum_{j=1}^{\infty} \mathbf{1}_{[0,t]}(A_{i,j}) \tag{2.7}$$

as the number of jobs of τ_i released before the instant t. We call

$$\lambda_i = \mathbf{E}[N_i(1)]$$

the *intensity* of N_i.

Definition 2.13 (Stationary renewal process)**.** *A renewal process N is said stationary if $N(t+s) - N(s) \overset{(d)}{=} N(t)$ for any $t, s > 0$.*

Theorem 2.1 (Section 1.4 [Sigman, 2009], [Sigman, 2006])**.** *A renewal process N with an inter-arrival time distribution G of mean λ^{-1} is stationary when its offset O is distributed with*

$$G^0(x) = \lambda \int_0^x (1 - G(y))dy \tag{2.8}$$

where G^0 is called the recurrence distribution of the renewal process N.

Definition 2.14 (Stationary queueing model)**.** *We call a* $\sum_i G_i / \sum_i G_i / 1 / SP$ *queueing model stationary when its offsets* $O_i, i = 1, \ldots, n$ *satisfy* (2.8)*.*

Corollary 2.1 (Periodic arrivals)**.** *In the deterministic case, i.e.,* $\mathbf{P}(T_i = \lambda_i^{-1}) = 1$*, if* O_i *has a uniform distribution between* 0 *and* λ_i^{-1}*, i.e.,* $G_i^0(x) = \lambda_i \min(x, \lambda_i^{-1})$*,* N_i *is stationary.*

Proof. According to Theorem 2.1, in order for N_i to be stationary, is that the distritbution of O_i is $G_i^0(x) = \lambda_i \int_0^x (1 - G_i(y))dy$. In a periodic system, we have $1 - G_i(x) = \mathbf{1}_{[0,\lambda_i^{-1})}(y)$. However,

$$\begin{aligned} G_i^0(x) &= \lambda_i \int_0^x \mathbf{1}_{[0,\lambda_i^{-1}]}(y)dy \\ &= \begin{cases} \lambda_i \int_0^x 1dy & if \quad x \leqslant \lambda_i^{-1} \\ \lambda_i \int_0^{\lambda_i^{-1}} 1dy + \lambda_i \int_{\lambda_i^{-1}}^x 0dy & if \quad x > \lambda_i^{-1} \end{cases} \\ &= \begin{cases} \lambda_i x & if \quad x \leqslant \lambda_i^{-1} \\ 1 & if \quad x > \lambda_i^{-1} \end{cases} \\ &= \lambda_i \min(x, \lambda_i^{-1}) \end{aligned}$$

which is the uniform distribution function according to Definition 2.8 □

Theorem 2.2 (p.394 [Stirzaker and Grimmett, 1992])**.** *Let* N *be a stationary renewal process of intensity* λ*. Then,*

$$\frac{N(t)}{t} \underset{t \to \infty}{\to} \lambda \tag{2.9}$$

Poisson process

The increments $N(t+s) - N(s)$ of a stationary renewal process N only depend on the value of $t > 0$, where $t > s > 0$. However, any inter-arrival time is dependent on the history of the process, *i.e.*, all the information about past arrival times. Only one family of renewal processes is memoryless, *i.e.*, not dependent on the past: the Poisson processes.

Definition 2.15 (Poisson point process)**.** *Let N be a stationary renewal process of intensity λ. N is called a Poisson point process if $\mathbf{E}[N(t)] = \lambda t$, and we have*

$$P(N(t) = k) = e^{-\lambda}\frac{(\lambda t)^k}{k!}$$

i.e., $N(t)$ is a Poisson variable of mean λt. Furthermore, the inter-arrival times are exponentially distributed with mean $1/\lambda$. We say that λ is the intensity of N.

Corollary 2.2. *Poisson point processes are stationary.*

Proof. According to Theorem 2.1, in order for a renewal process N of inter-arrival distribution G to be stationary, the distritbution of its offset is $G^0(x) = \lambda\int_0^x(1 - G(y))dy$. However in the case of a Poisson process of intensity λ, $G(x) = 1 - e^{-\lambda x}$, hence

$$G^0(x) = \lambda\int_0^x e^{-\lambda y}dy = 1 - e^{-\lambda x}$$

which means that the offset is distributed as all inter-arrival times. □

Remark. *Poisson point processes are the only memoryless renewal processes, i.e., the probability that at a time t, the next arrival is at time $t + s$ only depends on s. In other words, the arrival times probabilities do not depend on the past at any instant t. We classify exponential inter-arrival in the stationary context as memoryless inter-arrival times. Indeed, the exponential distribution is the only distribution having the property of not depending on the arrival times of jobs: for any renewal process, the probability that the inter-arrival time T of a job exceeds a value $t + s$ given that an interval of time of length s as past since the arrival of this job, i.e., $\mathbf{P}(T > t + s \mid T > s)$ depends on t and s, except of the exponential distribution where $\mathbf{P}(T > t + s \mid T > s) = \mathbf{P}(T > s)$. This is why this property is called memoryless: it does not need to know anything from the past. In general, this is not the case, and we discuss the general case in the end of the next chapter.*

Poisson processes are renewal processes with many interesting properties that we use throughout this thesis.

Theorem 2.3 (Superposition, Section 1.4.2 [Baccelli and Brémaud, 2013])**.** *Let N_1 and N_2 be two independent Poisson point processes of respective intensities λ_1 and λ_2. Then $N_1 + N_2$ is a Poisson process of intensity $\lambda_1 + \lambda_2$.*

Proof. Since N_1 and N_2 are both stationary, the stationarity of $N_1 + N_2$ comes from the fact that its jumps are also i.i.d. For this, we need to show that $\min(O_1, O_2)$ is exponentially distributed with parameter $\lambda_1 + \lambda_2$. We have

$$\begin{aligned}\mathbf{P}(\min(O_1, O_2) > t) &= \mathbf{P}(O_1 > t, O_2 > t) \\ &= \mathbf{P}(O_1 > t)\mathbf{P}(O_2 > t) && (O_i\text{'s are independent}) \\ &= e^{-\lambda_1 t} e^{-\lambda_2 t}\end{aligned}$$

then we conclude with Corollary 2.2,*i.e.*, the fact that $O_i \overset{(d)}{=} T_i$. □

Lemma 2.3 (Marked Poisson process)**.** *Let N be a Poisson point process counting the arrivals of the jobs of n different tasks respectively arriving at a rate $\lambda_1, \ldots, \lambda_n$. Let $\bar{I}_{k,l}$ be the index of the task of the l-th job of level k, i.e., the job of arrival time $\bar{A}_{k,l} = \inf\{t > 0 : \sum_{i=1}^{k} N_i(t) = l\}$. Then*

$$\mathbf{P}(\bar{I}_{k,l} = i) = \frac{\lambda_i}{\sum_{i=1}^{k} \lambda_i} \tag{2.10}$$

In other words, the probability that this job is from task $\tau_i, i = 1, \ldots, n$ is $\lambda_i / \sum_{i=1}^{k} \lambda_i$.

Proof. The $\bar{I}_{k,l}$ are independent from the $C_{k,l}, T_{k,l}$ and *a fortiori* of $A_{k,l}$. Hence, we can use [Baccelli and Brémaud, 2013, Remark 1.4.2.] for each level of priority k which gives us immediately (2.10). □

Definition 2.16 (Poisson arrival)**.** *Let Γ be a stationary task set. We say that Γ has Poisson arrivals if the inter-arrival times of tasks are exponentially distributed.*

Definition 2.17 (Stationary Real-time System)**.** *Let $(\Omega, \mathcal{A}, \mathbf{P})$ a probability space, and $\{\theta_t\}$ be a sequence of projections on $(\Omega, \mathcal{A}, \mathbf{P})$. Let Γ be a task set such that all*

the couples $(C_{i,j}, T_{i,j}), j \in \mathbb{N}$ *are independent and identically distributed (i.i.d.), and* $(O_i, i = 1, \ldots, n)$ *satify* (2.8). *We call* $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ *a stationary real-time system. We say that such system is deterministic if the timing variables in* Γ *do not depend on the sequence of environments in* Ω.

Demand process

Let N be a stationary renewal processes of intensity λ, and $X_1, X_2, \ldots$ an i.i.d. sequence of bounded execution times of a task $\tau \in \Gamma$. We define the *demand process* of the task τ as

$$W(t) = \sum_{j=1}^{N(t)} X_j \tag{2.11}$$

the accumulation of execution times required by the task τ until the instant $t > 0$.

Lemma 2.4 (Wald's lemma [Wald, 1944])**.** *Let* N *be a positive integer-valued variable independent from the i.i.d. timing variables* $X, X_1, X_2, \ldots$. *Then for any* $t > 0$,

$$\mathbf{E}\left[\sum_{j=1}^{N} X_j\right] = \mathbf{E}[N]\mathbf{E}[X]$$

Lemma 2.5 (Law of large numbers)**.** *Let* $X_1, X_2, \ldots$ *be an i.i.d. sequence of timing variables such that* $\mathbf{E}[|X_1|] < \infty$. *Then*

$$\frac{1}{n}\sum_{i=1}^{n} X_i \underset{n\to\infty}{\to} \mathbf{E}[X_1]$$

Theorem 2.4. *Let* W *be defined as in* (2.11). *Then*

$$\frac{W(t)}{t} \to \lambda \mathbf{E}[X] \tag{2.12}$$

Proof. We have $\frac{W(t)}{t} = \frac{N(t)}{t}\frac{1}{N(t)}\sum_{j=1}^{N(t)} X_j$. Theorem 2.9 gives us $\frac{N(t)}{t} \underset{t\to\infty}{\to} \lambda$ and as $N(t) \underset{t\to\infty}{\to} \infty$ the law of large numbers (Lemma 2.5) gives us $\frac{1}{N(t)}\sum_{j=1}^{N(t)} X_j \to \mathbf{E}[X]$ since the X_j's are i.i.d. and bounded, we have $\mathbf{E}[|X|] < \infty$. □

Until Chapter 6, we consider the stationary real-time system $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$.

2.3 Time demand analysis

Due to the static-priority policy, the response times of a task τ_k depend only those of higher priority tasks. We call job of *level* k any job of a task of higher or equal priority than τ_k, *i.e.*, any job $\tau_{i,j}, 1 \leqslant i \leqslant k$, $j \in \mathbb{N}$. We suppose that priorities are distinct. Let

$$\bar{u}_k = \sum_{i=1}^{k} \lambda_i \mathbf{E}[C_i]$$

be the *k-level mean utilization* of Γ, the *total mean utilization* $\bar{u} = \bar{u}_{|\Gamma|}$, and the *maximum utilization* of level k,

$$\bar{u}_k^{max} = \sum_{i=1}^{k} \lambda_i c_i^{max}$$

Definition 2.18. *Let* $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ *be a stationary real-time system with total mean utilization* $\bar{u}$. Γ *is said stable if* $\bar{u} < 1$.

Many schedulability tests rely on the utilization of the system, that is, sufficient conditions that ensure that the system is indeed schedulable. However, those schedulability tests are not suited for probabilistic real-time systems, as they do not take advantage of the entire distributions of execution times and inter-arrival times. Hence, the domain of feasibility of probabilistic real-time systems is yet to be defined and is an active topic as presented in Section 1.3.2. In Chapter 3, we prove that systems such that $\bar{u} > 1$ are not feasible.

In the remainder of this thesis, all variables written with a bar, *i.e.*, $\bar{x}$ refer to a priority level.

Let us define the three following stochastic processes:

(i) $N_k(t) = \sum_{l=1}^{\infty} \mathbf{1}_{[0,t]}(A_{k,l})$ as the number of jobs of τ_k released before $t > 0$,

(ii) the k-level demand $\bar{W}_k(t)$ as the accumulation of the execution times required by jobs of priority higher or equal than τ_k, regardless of potential deadline misses, released before the instant t to complete,

$$\bar{W}_k(t) = \sum_{i=1}^{k} \sum_{j=1}^{N_i(t)} C_{i,j}$$

(iii) the k-level backlog $\beta_k(t)$ as the remaining workload of level k at $t \geqslant 0$, defined by

$$\beta_k(t) = \bar{W}_k(t) - \int_0^t \mathbf{1}_{\{\beta_k(s)>0\}} ds$$

The demand represents the workload required by the jobs arriving over time. The goal of schedulability tests is then to check in which proportion of Ω this demands fits the processor time (otherwise called *budget*) given to those jobs.

2.3.1 Pessimism

In probabilistic systems, some authors like Diaz [Díaz et al., 2004] introduced the concept of pessimism for probabilistic systems in a formal way, which is itself called stochastic dominance in the probability field.

Definition 2.19 (Pessimism [Díaz et al., 2004])**.** *We say that X is more pessimistic than Y if and only if $\mathbf{P}(Y > t) \leqslant \mathbf{P}(X > t)$ for all $t > 0$, and we write $Y \leqslant_{st} X$.*

This property is also called *stochastic dominance*. Pessimism is a weaker kind of dominance than the usual inequality operator, as shown in the following lemma.

Lemma 2.6. *Let X and Y be two timing variables. If $X \leqslant Y$, then $X \leqslant_{st} Y$.*

Proof. First of all, since $X \leqslant Y$, we get for all $t > 0$ that $\mathbf{P}(Y > t \mid X > t) = 1$ which implies that

$$\forall t > 0, \mathbf{P}(X > t, Y > t) = \mathbf{P}(X > t) \tag{2.13}$$

Then

$$\mathbf{P}(X > t \mid Y > t) = \frac{\mathbf{P}(X > t, Y > t)}{\mathbf{P}(Y > t)} = \frac{\mathbf{P}(X > t)}{\mathbf{P}(Y > t)} \qquad \text{(according to (2.13))}$$

Hence $\forall t > 0, \frac{\mathbf{P}(X>t)}{\mathbf{P}(Y>t)} \leqslant 1$ and we have the result. □

2.3.2 Blocking time

We denote $C_{i,N_i(t)}$ the execution time of the last job released before the instant $t > 0$. Depending on the discarding policy, jobs wait a certain amount of time between their release and their actual execution. For a task τ_i, we call blocking time the process defined for all $t > 0$ by

$$\begin{cases} B_0(t) &= 0 \\ B_k(t) &= \sum_{i=1}^{k} \min(\beta_i(t) - B_{i-1}(t), C_{i,N_i(t)}) \end{cases}$$

as the amount of time the most recent job before $t > 0$ $\tau_{i,N_i(t)}$ has to wait before its execution. In Figure 2.3 for example, the dotted area represents the blocking time of τ_2.

In opposition to the backlog, the blocking time takes into account the discarding policy. Note that the following inequality allows us to bound the blocking time, which we do in the next chapter.

Proposition 2.1. *For all $t > 0$,*

$$B_k(t) \leqslant \min\left(\beta_k(t), \sum_{i=1}^{k} C_{i,N_i(t)}\right) \tag{2.14}$$

Proof. First of all, since $B_0(t) = 0$, we have $B_1(t) = \min(\beta_1(t), C_{1,N_1(t)})$. Then, by

induction, suppose that (2.14) holds. $B_{k+1}(t)$ is such that

$$\begin{aligned} B_{k+1}(t) &= B_k(t) + \min(\beta_{k+1}(t) - B_k(t), C_{k+1,N_{k+1}(t)}) \\ &\leqslant \min\left(\beta_{k+1}(t), B_k(t) + C_{k+1,N_{k+1}(t)}\right) \\ &\leqslant \min\left(\beta_{k+1}(t), \sum_{i=1}^{k+1} C_{i,N_i(t)}\right) \end{aligned}$$

thus we have (2.14)

□

Hence we have the two following variable that are more pessimistic than $B_i(t)$ thanks to the following proposition.

Corollary 2.3. *For all $t > 0$, $B_k(t) \leqslant_{st} \beta_k(t)$.*

Proof. Apply Lemma 2.1 and Lemma 2.6. □

Corollary 2.4. *For all $t > 0$, $B_k(t) \leqslant \sum_{i=1}^{k} c_i^{max}$.*

Proof. Apply Lemma 2.1 and the fact that the execution time C_i is bounded by c_i^{max}. □

2.3.3 Response times

Due to the static-priority policy, the response time of task τ_i depends on those of higher priority tasks.

Definition 2.20 (Response time)**.** *The response time $R_{k,l}$ of a job $\tau_{k,l}$ is the size of the smallest interval after its arrival time $A_{k,l}$ where the blocking time of level i is zero, i.e.,*

$$R_{k,l} = \inf\{t > 0 : B_k(A_{k,l} + t) = 0\} \tag{2.15}$$

Example 2.1. *Consider a task set $\{\tau_1, \tau_2\}$, and that τ_2 is activated at $t = 0$, i.e., $O_2 = 0$, and $C_{2,1} = y = 3$. Let $\mathbf{P}(C_1 = 1) = 1/2, \mathbf{P}(C_1 = 2) = 1/2$, $\mathbf{P}(T_1 = 2) = 1/2$, $\mathbf{P}(T_1 = 4) = 1/2$, and suppose $B_1(0) = x + y = 8$, $C_{1,1} = 1$,*

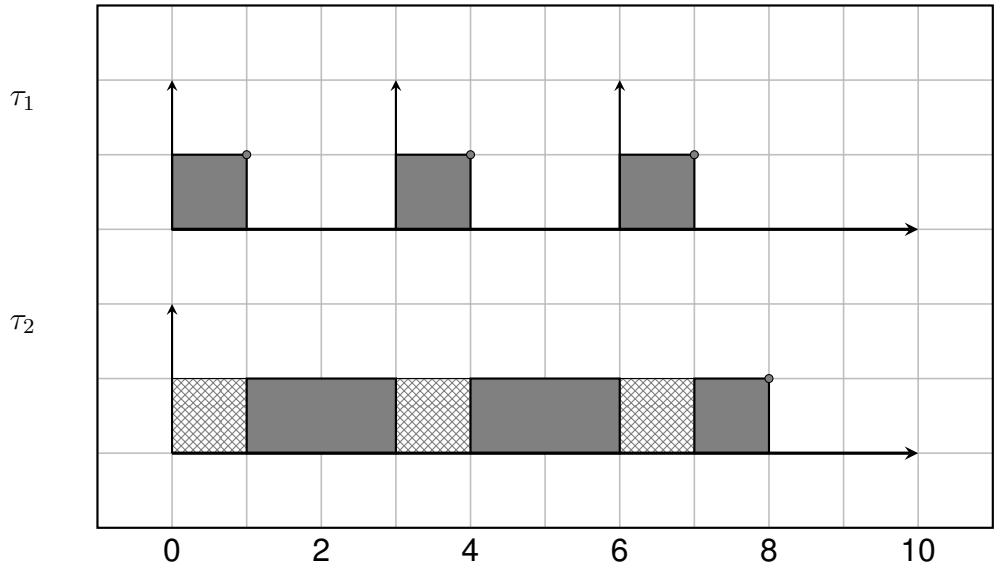


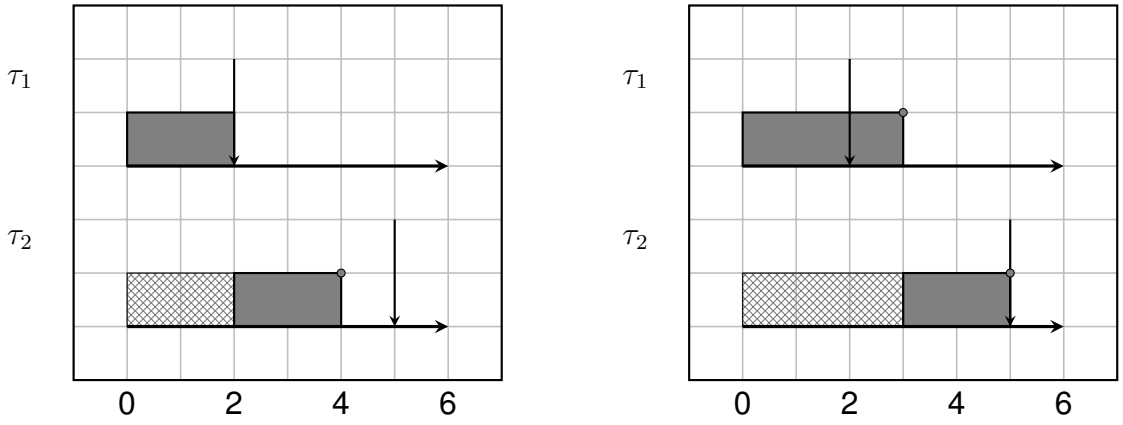
Figure 2.2: $\tau_{2,1}$ is the first job of τ_2 and is released at time 0. Its response time is equal to 8. It starts executing at 1, its blocking time is 1 as it is blocked by the job $\tau_{1,1}$ also released at time 0. It is preempted by $\tau_{1,2}$ at time 3, and $\tau_{1,3}$ at time 6.

$C_{1,2} = 2$, $C_{1,3} = 2$, *and* $O_1 = 2, A_{1,2} = 6, A_{1,3} = 10$ *and* $A_{1,4} = 14$. *Then the response time* $R_{2,1}$ *is the first instant* t *when* $B_1(t) = 0$*:*

$$
\begin{aligned}
B_1(1) &= W_1(0) = x + y \\
B_1(3) &= W_1(0) + C_{1,1} - 3 = x + y - 2 \\
B_1(7) &= W_1(0) + C_{1,1} + C_{1,2} - 7 = x + y - 4 \\
B_1(11) &= W_1(0) + C_{1,1} + C_{1,2} + C_{1,3} - 11 = x + y - 6 \\
B_1(13) &= W_1(0) + C_{1,1} + C_{1,2} + C_{1,3} - 13 = x + y - 8 = 0
\end{aligned}
$$

Hence $R_{2,1} = 13$. *This is illustrated in Figure 4.2. To build all possible values of* $R_{2,1}$*, one must do the same for all combinations of possible values of* $(A_{1,1}, A_{1,2}, A_{1,3}, A_{1,4})$ *and* $(C_{1,1}, C_{1,2}, C_{1,3}, C_{1,4})$.

The discarding policy of a scheduling policy is the decision taken when a deadline is missed. When firm deadlines are missed, jobs are instantly discarded. Thus, for example, a job can be discarded when it misses its deadline. This is the most common discarding policy. However, others can be imagined: no discarding from a certain amount of time, discard the next job, etc. This is illustrated in



(a) The job of τ_1 is discarded.

(b) The job of τ_1 is not discarded.

Figure 2.3: Example of a schedule with the discarding policy and without. The down-arrow represents the deadline of a task. The black circle means that the task has finished its execution. The gray-squared areas indicates that a task awaits for processing resources

Figure 2.3. Pessimism has been introduced to demonstrate that response time analyses only depend on the blocking times.

Theorem 2.5 (Theorem 1 in [López et al., 2008])**.** *Let R be the response time of a job with blocking time b and R' be the response time of this same job with blocking time b'. Then if $b' \geqslant b$, $R \leqslant_{st} R'$,i.e., R' is more pessimistic than R.*

Proof. We have $R = b + X$ and $R' = b' + X$. If $b' \geqslant b$, we get $R \leqslant R'$, hence applying Lemma 2.6 gives us the result. □

Corollary 2.5. *Response time analyses considering a non-discarding policy are more pessimistic than any other discarding policy.*

Proof. Direct application of Theorem 2.5, since non-discarding schedule provide the largest possible blocking times when no job discarding is applied. □

Pessimist response time analysis

As introduced in [Joseph and Pandya, 1986], the response time $R_{k,l}$ of a job $\tau_{k,l}$ is the smallest instant after its arrival time $A_{k,l}$ lower than the time it is given to run the level k demand. The point of the following theorem is to provide a

more pessimistic variable than $R_{k,l}$. A formal translation of this definition couples with Corollary 2.5 gives us the following results.

Theorem 2.6 (Response time bound)**.** *Let* $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ *follow a* $\sum_i M_i / \sum_i G_i / 1 / SP$ *queueing model. The response time* $R_{k,l}$ *is such that*

$$R_{k,l} \leqslant_{st} \inf\{t \in (0, T_k) : \beta_k(A_{k,l}) + \bar{W}_k(t) - \bar{W}_k(0) \leqslant t\} \tag{2.16}$$

Proof. First of all, $B_k(t) \leqslant \beta_k(t)$ for all $t > 0$ according to Proposition 2.1. Hence according to Definition 2.20,

$$R_{k,l} \leqslant \inf\{t \in (0, T_{k,l+1}) : \beta_k(A_{k,l} + t) = 0\}$$

Then for any $a > 0$,

$$\begin{aligned}\beta_k(t+a) &= \bar{W}_k(t+a) - \bar{W}_k(a) + \bar{W}_k(a) - \int_0^a \mathbf{1}_{\{\beta_k(s)>0\}} ds - \int_a^{a+t} \mathbf{1}_{\{\beta_k(s)>0\}} ds \\ &= \beta_k(a) + \bar{W}_k(t+a) - \bar{W}_k(a) - \int_a^{a+t} \mathbf{1}_{\{\beta_k(s)>0\}} ds\end{aligned}$$

thus according to Definition 2.20, $R_{k,l}$ can be stochastically dominated by

$$\inf\left\{t \in (0, T_{k,l+1}) : \beta_k(A_{k,l}) + \bar{W}_k(A_{k,l} + t) - \bar{W}_k(A_{k,l}) \leqslant \int_{A_{k,l}}^{A_{k,l}+t} \mathbf{1}_{\{\beta_k(s)>0\}} ds\right\} \tag{2.17}$$

since $R_{k,l}$ is the first instant after $A_{k,l}$ that the backlog $\beta_{k,l}$ is null, it means that on the interval $s \in [A_{k,l}, A_{k,l} + R_{k,l}]$ we have $\mathbf{1}_{\{\beta_k(s)>0\}} = 1$, which allows to rewrite (2.17) as

$$R_{k,l} = \inf\{t > 0 : \beta_k(A_{k,l}) + \bar{W}_k(A_{k,l} + t) - \bar{W}_k(A_{k,l}) \leqslant (A_{k,l} + t - A_{k,l})\} \tag{2.18}$$

Furthermore,

$$\bar{W}_k(a+t) - \bar{W}_k(a) = \sum_{i=1}^{k} \sum_{j=N_k(a)+1}^{N_k(t+a)} C_{i,j}$$

and since the N_i's are renewal processes and the $(C_{i,j})_j$ are i.i.d., we get

$$\bar{W}_k(a+t) - \bar{W}_k(a) \overset{(d)}{=} \bar{W}_k(t) - \bar{W}_k(0) \tag{2.19}$$

Finally with (2.18), (2.19) and the fact that $T_{k,l+1}$ is independent from $A_{k,l}, \beta_k$ and W_k, and is with the same distribution as T_k we get the result. □

This last theorem allows to consider the simultaneous activation of tasks as the worst-case as we see in the following.

Theorem 2.7 (WCRT)**.** *Consider a stationary real-time system and let* $R_k^{max} = \inf\{t \in (0, T_k) : \sum_{i=1}^{k-1} c_i^{max} + \bar{W}_k(t) \leqslant t\}$*. Then,*

$$\forall l \in \mathbb{N}, R_{k,l} \leqslant_{st} R_k^{max}$$

Proof. With the discarding policy, the blocking time of level i is always bounded by $\sum_{k=1}^{i} c_k^{max}$. Hence applying Theorem 2.6 with $\beta_i(A_{i,j}) = \sum_{k=1}^{i} c_k^{max}$ gives the result. □

This method is called TDA. TDA holds for any static-priority preemptive single-processor model with the discarding policy discussed in Section 2.3.3, thanks to the following theorem.

Theorem 2.8 (Theorem 3 in [Burns and Baruah, 2008])**.** *Deterministic TDA of static-priority preemptive single processor systems with independent tasks is sustainable.*

This last theorem is the reason why the probabilistic analysis is possible for real-time systems and that TDA works on stationary real-time systems just like any periodic deterministic real-time system.

2.3.4 Deadline miss probabilities

The DMP of a job $\tau_{k,l}$ is the probability that its response time is greater than its deadline on any possible sequence of environments. In order to compute it, we compute the exceedence function of the response time $R_{k,l}$

$$H_{k,l}(t) = \mathbf{P}\left(R_{k,l} > t\right) \tag{2.20}$$

Computing this exceedence functions requires to know the distribution of all the variables that it involves. For this matter, we build a pessimistic approximation of the response time of the task τ_i denoted R_i^{max} in Chapter 3 with a *fluid model*. With $H_{i,j}$ we can tell if the job $\tau_{i,j}$ is schedulable or not. Although, computing all the $H_{i,j}, j \in \mathbb{N}$ is difficult. In order to be safe, *i.e.*, make a pessimist analysis, we define the WCRT as a bound R_i^{max} such that for all $j \in \mathbb{N}$, $R_{i,j} \leqslant_{st} R_i^{max}$, *i.e.*,

$$\sup_{j \in \mathbb{N}} H_{i,j}(t) \leqslant \mathbf{P}\left(R_i^{max} > t\right)$$

Without further information we cannot say if such bound exists and is finite. However, if we can compute the distribution of the WCRT R_i^{max}, we then have a bound of deadline miss probabilities.

Definition 2.21. Γ *is said feasible if and only if there exists at least one scheduling policy such that the WCRT* R_i^{max} *is finite and such that all tasks* $\tau_i \in \Gamma$ *satisfy*

$$p_i^{max} = \mathbf{P}\left(R_i^{max} > D_i\right) \leqslant \alpha_i \tag{2.21}$$

In those terms, checking if a task is schedulable is checking if the WCRT R_i^{max} is finite and $p_i^{max} \leqslant \alpha_i$ for all tasks $\tau_i \in \Gamma$ is the goal of Chapters 3 and 4.

2.4 Processor model

A uniform multiprocessor system is a set of processors $\Pi = \{\kappa_1, \ldots, \kappa_m\}$ composed of processors $\kappa \in \Pi$ of respective speed $s(\kappa) \in [1, s^{max}]$, *i.e.*, κ can process $s(\kappa)$ workload units in one unit of time. We consider Π ordered by decreasing speeds, *i.e.*, $s(\kappa_i) > s(\kappa_j)$ if $i < j$. Let $\kappa \in \Pi$ be a processor at a given state in time and a task $\tau_i \in \Gamma$.

We denote the *local mean utilization* of the task τ_i on the processor κ by

$$u_i(\kappa) = \frac{\lambda_i \mathbf{E}[C_i]}{s(\kappa)} \tag{2.22}$$

and suppose for all $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$ that $u_i(\kappa_j) < 1$. We define $\Gamma_i^+(\kappa)$ (resp. $\Gamma_i^-(\kappa)$) to be the set of tasks of priority higher (resp. lower) or equal to the priority of τ_i active on the processor κ at a given time, and let

$$\bar{u}_i(\kappa) = \sum_{\tau_j \in \Gamma_i^+(\kappa)} u_i(\kappa) \tag{2.23}$$

be the *local mean utilization* of level i in the processor κ and, respectively, let

$$\bar{v}_i(\kappa) = \frac{1}{s(\kappa)} \left(\sum_{\tau_j \in \Gamma_i^+(\kappa)} \lambda_j \mathbf{E}[C_j^2] \right)^{1/2}$$

be the *local deviation* of level i on the processor κ, that is the sum of the utilization (resp. deviation) of the tasks in $\Gamma_i^+(\kappa)$. Let

$$\bar{u}_i^{max}(\kappa) = \frac{1}{s(\kappa)} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \lambda_j c_j^{max}$$

be the *local maximum utilization* of level i of κ and

$$\bar{v}_i^{max}(\kappa) = \frac{1}{s(\kappa)^2} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \lambda_j (c_j^{max} - c_j^{min})^2$$

the *local maximum deviation* of level i in κ.

Definition 2.22 (Stationary Multiprocessor system)**.** *Let* $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ *be a stationary real-time system as defined in Definition 2.17, and* Π *be a multiprocessor system. We call* $(\Omega, \mathbf{P}, \Gamma, \Pi, \{\theta_t\})$ *a stationary multiprocessor system. Without any loss of generality, we refer to it as a stationary real-time system.*

2.5 Rate Monotonic

Real-time scheduling is the decision process deciding which job should be executed. Online priority-driven scheduling algorithms are typically implemented as follows: at each time instant, they allocate an available processor to the highest-priority job. Static-priority algorithms satisfy the property that for two tasks τ_i and τ_j, whenever τ_i and τ_j are both have active, it is always the case that the jobs of one task have priority over the other. With dynamic priority algorithms in the other hand, it is possible that some tasks τ_i and τ_j both have active jobs simultaneously, but in some case the job of τ_i has a higher priority than the job of τ_2 and in other cases the opposite. Scheduling algorithms that allow such "switching" of priorities between jobs are known as dynamic-priority algorithms. We cover in this thesis only static-priority policies.

The two famous and widely used scheduling policies RM and EDF are proven optimal in some well defined models. RM is a scheduling algorithm used to prioritize the scheduling of processes based on their relative rate in the case of periodic systems. The algorithm assigns higher priority to a tasks that are given higher maximum rates, meaning that the most important processes are scheduled more often than lower rate processes. Meaning that τ_i has priority over τ_j if $\lambda_i > \lambda_j$. This type of scheduling algorithm is useful for ensuring that the most important tasks are completed on time and with minimal delays. EDF is used for its dynamic computation of priorities : each instance of a program has its own priority as a function of its deadline. It is also shown optimal in the non preemptive single processor case for example, and many of its variants are studied in [Liu and Layland, 1973, Baruah and Baker, 2008, Baruah and Goossens, 2008].

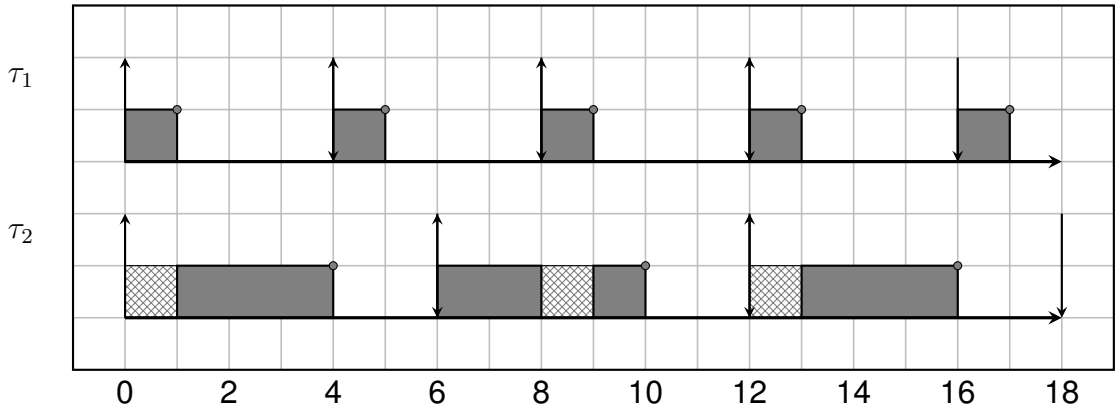


Figure 2.4: Two tasks with implicit deadline using the RM policy

Theorem 2.9 (Hyperbolic bound [Bini et al., 2003])**.** *In the case of a single processor preemptive RM scheduling, with periodic inter-arrival times and implicit deadlines, if*

$$\prod_{i=1}^{n}(\lambda_i c_i^{max} + 1) \leqslant 2 \tag{2.24}$$

the system is schedulable with a permitted failure rate equal to zero.

In the case of multiprocessor scheduling, *i.e.*, $m \geqslant 2$, the RM policy is verified schedulable for identical multiprocessor systems global scheduling when

$$\bar{u}_n^{max} \leqslant \frac{m^2}{3m-1} \;;\; \forall \tau_i \in \Gamma, \lambda_i c_i^{max} \leqslant \frac{m}{3m-2} \tag{2.25}$$

and by allocating jobs to any available processor, all tasks are schedulable with a permitted failure rate equal to zero [Andersson et al., 2001]. In [Baruah and Goossens, 2003] authors extend this result to uniform heterogeneous multiprocessor systems. The system is schedulable with permitted failure rates equal to zero when

$$\bar{u}_n^{max} \leqslant \frac{1}{2}\left(\sum_{\kappa\in\Pi} s(\kappa) - (1+\Lambda)\max_{\tau_i\in\Gamma}\{\lambda_i c_i^{max}\}\right) \tag{2.26}$$

where $\Lambda = \max_{\kappa_j\in\Pi} \frac{1}{s(\kappa_j)} \sum_{i=j+1}^{m} s(\kappa_i)$ measures the degree by which Π differs from an identical multiprocessor system.

Finally, for the restricted migration the utilization bound of RM on a multiproces-

sor system with identical processors of speed s is proven [Goossens et al., 2012] to be

$$\bar{u}_n^{max} \leqslant ms - (m-1) \max_{\tau_i \in \Gamma} \{\lambda_i c_i^{max}\}$$

3 | Fluid model

Contents

Utilization-based schedulability conditions for single-core static-priority preemptive scheduling policies are widely studied [Davis et al., 2016]. The seminal work of Liu and Layland [Liu and Layland, 1973] introduces a sufficient condition for the feasibility of a real-time system using its *maximal utilization*. Nevertheless, a real-time system not satisfying this sufficient condition may remain *schedulable* with a given probability (see Eq. (2.21) in Section 2). Moreover, while probabilistic methods have been focused towards fitting in this sufficient condition by providing less pessimistic analyses, their domain of feasibility needs to be defined as well. In

this thesis, we build necessary feasibility conditions for static-priority scheduling policies based on the mean utilization of the real-time system. We demonstrate that a mean utilization smaller than 1 is mandatory for *response times* to be finite, *c.f.*, Propositions 4.2 and 4.4. We call systems with a mean utilization smaller than 1, *stable* real-time systems.

3.1 Stochastic analysis background

The elements presented in this chapter are widely inspired from the books of François Baccelli and Pierre Brémaud, *Elements of queueing theory*, [Baccelli and Brémaud, 2013] and, Hong Chen and David D. Yao, *Fundamentals of queueing networks: Performance, asymptotics, and optimization*, [Chen and Yao, 2001].

3.1.1 Brownian motions

In order to statistically describe the behavior of real-time systems we use a process called Brownian Motion. It allows to provide theorems similar to the *central limit theorem* for stochastic processes. All definitions and results presented in this section can be found in [Le Gall, 2016].

Definition 3.1 (Standard Brownian motion)**.** *A standard Brownian Motion is a process* $B = (B(t), t > 0)$ *such that*

- $B(0) = 0$,
- $B(t+s) - B(s) \sim \Phi_{0,t}$, *for* $t, s > 0$,
- $B(t) - B(s)$ *is independent of* $B(u) - B(v)$ *for* $t > s > u > v > 0$,
- B *is continuous.*

Lemma 3.1 (Re-scaling property of Brownian motions)**.** *Let* B *be a standard Brownian motion. For any* $a > 0$, $a^{-1/2}B(at) \overset{(d)}{=} B(t)$.

This is an important property that we use in the next section when introducing the heavy-traffic theorem.

Definition 3.2 (Brownian motion)**.** *A Brownian motion of drift u and deviation $v > 0$ is a process W such that there exists a standard Brownian motion B such that*

$$W(t) - W(0) = ut + vB(t) \tag{3.1}$$

for all $t > 0$. For each $t > 0$, the distribution function of $W(t) - W(0)$ is Φ_{ut,v^2t}.

Theorem 3.1 (First-passage time of a Brownian motion, [Abundo, 2016])**.** *Let W be a Brownian motion of drift $u > 0$ and deviation $v > 0$. Let $\mathcal{I}(x)$ be the first-passage time of W on 0 when $W(0) = x > 0$, i.e.,*

$$\mathcal{I}(x) = \inf\{t > 0 : W(t) = 0\}$$

Then $\mathcal{I}(x)$ has an inverse Gaussian distribution of mean x/u and shape $(x/v)^2$.

Definition 3.3 (Reflected Brownian motion, Theorem 6.1 [Chen and Yao, 2001])**.** *A process β is called a reflected Brownian motion of drift u and deviation $v > 0$ if there exists a Brownian motion W of drift u and deviation $v > 0$ such that for all $t > 0$,*

$$\beta(t) = W(t) + \sup_{s\in[0,t]} (-W(s))^+$$

Theorem 3.2 (Theorem 6.2 [Chen and Yao, 2001])**.** *Let β be a reflected Brownian motion of drift u and deviation $v > 0$. If $u < 0$, then the distribution of $\lim_{t\to\infty} \beta(t)$ is an exponential distribution of parameter*

$$\eta = -\frac{2u}{v^2}$$

and is called the steady-state distribution of β.

3.1.2 Backlog process

We now consider the evolution of the execution of tasks and the demand. Let W be a demand process. We introduce the *backlog process* as the remaining demand at a given instant

$$\beta(t) = W(t) - \int_0^t \mathbf{1}_{\{\beta(s)>0\}} ds \tag{3.2}$$

describing the remaining demand at the instant $t > 0$, after having been executed at most t units of time. The term $\int_0^t \mathbf{1}_{\{\beta(s)>0\}} ds$ is stochastic and represents the amount of time the system is not *idle* considering the work-conserving assumption. This backlog process satisfies the relation

$$\beta(a_{n+1}) = \left(\beta(a_n) + c_{n+1} - t_{n+1}\right)^+, \forall n \in \mathbb{N} \tag{3.3}$$

where the $(a_n), (c_n), (t_n)$ are respectively sequences of arrival times, execution times and inter-arrival times. Eq. (3.3) is at the basis of the *shrink and convolve* method used by many, *e.g.*, [Díaz et al., 2004, Kim et al., 2005, Palopoli et al., 2012, Villalba Frias, 2018, von der Brüggen et al., 2021]. Under some conditions, this backlog process has asymptotic properties according to the following theorem.

Theorem 3.3 (Steady-state backlog [Loynes, 1962])**.** *Let β be the backlog process of a $M/G/1$ queueing model as defined in (3.2). If $\mathbf{E}[c_0] < \mathbf{E}[t_0]$, the limit $\tilde{\beta} = \lim_{t\to\infty} \beta(t)$ exists, is finite and is equal to*

$$\tilde{\beta} = \left(\sup_n \sum_{j=1}^{n} c_j - t_j \right)^+$$

In the next section, we use this backlog process at priority levels, *i.e.*, by considering the demand of not only a task but also of higher priority tasks. This allows to model the arrival of jobs of a static-priority scheduling policy. Indeed, at priority level, we know that while there is workload from higher priority, it is sufficient to check the workload of priority level k and the time that passed between

Table 3.1: Task set used in the simulations of the experimental results of Section 4.4

τ_k	λ_k^{-1}	$[c_k^{min}, c_k^{max}]$	dF_k	$\bar{u}_k$	$\bar{u}_k^{max}$
τ_1	4	$(1, 2)$	$(0.5, 0.5)$	0.375	0.5
τ_2	6	$(1, 2)$	$(0.5, 0.5)$	0.625	0.833
τ_3	8	$(1, 2, 3)$	$(0.5, 0.3, 0.2)$	0.838	1.208
τ_4	10	$(1, 2, 3)$	$(0.6, 0.2, 0.2)$	0.998	1.508
τ_5	12	$(1, 2, 3, 4)$	$(0.5, 0.3, 0.1, 0.1)$	1.148	1.841

two arrival times of jobs of level k. In the end of this chapter, we focus on *idle times*, the times where the backlog is 0, and it is important to remark that any scheduling policy considering priority levels instead of specific tasks will provide the same idle times. The reason of this is that what is important at the end is the amount of workload demanded to the system and the amount of time that passes between job arrivals. In other words, for a sorted queue, *i.e.*, by priority level, we can use FIFO results on priority level backlog in the next section. See Figure 2.1.

3.2 Memoryless backlog

In this chapter, we present an analytical approximation of the demand of probabilistic real-time systems, using a *fluid* model associated to the actual demand of the system. Fluid models are widely used in queueing theory [Baccelli and Brémaud, 2013] in order to determine asymptotic results. In real-time systems, a famous example is the DP-FAIR scheduling algorithm [Levin et al., 2010], that uses the fluid model of the backlogs to determine scheduling decisions for homogeneous multiprocessor systems.

Let $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ be a stationary real-time system following the queueing model $\sum_i M_i / \sum_i G_i / 1/\mathrm{SP}$, such that Γ is ordered by decreasing priority order, *i.e.*, τ_i has priority over τ_{i+1}. Let us remind the three following stochastic processes:

(i) $N_k(t) = \sum_{l=1}^{\infty} \mathbf{1}_{[0,t]}(A_{k,l})$ as the number of jobs of τ_k released before $t > 0$, of mean $\mathbf{E}[N_k(t)] = \lambda_k t$, see Definition 2.15. N_k is right-continuous with left limit (RCLL) and integer-valued.

(ii) the k-level demand $\bar{W}_k(t)$ as the workload required by jobs of priority higher

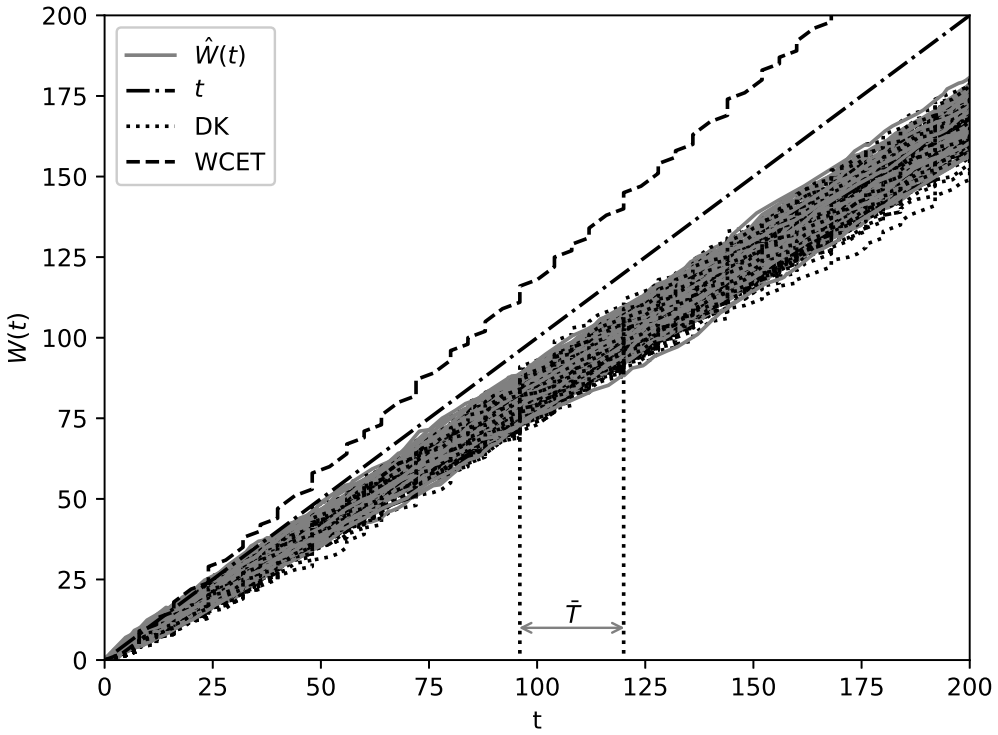


Figure 3.1: Level 3 demand of 1 000 instances of the Diaz and Kim (DK) model, the heavy-traffic demand process and the classical deterministic worst-case analysis (WCET) considering only the maximal execution time for each task, for $\bar{u} = 0.838$ and $\bar{u}^{max} = 1.208$ and hyper-period $\bar{T}$, for Γ defined in Table 3.1

or equal than τ_k, regardless of potential deadline misses, released before the instant t to complete, $\bar{W}_k(0) \geqslant 0$,

$$\bar{W}_k(t) = \sum_{i=1}^{k} \sum_{j=1}^{N_i(t)} C_{i,j}$$

of mean $\mathbf{E}[\bar{W}_k(t)] = \sum_{i=1}^{k} \mathbf{E}[N_i(t)]\mathbf{E}[C_i] = \bar{u}_k t$, see [Janssen and Manca, 2006, (6.53)]. $\bar{W}_k$ is RCLL and positive.

(iii) the k-level backlog $\beta_k(t)$ as the remaining workload of level k at $t \geqslant 0$ when Γ is ordered in a decreasing priority order and scheduled with a preemptive

static-priority scheduling policy, defined by

$$\beta_k(t) = \bar{W}_k(t) - \int_0^t \mathbf{1}_{\{\beta_k(s)>0\}} ds \tag{3.4}$$

where $\int_0^t \mathbf{1}_{\{\beta_k(s)>0\}} ds$ is the total busy time of level k [Lehoczky, 1990, p. 2] before $t > 0$. β_k is RCLL and positive.

The backlog process β_i is such that for $t \in [A_{i,j}, s)$ and $s < \inf\{A_{p,l} : A_{p,l} > A_{i,j}\}$,

$$\beta_k(t) = (\beta_k(A_{i,j}) + C_{i,j} - (t - A_{i,j}))^+$$

which is known as Lindsley's equation [Lindley, 1952]. β_k is not a Markovian process in general, but a stochastic recurrence [Baccelli and Brémaud, 2013].

Example 3.1 (Backlog)**.** *Let us consider the task set* $\{\tau_1, \tau_2\}$*, with* $\mathbf{P}(C_1 = 1) = 1$*,* $\mathbf{P}(T_1 = 2) = 1$*,* $\mathbf{P}(C_2 = 1) = 1/2$*,* $\mathbf{P}(C_2 = 2) = 1/2$*,* $\mathbf{P}(T_2 = 3.1) = 1/2$*,* $\mathbf{P}(T_2 = 4) = 1/2$*. Suppose both tasks* τ_1 *and* τ_2 *are activated at time* $t = 0$ *and* $T_{2,2} = 3.1$*,* $C_{2,1} = 2$*,* $C_{2,2} = 1$*. Then,* $\bar{W}_2(0) = 1 + 2$*,*

$$\begin{aligned}
\beta_2(2) &= \bar{W}_2(2) - \textstyle\int_0^2 \mathbf{1}_{\{\beta_2(s)>0\}} ds \\
&= (1 + 2 + 1) - 2 = 2 \\
\beta_2(3) &= \bar{W}_2(3) - \textstyle\int_0^3 \mathbf{1}_{\{\beta_2(s)>0\}} ds \\
&= (1 + 2 + 1) - 3 = 1 \\
\beta_2(3.1) &= \bar{W}_2(3.1) - \textstyle\int_0^{3.1} \mathbf{1}_{\{\beta_2(s)>0\}} ds \\
&= (1 + 2 + 1 + 1) - 3.1 = 1.9 \\
\beta_2(3.2) &= \bar{W}_2(3.2) - \textstyle\int_0^{3.2} \mathbf{1}_{\{\beta_2(s)>0\}} ds \\
&= (1 + 2 + 1 + 1) - 3.2 = 1.8
\end{aligned}$$

At the instant $t = 3.2$*, the backlog of level* 2 *is* 1.8*.*

The process β_k describes the remaining demand without considering deadline misses, *i.e.*, while jobs are discarded their demand remains in the backlog analysis. Thus, the process β_k is an upper-bound of the blocking time, see Eq. (4.4), *i.e.*, the

response time analysis that we provide in Chapter 4 is pessimistic as defined in [Díaz et al., 2004].

We express in the following the backlog process in a more convinient way.

Lemma 3.2 (Theorem 6.1 [Chen and Yao, 2001])**.** *For any right-continuous with left limits process X, there exists a unique pair of processes (Z, Y) such that*

(S1) $Z = X + Y \geqslant 0$,

(S2) $\int_0^\infty Z(t)dY(t) = 0$,

(S3) $dY(t) \geqslant 0, Y(0) = 0$.

Furthermore, $Y(t) = \sup_{s\in[0,t]}(-X(s))^+$.

Lemma 3.2 solves a reflexion mapping problem called the One dimensional Skorokhod problem. We use is in the following to express the right limit of the backlog process.

Theorem 3.4 (Section 6.2 [Chen and Yao, 2001])**.** *The backlog of level k is such that*

$$\beta_k(t) = \bar{W}_k(t) - t + \sup_{s\in[0,t]}(s - \bar{W}_k(s))^+ \tag{3.5}$$

for all $t > 0$.

Proof. Let $X(t) = \bar{W}_k(t) - t$ and $Y(t) = \int_0^t \mathbf{1}_{\beta_k(s)=0}ds$. Clearly, $\beta_k(t) = X(t) + Y(t)$. We call Y the idle time process. The following relations must hold: For all $t > 0$,

(i) $\beta_k \geqslant 0$,

(ii) $\int_0^\infty \beta_k(t)dY(t) = 0$,

(iii) $dY(t) \geqslant 0, Y(0) = 0$.

In other words, $dY(t) \geqslant 0$ means that Y is nondecreasing, since the idle time process is measured as a cumulation of a positive quantity over time; and $\int_0^\infty \beta_k(t)dY(t) = 0$ reflects the fact that the idle time cannot cumulate when the backlog is positive. From Lemma 3.2, we check the $(S1) - (S3)$ conditions, hence we know that $Y(t) = \sup_{s\in[0,t]}(s - \bar{W}_k(s))^+$ and we get Eq. (3.5). □

3.2.1 The Loynes theorem

In this section, we define the domain of the *steadiness*, provide an expression of the instant when the system goes from transient to steady, and prove that this time instant exists and is finite under some conditions.

Definition 3.4 (Steady-state backlog)**.** *For a stable real-time system* Γ*, the steady-state backlog is defined by*

$$\tilde{\beta}_k = \lim_{t\to\infty} \beta_k(t)$$

and we denote

$$\pi_k(x) = \mathbf{P}\left(\tilde{\beta}_k \leqslant x\right)$$

the distribution function of the steady-state backlog of level k*.*

We use background results of queueing theory presented in 3.2.1 and 3.2.2, provide the exact formulation of the steady-state backlog distribution π_k and illustrate this result in the deterministic inter-arrival case already studied by Diaz *et al.* [Díaz et al., 2002] in Section 3.3.

As the demand and backlog processes $\bar{W}_k$ and β_k are well studied in queueing theory [Baccelli and Brémaud, 2013, Chen and Yao, 2001, Jeanblanc et al., 2009], we provide the formula of the steady-state of the system, by adapting Theorem 3.1 for the $\sum_i M_i/\sum_i G_i/1/\mathrm{SP}$ queueing model.

Proposition 3.1. *Let* $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ *follow a* $\sum_i M_i/\sum_i G_i/1/SP$ *queueing model. Let*

$$\bar{A}_{k,l} = \inf\{t > 0 : \sum_{i=1}^{k} N_i(t) = l\}$$

be the activation time of the l*-th job of level* k*, and* $\bar{I}_{k,l}$ *the index of the task of the* l*-th job of level* k*, and let*

$$\bar{C}_{k,l} = C_{\bar{I}_{k,l}, N_{\bar{I}_{k,l}}(\bar{A}_{k,l})}$$

be the execution time of the l*-th job of level* k*. Then,*

- *if* $\bar{u}_k < 1$ *the steady-state backlog* $\tilde{\beta}_k$ *exists, is finite and is equal to*

$$\tilde{\beta}_k = \left(\sup_n \sum_{l=1}^{n} \bar{C}_{k,l} - (\bar{A}_{k,l+1} - \bar{A}_{k,l}) \right)^+ \tag{3.6}$$

  where $x^+ = \max(0, x)$*, c.f., [Baccelli and Brémaud, 2013, Property 2.2.1] and Theorem 3.3. In addition, there is an infinite number of idle times, c.f., [Baccelli and Brémaud, 2013, Property 2.2.5],*

- *if* $\bar{u}_k = 1$*, then the existence of a finite steady-state* $\tilde{\beta}_k$ *is uncertain,*

- *If* $\bar{u}_k > 1$*, there exists a finite number of idle times of level* k *and no finite steady-state, c.f., [Baccelli and Brémaud, 2013, Property (2.2.2)], backlogs are always transient.*

Proof. First of all, the Example 3.1.3 [Baccelli and Brémaud, 2013] shows that a stationary point process with priority class jobs is still associated to a point process of intensity $\bar{u}_k$. According to Theorem 2.3, the superposition of Poisson point processes is still a Poisson process, hence, the superposition of the arrival of all jobs of level k is a stationary point process. Let $\bar{A}_{k,l} = \inf\{t > 0 : \sum_{i=1}^{k} N_i(t) = l\}$ be the activation time of the l-th job of level k, and $\bar{I}_{k,l}$ the index of the task of the l-th job of level k, and let

$$\bar{C}_{k,l} = C_{\bar{I}_{k,l}, N_{\bar{I}_{k,l}}(\bar{A}_{k,l})}$$

be the execution time of the l-th job of level k. In that way, the arrival of jobs of level k form a $M/G/1$ model. We use the property of marked Poisson processes shown in Lemma 2.3:

$$\forall l, \mathbf{P}(\bar{I}_{k,l} = i) = \frac{\lambda_i}{\sum_{i=1}^{k} \lambda_i} \tag{3.7}$$

The $\bar{I}_{k,l}$ are independent from the $C_{i,j}, j \geqslant 1$ (not $\bar{C}_{k,l}$!), $T_{i,j}, j \geqslant 1$ and *a fortiori* of $A_{i,j}, j \geqslant 1$. Then we have for any $A \subset \mathbb{R}_+$

$$\begin{aligned}\mathbf{P}(\bar{C}_{k,l} \in A) &= \sum_{i=1}^{k} \mathbf{P}(C_{i,N_i(\bar{A}_{k,l})} \in A \mid \bar{I}_{k,l} = i)\mathbf{P}(\bar{I}_{k,l} = i) \\ &= \sum_{i=1}^{k} \mathbf{P}(C_i \in A \mid \bar{I}_{k,l} = i)\mathbf{P}(\bar{I}_{k,l} = i) \\ &= \frac{1}{\sum_{i=1}^{k} \lambda_i} \sum_{i=1}^{k} \lambda_i \mathbf{P}(C_i \in A)\end{aligned}$$

Then we use Theorem 3.3 on this superposed process with the execution times $\bar{C}_{k,l}$ and inter-arrival times $\bar{A}_{k,l+1} - \bar{A}_{k,l} \overset{(d)}{=} \min_{i=1,\dots,k} T_i$ which is exponential of parameter $\sum_{i=1}^{k} \lambda_i$, marked by the $\bar{I}_{k,l}$. $\bar{C}_{k,l}$ and $\bar{A}_{k,l+1} - \bar{A}_{k,l}$ are dependent through the mark $\bar{I}_{k,l}$, but this independence is not required in the proof of Theorem 3.3, *c.f.*, Example 1.4.4 [Baccelli and Brémaud, 2013]. □

Our goal is to find the distribution of the steady-state backlog $\tilde{\beta}_k$. However, given the generality of this model, we cannot provide an exact description of the process β_k. The *heavy-traffic assumptions* allows us to find an approximation for the distribution of the steady-state backlog $\tilde{\beta}_k$ when the system utilization gets close to 1, *c.f.*, Figure 3.2, *i.e.*, we build a process $\beta_k^{(\infty)}$ such that its steady-state approximates $\tilde{\beta}_k$.

3.2.2 The heavy-traffic theorem

A first step in the approximation of the backlog process β_k is the approximation of the demand process $\bar{W}_k$. The following theorem provides a *fluid model*, that is a continuous version of the backlog process using asymptotic results. Fluid models are widely used in queueing theory [Chen and Yao, 2001] in order to determine asymptotic results. In real-time systems, a famous example is the DP-FAIR scheduling algorithm [Levin et al., 2010], that uses the fluid model of the backlogs to determine optimal decisions for homogeneous multiprocessor systems that we discuss in more details in Chapter 6. The idea behind heavy-traffic, is that we look at the workload processes in the long time, *i.e.*, we put the processes

in a limit situation, and analyze this limit.

Let us define the sequence of re-scaled processes

$$\left\{\begin{array}{rcl} \bar{W}_k^{(n)}(t) & = & \bar{W}_k(0) + n^{-1}\bar{W}_k(nt) + n^{-1/2}(\bar{W}_k(nt) - \bar{u}_k nt) \\ \beta_k^{(n)}(t) & = & \beta_k(0) + n^{-1}\beta_k(nt) + n^{-1/2}(\beta_k(nt) - (\bar{u}_k - 1)nt) \end{array}\right. \tag{3.8}$$

and look for their limit.

Theorem 3.5. *Let Γ be a stationary task set as defined in Section 2.1.2. The re-scaled demand process sequence $\bar{W}_k^{(n)}, n \geqslant 0$ is such that for all $t > 0$,*

$$\lim_{n\to\infty} \bar{W}_k^{(n)}(t) \stackrel{(d)}{=} \bar{W}_k^{(\infty)}(t)$$

where $\bar{W}_k^{(\infty)}$ is a Brownian motion of drift $\bar{u}_k = \sum_{i=1}^{k} \lambda_i \mathbf{E}[C_i]$ and deviation $\bar{v}_k^2 = \sum_{i=1}^{k} \lambda_i \mathbf{E}[C_i^2]$. See Figure 3.1 for an illustration.

Proof. The proof is based on the Laplace transform of the demand process. Let N_i be the associated Poisson process of arrivals of τ_i. Let $W_i(t) = \sum_{j=1}^{N_i(t)} C_{i,j}$ where $C_i, C_{i,1}, \ldots$ are the i.i.d. execution times of τ_i, and its Laplace transform

$$\begin{aligned} \mathbf{E}[e^{\xi W_i(t)}] &= \mathbf{E}\left[e^{\xi \cdot \sum_{j=1}^{N(t)} C_{i,j}}\right] \\ &= \sum_{n\geqslant 0} \mathbf{E}\left[e^{\xi \cdot \sum_{j=1}^{n} C_{i,j}} | N_i(t) = n\right] \mathbf{P}(N_i(t) = n) \end{aligned}$$

The Poisson processes N_i and the execution times are independent, and $N_i(t)$ is a Poisson variable of parameter $\lambda_i t$, *i.e.*,$\mathbf{P}(N_i(t) = n) = e^{-\lambda_i t}\frac{(\lambda_i t)^n}{n!}$, which leads us to

$$\begin{aligned} \mathbf{E}[e^{\xi W_i(t)}] &= \sum_{n\geqslant 0} \mathbf{E}\left[e^{\xi \cdot \sum_{j=1}^{n} C_{i,j}}\right] e^{-\lambda_i t}\frac{(\lambda_i t)^n}{n!} \\ &= e^{-\lambda_i t} \sum_{n\geqslant 0} \frac{(\lambda_i t)^n}{n!} \prod_{j=1}^{n} E[e^{\xi C_{i,j}}] \end{aligned}$$

Finally, the variables $C_i, C_{i,1}, C_{i,2} \ldots$ are identically distributed, hence $\prod_{j=1}^{n} E[e^{\xi C_{i,j}}] =$

$\mathbf{E}[e^{\xi C_i}]^n$, which leads to

$$\begin{aligned}\mathbf{E}[e^{\xi W_i(t)}] &= e^{-\lambda_i t}\sum_{n\geqslant 0}\mathbf{E}[e^{\xi C_i}]^n\frac{(\lambda_i t)^n}{n!}\\ &= e^{\lambda_i t(\mathbf{E}[e^{\xi C_i}]-1)}\end{aligned} \tag{3.9}$$

In order to find the variance of $W_i(t)$ we use the following lemma:

Lemma 3.3 (Eve's law [Blitzstein and Hwang, 2019])**.** *Let X and Y be variables with finite variance. Then,*

$$\mathrm{Var}(Y) = \mathbf{E}[\mathrm{Var}(Y \mid X)] + \mathrm{Var}(\mathbf{E}[Y \mid X])$$

Applying this last lemma to $Y = W_i(t)$ and $X = N_i(t)$ gives us

$$\begin{aligned}\mathrm{Var}(W_i(t)) &= \mathbf{E}[\mathrm{Var}(W_i(t) \mid N_i(t))] + \mathrm{Var}(\mathbf{E}[W_i(t) \mid N_i(t)])\\ &= \mathbf{E}[N_i(t)\mathrm{Var}(C_i)] + \mathrm{Var}(N_i(t)\mathbf{E}[C_i])\\ &= \mathrm{Var}(C_i)\mathbf{E}[N_i(t)] + \mathbf{E}[C_i]^2\mathrm{Var}(N_i(t))\\ &= \mathrm{Var}(C_i)\lambda_i t + \mathbf{E}[C_i]^2\lambda_i t\\ &\qquad\qquad \text{(Because } N_i(t) \text{ is a Poisson variable of mean } \lambda_i)\\ &= \lambda_i t\mathbf{E}[C_i^2]\end{aligned}$$

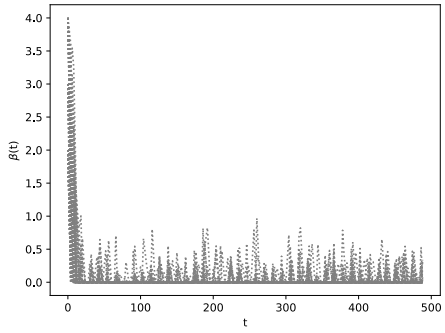
let $u_i = \lambda_i\mathbf{E}[C_i]$, $v_i^2 = \lambda_i\mathbf{E}[C_i^2]$ and $\tilde{W}_i(t) = \frac{W_i(t)-u_i t}{\sqrt{tv_i^2}}$. By noticing that

$$\mathbf{E}[e^{\xi\tilde{W}_i(t)}] = \mathbf{E}[e^{\xi W_i(t)/\sqrt{tv_i^2}}]e^{-\xi t u_i/\sqrt{tv_i^2}}$$

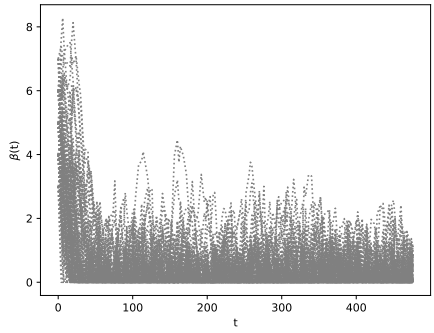
The Taylor expansion on $e^{\xi C_i}$ when $t \underset{t\to\infty}{\to} \infty$ is well defined as C_i is bounded. Then

$$\mathbf{E}[e^{C_i\xi/\sqrt{tv_i^2}}] = \mathbf{E}[1 + \frac{\xi}{\sqrt{tv_i^2}}C_i + \frac{\xi^2}{2tv_i^2}C_i^2 + \circ(\xi^2)]$$

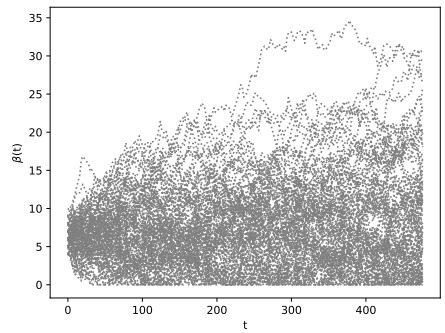
which leads with Eq. (3.9) to the convergence $\mathbf{E}[e^{\xi\tilde{W}_i(t)}] \underset{t\to\infty}{\to} e^{\xi^2/2}$, where we recognize the Laplace transform of a Gaussian variable $\mathcal{N}(0,1)$. Which means



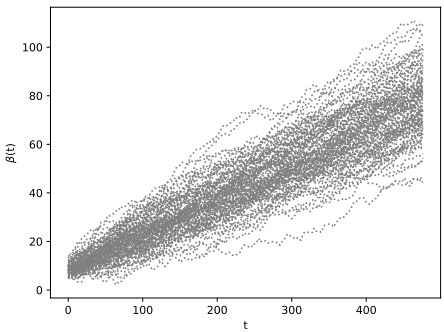
(a) $\bar{u} = 0.625, \bar{u}^{max} = 0.833$



(b) $\bar{u} = 0.838, \bar{u}^{max} = 1.208$



(c) $\bar{u} = 0.998, \bar{u}^{max} = 1.508$



(d) $\bar{u} = 1.018, \bar{u}^{max} = 1.608$

Figure 3.2: The backlog process of systems with different mean utilization, initialized with $\bar{W}(0) = \sum_{\tau_i \in \Gamma} C_i$

that $\lim_{t\to\infty} \frac{W_i(t)-u_i t}{\sqrt{t}} \sim \Phi_{0,v_i^2}$. Taking nt instead of t gives us the result. Finally, we conclude with (2.12) that gives us $n^{-1}W_i(nt) = t\frac{W_i(nt)}{nt} \underset{n\to\infty}{\to} ut$. Combine those two to get

$$W_i^{(\infty)}(t) - W_i^{(\infty)}(0) \overset{(d)}{=} u_i t + v_i B(t)$$

where B is a standard Brownian motion.

The demand processes $W_1^{(\infty)}, \ldots, W_n^{(\infty)}$ are independent, thus we get that $\bar{W}_k^{(\infty)} = \sum_{i=1}^{k} W_i^{(\infty)}$ is the sum of k independent Brownian motions, which is also a Brownian motion. Finally we get that for each $t > 0$, $\bar{W}_k^{(\infty)}(t) - \bar{W}_k^{(\infty)}(0) \sim \Phi_{\bar{u}_k t, \bar{v}_k^2 t}$. □

In [Lehoczky, 1996], the author uses the *heavy-traffic approximation* providing the distribution of the lateness of jobs in a system with exponential inter-arrival times. To illustrate the heavy-traffic approximation, one can think of water continuously flowing into a sink at a rate λ_i and the execution times as the rate $1/\mathbf{E}[C_i]$ the water

leaves the sink. It is usually understood as true when the system is at full processor utilization, because the theorems of heavy-traffic theory are exact when $\bar{u} \to 1$. However we use it as a way to build an upper-bound of the backlog process. Indeed, the heavy-traffic assumption should be seen as a bound, or more specifically, a way to suppose that the system utilization is at its maximum (*i.e.*, $\bar{u} = 1$), providing *upper-bounds* that are exact when the processor utilization at 100%. Figure 3.3b illustrates this upper-bound becoming exact in Figure 3.3c and Figure 3.3d.

In this section, we use the heavy-traffic assumption to find the steady-state backlog $\tilde{\beta}_k$ as an approximation upper-bounding the *blocking time* of the system in its steady-state, see Eq. (4.4). The approximation in Theorem 3.5 leads to the standard Brownian motion which is continuous. It means that instead of looking at the demand for a large amount of time, we consider a re-scaled version of the demand in order to build a good approximation. Theorem 3.5 can be written as

$$\bar{W}_k^{(\infty)}(t) = \bar{W}_k(0) + \bar{u}_k t + \bar{v}_k B(t) \tag{3.10}$$

where B is a standard Brownian motion, as defined in Eq. (3.1).

In order to find the steady-state backlog $\tilde{\beta}_k$, we work with the heavy-traffic demand $\bar{W}_k^{(n)}$.

Proposition 3.2. *The rescaled backlog process of level k is defined by*

$$\beta_k^{(n)}(t) = \bar{W}_k^{(n)}(t) - t + \sup_{s\in[0,t]} (s - \bar{W}_k^{(n)}(s))^+ \tag{3.11}$$

Proof. From its definition in Eq (3.8), we have

$$\begin{aligned}
\beta_k^{(n)}(t) &= \bar{W}_k(0) + \frac{\bar{W}_k(nt)}{n} - \frac{1}{n}\int_0^{nt} \mathbf{1}_{\beta_k(s)>0} ds + \frac{1}{\sqrt{n}}\left(\bar{W}_k(nt) - \int_0^{nt} \mathbf{1}_{\beta_k(s)>0} ds - (\bar{u}_k - 1)nt\right) \\
&= \bar{W}_k(0) + \frac{\bar{W}_k(nt)}{n} - \frac{1}{n}\int_0^{nt} \mathbf{1}_{\beta_k(s)>0} ds + \sqrt{n}\left(\frac{\bar{W}_k(nt)}{n} - \bar{u}_k t\right) - \frac{1}{\sqrt{n}}\int_0^{nt} \mathbf{1}_{\beta_k(s)>0} ds + \sqrt{n}t \\
&= \bar{W}_k^{(n)}(t) - \frac{1}{n}\left(nt - \int_0^{nt} \mathbf{1}_{\beta_k(s)=0} ds\right) - \frac{1}{\sqrt{n}}\left(nt - \int_0^{nt} \mathbf{1}_{\beta_k(s)=0} ds\right) + \sqrt{n}t \\
&= \bar{W}_k^{(n)}(t) - t + (1 + \frac{1}{\sqrt{n}})\frac{1}{\sqrt{n}}\int_0^{nt} \mathbf{1}_{\beta_k(s)=0} ds
\end{aligned} \tag{3.12}$$

Then let $X^{(n)}(t) = \bar{W}_k^{(n)}(t) - t$ and $Y^{(n)}(t) = (1+\frac{1}{\sqrt{n}})\frac{1}{\sqrt{n}}\int_0^{nt} \mathbf{1}_{\beta_k(s)=0} ds$. $\beta_k^{(n)}, X^{(n)}$ and $Y^{(n)}$ should satisfy $(S1)-(S3)$, thus by applying Lemma 3.2 to $X^{(n)}$ wet get that the pair $(\beta_k^{(n)}, Y^{(n)})$ is unique and $Y^{(n)}(t) = \sup_{s\in[0,t]}(s - \bar{W}_k^{(n)}(s))^+$. □

Theorem 3.6. *Consider the same hypotheses as Theorem 3.5 and let $\bar{u}_k < 1$. The heavy-traffic backlog process $\beta_k^{(\infty)} = \lim_{n\to\infty} \beta_k^{(n)}$ is a reflected Brownian motion of drift $\bar{u}_k - 1$ and deviation $\bar{v}_k$, and*

$$\beta_k^{(\infty)}(t) = \bar{W}_k(t) - t + \sup_{s\in[0,t]} (s - \bar{W}_k^{(\infty)}(s))^+$$

Proof. We have

$$\begin{aligned}\beta_k^{(\infty)}(t) &= \lim_{n\to\infty} \bar{W}_k^{(n)}(t) - t + \lim_{n\to\infty}(1+\frac{1}{\sqrt{n}})\frac{1}{\sqrt{n}}\int_0^{nt} \mathbf{1}_{\beta_k(s)=0} ds \quad \text{(From Eq. (3.12))} \\ &= \bar{W}_k^{(\infty)}(t) - t + \lim_{n\to\infty} \frac{1}{\sqrt{n}}\int_0^{nt} \mathbf{1}_{\beta_k(s)=0} ds\end{aligned}$$

Finally, let $Y(t) = \lim_{n\to\infty} \frac{1}{\sqrt{n}}\int_0^{nt} \mathbf{1}_{\beta_k(s)=0} ds$. $(\beta_k^{(\infty)}, Y)$ satisfy $(S1)-(S3)$ from Lemma 3.2, hence $Y(t) = \sup_{s\in[0,t]}(s - \bar{W}_k^{(\infty)}(s))^+$. $\bar{W}_k^{(\infty)}(t) - t, t \geqslant 0$ being a Brownian motion of drift $\bar{u}_k - 1$ and deviation $\bar{v}_k$, we conclude with Definition 3.3.

□

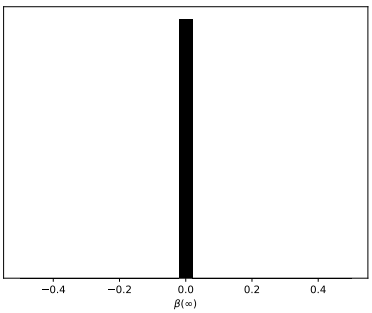
It is shown in [Chen and Yao, 2001, Remark 6.17, p. 148], that the error of the estimation stated in Eq. (3.11) is

$$\sup_{s\in[0,t]} |\beta_k(s) - \beta_k^{(\infty)}(s)| = \mathcal{O}\left((t\log\log t)^{1/4}(\log t)^{1/2}\right) \tag{3.13}$$

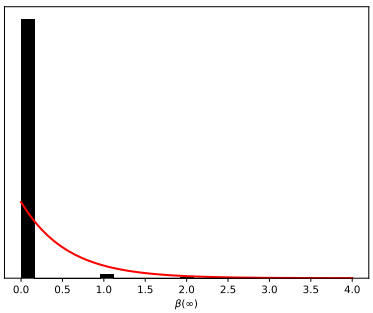
Proposition 3.3. *Consider the same hypotheses as Theorem 3.5, with exponential inter-arrival times of rates $\lambda_i, i = 1,\ldots,k$. Let $\bar{u}_k < 1$. Consider the parameter*

$$\eta_k = \frac{2(1-\bar{u}_k)}{\bar{v}_k^2} \tag{3.14}$$

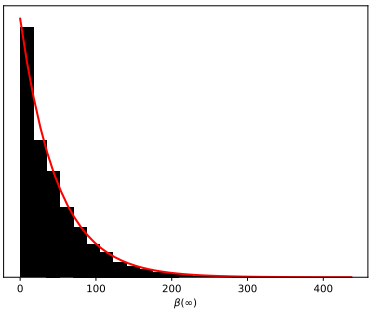
Then, the distribution function π_k of the steady-state backlog $\tilde{\beta}_k$ is defined for $x > 0$



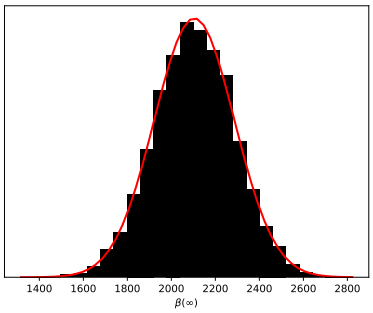
(a) $\bar{u} = 0.625, \bar{u}^{max} = 0.833$



(b) $\bar{u} = 0.838, \bar{u}^{max} = 1.208$



(c) $\bar{u} = 0.998, \bar{u}^{max} = 1.508$



(d) $\bar{u} = 1.018, \bar{u}^{max} = 1.608$

Figure 3.3: Steady-state backlog simulations with different mean utilizations in the Diaz and Kim model. In black the histogram of simulations of $\beta(n)/\sqrt{n}$ for $n = 10000$, in red the probability function of $\tilde{\beta}$.

by

$$\pi_k(x) = 1 - \exp\left(-\eta_k x\right) \tag{3.15}$$

Proof. Apply Theorem 3.2 and Theorem 3.6.

□

3.3 Periodic backlog

We consider in this section a stationary real-time system $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ following the $\sum_i D_i / \sum_i G_i / 1 / SP$ queueing model. In opposition to memoryless scheduling, periodic systems use knowledge of the past to take scheduling decisions. In the previous section do not treat the periodic case, although we show in this section some exisiting results and build heuristics for the periodic case in the next chapter.

For the case where the tasks of the real-time system Γ are periodic and have deterministic deadlines, *i.e.*, $T_k = \lambda_k^{-1} \in \mathbb{N}$ and $G_k(x) = \mathbf{1}_{[\lambda_k^{-1},\infty)}(x)$ for all $\tau_k \in \Gamma$ and $x > 0$, also known as the Diaz and Kim (DK) model [Díaz et al., 2002, Kim et al., 2005, Díaz et al., 2004], the authors approximate the distribution function π_k, resolving linear system equations and compute response times distributions with the help of convolutions.

The fastest computational complexity of convolutions is $\mathcal{O}(N \log N)$ when N is the number of values that a probabilistic variable can take. Computing the exact values of π_k quickly becomes an expensive operation when the number of tasks or the number of possible execution times gets larger, even with methods that soften those computations like Markovic *et al.* [Marković et al., 2021], Milutinovic *et al.* [Milutinovic et al., 2015] or Maxim *et al.* [Maxim et al., 2012] for example. Moreover, the computation of response times has the same problem, as the number of possible values of $\tilde{\beta}_k$ quickly becomes large. With Eq. (3.16) we have an explicit formula of the distribution of $\tilde{\beta}_k$, with Eq. (3.13) we know the error of the heavy-traffic approximation, and with Theorem 3.1 we have an analytical expression of the backlog in the deterministic case.

Indeed, let $\bar{T}_k = \operatorname{lcm}\left(\lambda_1^{-1}, \ldots, \lambda_k^{-1}\right)$ be the *hyper-period* of level k of Γ. In the DK model [Díaz et al., 2002, Kim et al., 2005], the authors consider the k-level backlog $\beta_k(t\bar{T}_k), t \in \mathbb{N}$, *i.e.*, the remaining demand of level k at the beginning of the t-th hyper-period. Diaz *et al.* [Díaz et al., 2002] have proven that the sequence $\left(\beta_k\left(t\bar{T}_k\right)\right)_{t\in\mathbb{N}}$ is a *stationary* Markov chain when $\bar{u}_k < 1$. The sequence $\left(\beta_k(t\bar{T}_k)\right)_{t\in\mathbb{N}}$ is defined by $\bar{W}_k(0) \geqslant 0$ and

$$\beta_k\left((t+1)\bar{T}_k\right) = \left(\beta_k(t\bar{T}_k) + \bar{W}_k(\bar{T}_k) - \bar{T}_k\right)^+$$

for $t \in \mathbb{N}$.

Similarly to memorlyess systems, we get the asymptotic backlog as follows.

Proposition 3.4. *Let* $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ *follow a* $\sum_i D_i / \sum_i G_i / 1 / SP$ *stationary queue-*

ing model. Let $\bar{T}_k = \operatorname{lcm}\left(\lambda_1^{-1}, \ldots, \lambda_k^{-1}\right)$ *be the hyper-period of level* k *of* Γ*. Let*

$$W_{k,l} = \sum_{i=1}^{k} \sum_{j=N_i((l-1)\bar{T}_k)}^{N_i(l\bar{T}_k)} C_{i,j}$$

be the demand of level k *released during the* l*-th hyperperiod, i.e., the interval* $[(l-1)\bar{T}_k, l\bar{T}_k]$*. Then if* $\bar{u}_k < 1 - \frac{k}{2\bar{T}_k}$ *the steady-state backlog* $\tilde{\beta}_k$ *exists, is finite and is equal to*

$$\tilde{\beta}_k = \left(\sup_n \sum_{l=1}^{n} W_{k,l} - n\bar{T}_k \right)^+ \tag{3.16}$$

where $x^+ = \max(0, x)$*, c.f., [Baccelli and Brémaud, 2013, Property 2.2.1] and Theorem 3.3. In addition, there is an infinite number of idle times, c.f., [Baccelli and Brémaud, 2013, Property 2.2.5].*

Proof. First of all, the Example 3.1.3 [Baccelli and Brémaud, 2013] shows that a stationary point process with priority class jobs is still associated to a point process of intensity $\bar{u}_k$. Next, we observe that for periodic systems, the workload released in the intervals $[(l-1)\bar{T}_k, l\bar{T}_k], l \in \mathbb{N}$ are i.i.d.. Indeed,

$$\begin{aligned}
W_{k,l} &= \sum_{i=1}^{k} \sum_{j=N_i((l-1)\bar{T}_k)}^{N_i(l\bar{T}_k)} C_{i,j} \\
&\overset{(d)}{=} \sum_{i=1}^{k} \sum_{j=1}^{N_i(l\bar{T}_k)-N_i((l-1)\bar{T}_k)} C_{i,j} && (C_{i,j}, j \geqslant 1 \text{ are i.i.d.}) \\
&\overset{(d)}{=} \sum_{i=1}^{k} \sum_{j=1}^{N_i(\bar{T}_k)} C_{i,j} && (N_i\text{'s are stationary}) \\
&\overset{(d)}{=} W_{k,1}
\end{aligned}$$

Then, in order to use Theorem 3.3 "at hyperperiod level" with the $W_{k,l}$ as execution times and $\bar{T}_k$ as inter-arrival times, we need to check $\mathbf{E}[W_{k,1}] < \bar{T}_k$. We

have

$$\begin{aligned}
\mathbf{E}[W_{k,1}] &= \sum_{i=1}^{k} \mathbf{E}[N_i(\bar{T}_k)]\mathbf{E}[C_i] && \text{(According to Lemma 2.4)} \\
&= \sum_{i=1}^{k} \mathbf{E}[\lfloor \lambda_i(O_i + T_k) \rfloor]\mathbf{E}[C_i] \\
&\leqslant \sum_{i=1}^{k} (\frac{1}{2} + \lambda_i \bar{T}_k)\mathbf{E}[C_i] && \text{(Because } O_i \text{ is uniform in } [0, \lambda_i^{-1}]) \\
&= \frac{k}{2} + \bar{u}_k \bar{T}_k
\end{aligned}$$

which leads to the condition $\frac{k}{2(1-\bar{u}_k)} < \bar{T}_k$. We conclude with Theorem 3.3. □

The representation in Eq. (3.16) is an efficient method to approximate the stationary distribution π_k of the backlog process β_k. Indeed, let us take an integer $n > 0$, and generate a sample $(W_{k,l})_{l=1}^{n}$ independent and identically distributed sequence with the distribution of $\bar{W}_k\left(\bar{T}_k\right)$. Eq. (3.16) provides the variable of distribution π_k found in Diaz *et al.* [Díaz et al., 2002] and Kim *et al.* [Kim et al., 2005]. It also means that the variable $\max_{1 \leqslant j \leqslant n} \left(\sum_{l=1}^{j} \left(W_{k,l} - \bar{T}_k\right)\right)^+$ is an approximation of $\tilde{\beta}_k$ when n is large enough. This method is not expensive in complexity as it requires only to build the distribution function of $\bar{W}_k\left(\bar{T}_k\right)$ once.

In the periodic case, we are able to find a bound of deadline miss probabilities of the RM policy with the following.

Proposition 3.5 (Hoeffding DMP for periodic inter-arrival times with Rate Monotonic)**.** *Suppose that jobs of τ_i arrive periodically with rate λ_i. If $\bar{u}_k < 1$ and $1/\lambda_k > \frac{\sum_{i=1}^{k} \mathbf{E}[C_i]}{2(1-\bar{u}_k)}$ then*

$$\mathbf{P}(R_k^{max} > 1/\lambda_k) \leqslant \exp\left(-\frac{(1-\bar{u}_k)^2}{\lambda_k \bar{v}_k^{max}}\right)$$

where $\bar{u}_k = \sum_{i=1}^{k} \lambda_i \mathbf{E}[C_i]$ is the mean utilization of level k and $\bar{v}_k^{max} = \sum_{i=1}^{k} \lambda_i (c_i^{max} - c_i^{min})^2$ is the maximum deviation of level k.

Proof. Suppose the system is periodic with rate λ_i. According to [von der Brüggen

et al., 2018, Theorem 6], the Hoeffding inequality applied to a static-priority policy gives us

$$\mathbf{P}(R_k^{max} > 1/\lambda_k) \leqslant \inf_{\substack{t\in(0,1/\lambda_k)\\ t>\mathbf{E}[\bar{W}_k(t)]}} \exp\left(-2\frac{(t-\mathbf{E}[\bar{W}_k(t)])^2}{\sum_{i=1}^{k}(c_i^{max}-c_i^{min})^2 N_i(t)}\right) \tag{3.17}$$

where $N_i(t) = \lceil \lambda_i t \rceil$ is the number of jobs of the task τ_j released before $t > 0$ when all tasks are activated at $t = 0$, *i.e.*, $O_i = 0, i = 1, \ldots, k$. According to Lemma 2.4 we have $\mathbf{E}[\bar{W}_i(t)] = \sum_{j=1}^{i} \mathbf{E}[N_j(t)]\mathbf{E}[C_j]$. Since

$$\lambda_i t \leqslant \mathbf{E}[N_i(t)] \leqslant \lambda_i t + \frac{1}{2} \tag{3.18}$$

and $\bar{u}_k < 1$, we have the relation $\bar{u}_k t + \frac{1}{2}\sum_{i=1}^{k}\mathbf{E}[C_i] \geqslant \mathbf{E}[\bar{W}_k(t)] \geqslant \bar{u}_k t$. Hence, $t > \frac{\sum_{i=1}^{k}\mathbf{E}[C_i]}{1-\bar{u}_k}$ implies $t > \mathbf{E}[\bar{W}_k(t)]$. Suppose $1/\lambda_k > \frac{\sum_{i=1}^{k}\mathbf{E}[C_i]}{2(1-\bar{u}_k)}$ and $t \in (\frac{\sum_{i=1}^{k}\mathbf{E}[C_i]}{2(1-\bar{u}_k)}, 1/\lambda_k)$. With Eq. (3.18) we get

$$\frac{(t-\mathbf{E}[\bar{W}_k(t)])^2}{\sum_{i=1}^{k}(c_i^{max}-c_i^{min})^2 N_i(t)} \geqslant \frac{t(1-\bar{u}_k)^2}{\bar{v}_k^{max} + t^{-1}\sum_{i=1}^{k}(c_i^{max}-c_i^{min})^2} \tag{3.19}$$

Finally the infinimum in Eq. (3.17) is reached for $t = 1/\lambda_i$, and we are using the RM policy, thus we have $\lambda_i \leqslant \lambda_j$ for $j \leqslant i$ since we assume working under the RM policy, hence we get

$$\bar{v}_k^{max} + \lambda_k \sum_{i=1}^{k}(c_i^{max}-c_i^{min})^2 \leqslant 2\bar{v}_k^{max} \tag{3.20}$$

which gives us the result with Eq. (3.19). □

3.4 Schedulability test

The difference between a non-discarding schedule and a discarding one is significant. Indeed, the analysis provided in this chapter is agnostic from discarding jobs.

However, the non-discarding and the discarding schedule of a schedulable task set Γ are the same, because all deadlines are satisfied a schedulable task set. Based on this fact, the appropriate way to test if there will be discarded jobs in steady-state is to check if the worst-case blocking time and the steady-state backlogs satisfy

$$\forall i, \mathbf{P}\left(\tilde{\beta}_i \leqslant \sum_{j=1}^{i} c_j^{max}\right) > 1 - \varepsilon$$

for a small enough ε, which means that the discarding policy and the non-discarding policies are equivalent for all tasks.

3.5 Potential extensions

3.5.1 Extension to EDF and FIFO

The first step into dynamic scheduling, as for example EDF is to study the backlog processes considering that the priority is at the job level (in opposition to task level). Indeed, for static-priority policies, those variables are simply the backlog and demand of the lowest priority level. However, for EDF, levels of priority need to be defined not only for tasks but for jobs. In [Díaz et al., 2002], authors use the concept of *ground jobs* which are jobs released at an instant where the system is idle, and as shown in the Loynes theorem, when $\bar{u} < 1$, idle times are finite. This means that an analysis mixing the concept of ground jobs and idle times as defined in this thesis can provide an extension of our results for dynamic-priority scheduling. In [Lehoczky, 1996] authors show that if $\bar{u} < 1$, $\beta_k^{(\infty)}$ is a reflected Brownian motion of drift $-\gamma$ and deviation $2\sum_i \lambda_i$ where $0 < \gamma < 1$ is such that for each $n > 0$, the utilization of the process $(\beta_k(nt)/\sqrt{n})_t$ can be written $1 - \gamma/\sqrt{n}$, for the EDF and FIFO policies. This means that even without showing the exact parameters of the first-passage time distributions, we can assume that response times belong in the same domain of distribution functions.

3.5.2 Extension to general stationary inter-arrival times

We studied two types of stationary task sets with static priorities: periodic and memoryless. However, considering renewal processes with other distributions than exponential for inter-arrival times is challenging. The main reason being that the superposition of stationary renewal processes are not renewal processes in general because inter-arrival times of each task become inter-dependent. We present in this section some theoretical background that could help generalize the results of this chapter. Let us recall that the stationary renewal process N_i counts the number of jobs of τ_i through time, and $N = \sum_i N_i$ counts all jobs regardless of which task those jobs are.

Palm-Khintchin theorem The first property of renewal processes that could be use to generalize the results of this chapter is the equivalent of the central limit theorem for renewal processes.

Theorem 3.7 (Palm-Khintchine)**.** *Let $N_1, \dots, N_n$ be n independent renewal processes with distinct inter-arrival rates λ_i. Then the superposition $N = \sum_{i=1}^{n} N_i$ is asymptotically a Poisson process when $n \to \infty$, if the following assumptions hold:*

(i) $\sum_{i=1}^{n} \lambda_i < \infty$ *when* $n \to \infty$,

(ii) $\lambda_i \leqslant \frac{1}{n} \sum_{i=1}^{n} \lambda_i, \forall i$

Then N converges to a Poisson process of inter-arrival rate $\lambda = \bar{\lambda}_\infty$ when $n \to \infty$.

However this convergence can occur very slowly and might not be suited for all systems. This property is used in telecommunications and IoT, *e.g.*, [Metzger et al., 2019], and is the reason why Poisson processes are a widely used and studied renewal process.

Approximating the superposition with a stationary renewal processes A result from 2001 [Torab and Kamen, 2001] shows that approximating the superposition N of renewal processes by supposing that inter-arrival are indeed independent is possible and that there is a method that can minimize the error of such hypotheses. What we look for is preserve the stationarity of renewal processes when they are superposed. One property of renewal processes is that any renewal process can be modified to be stationary by adding a *delay*, or as we call it an offset, with a very specific distribution.

We denote by $\bar{G}_k$ the distribution function of the first arrival of the superposed process N. First of all, we know from [Baccelli and Brémaud, 2013, Example 1.4.1, p.35] and [Lawrance, 1973] that

$$\bar{G}_k(x) = 1 - \sum_{i=1}^{k} \frac{\lambda_i}{\bar{\lambda}_k} (1 - G_i(x)) \prod_{j \neq i} (1 - G_j^0(x)) dx$$

where $G_j^0(x) = \lambda_j \int_0^x (1 - G_j(x))dx$ is the distribution of the offset of τ_j as defined in (2.8). We call G_j^0 the distribution of the recurrent time of the process N_j, and suppose that the superposition is still stationary, hence characterizing the distribution of this first job arrival is enough to characterize all jobs arrivals.

The intensity used to approximate the distribution of N in [Torab and Kamen, 2001] comes from the recurrent times instead of the inter-arrival times. By remarking that $\frac{d}{dt}G_j^0 = \lambda_j(1 - G_j)$, and setting $g_i = \frac{d}{dt}G_i$ we get the intensity

$$\nu_i = \lambda_i \frac{1 - G_i}{1 - G_j^0}$$

instead of $\mu_i = \frac{g_i}{1-G_i}$. Finally, this new intensity is actually shown to minimize to quadratic error, and is of the form

$$\nu^*(t) = \frac{\sum_i \nu_i(t)(\mu_i(t) + \sum_{j \neq i} \nu_j(t))}{\sum_i \nu_i(t)}$$

The mean of the first arrival time is

$$\begin{aligned} \bar{m}_k = E[\bar{A}_{k,1}] &= \int (1 - \bar{G}_k(t))dt \\ &= \sum_{i=1}^{k} \frac{\lambda_i}{\bar{\lambda}_k} \int (1 - G_i(t)) \prod_{j \neq i} (1 - G_j^0(t))dt \end{aligned}$$

and by noticing that

$$\frac{d}{dt}\bar{G}_k = \sum_{i=1}^{k} \frac{\lambda_i}{\bar{\lambda}_k}(1 - G_i) \prod_{j \neq i} (1 - G_j^0) \left(\frac{g_i}{1 - G_i} + \sum_{j \neq i} \lambda_j \frac{1 - G_j}{1 - G_j^0} \right)$$

its variance is

$$\mathrm{Var}(\bar{A}_{k,1}) = \int (x - \bar{m}_k)^2 d\bar{G}_k(t)$$
$$= \sum_{i=1}^{k} \frac{\lambda_i}{\bar{\lambda}_k} \int (x - \bar{m}_k)^2 (1 - G_i(t)) \prod_{j \neq i} (1 - G_j^0(t)) \left(\frac{g_i(t)}{1 - G_i(t)} + \sum_{j \neq i} \lambda_j \frac{1 - G_j(t)}{1 - G_j^0(t)} \right) dt$$

which leads to the squared coefficient of variation $\gamma_{a,k}^2$ equal to

$$\sum_{i=1}^{k} \frac{\lambda_i}{\bar{\lambda}_k} \int (\frac{x}{\bar{m}_k} - 1)^2 (1 - G_i(x)) \prod_{j \neq i} (1 - G_j^0(x)) \left(\frac{g_i(x)}{1 - G_i(x)} + \sum_{j \neq i} \lambda_j \frac{1 - G_j(x)}{1 - G_j^0(x)} \right) dx$$

which would allow to provide an approximation for any stationary task set. We test the hypotheses that we can actually use this same method in the following chapter.

4 | Response time approximation using fluid models

Contents

Probabilistic methods for the analysis of response times have many applications in real-time systems [Davis and Cucu-Grosjean, 2019]. Two main directions have been explored: static methods for the exact computation and approximation of response time distributions [Díaz et al., 2002, Kim et al., 2005, Maxim and Cucu-Grosjean, 2013, Manolache et al., 2001] *a priori*, and the measurement-based application of the EVT method [Liu et al., 2013, Lu et al., 2012] approximating the distribution of the maximum values of response times *a posteriori*. Often, probabilities are considered for execution times and few papers consider probabilistic inter-arrival times and

deadlines [Maxim and Cucu-Grosjean, 2013, Lehoczky, 1996, Gaujal et al., 2020b]. Moreover, the method introduced in [Díaz et al., 2002, Kim et al., 2005] requires a large amount of convolutions which have a high space and time complexity, and the analysis provided by Lehoczky [Lehoczky, 1996] is not suited to express response time distributions of static-priority scheduling.

The contributions of this chapter are based on queueing theory results [Sparaggis and Towsley, 1994, Huang et al., 2015, Sethuraman and Squillante, 1999, Baccelli and Brémaud, 2013, Chen and Yao, 2001]. To the best of our knowledge, no result from the queueing theory is focused on general execution times and inter-arrival times, multi-class clients (*i.e.*, different tasks) and the quantization of deadline misses of such systems. The only results based on queueing theory for real-time systems have been published within the thread of papers related to [Lehoczky, 1996], where the author approximates the number of simultaneously activated jobs by a *reflected at the origin Brownian motion*. There is a proportional relation between the number of activated jobs and the workload of a system by applying the *Little formula* [Baccelli and Brémaud, 2013, Eq. (3.1.16)].

However, the author makes strong hypotheses restricting the model. The exponential distribution of inter-arrivals and execution times suggested in [Lehoczky, 1996] is a strong hypothesis. Furthermore, his model has another important limitation as it considers systems of jobs of only one task, which does not allow a response time analysis. To overcome this limitation, we consider a *multi-class* analysis describing tasks with parameters with different distributions. Thus, we extend the model proposed by [Lehoczky, 1996] and consider a more general case: all jobs are instances of various independent tasks scheduled with a static-priority policy.

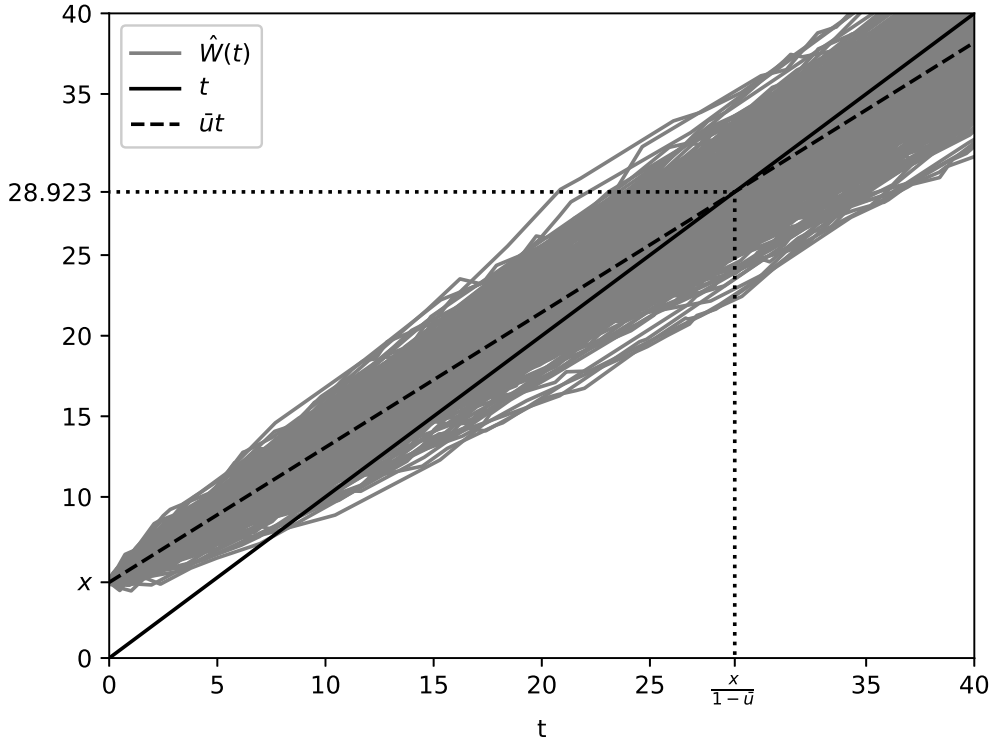


Figure 4.1: Trajectories of $\bar{W}^{(\infty)}$ and average of the first idle time for $\bar{u} = 0.838$ and $x = 4.85$.

4.1 Heavy-traffic approximation

In the following, we consider the conditional probability that the system starts with a level k demand $x \geqslant 0$ and the job $\tau_{k,1}$,

$$\mathbf{P}_k^x(\cdot) = \mathbf{P}(\cdot \mid \bar{W}_k(0) = x, O_k = 0)$$

Whenever we need to suppose $\bar{W}_k(0) = x, O_k = 0$, we say that we work *under the probability* $\mathbf{P}_k^x$.

As previously defined, response times are idle times. We use this representation of response times to provide an approximation using the fluid model introduced in the previous chapter.

Definition 4.1 (Heavy-traffic response time)**.** *Let* $R_{k,l}^{(n)}$ *be the first idle time after*

the arrival of the l-th job of the task τ_k in the sequence of $M/G/1$ queues defined in Eq. (3.8), i.e.,

$$R_{k,l}^{(n)} = \inf\{t > 0 : \beta^{(n)}(A_{k,l} + t) = 0\}$$

The heavy-traffic response time does not take discarding into account, but as shown in the following, it bounds the response times of each $M/G/1$ queue of the rescaled sequence of queueing models.

Lemma 4.1 (Heavy-traffic response time bound)**.** *Let $(R_{k,l}^{(n)})_n$ be the sequence of response times defined in Definition (4.1). Let $(B_{k,l}^{(n)})_n$ be the associated sequence of blocking time processes. Then,*

$$\inf\{t > 0 : B^{(n)}(A_{k,l} + t) = 0\} \leqslant_{st} R_{k,l}^{(n)}$$

Now that we know that the sequence of response times considering the discarding policy are bounded by the heavy-traffic response time sequence, we look in the following at the limit of this sequence in order to use the Brownian approximation introduced in Chapter 3.

4.1.1 First idle time

We study in this section the first idle time of the limit demand process, *i.e.*,the first-passage to 0 of a Brownian motion.

Definition 4.2 (Idle time)**.** *Let $\mathcal{I}_k(x)$ be the first idle time of level k, i.e.,*

$$\mathcal{I}_k(x) = \inf\left\{t > 0 : \beta_k^{(\infty)}(t) = 0\right\} \tag{4.1}$$

when the initial demand of level k is equal to $x > 0$.

Lemma 4.2. *The distribution of $\mathcal{I}_k(x)$ is an inverse Gaussian distribution with probability function*

$$\psi_k(t, x) = \psi\left(t; \frac{x}{1 - \bar{u}_k}, \frac{x^2}{\bar{v}_k^2}\right), t > 0 \tag{4.2}$$

where ψ is given in Definition 2.12, and exceedence function

$$\Psi_k(t,x) = \mathbf{P}(\mathcal{I}_k(x) > t) = \Psi(x,t;\bar{u}_k,\bar{v}_k)$$

where

$$\Psi(x,t;u,v) = \Phi\left(-\frac{(1-u)t-x}{v\sqrt{t}}\right) - e^{-2x\frac{1-u}{v^2}}\Phi\left(-\frac{(1-u)t+x}{v\sqrt{t}}\right) \tag{4.3}$$

Proof. The idle time $\mathcal{I}_k(x)$ defined in Eq. (4.1) is a quantity called *first-passage time* of a Brownian motion [Molini et al., 2011, Eq. (28)]. Until the first idle time, we know that $\beta_k^{(\infty)}(t) = x + \bar{W}_k^{(\infty)}(t) - t$, because $\mathbf{1}_{\beta_k(s)>0} = 1$ for $0 \leqslant s < t \leqslant \mathcal{I}_k(x)$. Then from Eq. (3.10) we have $\mathcal{I}_k(x) \sim \inf\{t > 0 : B(t) = t(1-\bar{u}_k)/\bar{v}_k - x/\bar{v}_k\}$ where B is a standard Brownian motion. When $\bar{W}_k(0) = x$, the distribution of $\mathcal{I}_k(x)$ is an inverse Gaussian distribution of mean $x/(1-\bar{u}_k)$ and shape $x^2/\bar{v}_k^2$ according to Theorem 3.1. See Figure 4.1 and [Jeanblanc et al., 2009, p. 146], for more details. □

4.1.2 Heavy-traffic time demand analysis

Response times depend on properties of real-time systems such as the scheduling policy, the preemptiveness, *etc.*. Our motivation is to exploit those properties leading response times to the domain of a certain probability distribution. In this section, the inverse Gaussian distribution is emphasized as the appropriate distribution for an approximation of response times in the context of static-priority scheduling policies, using asymptotic results of queueing theory, *c.f.*, Propositions 4.7. We propose two different approximations, a worst-case approximation before the system is in its *steady-state* (*c.f.*, Proposition 4.5, see Definition 3.4) and another one when the system is *steady*, *c.f.*, Propositions 3.3 and 4.6.

Let $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\})$ be a stationary real-time system, with Γ ordered by decreasing priority order, *i.e.*, τ_i has priority over τ_{i+1}.

According to Proposition 2.1, the blocking time is such that

$$B_k(t) \leqslant \min\left(\beta_k(t), \sum_{i=1}^{k} C_{i,N_i(t)}\right) \tag{4.4}$$

for all $t > 0$, which makes our TDA build upper-bounds of response times according to Theorem 2.5. This property is used in Lemma 4.1 to provide a sequence of pessimistic response time analysis. We now look at the limit of those.

Proposition 4.1. *For all $x > 0$,*

$$\mathbf{P}\left(R_{k,l}^{(n)} > t \mid \beta_k^{(n)}(A_{k,l}) = x\right) \underset{n\to\infty}{\to} \mathbf{P}(\mathcal{I}_k(x) > t) \tag{4.5}$$

Proof. We have shown in Theorem 3.6 that $\beta^{(n)}$ converges in distribution to $\beta^{(\infty)}$, thus we conclude that conditionnaly to $\bar{W}_k(0) = x$ and $O_k = 0$,

$$R_{k,1}^{(n)} = \inf\{t > 0 : \beta_k^{(n)}(t) = 0\}$$

converges in distribution to $\mathcal{I}_k(x)$. Then from Eq. (4.8), any job $\tau_{k,l}$ has a heavy-traffic response time distribution that can be expressed from the response time $R_{k,1}^{(\infty)}$,

$$\mathbf{P}\left(R_{k,l}^{(\infty)} > t \mid \beta_k^{(\infty)}(A_{k,l}) = x\right) = \mathbf{P}_k^x\left(R_{k,1}^{(\infty)} > t\right) \tag{4.6}$$

Let us condition this probability for specific values of execution times. As stated in Eq. (4.6), the proper conditioning on backlogs provides the distribution of the response time of $\tau_{k,1}$. Furthermore, when the backlog $\beta_k^{(\infty)}(A_{k,l}) = x$, the response time $R_{k,l}^{(\infty)}$ is the time it takes for all level k jobs to finish plus the time it takes for level k to stay idle for x instants in the interval $[A_{k,l}, A_{k,l} + R_{k,l}]$. It means that we can artificially set the initial demand to x and look at $\mathcal{I}_k(x)$, the first idle time of level k, as represented in Figure 4.2. In other words,

$$\mathbf{P}_k^x\left(R_{k,1}^{(\infty)} > t\right) = \mathbf{P}_k^x\left(\sup_{s\in[0,t]} s - \bar{W}_k^{(\infty)}(s) < 0\right) = \mathbf{P}\left(\mathcal{I}_k(x) > t\right) \tag{4.7}$$

which is sufficient to conclude. □

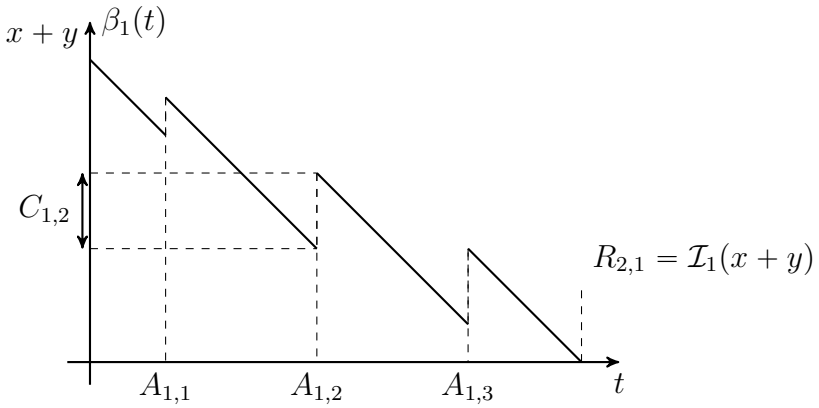


Figure 4.2: Representation of the response time $R_{2,1}$ as an idle time when $O_2 = 0, C_{2,1} = y$ and $\beta_1(0) = x$, in Example 2.1

Proposition 4.1 provides an analytical expression of this result for the heavy-traffic response time $R_{2,1}^{(\infty)}$.

The heavy-traffic demand $\bar{W}_k^{(\infty)}$ is a Brownian motion, hence $\bar{W}_k^{(\infty)}(t + A_{k,l}) - \bar{W}_k^{(\infty)}(A_{k,l}) \overset{(d)}{=} \bar{W}_k^{(\infty)}(t) - \bar{W}_k^{(\infty)}(0)$. Furthermore, $\bar{W}_k^{(\infty)}$ is continuous. Thus we define the heavy-traffic response time of a job $\tau_{k,l}$ as

$$R_{k,l}^{(\infty)} = \inf\left\{t > 0 : \beta_k^{(\infty)}(A_{k,l}) + W_k^{(\infty)}(t) - \bar{W}_k^{(\infty)}(0) = t\right\} \tag{4.8}$$

The Markovian property of Brownian motions allows to approximate the distribution of any response time $R_{k,l}$ in terms of the backlog $\beta_k^{(\infty)}(A_{k,l})$ and the first response time $R_{k,1}^{(\infty)}$, thus response times will be conditioned to backlogs and execution times, and represented as idle times following the inverse Gaussian distribution. In a second part, we provide an analytical expression of the heavy-traffic WCRT distribution, and in a third part we do the same for the steady-state heavy-traffic response time distribution. Finally, we explain how to simulate heavy-traffic response times of a task τ_k from the distribution functions F_k and G_k.

4.1.3 Conditioning response times

In this section we use the Markovian property of stationary renewal processes and Brownian motions in order to characterize the distribution of $R_{k,l}^{(\infty)}$.

From Eq. (4.8) and Proposition 4.1 we establish a necessary condition of the feasibility of Γ.

Proposition 4.2. *A non-stable stationary real-time system with implicit deadlines is not feasible under a static-priority scheduling policy.*

Proof. Let $\tau_k \in \Gamma$ be such that $\bar{u}_k > 1$. The Loynes theorem 3.1 states that there is a finite number of idle times of level k, which implies with Eq. (4.8) that heavy-traffic response times get infinite at some point, *i.e.*,for all $t > 0$, $\mathbf{P}\left(R_{k,l}^{(\infty)} > t\right) = 1$ for an infinite number of jobs. In other words, there is no permitted failure rate $\alpha_k \in (0, 1)$ such that τ_k is schedulable as defined in Eq. (2.21). As we consider a static-priority scheduling policy, then the lowest level backlog is larger than all k-level backlogs $\beta_k(t)$ at any time $t > 0$. □

Remark. *The probabilistic approach has some subtleties that need to be emphasized. Proposition 4.2 is a strong result. However, one can be tempted to build a counter example showing that with a mean utilization greater than* 1*, there are actually some jobs that are schedulable. The fact that jobs are discarded when they miss their deadline is confusing for the probabilistic approach, as some could say that if we discard, then we go back to a backlog equal to zero, hence the analysis restarts again just like any other jobs. This is wrong. In order to understand better what we mean here, we should consider the system without job discarding. Then the good translation of Proposition 4.2 is as follow:*

> *There is a strictly positive probability that only a finite number of jobs satisfy their deadlines if there is no discarding policy. While this probability exists, the system is not feasible.*

Now if we see schedulability as the measure of how far the discarding system is from the non discarding system as proposed in Section 3.4, we see that the probability $\mathbf{P}(\tilde{\beta}_k \leqslant \sum_{i=1}^{k} c_i^{max})$ *is* 0 *when the mean utilization is greater than* 1*, because* $\tilde{\beta}_k = \infty$ *in that case.*

In the rest of this thesis, the central quantity is the DMP of a task. We have seen that the backlog process is the main process to look at in our model for an

end-to-end analysis. Conditioning response times to their associated backlog is the natural step in our analysis. We define

$$p_k(x) = \mathbf{P}_k^x \left(R_{k,1}^{(\infty)} > D_k \right) \tag{4.9}$$

as the DMP of τ_k conditioned to an initial demand $x \geqslant 0$.

Proposition 4.3. *Let $\bar{u}_k < 1$, $0 < \bar{v}_k < \infty$ and $\gamma_k = \left(\frac{\bar{v}_k}{1-\bar{u}_k}\right)^2$. Then*

$$p_k(x) = 1 - \exp\left(-x \frac{\sqrt{1 + 2\lambda_k \gamma_k} - 1}{\gamma_k (1 - \bar{u}_k)} \right) \tag{4.10}$$

is the DMP of any job of τ_k released with an initial demand of level k equal to x.

Proof. We have

$$\begin{aligned} p_k(x) &= \mathbf{P}\left(\sup_{t \in (0, T_k)} t - \bar{W}_k^{(\infty)}(t) \leqslant x \right) && \text{(with (4.7))} \\ &= \int \mathbf{P}(\mathcal{I}_k(x) > t) dG_k(t) \\ &= \int \Psi_k(t, x)\, \lambda_k e^{-\lambda_k t} dt \end{aligned}$$

Since $\bar{u}_k < 1$ for all $x > 0$, we have

$$\lim_{t \to \infty} \Psi_k(t, x) = \mathbf{P}_k^x \left(R_{k,1}^{(\infty)} = \infty \right) = 0 \tag{4.11}$$

Furthermore, since $1 - e^{-\lambda_k t} = 0$ when $t = 0$, we get by integration by parts that

$$\int \Psi_k(t, x)\, \lambda_k e^{-\lambda_k t} dt = \lim_{t \to \infty} \Psi_{k-1}(t, x)(1 - e^{-\lambda_k t}) - \Psi_k(0, x)(1 - e^{-\lambda_k \times 0}) + \int \psi_k(t, x)\, (1 - e^{-\lambda_k t}) dt$$

because $\frac{d}{dt}\Psi_k(t, x) = -\psi_k(t, x)$. Finally since $\int \psi_k(t, x)\, dt = 1$ because $\psi_k(\cdot, x)$ is a

probability function for all $x > 0$, we get

$$p_k(x) = 1 - \int \psi_k(t, x)\, e^{-\lambda_k t} dt$$

where we recognize the Laplace transform of $\mathcal{I}_k(x)$, *i.e.*,

$$L(s) = \mathbf{E}[\exp(-s\mathcal{I}_k(x))] = \int \psi_k(t, x)\, e^{-st} dt$$

we conclude by developing this Laplace transform of an inverse Gaussian distribution of mean $\xi = \frac{x}{1-\bar{u}_k}$ and shape $\delta = \frac{x^2}{\bar{v}_k^2}$

$$L(s) = \exp\left[-\frac{\delta}{\xi}\left(\sqrt{1 + 2s\frac{\xi^2}{\delta}} - 1\right)\right]$$

we conclude with $s = \lambda_k$. □

In Sections 4.1.4 and 4.1.5, we prove that the proper initialization of the system puts the system in two specific cases: the worst-case and the steady-state.

4.1.4 Worst-case response time

Before the system reaches its steady-state, we say it is *transient*. In that case we cannot provide the exact distribution of the backlog in an analytical formulation. However, we can bound it by using the *worst-case blocking time*. We define the WCRT of the task τ_i as the heavy-traffic response time $R_{i,1}^{(\infty)}$ initialized with the worst-case blocking time.

In the following we approximate the WCRT R_k^{max} by the heavy-traffic WCRT $\mathcal{I}_k\left(\sum_{i=1}^{k} c_i^{max}\right)$.

Proposition 4.4. *If $\bar{u}_k < 1$, for all $l \in \mathbb{N}$ we have*

$$\inf\{t > 0 : B_k(A_{k,l}) + \bar{W}_i^{(\infty)}(t) - \bar{W}^{(\infty)}(0) \leqslant t\} \leqslant_{st} \mathcal{I}_k\left(\sum_{i=1}^{k} c_i^{max}\right) \tag{4.12}$$

Proof. First, let us consider $\bar{u}_k < 1$, as stated in Theorem 3.1 the backlog process converges to $\tilde{\beta}_k$ which is finite. Jobs are discarded if they miss their deadlines, and as we consider implicit deadlines, there can be at most one job per task activated simultaneously. Indeed, at any instant and for any task τ_k, $B_k(t) \leqslant \sum_{i=1}^{k} c_i^{max}$. This leads into considering a job $\tau_{k,l}$ with blocking time $B_k(A_{k,l}) = \sum_{i=1}^{k} c_i^{max}$ as the maximum backlog of level k, *c.f.*, Eq. (4.4), and use the property stated in Eq. (4.6) with an initial demand $\bar{W}_k(0) = c_k^{max}$. According to Theorem 2.6, the solution of

$$\inf\{t > 0 : \beta_k(A_{k,l}) + \bar{W}_k(t) - \bar{W}_k(0) \leqslant t\}$$

is more pessimistic. Hence we set R_k^{max} as defined in (4.12) as the WCRT of τ_k.

When $\bar{u}_k = 1$, idle times of level k may or may not be finite, thus we cannot conclude anything on the distribution of response times, *i.e.*, $\mathbf{P}(\exists l : \beta_k(A_{k,l}) = \infty) > 0$.

When $\bar{u}_k > 1$, the largest response time does not come from the synchronous activation and is in fact ∞. As we have already demonstrated in the proof of Proposition 4.2, response times of the task τ_k increase to ∞, due to the absence of idle times of level k. Then we conclude that the heavy-traffic WCRT is ∞. In this case, a full example is detailed in [Chen et al., 2022].

□

We set the heavy-traffic WCRT as the heavy-traffic response time with a backlog equal to $\sum_{i=1}^{k-1} c_i^{max}$. Hence we get the approximation

$$\mathbf{P}(R_k^{max} > t) \approx H_k^{max}(t)$$

that we develop in the following proposition.

Proposition 4.5. *Let $\tau_k \in \Gamma$, $\bar{u}_k < 1$ and let Ψ_k be as defined in (4.3). The*

exceedence function of the heavy-traffic WCRT of τ_k is

$$H_k^{max}(t) = \Psi_k \left(t, \sum_{i=1}^{k-1} c_i^{max} \right) \tag{4.13}$$

Proof. From Proposition 4.4 we know that when $\bar{u}_k < 1$, the heavy-traffic WCRT of $\tau_k \in \Gamma$ is $R_{k,1}^{(\infty)}$ under $\mathbf{P}_k^{b_{k-1}}$ with $b_{k-1} = \sum_{i=1}^{k-1} c_i^{max}$. □

Corollary 4.1. *Let $\tau_k \in \Gamma$, $\gamma_k = \left(\frac{\bar{v}_k}{1-\bar{u}_k}\right)^2$ and suppose $\bar{u}_k < 1$. The worst-case DMP of τ_k is*

$$p_k^{max} = 1 - \exp\left(- \sum_{i=1}^{k} c_i^{max} \frac{\sqrt{1 + 2\lambda_k \gamma_k} - 1}{\gamma_k (1 - \bar{u}_k)} \right) \tag{4.14}$$

Proof. Direct consequence of $p_k^{max} = p_k \left(\sum_{i=1}^{k} c_i^{max} \right)$ and Proposition 4.5. □

4.1.5 Steady-state response time

Let us denote the conditional probability that the distribution of the initial demand $\bar{W}_k(0)$ is the probability $d\mu$ and the first job released is $\tau_{k,1}$,

$$\mathbf{P}_k^{\mu}(\cdot) = \mathbf{P}(\cdot \mid \bar{W}_k(0) \sim d\mu, O_k = 0) = \int \mathbf{P}_k^{x}(\cdot) d\mu(x)$$

The backlog process of level k being stationary and with a stationary distribution π_k, in the steady-state Eq. (4.6) becomes

$$\mathbf{P}_k^{\pi_k} \left(R_{k,l}^{(\infty)} > t \right) = \mathbf{P}_k^{\pi_k} \left(R_{k,1}^{(\infty)} > t \right) \tag{4.15}$$

for any $l \in \mathbb{N}$. Then if it holds for any $l \in \mathbb{N}$, it holds for the all response times after the convergence of the backlog process. This is why in the steady-state, the distribution of heavy-traffic response times is unique. Consider the steady-state response time

$$\tilde{R}_k = \inf \left\{ t > 0 : \tilde{\beta}_k + \bar{W}_k^{(\infty)}(t) = t \right\} \tag{4.16}$$

As in Section 4.1.4, we get the distribution function of steady-state response times.

Proposition 4.6. *Let $\tau_k \in \Gamma$, $\bar{u}_k < 1$, $\tilde{H}_k$ be the exceedence function of the steady-state response time of τ_k, and let η_k be as defined in Proposition 3.3 and Ψ_k as defined in (4.3). Then for all $t > 0$,*

$$\tilde{H}_k(t) = \eta_k \int_0^\infty \Psi_k(t,x) e^{-\eta_k x} dx$$

Proof. We know from Proposition 3.3 that the steady-state level $(k-1)$ backlog distribution is π_{k-1}. We know from Proposition 4.1 that the stead-state response time $\tilde{R}_k$ is the first idle time with an initial demand of $\tilde{\beta}_k$. Finally by definition of conditional probabilities we have $\tilde{H}_k(t) = \int \Psi_k(t,z)\, d\pi_k(z)$, and $d\pi_k(z) = \eta_k e^{-\eta_k z} dz$. □

Corollary 4.2. *Let $\tau_k \in \Gamma$, $\gamma_k = \left(\frac{\bar{v}_k}{1-\bar{u}_k}\right)^2$ and suppose $\bar{u}_k < 1$. The steady-state DMP of τ_k is*

$$\tilde{p}_k = 1 - \left(1 + \frac{\sqrt{1+2\lambda_k\gamma_k} - 1}{2\bar{v}_k}\right)^{-1} \tag{4.17}$$

Proof. It is a direct consequence of Proposition 4.6 and

$$\begin{aligned} \tilde{p}_k &= \mathbf{E}[p_k(\tilde{\beta}_k)] \\ &= 1 - \mathbf{E}\left[\exp\left(-\tilde{\beta}_k \frac{\sqrt{1+2\lambda_k\gamma_k} - 1}{\gamma_k(1-\bar{u}_k)}\right)\right] \qquad \text{(with (4.10))} \end{aligned}$$

$\tilde{\beta}_k$ being an exponential variable of parameter η_k, we conclude by identifying the Laplace transform of a exponential variable,

$$L(s) = \mathbf{E}[\exp\left(-s\tilde{\beta}_k\right)] = \left(1 + \frac{s}{\eta_k}\right)^{-1}$$

Then with $s = \frac{\sqrt{1+2\lambda_k\gamma_k}-1}{\gamma_k(1-\bar{u}_k)}$ and $\gamma_k\eta_k(1-\bar{u}_k) = 2\bar{v}_k$ we get the result. □

4.2 Simulations

In Proposition 4.1 we prove that response times $R_{k,l}$ can be simulated from a sample of idle times of level $k-1$, a sample of execution times and the proper initialization backlog sample, the steady-case or the worst-case.

The procedure is as follows:

- *Generate* a backlog b equal to $\sum_{i=1}^{k} c_i^{max}$ (resp. with the distribution function π_k),
- *Generate* the response time $R_k = \mathcal{I}_k^b$.

See Figure 4.2 and Figure 4.3a.

Proposition 4.7. *The distribution function of R_k is H_k^{max} (resp. $\tilde{H}_k$).*

Proof. We can see that by construction Eq. (4.13) and Eq. (4.6) come from conditional probabilities. The representation in Proposition 4.1 indicates that the distribution of response times is conditioned by the values of the backlog and the execution time. □

4.3 Stability

The first idle time is also a probabilistic variable, but we can build an instant $t_{idle}(\varepsilon, x)$ such that for any $\varepsilon > 0$ and all $t > t_{idle}(\varepsilon, x)$, the probability that the first idle time of the lowest priority level is greater than t is less than ε. We call *first ε-idle time* the instant $t_{idle}(\varepsilon, x)$ from which we can guarantee the system is steady with probability $1-\varepsilon$ and initial backlog $x > 0$. We know that $\beta_k^{(\infty)}$ is a reflected Brownian motion reflected. This means that until it reaches zero, *i.e.*, an idle time, it is a simple Brownian motion. In fact, between two consecutive idle times the backlog process has the same dynamic. Thus, we define the stability of a real-time system according to its response times.

Definition 4.3 (Stability)**.** *A real-time system is said steady when its response times are stationary, i.e., there exists a sequence of exceedence functions $H_{k,l_1}, \ldots, H_{k,l_k}$ repeating indefinitely for any task $\tau_k \in \Gamma$.*

The following proposition bounds with a given probability the first idle time and the instant the system gets stable.

Proposition 4.8. *Let $n = |\Gamma|$, $\varepsilon \in (0,1)$ and $x \geqslant 0$. If $\bar{u}_n < 1$, the system of initial demand x is stable with probability at least $1-\varepsilon$ at the instant*

$$t_{idle}(\varepsilon, x) = \left(\frac{q_{1-\varepsilon} + \sqrt{q_{1-\varepsilon}^2 + 4\left(\frac{1-\bar{u}_n}{\bar{v}_n}\right) x}}{2\left(\frac{1-\bar{u}_n}{\bar{v}_n}\right)} \right)^2 \tag{4.18}$$

where $q_{1-\varepsilon} = \Phi^{-1}(1-\varepsilon)$ is the $(1-\varepsilon)$-quantile of a standard normal distribution.

Proof. First of all, between two idle times of the same level, say $\mathcal{I}$ and $\mathcal{I}'$, the backlog processes $\beta_k^{(\infty)}, k = 1, \ldots, n$ make excursions between passages to 0, *i.e.*, idle times. As it is a reflected Brownian motion, those excursions between idle times are i.i.d., since the distribution only depends on the intial value, which is 0 at each idle time. Furthermore, there is a finite number of possible jobs realed in the interval $[\mathcal{I}, \mathcal{I}']$. Hence, a finite number of reponse time distributions that repeat on every excursion. According to Definition 4.3, this makes the system stable when the backlog of each priority level reaches 0. According to static priority scheduling, $\mathcal{I}_1(x) \leqslant \cdots \leqslant \mathcal{I}_n(x)$ where $\mathcal{I}_n(x)$ represents the first idle time of the lowest priority level initialized with a demand $\bar{W}_n^{(\infty)}(0) = x$. Hence, after the instant $\mathcal{I}_n(x)$, all tasks have response times distributions that repeat according to the excursions of the backlog of their associated priority level. Let us now bound the probability that the idle time $\mathcal{I}_n(x)$ exceeds a given value $t > 0$. Note that

$$\begin{aligned}
\mathbf{P}(\mathcal{I}_n(x) > t) &= \mathbf{P}_n^x \left(\inf_{s \in [0,t]} \beta_n^{(\infty)}(s) > 0 \right) \\
&= \mathbf{P}_n^x \left(\sup_{s \in [0,t]} s - \bar{W}_n^{(\infty)}(s) \leqslant 0 \right) \qquad \text{(According to Theorem 3.6)}
\end{aligned}$$

According to Theorem 3.5, $\bar{W}_n^{(\infty)}$ is a Brownian motion of drift $\bar{u}_n$ and deviation $\bar{v}_n$, there exists a standard Brownian motion B such that $\bar{W}_n^{(\infty)}(t) - \bar{W}_n^{(\infty)}(0) = \bar{u}_n t + \bar{v}_n B(t)$. Hence

$$\begin{aligned} \mathbf{P}(\mathcal{I}_n(x) > t) &= \mathbf{P}\left(\sup_{s\in[0,t]} (1-\bar{u}_n)s - \bar{v}_n B(s) \leqslant x\right) \\ &\leqslant \mathbf{P}\left(B(t) > t\frac{1-\bar{u}_n}{\bar{v}_n} - \frac{x}{\bar{v}_n}\right) \\ &= 1 - \Phi\left(\frac{(1-\bar{u}_n)t - x}{\bar{v}_n\sqrt{t}}\right) \end{aligned} \tag{4.19}$$

Note that this inequality is also found in [Markovic et al., 2022]. Let us define $t_{idle}(\varepsilon, x)$ such that

$$1 - \Phi\left(\frac{(1-\bar{u}_n)t_{idle}(\varepsilon,x) - x}{\bar{v}_n\sqrt{t_{idle}(\varepsilon,x)}}\right) \leqslant \varepsilon$$

which implies that for $t > t_{idle}(\varepsilon, x)$ we should have

$$t > \bar{u}t + \Phi^{-1}(1-\varepsilon)\bar{v}_n\sqrt{t} + \bar{v}_n x \tag{4.20}$$

see Figure 4.4. This is a second order polynomial equation, which admits no solution when $\bar{u} > 1$, and, when $\bar{u} \leqslant 1$ and $\varepsilon \in (0,1)$, the smallest solution of Eq. (4.20) is Eq. (4.18). □

Remark. *Two important remarks:*

- ***(i)*** *When $\bar{u} < 1$, the excursions of the processes $(\beta_k^{(\infty)}(t + \mathcal{I}_k(x))), t > 0$ between two passages at 0 are the longest excursions possible. Indeed, the trajectories of Brownian motions are continuous, which means that, with a intermediate value theorem argument, an excursion between two passages on x is necessarily smaller than the excursion of the same trajectory between passages through 0 .*
- ***(ii)*** *To go deeper into the theory of excursions, Îto proves [Itô, 1972] that the sequences of idle times also forms a Poisson point process in certain conditions.*

Note that for all $x \geqslant 0$, $\lim_{\varepsilon \to 0} t_{idle}(\varepsilon, x) = \infty$, $\lim_{\varepsilon \to 1} t_{idle}(\varepsilon, x) = 0$, and

- $\bar{u}_n < 1 \implies t_{idle}(\varepsilon, x) < \infty$,
- $\bar{u}_n = 1 \implies t_{idle}(\varepsilon, x) = \infty$,
- $\bar{u}_n > 1 \implies t_{idle}(\varepsilon, x)$ is not defined.

Remark. *We provide the analysis for the lowest priority level, but in fact each priority level has its own first ε-idle time.*

Finally we consider the *maximum first ε-idle time* $t_{idle}^{max}(\varepsilon)$ by considering a synchronous activation, *i.e.*, $\bar{W}_n(0) = \sum_{\tau_i \in \Gamma} c_i^{max}$. See Proposition 4.4 for a more detailed explanation. The maximum first ε-idle time is then

$$t_{idle}^{max}(\varepsilon) = t_{idle}\left(\varepsilon, \sum_{\tau_i \in \Gamma} c_i^{max}\right) \tag{4.21}$$

We can now say that, with a level of confidence ε, that the system is steady at time $t_{idle}^{max}(\varepsilon)$ in the worst-case.

4.4 Experimental results

The purpose of this section is to illustrate that the response times generated from Proposition 4.7 provide a good approximation, by comparing distribution functions of simulations and its associated EVT estimation, and generated heavy-traffic response times distributions. Closer are the curves, better is the estimation. Those results are not exhaustive and are an illustration, we do not cover in this work the sensitivity of the model for different values of $\bar{u}$. We use the data generated by SimSo to apply EVT on response times (using the Scipy framework[1] on Python). Finally we compare our results with SimSo simulations and EVT estimations. Without loss of generality the periods are deterministic in these simulations.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gereme.html

In order to illustrate the stability described in this thesis, we use the task set Γ provided in Table 3.1. The level 2 maximum utilization is smaller than 1, hence backlogs of level 2 converge quickly to 0, see Figure 3.3a. The level 3 maximum utilization is greater than 1 and the level 3 mean utilization is smaller than 1. Thus, τ_3 is the task of interest, see Figure 3.2b and Figure 3.3b. The level 4 mean utilization is close to 1, hence Figure 3.2c and Figure 3.3c illustrate the behavior of backlogs when the utilization approaches its phase transition. The level 5 mean utilization is greater than 1, which illustrates the *explosion* (infinite response times) of the system, see Figure 3.2d and Figure 3.3d.

In Figure 3.2 we observe that the demand $W(t)$ follows the line $\bar{u}t + x$ (its mean). In Figure 3.2 and 3.3 we see what happens when the system mean utilization is smaller, close and greater than 1 : for smaller values of $\bar{u}$ the system stays with zero backlog at some point, see Figure 3.2b, but for values greater than 1 the system explodes, see Figure 3.2d and 3.3d. In Figure 3.2c and 3.3c we see that even for $\bar{u}$ close to 1, the system always admits finite idle times. Most importantly, we see in Figure 3.2b and 3.3b that when $\bar{u} < 1$ and $\bar{u}^{max} > 1$, the analysis holds and provides quantifiable response times.

In Figure 4.4, we see to what corresponds idle times and ε-idle times graphically. In Figure 4.1 the average idle time corresponds to the point where the line t and $\bar{u}t + x$ meet, and the distribution of the idle times correspond to the frequency the demand process meets the line t for each instant $t > 0$.

Finally, in Figure 4.3a, we can see the simulations presented in Section 4.2 and a comparison with response times simulated via SimSo [Chéramy et al., 2014]. The two *ground truth* samples are the two subsets *SimSo-transient* and *SimSo-steady*, which are composed respectively of simulated response time released before and after the maximum ε-idle time $t_{idle}^{max}(\varepsilon) = 154$ where $\varepsilon = 10^{-6}$. The maximum idle time $t_{idle}^{max}(\varepsilon)$ is computed via Monte-Carlo approximations with the representation in Eq. (4.21).

The EVT estimation (green curve) from these SimSo simulations and the steady-state and WCRTs suggested in this thesis are compared via their distribution functions in Figure 4.3b. In this case, where $\bar{u}$ is not too close to 1 and $\bar{u}^{max}$

greater than 1, we can see that the *Worst-case* response times, the EVT estimation and the steady-state response times are greater (in the stochastic sense [Díaz et al., 2004]) than the *true* response times simulated with SimSo. The heavy-traffic *Worst-case* response time seems to be an upper-bound in practice, and the *Steady-state* response time is quite accurate.

Closer is the *Steady-state* to the *SimSo-steady* curve, more accurate the approximation is. We can see a big difference between the *Worst-case* curve and the *Steady-state* curve. However the proposed analysis does not permit to quantify analytically this difference.

4.5 Conclusion

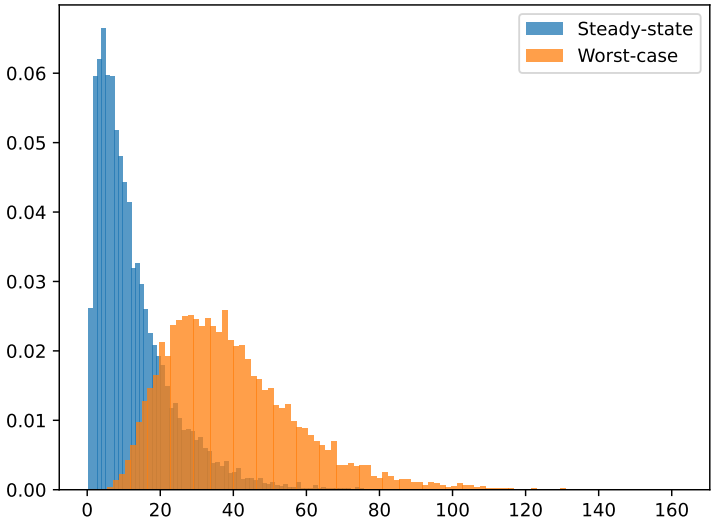
We have seen that real-time systems can reach steadiness over a finite and quantifiable amount of time, and that a necessary condition to assure this stability is that the mean utilization is lower than 1. In practice, systems with a mean utilization $\bar{u} > 0.9$ have many deadline misses, which requires from system designers to quantify deadline miss probabilities carefully by using the distributions of response times provided in this chapter.

No schedulability tests considering the steadiness of response times exist. This is a natural step in our opinion for the use of the analysis provided in this work. For example, the approach we have presented in this chapter is well suited for an application of a Monte-Carlo response time analysis [Bozhko et al., 2021] which has recently been proven efficient.

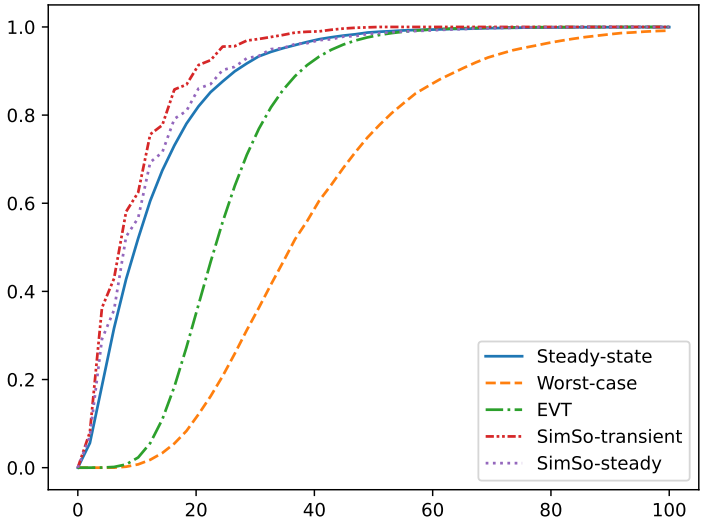
We have expressed the probability function of response times in a specific family of distributions and provided a method to generate them. However, the distribution functions of execution time are usually unknown. The methods built in this chapter could be used in empirical and measurement-based methods, for example using clustering methods [Friebe et al., 2020, Zagalo et al., 2020].

Yet these deadline miss probabilities depend on the Brownian approximation

which converges slowly. A next step in this analysis would be to check the sensibility to the size of the task set.



(a) Histogram of the generated response times of τ_3 in Table 3.1, as defined in Section 4.2.



(b) Response time distribution functions of τ_3 in Table 3.1.

Figure 4.3: Simulations of a 10 000 instances for the SimSo simulations, steady and transient, EVT estimation of the WCRT of the SimSo simulations and simulations of heavy-traffic WCRT (H_3^{max}) and steady-state response-time ($\tilde{H}_3$) when $\bar{u} = 0.838$

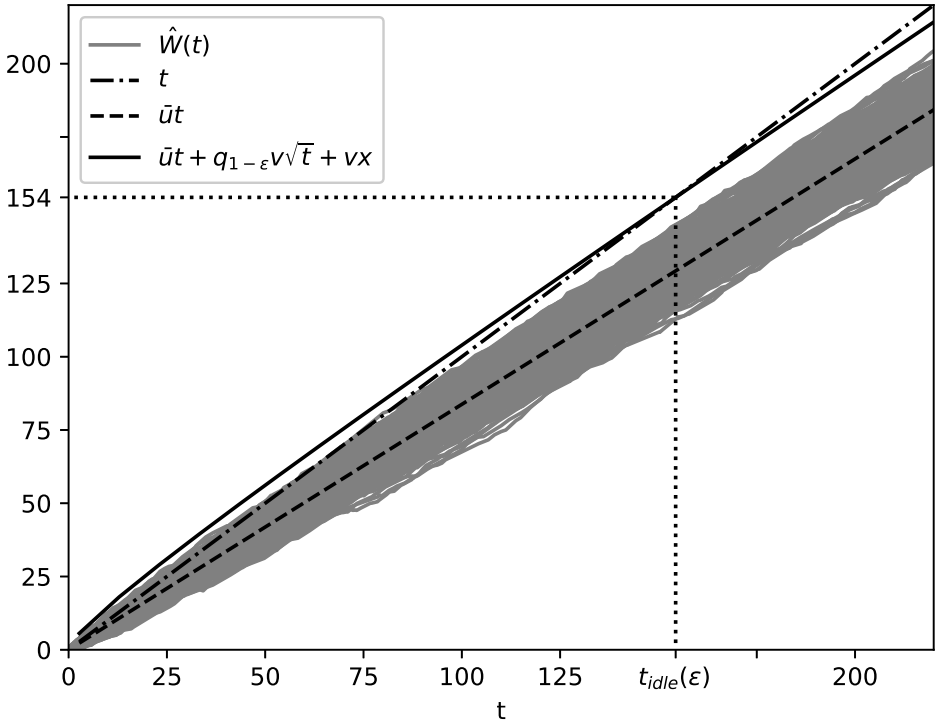


Figure 4.4: First maximum ε-idle time for $\varepsilon = 10^{-6}$, for $\bar{u} = 0.838$.

5 | Parametric estimation of transient response times

Contents

Concentration inequalities have been widely studied these last years to bound DMP [von der Brüggen et al., 2021,Palopoli et al., 2012,Chen et al., 2018]. Currently, the most efficient bound is the Hoeffding DMP [von der Brüggen et al., 2018]. These bounds compute DMP only from the parameters of the studied task set. The method built in this chapter uses knowledge on the task set and infers response time data to compute the Maximum Likelihood Estimate (MLE).

We justify in Chapter 4 that response times are first-passage times of Brownian motion, and hence the inverse Gaussian family is proposed as the appropriate family for response time approximations. Thus, the inverse Gaussian family is a natural choice for a statistical modelling of positive and right-skewed distributions, see [Folks and Chhikara, 1978, Tweedie, 1957]. It is used in many research fields, such as industrial degradation modelling [Ye and Chen, 2014], psychology [Schwarz, 2001, Palmer et al., 2011], and many others like hydrology, market research, biology, ecology, and so on *c.f.*, [Seshadri, 2012].

In this chapter, we propose a suited parameterization of the inverse Gaussian distribution in Section 5.1, using an adapted Expectation-Maximization (EM) algorithm, we estimate the parameters of a mixture of inverse Gaussian distributions. This allows to estimate response times and allows parametric inference. Finally in Section 5.2.1 we illustrate the convergence of the EM algorithm and compare it to the Hoeffding DMP (see Lemma 3.5) with simulations, compare it to the classic EM algorithm in terms of computation time.

As we proved in the previous chapter, response times can be approximated with inverse Gaussian distributions as long as inter-arrival times are exponential. In this chapter we test the assumption that bounds on response times may be approximated by inverse Gaussian distributions for periodic inter-arrival times. Thus, we test this assumption in this chapter. Let $(\Omega, \mathbf{P}, \Gamma, \{\theta_t\}_t)$ be a stationary real-time system following a $\sum_i D_i / \sum_i G_i / 1 / \mathrm{SP}$ queueing model.

5.1 Inverse Gaussian mixture model for response times

Let R_i be the response time of $\bar{u}_i$. Its distribution function is the mixture of the distribution functions of the response times $R_{i,j}$. The distribution of R_i is composed of k_i components. Formally, this means that we approximate the probability density

function of the response time R_i with a variable R_i of probability density function

$$h_i(x; \boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \sum_{k=1}^{k_i} \pi_{i,k} \psi_i \left(x; \theta_{i,k}\right) \tag{5.1}$$

where $\psi_i(x, \theta)$ is the inverse Gaussian probability function of mean $\theta/(1 - \bar{u}_i)$ and shape $\theta^2/\bar{v}_i^2$.

In real-time systems, the interest of the analytical approach is to measure the DMP p_i with a closed expression. For example, a task τ_i should not miss its deadline with a permitted failure rate α_i, and the inequality in Eq. (2.21) is approximated with the mixture Eq. (5.1).

5.1.1 Re-parameterized inverse Gaussian distribution for response times

The purpose of this section is to provide the efficient distribution family for an approximation of response times and an adapted EM algorithm to estimate the parameters of this approximation. This adapted re-parameterization of the inverse Gaussian distribution reduces the number of parameters of the model. Furthermore, as underlined in [Punzo, 2019], the log-likelihood of the inverse Gaussian distribution has flat regions, thus the EM algorithm has tiny variations. Reducing the number of parameters addresses a part of this problem. A second reduction of this problem is the use of the Aitken acceleration procedure [Aitken, 1926].

In [Punzo, 2019], the author introduces a modified version of the inverse Gaussian distribution of parameters (ξ, δ), using its mode

$$\mu = \left(\xi^2 + \frac{9\xi^4}{4\delta^2}\right)^{1/2} - \frac{3\xi^2}{2\delta}$$

and its *variability coefficient*

$$\gamma = \frac{\xi^2}{\delta}$$

instead of the mean and shape. This re-parameterized inverse Gaussian distribution of parameters (μ, γ) is defined by the probability density function

$$\tilde{\psi}(x; \mu, \gamma) = \sqrt{\frac{\mu(3\gamma + \mu)}{2\pi\gamma x^3}} \exp\left\{ -\frac{\left(x - \sqrt{\mu(3\gamma + \mu)}\right)^2}{2\gamma x} \right\} \tag{5.2}$$

With the re-parameterized inverse Gaussian distribution applied to the form that takes the parameters of the distributions of response times, one can see that only the mode is sensitive to the mixture provided in Eq. (5.1). The variability coefficient of an inverse Gaussian distribution of mean $\theta/(1-\bar{u}_i)$ and shape $(\theta/\bar{v}_i)^2$ is

$$\gamma_i = \frac{\bar{v}_i^2}{(1 - \bar{u}_i)^2} \tag{5.3}$$

and its mode is

$$\mu_i(\theta) = \sqrt{\left(\frac{\theta}{1 - \bar{u}_i}\right)^2 + \frac{9\gamma_i^2}{4}} - \frac{3\gamma_i}{2} \tag{5.4}$$

such that $\psi_i(x; \theta)$ is the probability density function of a re-parameterized inverse Gaussian distribution of mode $\mu_i(\theta)$ and variability γ_i.

5.1.2 Maximum likelihood estimation of response time distributions

In this section we present an adaptation of the MLE proposed by [Punzo, 2019] for real-time systems. We have implemented both methods in the Python language in the library *rInverseGaussian* [Zagalo and Verbytska, 2022].

When $k_i = 1$, we have the follwing proposition.

Proposition 5.1. *Let $(R_{i,j})_{j=1,\ldots,N}$ be a N-sample of response times of the task $\tau_\gamma \in \Gamma$. When $k_i = 1$, $\pi_{i,1} = 1$ and we have the MLE*

$$\hat{\theta}_i = \frac{1 - \bar{u}_i}{N} \sum_{j=1}^{N} R_{i,j} \tag{5.5}$$

Proof. See [Folks and Chhikara, 1978] for the classical MLE of IG distributions. We know from [Folks and Chhikara, 1978, Eq. (10)] that the mean of an IG distribution is the empirical mean $\frac{1}{N}\sum_{j=1}^{N} R_{\gamma,j}$. Furthermore, we are looking for estimating the mean $\frac{\theta}{1-u_\gamma}$. Since we already know u_γ, we get the result by estimating $\frac{\theta}{1-u_\gamma}$ with $\frac{1}{N}\sum_{j=1}^{N} R_{\gamma,j}$. □

The complete-likelihood of mixture models [McLachlan et al., 2019, Bouveyron et al., 2019] can be written as

$$L_c(\boldsymbol{Z}_i, \boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \prod_{j=1}^{n}\prod_{k=1}^{k_i}[\pi_{i,k}\psi_i(r_j;\theta_{i,k})]^{Z_{i,j,k}} \tag{5.6}$$

and the complete log-likelihood $\ell_c = \log L_c$ is

$$\ell_c(\boldsymbol{Z}_i, \boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \ell_{c_1}(\boldsymbol{Z}_i, \boldsymbol{\pi}_i) + \ell_{c_2}(\boldsymbol{Z}_i, \boldsymbol{\theta}_i) \tag{5.7}$$

where

$$\ell_{c_1}(\boldsymbol{Z}_i, \boldsymbol{\pi}_i) = \sum_{j=1}^{n}\sum_{k=1}^{k_i} Z_{i,j,k}\log\pi_{i,k} \tag{5.8}$$

and

$$\ell_{c_2}(\boldsymbol{Z}_i, \boldsymbol{\theta}_i) = \sum_{j=1}^{n}\sum_{k=1}^{k_i} Z_{i,j,k}\log\psi_i(r_j;\theta_{i,k}) \tag{5.9}$$

which leads to the following EM algorithm:

E-step For the $(s+1)$th step of the EM algorithm, $\boldsymbol{z}_i^{(s)}$ the conditional expectation of $\boldsymbol{Z}_i$ given $(\boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \left(\boldsymbol{\pi}_i^{(s)}, \boldsymbol{\theta}_i^{(s)}\right)$ is given by

$$z_{i,j,k}^{(s)} = \frac{\pi_{i,k}^{(s)}\psi_i\left(r_j;\theta_{i,k}^{(s)}\right)}{h_i\left(r_j;\boldsymbol{\pi}_i^{(s)},\boldsymbol{\theta}_i^{(s)}\right)} \tag{5.10}$$

M-step For the $(s+1)$th step of the EM algorithm, $\ell_{c_1}(\boldsymbol{z}_i^{(s)}, \cdot)$ is maximized by

$$\pi_{i,k}^{(s+1)} = \frac{1}{n}\sum_{j=1}^{n} z_{i,j,k}^{(s)}, k = 1,\ldots,k_i \tag{5.11}$$

and maximizing ℓ_{c_2} with respect to $\boldsymbol{\theta}$ is maximizing each of the k_i expressions

$$\sum_{j=1}^{n} z_{i,j,k}^{(s)} \log \psi_i(r_j; \theta_{i,k}), k = 1, \ldots, k_i \tag{5.12}$$

using Newton-like algorithms to solve

$$\nabla \ell_c = 0 \tag{5.13}$$

Then with $\frac{\partial \log \psi_i}{\partial \theta}(x;\theta) = \frac{\partial \mu_i}{\partial \theta}(\theta)\frac{\partial \log \tilde{\psi}}{\partial \mu}(x;\mu_i(\theta),\gamma_i)$ and the derivatives

$$\begin{aligned} \frac{\partial \log \tilde{\psi}}{\partial \mu}(x;\mu,\gamma) &= -\frac{3}{2x} - \frac{\mu}{x\gamma} + \frac{1}{3\gamma+\mu} + \frac{3\gamma}{2\mu(3\gamma+\mu)} + \frac{\sqrt{\mu}}{2\gamma\sqrt{3\gamma+\mu}} + \frac{\sqrt{3\gamma+\mu}}{2\gamma\sqrt{\mu}} \\ \frac{\partial \mu_i}{\partial \theta}(\theta) &= \frac{\theta}{(1-\bar{u}_i)^2}\left(\left(\frac{\theta}{1-\bar{u}_i}\right)^2 + \frac{9\gamma_i^2}{4}\right)^{-1/2} \end{aligned} \tag{5.14}$$

Eq. (5.13) is equivalently solved by Eq. (5.11) and the solutions of

$$\sum_{j=1}^{n} z_{i,j,k}^{(s)} \frac{\partial \log \psi_i}{\partial \theta}(r_j;\theta_k) = 0, \forall k = 1, \ldots, k_i \tag{5.15}$$

In order to stop the algorithm, the author in [Punzo, 2019] proposes the *Aitken acceleration*. The Aitken acceleration at iteration $s+1$ is given by

$$a^{(s+1)} = \frac{\ell^{(s+2)} - \ell^{(s+1)}}{\ell^{(s+1)} - \ell^{(s)}} \tag{5.16}$$

where $\ell^{(s)}$ is the observed-data log-likelihood from iteration s. The limit ℓ_∞ of the sequence of values of the log-likelihood is

$$\ell_\infty^{(s+2)} = \ell^{(s+1)} + \frac{\ell^{(s+2)} - \ell^{(s+1)}}{1 - a^{(s+1)}} \tag{5.17}$$

The EM algorithm is considered to have converged if

$$|\ell_\infty^{(s+2)} - \ell_\infty^{(s+1)}| < \varepsilon \tag{5.18}$$

with a tolerance $\varepsilon > 0$.

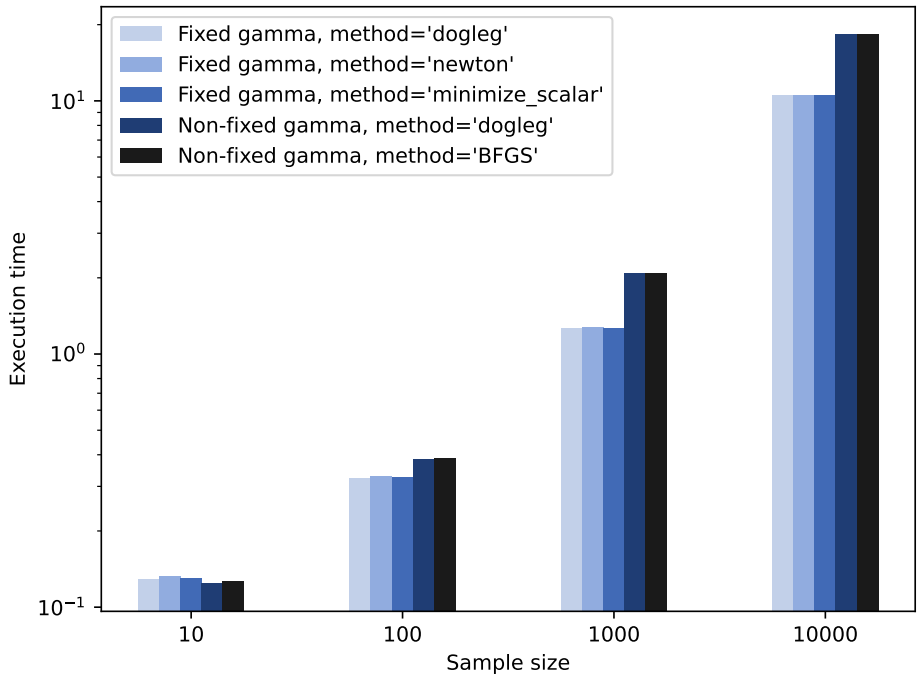


Figure 5.1: Computation time of the EM algorithm with the knowledge of $\bar{u}_i$ and $\bar{v}_i$ (fixed gamma) and without (non-fixed gamma)

Finally, we initialize the algorithm with a k-means clustering for $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{\pi}^{(0)} = 1/n$.

5.1.3 Bayesian information criteria

The Bayesian information criteria (BIC, [Schwarz, 1978]) is used to chose the number of components of the mixture, which has been proven consistent for mixture models [Raftery, 1995, Fraley and Raftery, 2002, Dasgupta and Raftery, 1998]. The number of parameters of a mixture of k components being $2k - 1$, the number of components chosen is equal to

$$k_i = \underset{k}{\operatorname{argmax}}\, 2\ell_n(\boldsymbol{\pi}_i, \boldsymbol{\theta}_i) - (2k - 1)\log n \tag{5.19}$$

where ℓ_n is the observed-data log-likelihood. The number of parameters being reduced from $3k_i - 1$ to $2k_i - 1$, the computation time of this EM algorithm is also reduced (see Figure 5.1).

5.1.4 Model validation

We use the relation of inverse Gaussian distributions with the χ_1^2 distribution to check the quality of the MLE. Indeed, if X is an inverse Gaussian variable of mean ξ and shape δ, then

$$\frac{\delta(X-\xi)^2}{\xi^2 X}$$

is distributed as a Chi-squared distribution of one degree of freedom [Tweedie, 1957]. Let $\mathbf{P}_k^{\mathrm{IG}}$ be the probability conditionally that the response time R_i is in the k-th component in the inverse Gaussian estimation, and

$$g_i(x;\theta) = \frac{\left(x - \frac{\theta}{1-\bar{u}_i}\right)^2}{\gamma_i x}, x > 0 \tag{5.20}$$

In our case, for each component $k = 1, \ldots, k_i$ of the mixture Eq. (5.1), after classification we should have that

$$g_i\left(R_i; \hat{\theta}_{i,k}\right) \sim \chi_1^2 \tag{5.21}$$

under the probability $\mathbf{P}_k^{\mathrm{IG}}$. Therefore, we use Eq. (5.21) to validate the MLE, and provide the DMP we are looking for. The larger quantiles values are the ones that real-time designers are interested in to determine whether a task is schedulable or not, see Eq. (2.21). We use Eq. (5.21) to determine whether a task is schedulable in its transient state or not.

5.1.5 Deadline miss probability

Proposition 5.2. *The deadline miss probability of the inverse Gaussian estimation is*

$$p_i^{IG} = \sum_{k=1}^{k_i} \pi_{i,k} \Psi(1/\lambda_i, \theta_{i,k}; \bar{u}_i, \bar{v}_i) \tag{5.22}$$

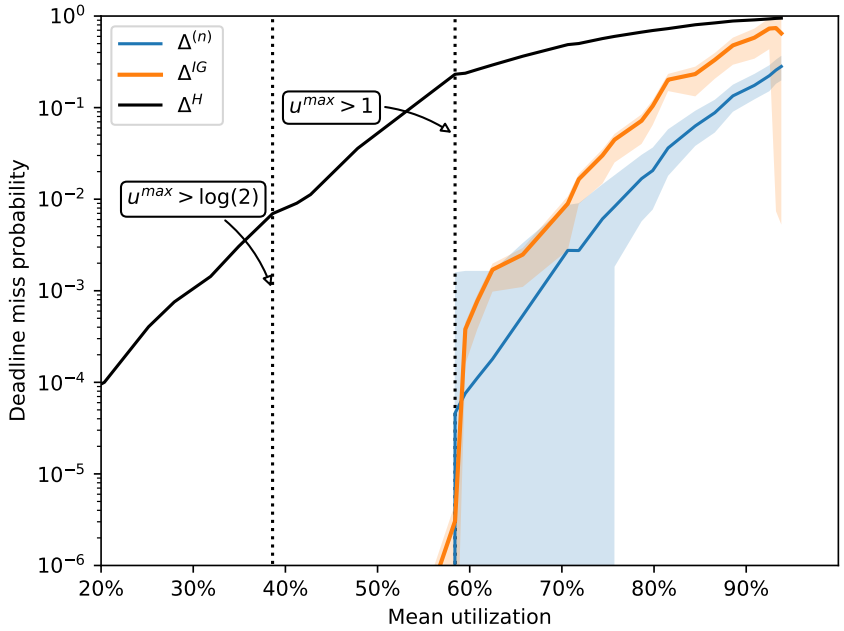


Figure 5.2: Utilizations $\bar{u}_i$ against the deadline miss probabilities of the inverse Gaussian estimation $p_i^{IG} = \Delta^{IG}(\bar{u}_i)$, the Hoeffding DMP $p_i^H = \Delta^H(\bar{u}_i)$ and the empirical deadline miss probability $p_i^{(n)} = \Delta^{(n)}(\bar{u}_i)$ over a sample of $n = 100\,000$ response times per task simulated on SimSo. In solid line the average value, and the colored area represents the values between the 25%-percentile and the 75%-percentile.

where

$$\Psi(t, b; u, v) = \Phi\left(-\frac{(1-u)t - b}{v\sqrt{t}}\right) - e^{-2b\frac{1-u}{v^2}}\Phi\left(-\frac{(1-u)t + b}{v\sqrt{t}}\right) \tag{5.23}$$

and Φ *is the standard normal distribution function.*

Proof. We have $p_i^{\mathrm{IG}} = \sum_{k=1}^{k_i} \pi_{i,k} \mathbf{P}_k^{\mathrm{IG}}(R_i > D_i)$ and since $g_i(\cdot;\theta)$ is positive and, decreasing for $x \leqslant \frac{\theta}{1-\bar{u}_i}$ and increasing for $x > \frac{\theta}{1-\bar{u}_i}$, we obtain the result. □

One may see in Figure 5.2 a comparison between the empirical DMP, the inverse Gaussian method in Eq. (5.22) and the Hoeffding DMP.

In the following, we test by simulation if p_i^{IG} is a good estimation of p_i and if the Hoeffding DMP is a safe bound of the inverse Gaussian estimation, *i.e.*, $p_i^{\mathrm{IG}} \leqslant p_i^{\mathrm{H}}$.

5.2 Experimental results

The seminal work of Liu and Layland [Liu and Layland, 1973] provides a sufficient condition for the schedulability of any system with finite supports of execution times using the maximal utilization $\bar{u}_i^{max}, i \geqslant 1$. Whenever

$$\bar{u}_n^{max} < n(2^{1/n} - 1) \tag{5.24}$$

the task set Γ is proven schedulable for $\alpha_1 = \cdots = \alpha_n = 0$ [Liu and Layland, 1973, Theorem 5]. Moreover, while $\bar{u}_n^{max} < 1$ there exists a dynamic-priority scheduling policy that can satisfy the schedulability of the system [Liu and Layland, 1973]. Hence there are two phase transitions, one at $\bar{u}_n^{max} > \log(2) = \lim_n n(2^{1/n} - 1)$ where deadline misses *can* happen, and one at $\bar{u}_n^{max} > 1$ where deadline misses *must* happen. As proven in [Zagalo et al., 2022a], the necessary condition for the schedulability of a task $\bar{u}_i$ Eq. (2.21) is that $\bar{u}_i < 1$. Hence, there is a gap to fill in the theory between the necessary condition $\bar{u}_i < 1$ and the sufficient condition $\bar{u}_i^{max} < \log(2)$. In particular in the case where $\bar{u}_i < 1$ and $\bar{u}_i^{max} > 1$ as we see in Figure 5.2.

5.2.1 Simulations

In this section, we verify our method with simulated data. The simulated data are generated using SimSo [Chéramy et al., 2014], a Python framework used to generate arrival times of jobs and scheduling policies. A modified version of SimSo [Cheramy, 2014] generates random inter-arrival times and random execution times [Zagalo and Auvray, 2022][1]. We study the quality of the estimation as a function of utilization level. We show that the larger the utilization, the better the estimation. We also measure by how much the inverse Gaussian bound is larger than the empirical DMP,

$$p_i^{(n)} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{1}_{R_{i,j} > T_{i,j+1}} \tag{5.25}$$

[1] https://github.com/kevinzagalo/simso/blob/main/generator/task_generator.py

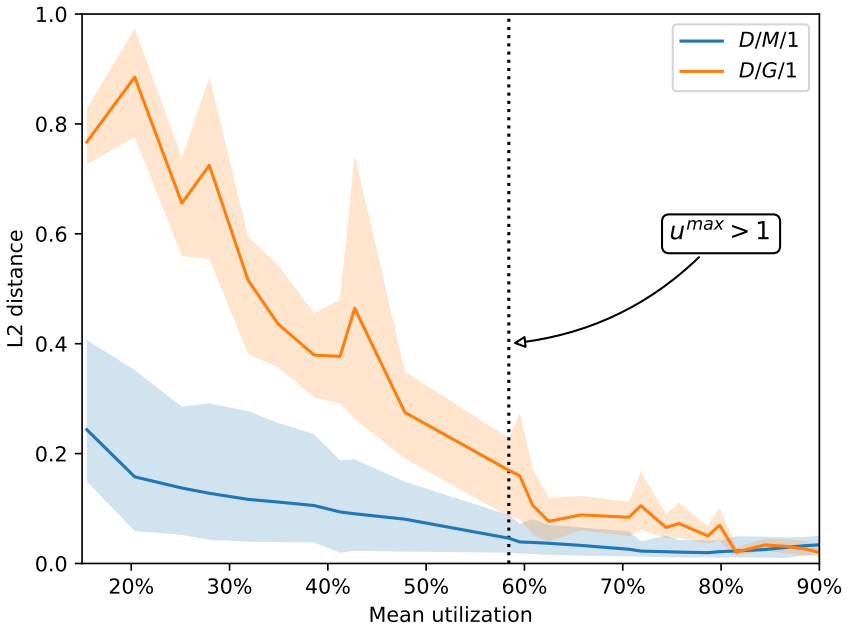


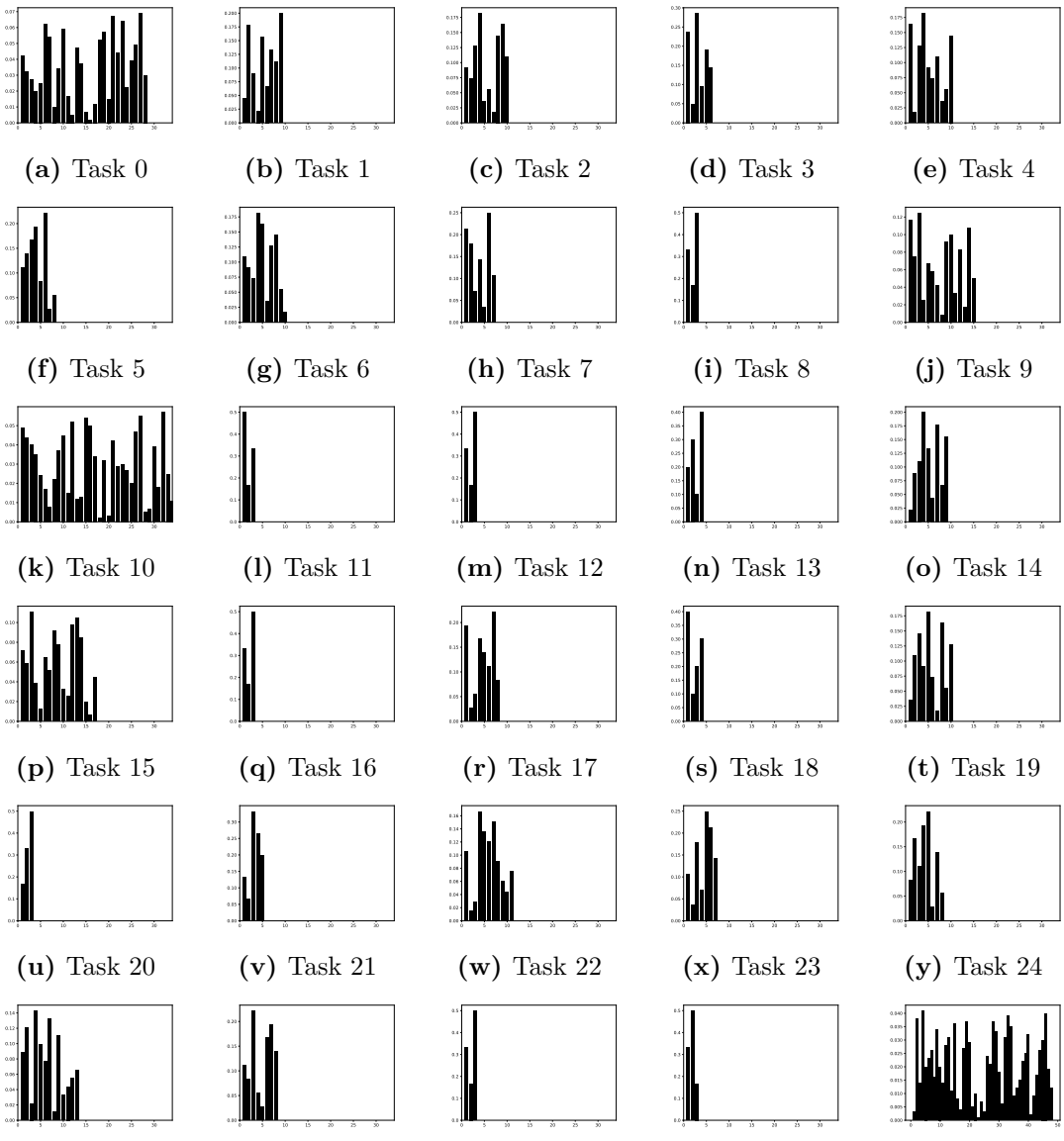
Figure 5.3: L2 distance between the empirical distribution of the simulations and the MLE distribution, of 1000 instances of the schedule, for the task set shown in Table 5.1. In solid line the average value, and the colored area represents the values between the 25%-percentile and the 75%-percentile.

We consider a task set where the probability density functions $(f_i)_i$ of the execution times $(C_i)_i$ are known, see Figure 5.4. From SimSo we generate the response times of tasks with the RM scheduling policy from the probability functions $(f_i, i = 1, \ldots, 28)$. Their parameters are given in Table 5.1. The distributions of execution times used in the simulations are generated with UUnifast [Bini and Buttazzo, 2005], to emphasize the fact that f_i can be any distribution ($D/G/1$ queue). Two methods are used: one with a finite support where the maximal utilization $\bar{u}_i^{max}$ is finite, and another one with an infinite support with exponential distributions where the maximal utilization is not defined. This schedule is instantiated 100 times, thus in Figure 5.3(a), the box-plots of each task are based on 100 estimators. Also note in Figure 5.3 that the variability of the estimates decreases with the priority level. Because of the static-priority structure of RM, we can see in Figure 5.3 that the error of the estimation decreases with the priority level. The first task is never preempted, so its response time is always equal to its execution time. Therefore

the estimation of its response time cannot be accurate in general.

In a second step, a task set with exponentially distributed execution times is simulated for comparison ($D/M/1$ queue), as it is a special case widely studied in queueing theory [Pack, 1977]. This is a baseline for determining the rate of convergence of the response times estimation as a function of priority levels. This baseline confirms that the rate of convergence depends on the type of distributions used for execution times, but that there is a phase transition at $\bar{u}_i^{max} > 1$, independent from the type of distribution used for execution times. The parameters of the task set are given in Table 5.1.

In Figure 5.2, we have the mean utilization $(\bar{u}_i)_i$ on the x-axis and $(p_i^{(n)}, p_i^{\mathrm{H}}, p_i^{\mathrm{IG}})_i$ on the y-axis. We can see that when $\bar{u}_i^{max} < \log(2)$, it is useless to compare the methods because they have already been proved schedulable in Eq. (5.24). Moreover, all DMP increase when $\bar{u}_i^{max} > 1$.



(a) Task 0 **(b)** Task 1 **(c)** Task 2 **(d)** Task 3 **(e)** Task 4

(f) Task 5 **(g)** Task 6 **(h)** Task 7 **(i)** Task 8 **(j)** Task 9

(k) Task 10 **(l)** Task 11 **(m)** Task 12 **(n)** Task 13 **(o)** Task 14

(p) Task 15 **(q)** Task 16 **(r)** Task 17 **(s)** Task 18 **(t)** Task 19

(u) Task 20 **(v)** Task 21 **(w)** Task 22 **(x)** Task 23 **(y)** Task 24

(z) Task 25 **(aa)** Task 26 **(ab)** Task 27 **(ac)** Task 28 **(ad)** Task 29

Figure 5.4: Execution time probability functions $f_i, i = 0, \ldots, 29$, used in SimSo, see Section 5.2.1. Integer-valued, their support all start at 1, the maximum is reached by Task 10 at 35.

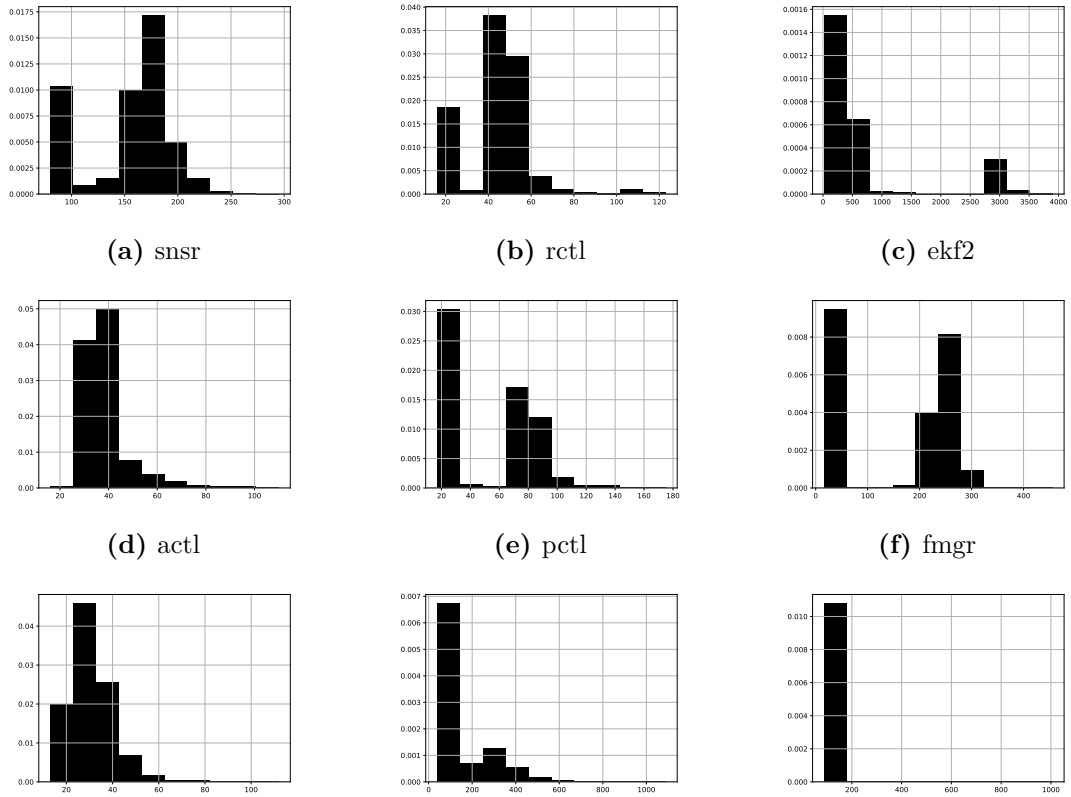
Table 5.1: Parameters of the task set used for the simulations in Section 5.2.1.

Task τ_i	0	1	2	3	4	5	6	7	8	9
Mean m_i (ms)	15.481	5.556	5.708	3.38	5.198	4.057	4.998	3.786	2.167	7.453
Std s_i (ms)	17.1957	5.6996	5.9963	3.323	5.5314	4.0299	5.1184	3.9005	1.8259	8.3373
Periods t_i (ms)	100	114	119	121	132	133	136	144	145	146
Deadlines d_i (ms)	100	114	119	121	132	133	136	144	145	146
Mean utilization $\bar{u}_i$	0.1548	0.2035	0.2515	0.2794	0.3188	0.3493	0.3861	0.4124	0.4273	0.4784
Maximum utilization u_i^{max}	0.28	0.3589	0.443	0.4926	0.5683	0.6285	0.702	0.7506	0.7713	0.874
Task τ_i	10	11	12	13	14	15	16	17	18	19
Mean m_i (ms)	16.812	1.833	2.167	2.7	5.448	8.46	2.167	4.665	2.4	5.604
Std s_i (ms)	19.1248	1.5271	1.8259	2.4495	5.4483	9.2277	1.8259	4.7411	2.2361	5.7741
Periods t_i (ms)	159	165	165	165	166	173	181	182	183	191
Deadlines d_i (ms)	159	165	165	165	166	173	181	182	183	191
Mean utilization $\bar{u}_i$	0.5841	0.5952	0.6083	0.6247	0.6575	0.7064	0.7184	0.744	0.7571	0.7865
Maximum utilization u_i^{max}	1.0879	1.1061	1.1242	1.1485	1.2027	1.301	1.3175	1.3615	1.3834	1.4357
Task τ_i	20	21	22	23	24	25	26	27	28	
Mean m_i (ms)	2.333	3.334	5.927	4.535	4.225	6.246	4.779	2.167	1.834	
Std s_i (ms)	1.9147	3.0555	6.0701	4.4073	4.1931	6.758	4.8654	1.8259	1.4149	
Periods t_i (ms)	193	200	201	214	215	268	296	298	315	
Deadlines d_i (ms)	193	200	201	214	215	268	296	298	315	
Mean utilization $\bar{u}_i$	0.7986	0.8152	0.8447	0.8659	0.8856	0.9089	0.925	0.9323	0.9381	
Maximum utilization u_i^{max}	1.4513	1.4763	1.531	1.5637	1.6009	1.6494	1.6764	1.6865	1.696	

5.2.2 Data

In this section we use the inverse Gaussian method on real data. We use a real case of 9 programs of an autopilot of a drone, PX4-RT [Khazen et al., 2022, Ntaryamira, 2021], a modified version of PX4 [Meier et al., 2015] with a real-time behavior, and a clock measuring preempting the operating system itself. PX4-RT is run on an ARM Cortex M4 CPU clocked at 180 MHz with 256 KB of RAM using a simulated environment from Gazebo [Koenig and Howard, 2004]. PX4-RT allows to measure execution times (Figure 5.5 and Table 5.2) and response times during the flight of a drone. It runs on top of NuttX, a Unix-like operating system. It provides an infrastructure for internal communications between all programs and off-board applications. Each task is a NuttX task launched at the beginning of the PX4 program. The tasks read data from sensors (snsr), estimate positions and attitudes using a Kalman filter (ekf2), control the position (pctl) and the attitude (actl) of the drone, the flight manager (fmgr), the hover thrust estimator (hte), handle the navigation (navr), command the state of the drone (cmdr), and the rate controller (rctl), which is the inner-most loop to control the body rates. These tasks are in continuous interference with the operating system NuttX. Because the operating system has the highest priority, the nine tasks studied are constantly preempted by NuttX. Unfortunately, it is difficult to have information about the interfering operating system programs. Unlike the simulation in Section 5.2.1, PX4-RT runs concurrently with other tasks which do not have timing requirements, making it a complex system with many unknown variables. We test in this section whether and when the proposed parametric estimation is suitable for such complex system.

In this case, the distribution functions of execution times cannot be provided. Therefore, we use the empirical distributions shown in Figure 5.5. Thus, the mean utilization $\hat{u}_i$ is computed with the empirical means of execution times, and the maximal utilization $\hat{u}_i^{max}$ with the empirical maximum of execution times, see Table 5.2 for a full description of the parameters. As shown in the previous section, the response times of the highest priority task *snsr* are not estimated.



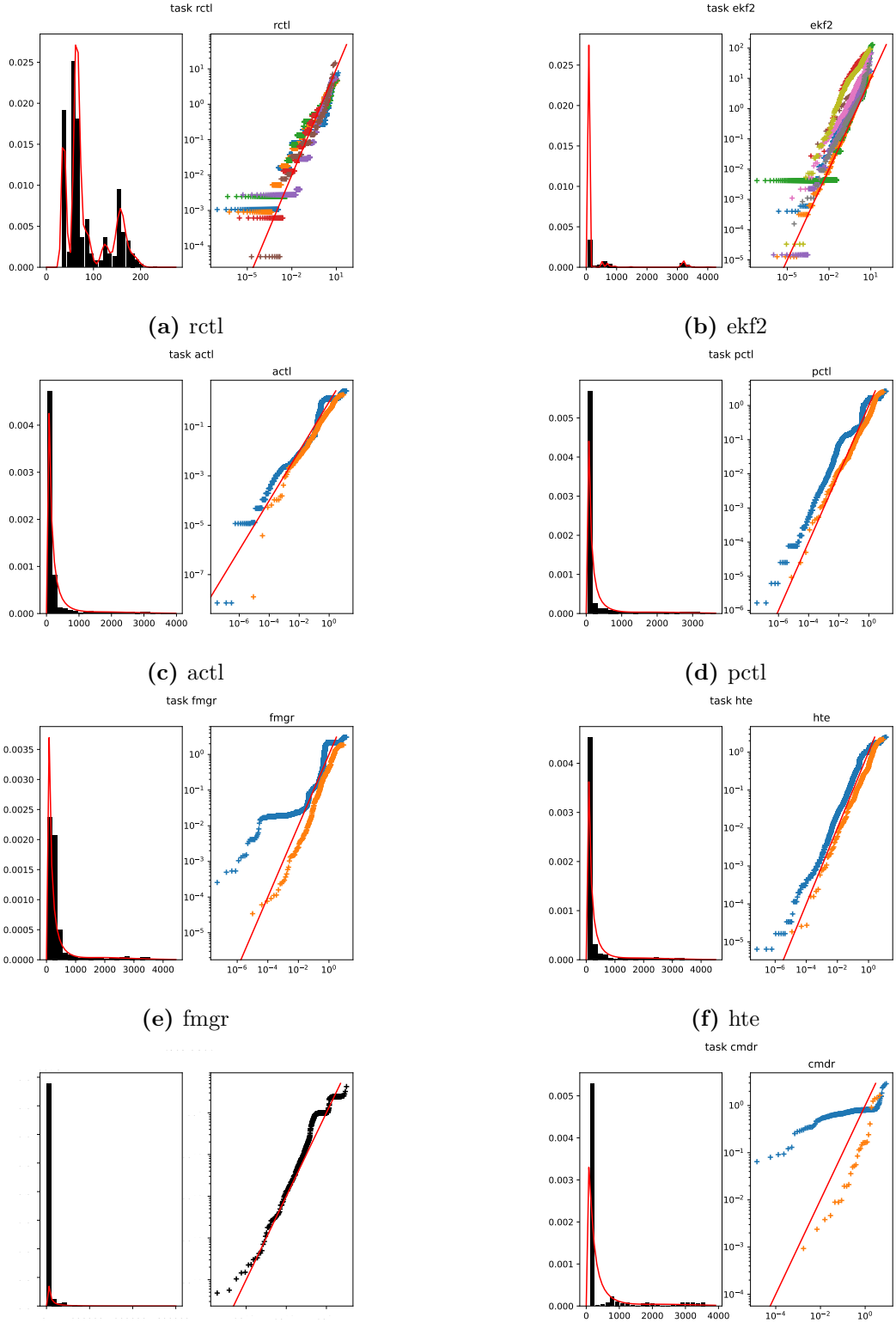
(a) snsr **(b)** rctl **(c)** ekf2

(d) actl **(e)** pctl **(f)** fmgr

(g) hte **(h)** navr **(i)** cmdr

Figure 5.5: Execution time empirical probability functions of the 9 studied tasks of the drone autopilot PX4-RT

These programs generate response times shown in Figure 5.6, on which we use the mixture model proposed in Eq. (5.1) with the EM algorithm provided in Section 5.1.2, see [Khazen et al., 2022] for a full description of the data. The QQplots in Figure 5.6 show that the estimation is good for the large quantiles, which is what is important to determine the schedulability of a system, *c.f.*, Eq. (2.21). We can identify in Figure 5.6 that for the *cmdr* and *fmgr* tasks the estimation is not good, which means that we do not have sufficient information about the programs interfere with them (operating system etc.), and that a schedulability test on this task would not be suitable with the method built in this chapter. Nevertheless, for the other tasks the approximation is appropriate and can therefore be used for a schedulability analysis.



(a) rctl **(b)** ekf2

(c) actl **(d)** pctl

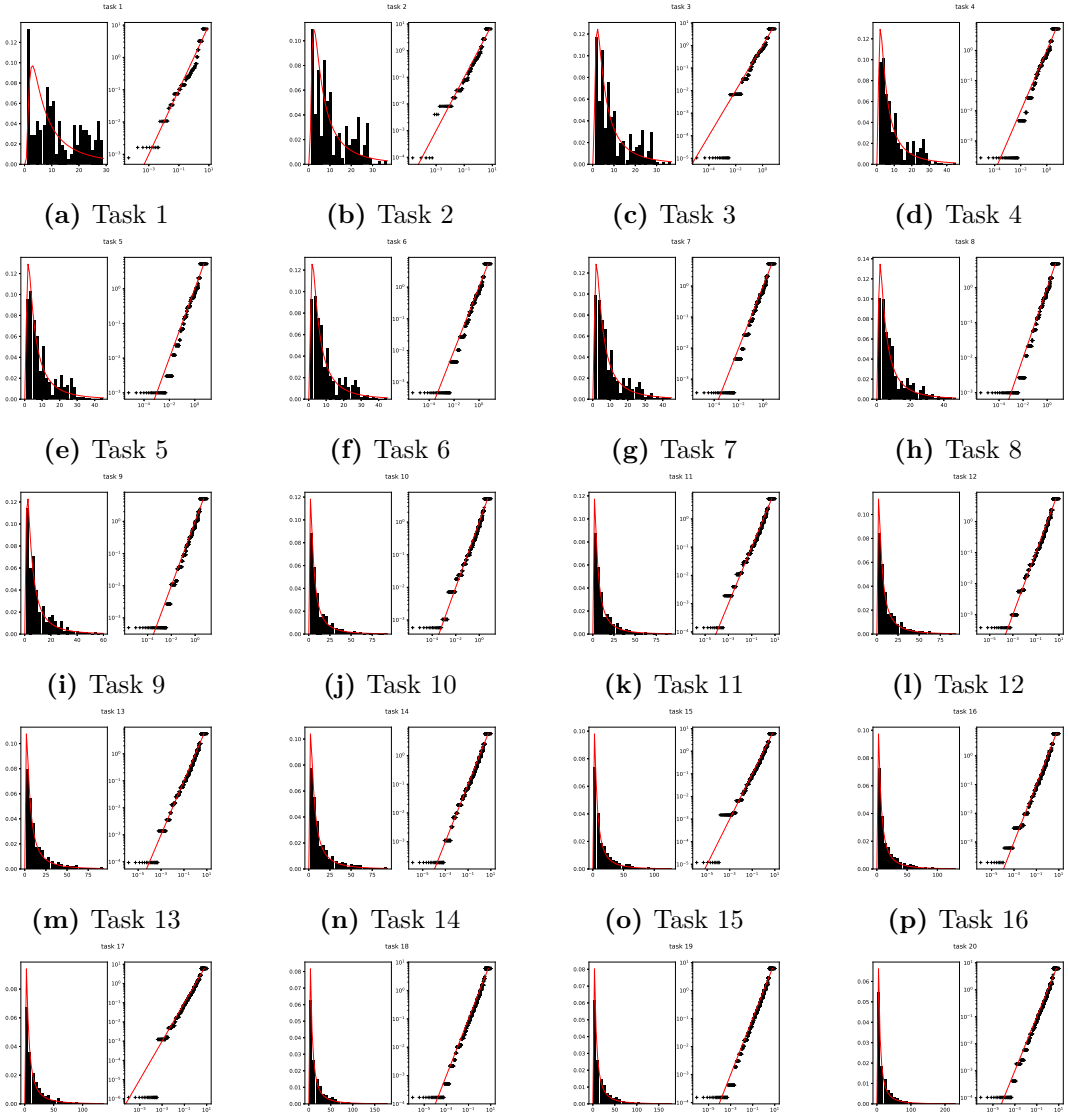
(e) fmgr **(f)** hte

(g) navr **(h)** cmdr

Figure 5.6: Response times empirical distributions of the PX4-RT autopilot and QQplots with the χ^2_1 quantiles from Eq. (5.21) for each component (*c.f.*,the different colors) of the estimated mixtures

Table 5.2: Empirical parameters, periods and deadlines used in the autopilot PX4-RT described in Section 5.2.2.

Task	snsr	rctl	ekf2	actl	pctl	fmgr	hte	navr	cmdr
Priority i	1	2	3	4	5	6	7	8	9
Number of components k_i	–	6	9	2	1	2	2	2	1
Empirical mean $\hat{m}_i$ (μs)	152.5	44.2	610.9	38.7	52.5	151.0	30.61	159.8	114.4
Empirical std $\hat{s}_i$ (μs)	40.5	16.4	982.8	9.5	32.6	114.4	10.9	133.9	61.6
Period t_i (ms)	3.0	4.0	4.1	5.0	5.2	6.0	7.0	50.0	100.0
Deadline d_i (ms)	3.0	4.0	4.1	5.0	5.2	6.0	7.0	50.0	100.0
Empirical mean utilization $\hat{u}_i$	0.05	0.06	0.21	0.22	0.23	0.25	0.26	0.26	0.26
Empirical maximum utilization $\hat{u}_i^{max}$	0.10	0.13	1.08	1.10	1.14	1.21	1.23	1.25	1.26
Empirical deadline miss probability $p_i^{(n)}$	0.0	0.0	0.001	0.0	0.0	0.0	0.0	0.0378	0.0
IG deadline miss probability p_i^{IG}	-	0.0	0.0006	0.0043	0.028	0.0022	0.0009	0.5487	0.0
Hoeffding bound p_i^{H}	-	10^{-168}	0.2512	0.1919	0.1881	0.1673	0.1274	10^{-7}	10^{-13}



(a) Task 1 **(b)** Task 2 **(c)** Task 3 **(d)** Task 4

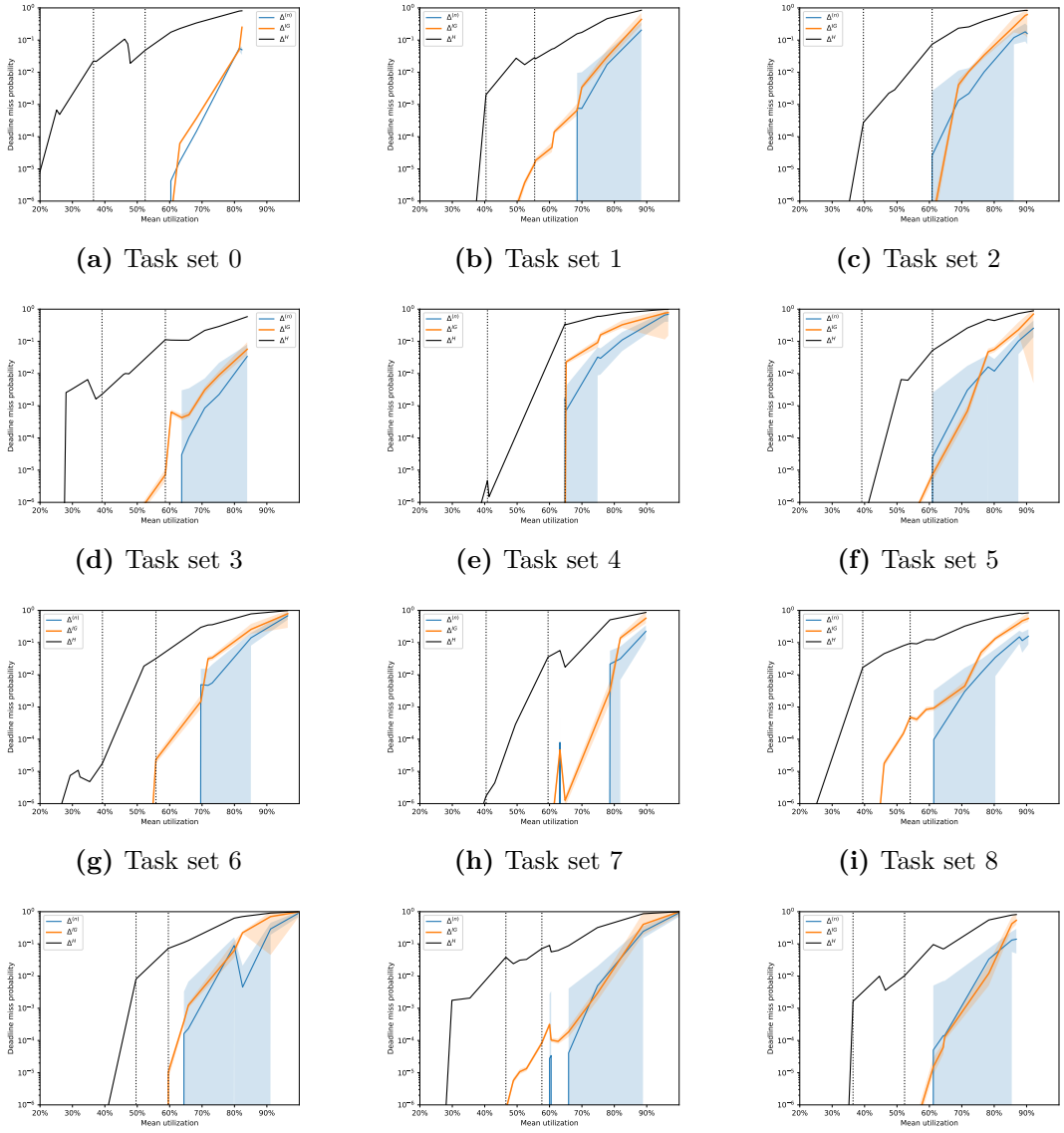
(e) Task 5 **(f)** Task 6 **(g)** Task 7 **(h)** Task 8

(i) Task 9 **(j)** Task 10 **(k)** Task 11 **(l)** Task 12

(m) Task 13 **(n)** Task 14 **(o)** Task 15 **(p)** Task 16

(q) Task 17 **(r)** Task 18 **(s)** Task 19 **(t)** Task 20

Figure 5.7: The response time MLEs and the histogram of simulations from SimSo, see Section 5.2.1



(a) Task set 0 **(b)** Task set 1 **(c)** Task set 2

(d) Task set 3 **(e)** Task set 4 **(f)** Task set 5

(g) Task set 6 **(h)** Task set 7 **(i)** Task set 8

(j) Task set 9 **(k)** Task set 10 **(l)** Task set 11

Figure 5.8: DMP for 12 randomly generated task sets. The estimation is always below the Hoeffding DMP, and the differences between the minimum and maximum DMP of the inverse Gaussian estimation are small. In solid line the average value, and the colored area represents the values between the 25%-percentile and the 75%-percentile.

6 | DMP-driven online partitioning

Contents

In this chapter we propose an application of the stochastic analyses proposed in the previous chapters to build a new class of allocation algorithms that use DMP-driven *bin-packing* that we define in Section 6.1. Let $(\Omega, \mathbf{P}, \Gamma, \Pi, \{\theta_t\}_t)$ be a stationary multiprocessor real-time system following a $\sum_i D_i / \sum_i G_i / m / \text{SP}$ queueing model.

We define DMP-driven bin-packing algorithms, which are allocation algorithms allowing to extend single processor scheduling policies to a *restricted migration* algorithm for multiprocessor systems that are DMP-aware. We restrict DMP-driven algorithms to the RM policy in this study. The bin-packing problem is NP-hard. For this reason, we build in this chapter an online processor allocation based on *forward induction* and use the results of previous chapters ensuring this allocation maintains the system stable.

The considered scheduling policy is *work-conserving*, *i.e.*, does not idle when there is workload to be executed, and preemptive, *i.e.*, a job may be preempted before the end of its complete execution, and its execution can be resumed with no cost.

The problem of the restricted scheduling on a multiprocessor system consists in two decisions: the dynamic allocation of processors and the static-priority assignment of tasks.

(i) Allocation: jobs are allocated to processors online; *i.e.*, immediately upon arrival, a job is assigned to a processor based on its expected DMP. The allocation decision is a global load-balancing optimization problem in which jobs are distributed among multiple identical processors by minimizing their DMP. Tasks *migrations* are allowed, *i.e.*, all jobs of a same task do not need to be executed on the same processor. However, we require in this study that a job that is allocated to one and only processor, and must finish its execution on the same processor. This is called *restricted migration*, see Figure 1.4. The restricted migration strategy provides a good compromise between the full migration and the partitioning strategies [Carpenter et al., 2004].

(ii) Priority assignment: the priority assignment is local and consists of solving a

Table 6.1: Maximum utilization bounds for migration strategies of the multiprocessor Rate-Monotonic with reasonable bin-packing

	full migrations	restricted migrations	no migrations
identical processors	$\left[\frac{m^2}{3m-1}, \frac{m+1}{2}\right]$	$ms - (m-1)\max_{\tau_i \in \Gamma} \lambda_i c_i^{max}$	$\left[m(2^{1/m}-1), \frac{m+1}{1+2^{\frac{1}{m+1}}}\right]$
uniform processors	Eq. (6.3)	–	$\sum_{j=1}^{m} s(\kappa_j) n_j (2^{1/n_j} - 1)$

sequencing problem. In this chapter, RM is applied.

The offline and online allocation differ in their complexity. Furthermore, in the case of a stationary system, offline and online allocation become significantly different. Offline allocation involves estimating or averaging arrival times, while online allocation uses the exact arrival time for each job.

6.1 Introduction

The RM scheduling policy is a static-priority scheduling algorithm which assigns each task a priority relatively to its periods - the smaller the period, the higher the priority. Let $u_i^{max} = \lambda_i c_i^{max}$ be the maximum utilization of the task $\tau_i \in \Gamma$. In the case of multiprocessor scheduling, *i.e.*, $m \geqslant 2$, the RM policy is extended in [Andersson et al., 2001] to identical multiprocessor systems global scheduling when

$$\bar{u}_n^{max} \leqslant \frac{m^2}{3m-1}; \ \forall \tau_i \in \Gamma, u_i^{max} \leqslant \frac{m}{3m-2} \tag{6.1}$$

and by allocating jobs to any available processor, all tasks are schedulable with a permitted failure rate equal to zero. In the case of restricted scheduling on identical processors of speed s, authors in [Goossens et al., 2012] prove that if

$$\bar{u}_n^{max} \leqslant ms - (m-1)\max_{\tau_i \in \Gamma} u_i^{max} \tag{6.2}$$

the system is schedulable with permitted failure rates equal to zero.

In [Baruah and Goossens, 2003] authors extend this result to uniform multiprocessor systems with global scheduling static-priority policies. The system is schedulable with permitted failure rates equal to zero when

$$\bar{u}_n^{max} \leqslant \frac{1}{2}\left(\sum_{\kappa\in\Pi} s(\kappa) - (1+\Lambda)\max_{\tau_i\in\Gamma} u_i^{max}\right) \tag{6.3}$$

where $\Lambda = \max_{\kappa_j\in\Pi} \frac{1}{s(\kappa_j)} \sum_{i=j+1}^{m} s(\kappa_i)$ measures the degree by which Π differs from an identical multiprocessor systems. However, this bound Eq. (6.3) is not proven sustainable according execution times and inter-arrival times as discussed in this same paper.

Finally, for partitioned scheduling algorithms, the task set Γ should be partitioned into m tasks subsets of respectively $n_1, \ldots, n_m$ tasks such that for any $\kappa_j \in \Pi$,

$$\bar{u}_n^{max}(\kappa_j) \leqslant n_j(2^{1/n_j} - 1) \tag{6.4}$$

However anomalies can be found in such partitioning, which can be corrected by giving priorities according to decreasing utilizations [Andersson and Jonsson, 2002], *i.e.*, τ_i has priority over τ_j on κ if $u_i^{max} > u_j^{max}$. This priority assignment is not only the appropriate way to partition a static-priority task set, but it is also sustainable if execution times are decreased and inter-arrival times are increased, which is suited for stationary real-time system. For the sake of simplicity, we keep referring to it by RM.

No bound still exists for the uniform restricted case. However, we use the global uniform bound as a baseline, knowing that there exists deadlines misses, while the system is feasible. Quantifying the deadline misses probabilities is the purpose of the first section of this chapter.

Let $(\Omega, \mathbf{P}, \Gamma, \Pi, \{\theta_t\})$ be a stationary real-time system, with the uniform multiprocessor $\Pi = \{\kappa_1, \ldots, \kappa_m\}$ composed of the processors $\kappa \in \Pi$ of speed $s(\kappa) \in [1, s^{max}]$, *i.e.*, κ can process $s(\kappa)$ workload units in one unit of time. We consider Π ordered by decreasing speeds, *i.e.*, $s(\kappa_i) > s(\kappa_j)$ if $i < j$. We consider also the task set $\Gamma = \{\tau_1, \ldots, \tau_n\}$ with stationary inter-arrival times and implicit deadlines, ordered

by decreasing utilization, *i.e.*, τ_i has priority over τ_j if $u_i^{max} > u_j^{max}$. We suppose for all $1 \leqslant i \leqslant n$ and $1 \leqslant k \leqslant m$ that $\lambda_i \mathbf{E}[C_i] < s(\kappa_k)$. We define $\Gamma_i^+(\kappa)$ (resp. $\Gamma_i^-(\kappa)$) to be the set of tasks of priority higher (resp. lower) or equal to the priority of τ_i active on the processor κ at a given time, and let

$$\bar{u}_i(\kappa) = \frac{1}{s(\kappa)} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \lambda_j \mathbf{E}[C_j] \tag{6.5}$$

be the *local mean utilization* of level i in the processor κ and, respectively, let

$$\bar{v}_i(\kappa) = \frac{1}{s(\kappa)} \left(\sum_{\tau_j \in \Gamma_i^+(\kappa)} \lambda_j \mathbf{E}[C_j^2] \right)^{1/2}$$

be the *local deviation* of level i on the processor κ, that is the sum of the utilization (resp. deviation) of the tasks in $\Gamma_i^+(\kappa)$. Let $u_i^{max} = c_i^{max}/t_i^{min}$ be the maximum utilization of τ_i,

$$\bar{u}_i^{max}(\kappa) = \frac{1}{s(\kappa)} \sum_{\tau_j \in \Gamma_i^+(\kappa)} u_i^{max}$$

be the *local maximum utilization* of level i of κ and

$$\bar{v}_i^{max}(\kappa) = \frac{1}{s(\kappa)^2} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \frac{(c_j^{max} - c_j^{min})^2}{t_j^{min}}$$

the *local maximum deviation* of level i in κ. Finally, we suppose that $\bar{u} < \sum_{j=1}^{m} s(\kappa_j)$ and $u_i(\kappa_j) < 1$ for all $i = 1, \ldots, n, j = 1, \ldots, m$. We remind that we say that a task τ_i is schedulable if its DMP is lower than its permitted failure rate α_i. We suppose the task set $\Gamma = \{\tau_1, \ldots, \tau_n\}$ is ordered such that $u_1^{max} > \cdots > u_n^{max}$.

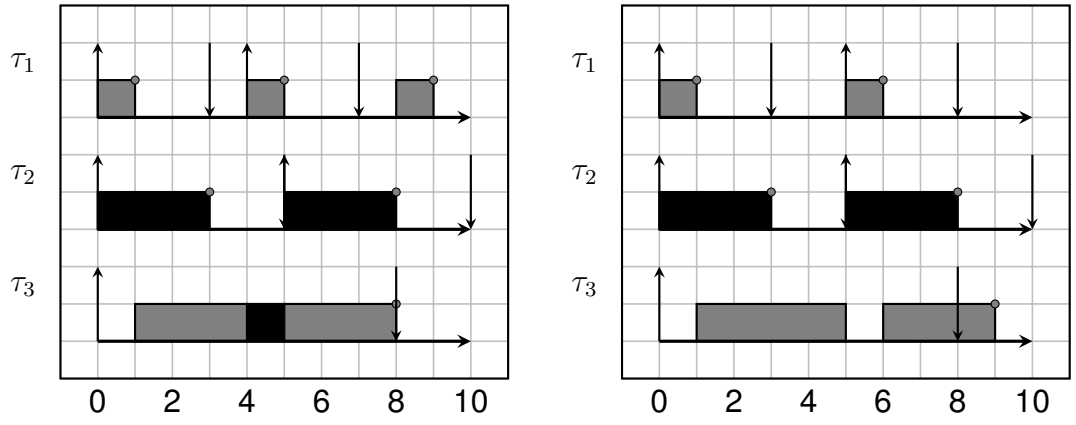
Bin-packing

The goal of bin-packing in multiprocessor scheduling is to assign a set of tasks to a set of processors online. It is used to solve one of the two sub-problems of multiprocessor scheduling listed above. The goal is to assign the tasks to the

processors in a way that the utilization is either increased or decreased. In bin-packing, each processor is viewed as a "bin" and the tasks are viewed as items that need to be placed into the bins. The problem is to find the optimal assignment of tasks to bins such that the total execution time is minimized. The choice of algorithm depends on the specific characteristics of the problem, such as the size of the task set and the number of processors available. In real-time systems, the three most used heuristic approaches are *Reasonable Best-Fit* (Algorithm 1), *Reasonable Worst-Fit* (Algorithm 2) and *Reasonable First-Fit* (Algorithm 3). In the Best-Fit algorithm, the tasks are iterated through and each task is assigned to the processor with the smallest amount of remaining capacity. In the Worst-Fit algorithm, the tasks are iterated through and each task is assigned to the processor with the most remaining capacity. In the Next-Fit algorithm, the tasks are iterated through and each task is assigned to the next available processor. The First-Fit algorithm differs from the Best-Fit and the Worst-Fit algorithms, in that it does not consider an optimization problem when making assignment decisions. The specific algorithm used to solve the bin-packing problem depends on the characteristics of the problem, such as the size of the task set and the number of processors available. A full review on utilization bounds RM and EDF combined with different bin-packing algorithms can be found in [Davis and Burns, 2011].

Anomalies in multiprocessor scheduling and sustainability

In order to generalize results of the deterministic analysis of periodic real-time systems to a probabilistic approach, we assert that a variation on the execution times of jobs and their inter-arrival times does not affect the scheduling process. A scheduling algorithm not sustainable has scheduling anomalies [Andersson and Jonsson, 2002], *e.g.*, Figure 6.1, *i.e.*, scenarios where the optimality of the single processor algorithms does not hold in the multiprocessor scheduling. This phenomenon has to be taken into account in order to use a stochastic analysis and in general to propose a multiprocessor scheduling policy. Unfortunately, online restricted scheduling algorithms are not sustainable with respect to the execution times [Ha and Liu,



(a) $T_1 = 4$ **(b)** $T_1 = 5$

Figure 6.1: Illustration of the non-sustainability of a static-priority scheduling algorithm using the Best-Fit bin-packing algorithm, a set of tasks $\Gamma = \{\tau_1, \tau_2, \tau_3\}$ such that $C_1 = 1, D_1 = 3$; $C_2 = 3, T_2 = 5, D_2 = 5$; $C_3 = 7, T_3 = 20, D_3 = 8$, and an identical multiprocessor $\Pi = \{\kappa_1, \kappa_2\}$ (one in grey, one in black).

1994]. Hence our goal is to decrease as much as possible the number of anomalies leading to deadline misses. In order to use results on deterministic anomaly-free bin-packing algorithms, we provide a heuristic of a *reasonable allocation algorithm*.

Definition 6.1 (From [Lopez et al., 2004])**.** *A Reasonable Allocation Algorithm (RAA) is one which fails to allocate a task only when there is no processor in the system which can hold the task.*

Since the restricted strategy proposed in this chapter is an online partitioning of the system, we can say that that a bin-packing strategy combined with the utilization-bound in Eq. (6.4) provides a restricted scheduling algorithm. There are bin-packing analyses providing solutions to the anomalies they can contain *e.g.*, [Murgolo, 1988]. The specific case of the RM policy with First-Fit is studied in [Andersson and Jonsson, 2002]. The purpose of this chapter is to build such bin-packing algorithm based on DMP that is reasonable.

6.1.1 Problem statement

We suppose a task τ_i is activated. When it is activated, it is potentially delayed by a backlog of distribution μ_{ij} in each processor κ_j, that is the workload of the

Algorithm 1 Reasonable Best-Fit when the task τ_i is activated

$q = 1$
for $\kappa \in \Pi$ **do**
  $n_\kappa = |\Gamma_n^+(\kappa)|$ ▷ The current number of tasks activated on κ
  $u_\kappa = \bar{u}_n^{max}(\kappa) + u_i^{max}/s(\kappa)$
  if $u_\kappa < (n_\kappa + 1)(2^{\frac{1}{n_\kappa+1}} - 1)$ **and** $q > 1 - u_\kappa$ **then**
    $q \leftarrow 1 - u_\kappa$
    $a \leftarrow (\tau_i, \kappa)$
if $q = 1$ **then**
  return $\varnothing$ ▷ Discard τ_i
else
  return a

Algorithm 2 Reasonable Worst-Fit when the task τ_i is activated

$q = 0$
for $\kappa \in \Pi$ **do**
  $n_\kappa = |\Gamma_n^+(\kappa)|$ ▷ The current number of tasks activated on κ
  $u_\kappa = \bar{u}_n^{max}(\kappa) + u_i^{max}/s(\kappa)$
  if $u_\kappa < (n_\kappa + 1)(2^{\frac{1}{n_\kappa+1}} - 1)$ **and** $q > u_\kappa$ **then**
    $q \leftarrow u_\kappa$
    $a \leftarrow (\tau_i, \kappa)$
if $q = 0$ **then**
  return $\varnothing$ ▷ Discard τ_i
else
  return a

Algorithm 3 Reasonable First-Fit when the task τ_i is activated

for $\kappa \in \Pi$ **do**
  $n_\kappa = |\Gamma_n^+(\kappa)|$ ▷ The current number of tasks activated on κ
  $u_\kappa = \bar{u}_n^{max}(\kappa) + u_i^{max}/s(\kappa)$
  if $u_\kappa < (n_\kappa + 1)(2^{\frac{1}{n_\kappa+1}} - 1)$ **then**
    return (τ_i, κ)
return $\varnothing$ ▷ Discard τ_i

unfinished jobs of higher priority tasks activated before τ_i. We denote the action of allocation τ_i on κ_j as the couple $a = (\tau_i, \kappa_j)$. For all $j = 1, \ldots, m$, let $\{\mu'_{kj}\}_k$ be the sequence of backlog distribution of jobs of level k activated on the processor κ_j induced by the allocation a. The goal of the allocation is then to focus not only the DMP $p_i(a, \mu_{ij})$ of τ_i conditionally to the backlog μ_{ij}, but also the DMP $\{p_k(a, \mu'_{kj})\}_{k>i}$ of the tasks blocked by τ_i induced by the choice of κ_j for the task τ_i. Thus the allocation problem is to find the processor κ^* minimizing both the probability $p_i(a, \mu_{ij}) \prod_{k>i} p_k(a, \mu'_{kj})$ and a quantity $q^{RAA}(a)$ of a given RAA that should be maximized, at each activation of the task τ_i, *i.e.*,

$$\kappa^* = \text{argmax}_{a=(\tau_i,\kappa)} - \log p_i(a, \mu_{ij}) - \sum_{k>i} \log p_i(a, \mu'_{kj}) - \log q^{RAA}(a) \tag{6.6}$$

which does not mean that we minimize the DMP of a particular task, but overall the metric is the DMP to decrease the number of deadline misses as much as possible.

DMP-driven decisions are not based in fact on the quality of the estimation of DMP, but more in the *order* induced by those. As in the priority assignment problem, the allocation problem builds some kind of hierarchy to decide which processor to allocate to which task. This is important to understand, and this is how probabilities can play a central role even in critical applications.

6.1.2 Local TDA

In Chapter 4 we used queueing theory arguments that any set of tasks active on a processor with a local mean utilization greater than 1 is not schedulable with a static-priority scheduling algorithm. Similarly, we use the fact that $\bar{u}_n(\kappa) < s(\kappa)$ for every processor $\kappa \in \Pi$ at any time, is a necessary condition for the schedulability of any global static-priority scheduling policy, and develop a allocation policy using forward induction to allocate jobs to processors by minimizing their DMP. The problem is stated as follows: find an optimal allocation policy with RM priorities focused on the DMP in the worst-case scenario.

Let κ be a processor at given instant in time, $\beta_i(\kappa)$ be the local backlog of level

i allocated to κ and $\hat{W}_i(t)$ the demand of level i at this same time. Suppose a job $\tau_{i,j}$ is allocated to a processor κ. According to Chapter 4, the TDA of this job is

$$\hat{R}_{i,j} = \inf \left\{ t > 0 : \beta_{i-1}(\kappa) + C_{i,j} + \hat{W}^j_{i-1}(t) \leqslant s(\kappa)t \right\} \tag{6.7}$$

where the $\hat{W}^j_i, j \in \mathbb{N}$ are independent copies of $\hat{W}_i$, itself a Brownian motion of drift $\bar{u}_i$ and deviation $\bar{v}_i$, *c.f.*, Chapter 3. This analysis is possible thanks to the following theorem.

Theorem 6.1 (Theorem 1 in [Andersson and Jonsson, 2002])**.** *The deterministic TDA of RM is sustainable according to execution times and periods if the local priority assignment is such that it orders tasks by decreasing local maximum utilization.*

Just like in the single processor case, this last theorem is the reason TDA holds as a probabilistic analysis. Any non-sustainable analysis in the deterministic case cannot be extended to a stochastic version.

Remark. *There are two differences with the single processor TDA:*

(i) *the time budget of the processor κ is $t \to s(\kappa)t$ rather than $t \to t$,*

(ii) *it is more pessimistic because considering the local demand $\hat{W}_{i-1}$ as the interference of tasks of level i is equivalent to assume that all jobs of higher priority tasks are going to be allocated in κ. This is false, but it gives us a safe bound to work with. The online method that we build in the next section is non-clairvoyant, one cannot say when and where future jobs will run, i.e., jobs released after this activation of τ_i. Hence, at this stage we cannot do better than over-estimate the local demand.*

6.2 DMP-RAA bin-packing

The online allocation has the benefit over the offline method to be adaptive. However, as explained in the previous section, online allocation for restricted scheduling

policies in not sustainable, in the sense that it generates anomalies with the variation of execution times. The goal of this section in to provide a penalty to a RAA such that not only the bin-packing decision is made according to RAA, but also adapting the decision to the induced DMP. We use this adaptive property to reward the good decisions of the scheduling algorithm with the forward induction method. Forward induction is a method of solving a decision problem by breaking it down into a sequence of decisions and working forward in time. This process begins with the first decision in the sequence and then continues to the next decision, considering the outcomes of the previous decision. The process ends when all decisions in the sequence have been evaluated. In order to do so, we define the good metric and provide a method to compute it. The DMP are our metric, although it does not mean that these decisions are optimal according to DMP. Although, we see that such decisions have the benefit of reducing the number of preemptions.

6.2.1 Forward induction

We suppose that we have a system with states $\mu = (\mu_1, \ldots, \mu_n) \in \mathcal{M}$, possible actions $a \in \mathcal{A}$, an immediate reward $r(\mu, a)$ of taking the action a in the state μ with a RAA. The goal of forward induction is to find a policy $\kappa^* : \mathcal{M} \to \mathcal{A}$ maximizing

$$\sum_{i=1}^{n} r(\mu_i, \kappa^*(\mu_i)) \tag{6.8}$$

where $\mathcal{M}$ is the set of possible states, and $\mathcal{A}$ the set of actions. This policy can be achieved online by satisfying the Bellman equation [Bellman, 1957]:

$$c_i(\mu) = \max_{a \in \mathcal{A}_i} r(\mu_i, a) + c_{i+1}(g_a(\mu)) \tag{6.9}$$

at each iteration, where $\mathcal{A}_i$ is the set of available actions; $r(\mu, a)$ is the expected immediate reward of the action a taken on state μ, $c_{i+1}(g_a(\mu))$ is the cost of getting to the state $g_a(\mu) = (\mu_1 P_a^1, \ldots, \mu_n P_a^n)$ on the next iteration, where $\mu_k P_a^k$ is the

next backlog distribution of level k if the action a is taken, with

$$\mu P(x) = \int \mu(dy) P(y, x)$$

and $P_a^k(y, x)$ is the probability to transition from a level k backlog y to x when taking the action a.

The optimal decision κ chooses the action that maximizes both the immediate expected reward and the expected reward further in time, *i.e.*,

$$\kappa^*(\mu) = \text{argmax}_{a \in \mathcal{A}_i(\mu)} \, r(\mu_i, a) + c_{i+1}(g_a(\mu)) \tag{6.10}$$

From the Bellman equation [Bellman, 1957], we know that satisfying Eq. (6.6) for all tasks minimizes all the DMP. In our case, the reward of a particular action (assigning a task to a specific processor) provides the asset to be optimized (DMP minimized). The allocation is the response to a specific state of backlogs distributed in several processors based on the DMP induced by the previous backlogs. In order to use forward induction, we formalize:

(i) A set of actions,

(ii) A set of states,

(iii) A reward function to evaluate the taken action at a specific state,

(iv) A transition probability for each action.

As only one job per task can be allocated to a processor (because of the discarding policy), we refer only to tasks and not to jobs. The jobs are referred to as activated tasks at a given time on a given processor. Another abuse of notation we use in the following, is that every time we mention a processor $\kappa \in \Pi$, we mean a processor at a given time. Since we are only interested in transitions of backlogs at a time t to the backlog induced by an allocation, there is no need to make all variables dependent on $t > 0$ for the sake of simplicity, when in the

end everything is characterized by transitions. Hence, we choose not to index every variable on time, and warn the reader that everything is presented conditionally to the state of processors at the activation dates of the tasks.

6.2.2 State space

The worst-case reasoning is the paradigm used in real-time systems in order to be safe, *i.e.*, make decisions according the worst-case scenario. We call *critical instant* of level i the scenario providing the worst-case blocking time of the task τ_i, *i.e.*, the largest backlog that can interfere with a job of τ_i. The scheduling policy is then adapted for this critical instant. We use in the following the result of the Chapter 4 on the critical instant of stationary real-time systems.

In single processor systems, the critical instant comes in the scenario of a simultaneous activation of all tasks. In multiprocessor systems however, the critical instant of level i is not always the simultaneous activation of all tasks [Guan and Yi, 2012], because of the fact that tasks can be activated on different processors. To address this problem, we think locally, *i.e.*, by processor, and use the information on the local utilization of each processor.

Proposition 6.1 (Local critical instant)**.** *Let $\kappa \in \Pi$ and $\tau_i \in \Gamma_n^+(\kappa)$. If $\bar{u}_i(\kappa) < 1$, the worst-case blocking time of level i in κ is*

$$B_i^{max}(\kappa) = \sum_{\tau_j \in \Gamma_i^+(\kappa)} C_j \tag{6.11}$$

thus the simultaneous activation of all tasks of lower priority than τ_i on κ. We call this scenario the local critical instant of level i of κ. $B_i(\kappa)$ is the sum[1] of the worst-case blocking time of the tasks of higher or equal priority than τ_i on the processor κ, i.e., the remaining workload to process by the processor κ before letting any job of $\tau_{i+1}, \tau_{i+2}, \ldots$ execute.

Proof. We suppose $\tau_i \in \Gamma_n^+(\kappa)$, *i.e.*, τ_i is active on κ, and $\bar{u}_i(\kappa) < 1$ and let

[1] With $\sum_{x \in \emptyset} x = 0$.

$\Gamma_i^+(\kappa) = \{\tau_{i_1}, \ldots, \tau_{i_p}\}$. Since the processor κ is fixed we can use the single processor reasoning introduced in Chapter 4. Thus the critical instant of level i is when all tasks activated on κ are released at the same time. Then the worst-case blocking time is

$$B_i^{max} = \sum_{k=1}^{p} C_{i_k} \tag{6.12}$$

c.f., Proposition 4.4. Thus the simultaneous activation of all tasks active on κ of lower priority than τ_i. For a speed greater than 1, it is sufficient to divide every execution time by the speed of the processor. Thus we have the result. □

Remark. *We cannot conclude as we did in Chapter 4 that the WCRT comes from this critical instant. However, we use the local critical instant to bound the demand at each allocation time.*

Let $\mu(\kappa) = (\mu_1(\kappa), \ldots, \mu_n(\kappa))$ be the vector of backlog distributions on a processor κ at a given time. In order to use forward induction, we exhibit an iterative behavior of the transitions of $\mu(\kappa)$ over time. Furthermore, backlogs cannot be measured in real-time, as execution times jobs are unknown until they finish. $t > 0$. Instead of looking at the exact values of the backlogs, we look at how the distribution of $B_i(\kappa)$ goes over time. Hence, the states considered for our decision are the distributions

$$\mathcal{M} = \{\mu(\kappa_j) : \forall i = 1, \ldots, n, \bar{u}_i(\kappa_j) < 1, j = 1, \ldots, m\}$$

In order to take online decisions, all possible worst-case blocking time distributions must be computed *offline*, before the run time scheduling. However, there is no need to do it for each processor.

Algorithm 4 Offline computation of the worst-case blocking time distributions

```
for k ∈ {1, ..., n} do
    for (τ_{i_1}, ..., τ_{i_k}) ∈ Γ^k do
        μ_{i_1,...,i_k} = f_{i_1} * ··· * f_{i_k}
```

This offline computation is exponential in complexity. Saving those distributions,

and refer to them in a table online allows to bypass the complexity of this problem for online decisions.

6.2.3 Reward function

In order to build an allocation decision that based on the DMP, we use the representation We adapt this method for each processor by summing only the utilizations of the tasks that are allocated to a processor κ_j. This representation permits in a two-step allocation to (i) approximate the local worst-case blocking time of a given processor, (ii) approximate the distribution of the future backlog, (iii) optimize the distribution of the future backlog of higher priority tasks allocated on the same processor by minimizing DMP, (iv) include the minimal DMP in the allocation of a bin-packing algorithm RAA.

As we want a policy minimizing DMP, conditionally to the worst-case blocking times of the system. We set the immediate expected reward function for the allocation a and backlog distribution $\mu \in \mathcal{M}$ as

$$r_i(\mu_i, a) = -\log p_i(a, \mu_i) \tag{6.13}$$

where $p_i(a, \mu)$ is the DMP of τ_i induced by the allocation a in the state μ, and $q^{RAA}(a) \in (0, 1]$ depends on the criteria of the bin-packing RAA to be minimized. For the action a, the tasks τ_k with $k \leqslant i$ are not considered, as they cannot be preempted or interfered by tasks with a lower priority.

Let $\mu(\kappa) \in \mathcal{M}$ be the state of the backlog distributions at a time $t > 0$. The immediate expected reward of an allocation $a = (\tau_i, \kappa)$ is $r_i(\mu_i(\kappa), a)$ and the cost of the allocation $a = (\tau_i, \kappa)$ on $\Gamma_i^-(\kappa)$ is

$$c_{i+1}(g_a(\mu(\kappa))) = \sum_{\tau_k \in \Gamma_i^-(\kappa)} r_k\left(\mu_k(\kappa) P_a^k, a\right)$$

Lemma 6.1. *The DMP induced by the allocation $a = (\tau_i, \kappa)$, where the backlog*

distribution of level i in κ is μ is

$$p_i(a,\mu) = \iint p_i^\kappa(t,x)dG_k(t)d\mu(x) \tag{6.14}$$

where $p_i^\kappa(t,x) = \Psi(x,t;\bar{u}_{i-1}(\kappa),\bar{v}_{i-1}(\kappa))$ and Ψ is as defined in Eq. (4.3).

Proof. We suppose the processor κ is fixed, and let $a = (\tau_i, \kappa)$. With the approximation of response times proposed in Chapter 4, we get the DMP conditionally to a backlog of level i fixed to a value x:

$$p_i^\kappa(T,x) = \mathbf{P}\left(\inf_{t\in[0,T]} W_{i-1}^\kappa(t) - s(\kappa)t > -x\right) \tag{6.15}$$

where W_{i-1}^κ is a Brownian motion such that $W_{i-1}^\kappa(0) = 0$, of drift $\bar{u}_{i-1}(\kappa)$ and deviation $\bar{v}_{i-1}(\kappa)$. Furthermore, we know from [Jeanblanc et al., 2009, p. 147] that for a given $T > 0$,

$$\mathbf{P}\left(\inf_{t\in[0,T_i]} W_{i-1}(t) - s(\kappa)t > -x\right) = \int p_i^\kappa(t,x)dG_i(t) \tag{6.16}$$

for any static-priority preemptive policy and any processor $\kappa \in \Pi$.

Finally, the DMP is computed by integrating on x the backlog distribution function μ. □

6.2.4 Action space

The global action space is $\mathcal{A} = \Gamma \times \Pi$, *i.e.*, an action is determined by a task and a processor. However, the system decides which task is allocated. The decision taken by the allocation is then to choose the right processor. As the state space, we reduce the action space by allowing only actions keeping the utilization of processors under one, and the DMP of each active task under its permitted failure rate. Hence,

when a task τ_i arrives, the set of possible actions is

$$\mathcal{A}_i = \Big\{\kappa \in \Pi : a = (\tau_i, \kappa) \in \mathcal{A}, \frac{u_i^{max}}{s(\kappa)} < 1 - \bar{u}_n^{max}(\kappa), \\ \forall \tau_k \in \Gamma_i^-(\kappa), r_k(\mu_k(\kappa) P_a^k, a) > -\log \alpha_k \Big\}$$

We note that the bound used to define the set of actions is an heuristic that we propose without proof but with extensive numerical evaluations against the state-of-the-art algorithms. The idea behind this bound is that as we ensure that there is no deadline misses with a maximum local utilization of κ lower than

$$\big(|\Gamma_n^+(\kappa)| + 1\big)\left(2^{\frac{1}{|\Gamma_n^+(\kappa)|+1}} - 1\right)$$

and that the system is feasible with a maximum local utilization lower than 1, then we quantify the number of deadline misses for an utilization between those two bounds.

6.2.5 Transition matrix

Transitions between states in $\mathcal{M}$ are driven by execution times distributions. Whenever a task is allocated to a processor, regardless of the current backlog of this processor, the DMP induced by the allocation of a task takes its execution time distribution into account by *convolving* it to the current worst-case blocking time distribution.

Proposition 6.2. *Let τ_i be a pending task, $\kappa \in \mathcal{A}_i$ and $a = (\tau_i, \kappa)$.*

(i) *If $\tau_k \in \Gamma_i^+(\kappa)$, the probability of transitioning from a backlog of level k equal to b, to b' by taking the action a is*

$$P_a^k(b, b') = \delta_b(b') \tag{6.17}$$

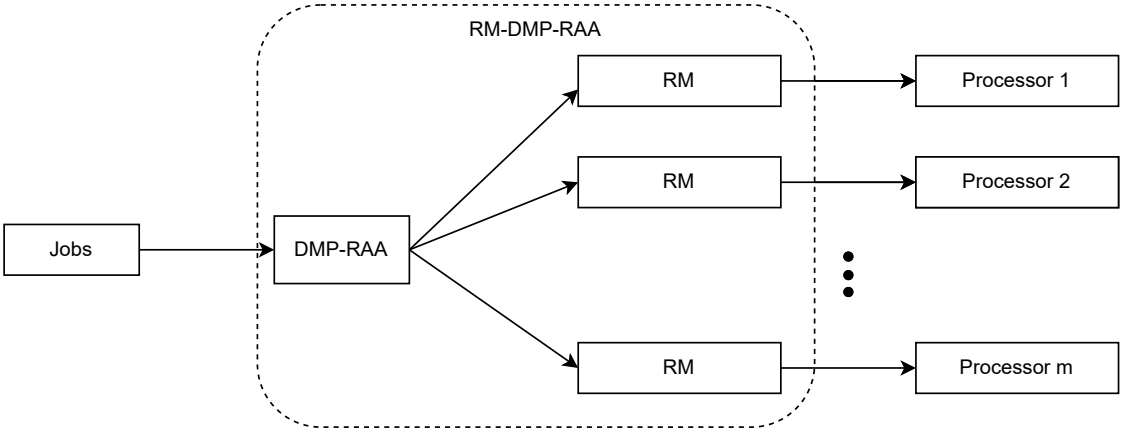


Figure 6.2: Scheme of the RM-DMP-RAA algorithm

where δ_x is the Dirac measure on x, i.e.,

$$\delta_x(A) = \begin{cases} 1 & if\ x \in A \\ 0 & otherwise \end{cases} \tag{6.18}$$

(ii) *if $\tau_k \in \Gamma_i^-(\kappa)$, the probability that backlog of level k goes from b to b' by taking the action a is*

$$P_a^k(b, b') = f_i\left(s(\kappa)(b' - b)\right) \tag{6.19}$$

where $f_i = dF_i/dx$ is the probability function of C_i.

Proof. Let τ_i be activated on the processor κ.

- First, if $\tau_k \in \Gamma_i^+(\kappa)$, the allocation $a = (\tau_i, \kappa)$ does not affect the active job of τ_k. Hence the backlog of level k denoted b remains the same and the only transition possible is from b to itself.
- Secondly, if $\tau_k \in \Gamma_i^-(\kappa)$, the execution time of τ_i is added to the backlog of level k on the processor κ. Hence, the transition probability from a backlog of level k denoted b and a backlog b' is $\mathbf{P}(b' = C_i/s(\kappa) + b) = f_i\left(s(\kappa)(b' - b)\right)$.

□

Algorithm 5 DMP-RAA when the task τ_i is activated

```
c ← 0
for κ ∈ A_i do
    a ← (τ_i, κ)
    if c < r_i(μ_i(κ), a) + c_{i+1}(g_a(μ(κ))) − log(q^{RAA}(a)) then
        c ← r_i(μ_i(κ), a) + c_{i+1}(g_a(μ(κ)))
        a* ← a
if c = 0 then
    return ∅                                   ▷ Discard τ_i
else
    return a*
```

6.3 Using analytical DMP

The DMP-driven algorithms takes any analysis providing a DMP for each tasks, and arranges jobs in a certain order based on the probability of missing a deadline. This means that, any approximation of deadline miss probability could supposedly work, whatever the quality of the approximation or the the fact that it bounds or not the *exact* DMP. What is important is the order provided by those DMP. The analytical approximation of the DMP is useful as it is easy and fast to compute, and the error is not, necessarily, a problem. However, the DMP model chosen must have strong mathematical background and be sustainable to small variations in the task set parameters. This is provided by the Hoeffding bound that we use in this section.

Proposition 6.3 (Local Hoeffding DMP)**.** *Let $\kappa \in \Pi$, $\tau_i \in \Pi$ and $a = (\tau_i, \kappa)$. We suppose $\bar{u}_i(\kappa) < 1$, and let*

$$p_i^H(\kappa) = \exp\left(-t_i^{min}\frac{(1-\bar{u}_i(\kappa))^2}{\bar{v}_i^{max}(\kappa)}\right) \tag{6.20}$$

be the Hoeffding DMP of τ_i on κ, where $\bar{v}_i^{max}(\kappa) = \sum_{\tau_j \in \Gamma_i^+(\kappa)} (c_j^{max} - c_j^{min})^2/t_j^{min}$ is the local maximum deviation of τ_i in κ.

If $t_i^{min} > \frac{1}{1-\bar{u}_i(\kappa)} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \mathbf{E}[C_j]/s(\kappa)$, then

$$p_i(a, \mu_i(\kappa)) \leqslant p_i^H(\kappa)$$

Proof. We use the same proof as in Proposition 3.5 with $\bar{W}_i^\kappa$ as a Brownian motion of drift $\bar{u}_i(\kappa)$ and deviation $\bar{v}_i(\kappa)$, hence for $t \in (0, t_i^{min})$, we obtain that

$$\bar{u}_i(\kappa)t + \sum_{\tau_j \in \Gamma_i^+(\kappa)} \frac{\mathbf{E}[C_j]}{s(\kappa)} > \mathbf{E}[\bar{W}_i^\kappa(t)]$$

for all $t > 0$. Thus, $t > \frac{1}{1-\bar{u}_i(\kappa)} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \frac{\mathbf{E}[C_j]}{s(\kappa)}$ implies that $t > \mathbf{E}[\bar{W}_i^\kappa(t)]$. Finally we get the same inequality with $\bar{u}_i(\kappa)$ and $\bar{v}_i^{max}(\kappa)$. For a more detailed proof one may see the proof of Proposition 3.5. □

Algorithm 6 DMP-RAA with Hoeffding DMP when the task τ_i is activated

```
c ← 0
for κ ∈ Π do
    if κ ∈ A_i^H then
        if c < t_i^min (1−ū_i(κ))^2 / (v̄_i^max(κ)) + Σ_{τ_k∈Γ_i^-(κ)} t_k^min (1−ū_k(κ)−λ_i E[C_i]/s(κ))^2 / (v̄_k^max(κ)+v_i^max/s(κ)^2) − log q^RAA(a) then
            c ← t_i^min (1−ū_i(κ))^2 / (v̄_i^max(κ)) + Σ_{τ_k∈Γ_i^-(κ)} t_k^min (1−ū_k(κ)−λ_i E[C_i]/s(κ))^2 / (v̄_k^max(κ)+v_i^max/s(κ)^2) − log q^RAA(a)
            a ← (τ_i, κ)
if c = 0 then
    return ∅                                   ▷ Discard τ_i
else
    return a
```

Thus, we can use the *stateless* immediate reward of the allocation $a = (\tau_i, \kappa)$

$$r(a) = -\log p_i^H(\kappa) - \log q^{RAA}(a) = t_i^{min} \frac{(1-\bar{u}_i(\kappa))^2}{\bar{v}_i^{max}(\kappa)} - \log q^{RAA}(a)$$

for the allocation $a = (\tau_i, \kappa)$, and the cost of a on higher priority tasks would be

$$\sum_{\tau_k \in \Gamma_i^-(\kappa)} t_k^{min} \frac{(1-\bar{u}_k(\kappa) - \lambda_i \mathbf{E}[C_i]/s(\kappa))^2}{v_k^{max}(\kappa) + v_i^{max}/s(\kappa)^2}$$

where $v_i^{max} = (c_i^{max} - c_i^{min})^2/t_i^{min}$ is the maximum deviation of τ_i. This is where analytical approximation of DMP shines: their complexity is $\mathcal{O}(1)$, hence the scheduling policy is as simple as it can be. We adapt RM-DMP-RAA with the Hoeffding DMP in Algorithm 6.

In order to make this algorithm reasonable, we add a rule coming from this

last theorem: for an allocation $a = (\tau_i, \kappa)$, we require that τ_i satisfies $t_i^{min} > \frac{1}{1-\bar{u}_i(\kappa)} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \frac{\mathbf{E}[C_j]}{s(\kappa)}$ and all tasks $\tau_k \in \Gamma_i^-(\kappa)$ satisfy

$$t_k^{min} > \frac{1}{1 - \bar{u}_k(\kappa) - \lambda_i \mathbf{E}[C_i]/s(\kappa)} \left(\frac{\mathbf{E}[C_i]}{s(\kappa)} + \sum_{\tau_j \in \Gamma_k^+(\kappa)} \frac{\mathbf{E}[C_j]}{s(\kappa)} \right)$$

Hence we redefine the set of possible actions for the Hoeffding DMP

$$\mathcal{A}_i^H = \left\{ \kappa \in \mathcal{A}_i : t_i^{min} > \frac{1}{1 - \bar{u}_i(\kappa)} \sum_{\tau_j \in \Gamma_i^+(\kappa)} \frac{\mathbf{E}[C_j]}{s(\kappa)}, \right.$$
$$\left. \forall \tau_k \in \Gamma_i^-(\kappa), t_k^{min} > \frac{1}{1 - \bar{u}_k(\kappa) - \lambda_i \mathbf{E}[C_i]/s(\kappa)} \left(\frac{\mathbf{E}[C_i]}{s(\kappa)} + \sum_{\tau_j \in \Gamma_k^+(\kappa)} \frac{\mathbf{E}[C_j]}{s(\kappa)} \right) \right\} \tag{6.21}$$

in order to keep it reasonable. Any new DMP approximation used with DMP-RAA should provide its own rules to be reasonable.

6.4 Simulations

The RM-DMP-RAA (Algorithm 5) algorithms are compared in the next section to other comparable scheduling policies. The task set used in these simulation is stationary, with parameters shown in Table 6.2. The performance of DMP-RAA with Hoeffding DMP is shown in Figure 6.3, and a comparison in the periodic and stationary case is shown in Figure 6.5. The task set is such that $\bar{u}_n^{max} < \sum_{\kappa \in \Pi} s(\kappa)$ and $\max_{\kappa \in \Pi} \max_{\tau_i \in \Gamma} \frac{u_i^{max}}{s(\kappa)} < 1$. First, we compare the full DMP and the DMP using Heoffding DMP. Then we compare the stationary and the periodic cases, *i.e.*, considering only t_i^{min} as inter-arrival times, with DMP-First-Fit ($q^{RAA}(a) = 1$). Then we compare DMP-RAA with RAA for the Best-Fit and Worst-Fit algorithms. Finally we compare DMP-Best-Fit with some state-of-the-art global scheduling algorithms.

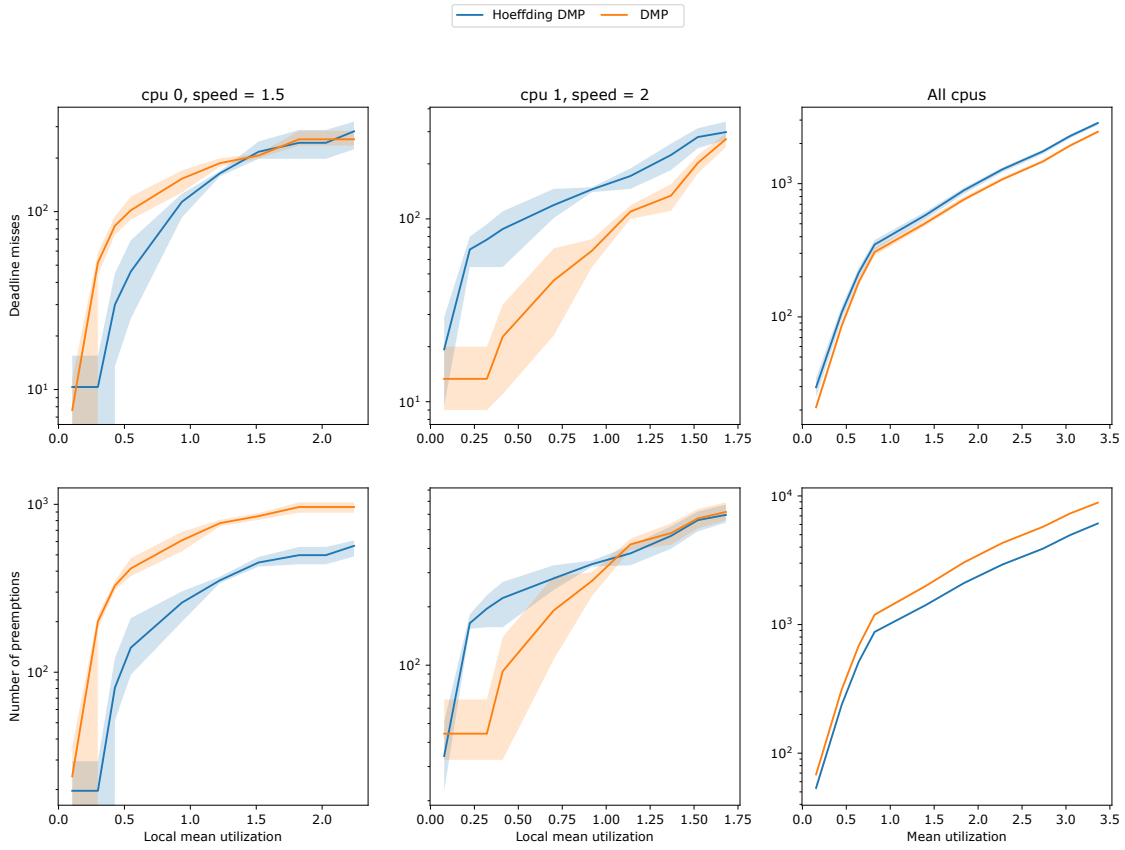


Figure 6.3: Comparison of 100 instances of DMP-First-Fit (yellow) and DMP-First-Fit with Hoeffding DMP (blue). The first row represents the number of deadline misses, the second row the number of preemptions. Each column represents a processor and the last column the accumulation of all processors. In solid line the average value and the colored area represent the values between the 25%-percentile and the 75%-percentile.

6.4.1 DMP-First-Fit vs. DMP-First-Fit with the Hoeffding DMP

We see in Figure 6.3 that, as the full DMP-First-Fit ($q^{RAA}(a) = 1$) method has a lower deadline miss ratio, at the end both algorithms provide equivalent metric. This means that the using an approximation of DMP, such as the Hoeffding bound, is a good tradeoff in performance/complexity.

Table 6.2: Parameters of the task set used in the simulation of Section 6.4

τ_i	$\{c_i^{min}, \ldots, c_i^{max}\}$	f_i	$\{t_i^{min}, \ldots, t_i^{max}\}$	dG_i/dt	$\lambda_i \mathbf{E}[C_i]$	$\lambda_i \mathbf{E}[C_i^2]$	$\bar{u}_i$	u_i^{max}	$\bar{u}_i^{max}$
τ_1	$(40, 52)$	$(0.8, 0.2)$	$(132, 142)$	$(0.5, 0.5)$	0.309	0.168	0.309	0.394	0.394
τ_2	$(30, 45)$	$(0.75, 0.25)$	$(144, 154)$	$(0.5, 0.5)$	0.227	0.283	0.536	0.312	0.706
τ_3	$(10, 20)$	$(0.9, 0.1)$	$(121, 131)$	$(0.5, 0.5)$	0.087	0.071	0.623	0.165	0.872
τ_4	$(3, 6)$	$(0.7, 0.3)$	$(40, 50)$	$(0.5, 0.5)$	0.087	0.042	0.71	0.15	1.022
τ_5	$(8, 13)$	$(0.7, 0.3)$	$(109, 119)$	$(0.5, 0.5)$	0.083	0.046	0.793	0.119	1.141

Table 6.3: Processor parameters used in Section 6.4

κ_i	$s(\kappa_i)$
κ_1	4
κ_2	3
κ_3	1.5

6.4.2 RM-DMP-First-Fit on periodic vs. stationary task sets

We see in Figure 6.4 a comparison of RM-DMP-First-Fit, where

$$q^{RFF}(a) = 1$$

with the Hoeffding DMP between the stationary and periodic inter-arrival times with the same inter-arrival rates. We see that its performance are quite equivalent. This justifies that inter-arrival times act on the response times distributions only through its rate, *e.g.*, Eq. (4.7).

6.4.3 DMP-RAA vs RAA

We test DMP-RAA against RAA on the two bin-packing algorithms Reasonable Best-Fit (RBF), where

$$q^{RBF}((\tau_i, \kappa)) = 1 - \frac{u_i^{max}}{s(\kappa)} - \bar{u}_n^{max}(\kappa)$$

see Figure 6.6 and Reasonable Worst-Fit (RWF) where

$$q^{RWF}((\tau_i, \kappa)) = \frac{u_i^{max}}{s(\kappa)} + \bar{u}_n^{max}(\kappa)$$

see Figure 6.7, on a task set and a processor set shown in Tables 6.2. On both figures we can see a slight improvement of the proportion of deadline misses (first row). The number of preemptions (second row) and the number of migrations (third row) stay equivalent to the original RAA $\in$ {RBF, RWF} allocation algorithm.

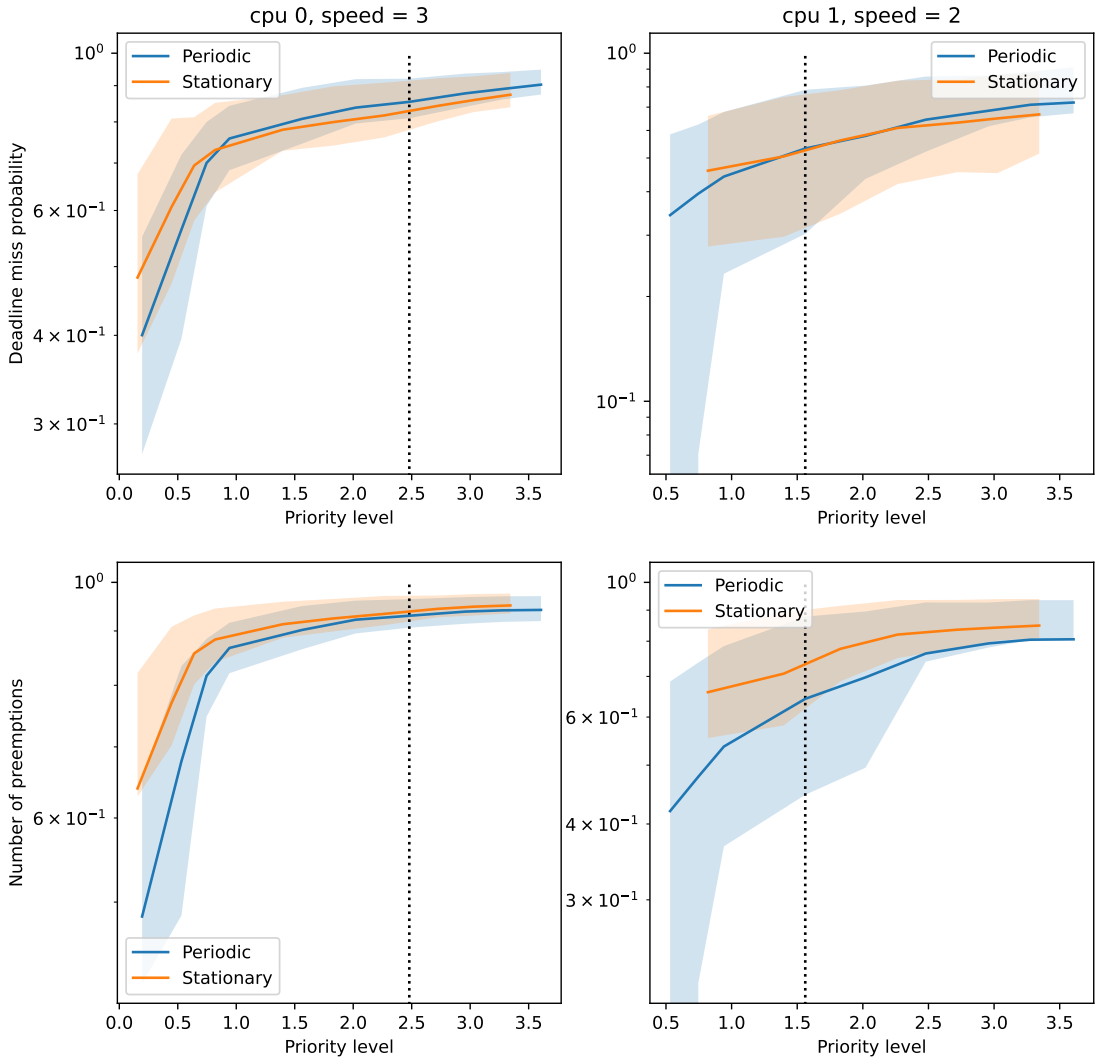


Figure 6.4: Comparison of RM-DMP-First-Fit on a stationary task set and its periodic equivalent. The first row represents deadline misses, the second row preemptions. Each column represents a processor. The dotted line represents deadline misses when the maximal local utilization exceeds 1. In solid line the average value and the colored area represent the values between the 25%-percentile and the 75%-percentile.

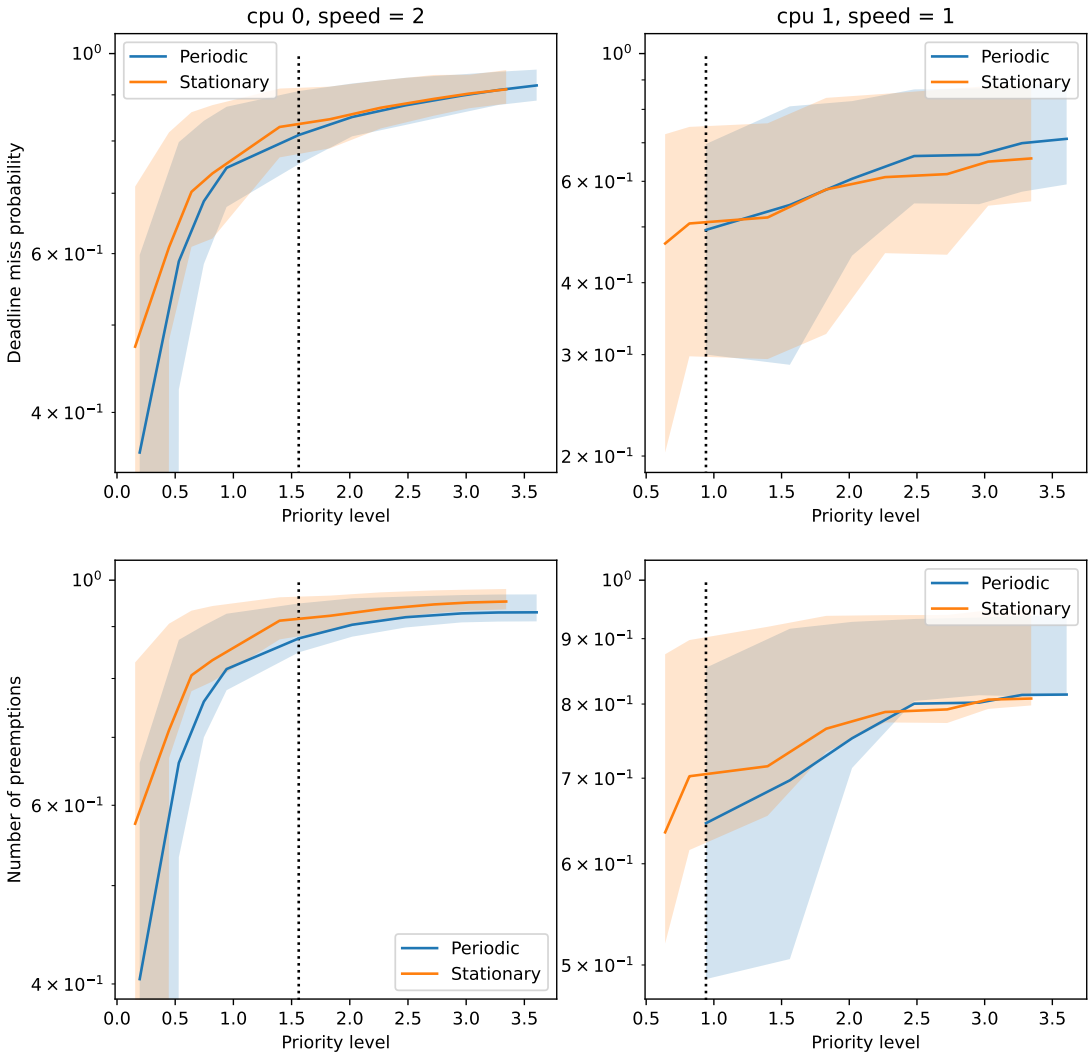


Figure 6.5: Comparison of RM-DMP-First-Fit with Hoeffding DMP on a stationary task set and its periodic equivalent. The first row represents the number of deadline misses, the second row the number of preemptions. Each column represents a processor. The dotted line represents deadline misses when the maximal local utilization exceeds 1. In solid line the average value and the colored area represent the values between the 25%-percentile and the 75%-percentile.

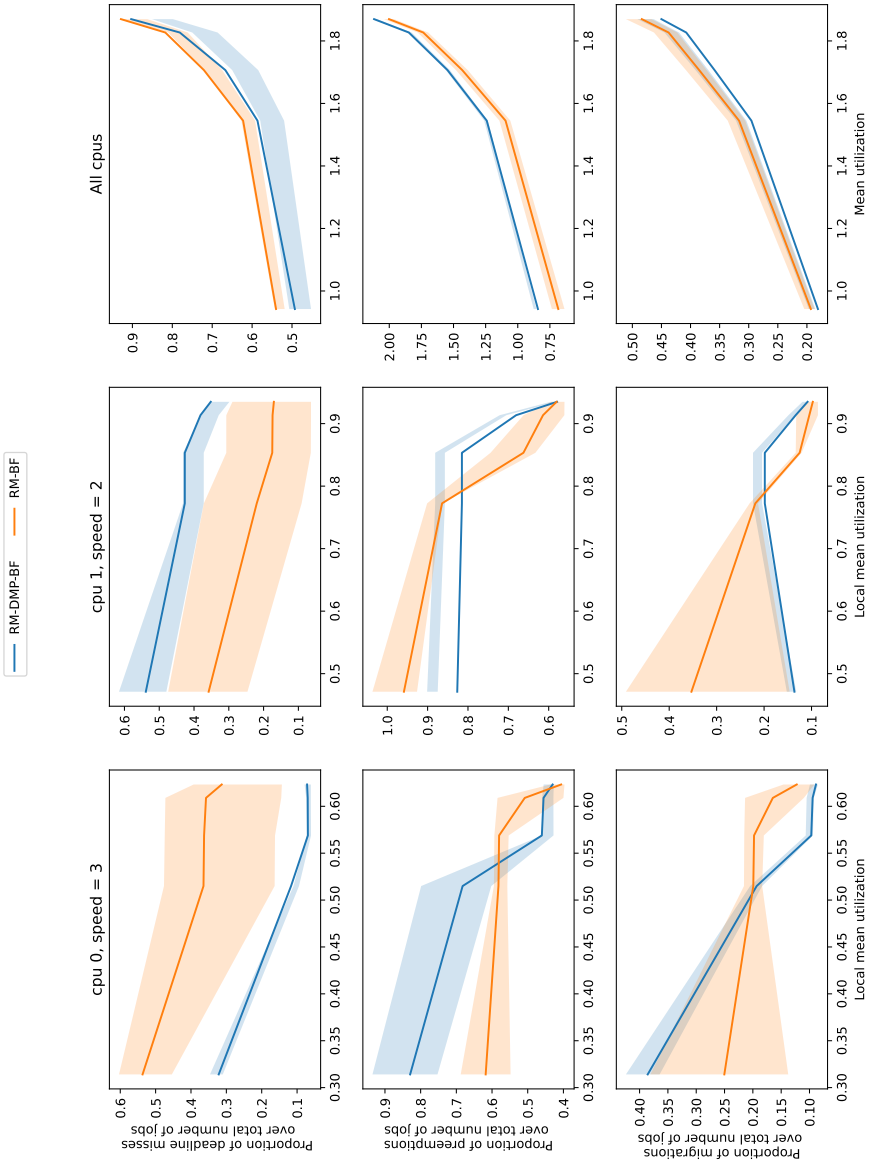


Figure 6.6: Comparison of RM-DMP-First-Fit and RM-First-Fit the a stationary task set shown in Tables 6.2. The first row represents deadline misses, the second preemptions the third one migrations. Each column rep rents a processor. In solid line the average value, and the colored area represents the values between the 25%-percentile and the 75%-percentile.

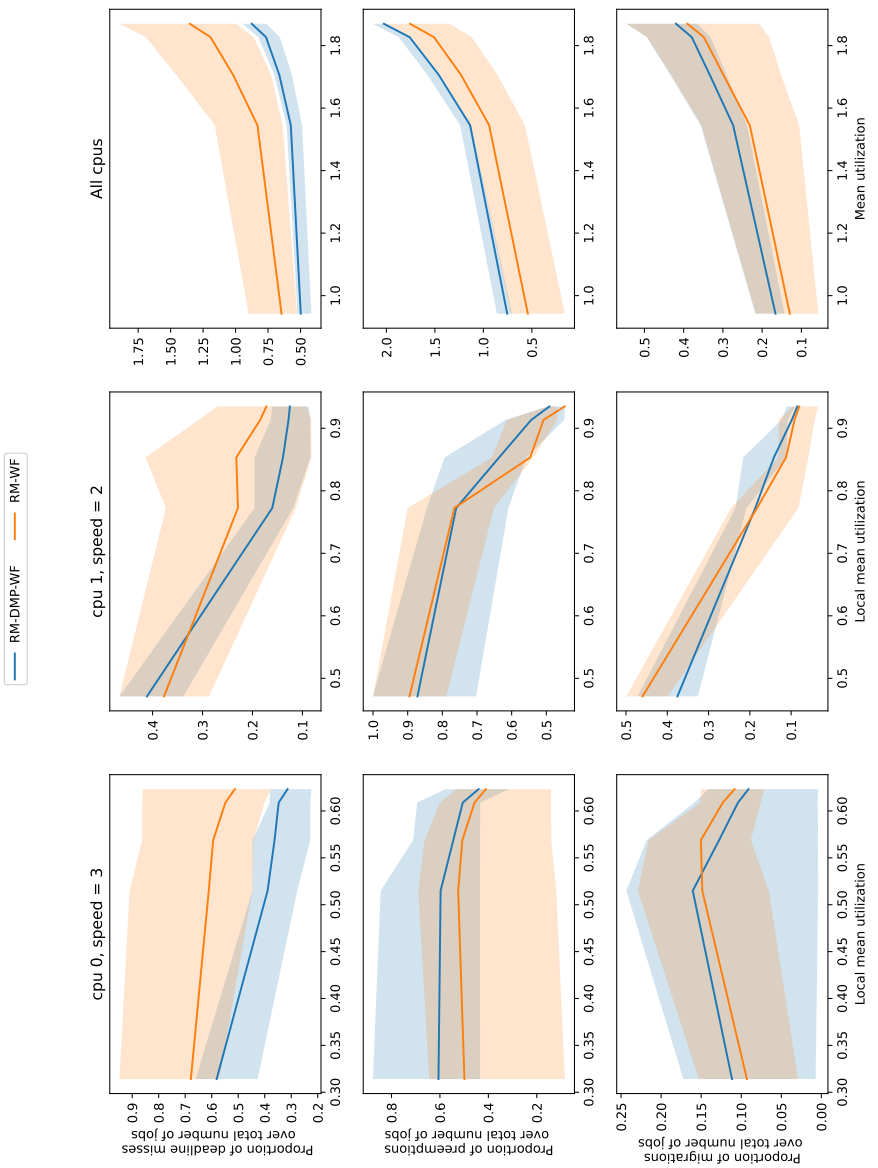


Figure 6.7: Comparison of RM-DMP-Worst-Fit and RM-Worst-Fit with the a stationary task set shown in Tables 6.2. The first row represents deadline misses, the second preemptions the third one migrations. Each column represents a processor. In solid line the average value, and the colored area represents the values between the 25%-percentile and the 75%-percentile.

6.4.4 RM-DMP-RAA vs. others

We see in Figure 6.8 and Figure 6.9 that DMP-driven algorithms are not systematically the scheduling algorithms with less deadline misses. Indeed, LLREF [Funk and Meka, 2009], DP-WRAP [Baruah and Carpenter, 2005, Levin et al., 2010], U-EDF [Baruah and Goossens, 2008] are proven optimal in some cases. For comparison we add global EDF and global RM. Those are global scheduling algorithms, hence the preemption and migration metric are not relevant for a comparison to a restricted migration scheduling. However, we see that RM-DMP has a good performance compared to global scheduling policies. Moreover, DP-WRAP (which is a Pfair-based algorithm) is optimal in the periodic case, see Figure 6.8. DP-WRAP is presented as a baseline.

The performances of DMP-driven bin-packing is quite limited. However, we provide it as a simple method to implement with not much overhead than a classical bin-packing algorithm, that slightly improves the metrics used in this thesis.

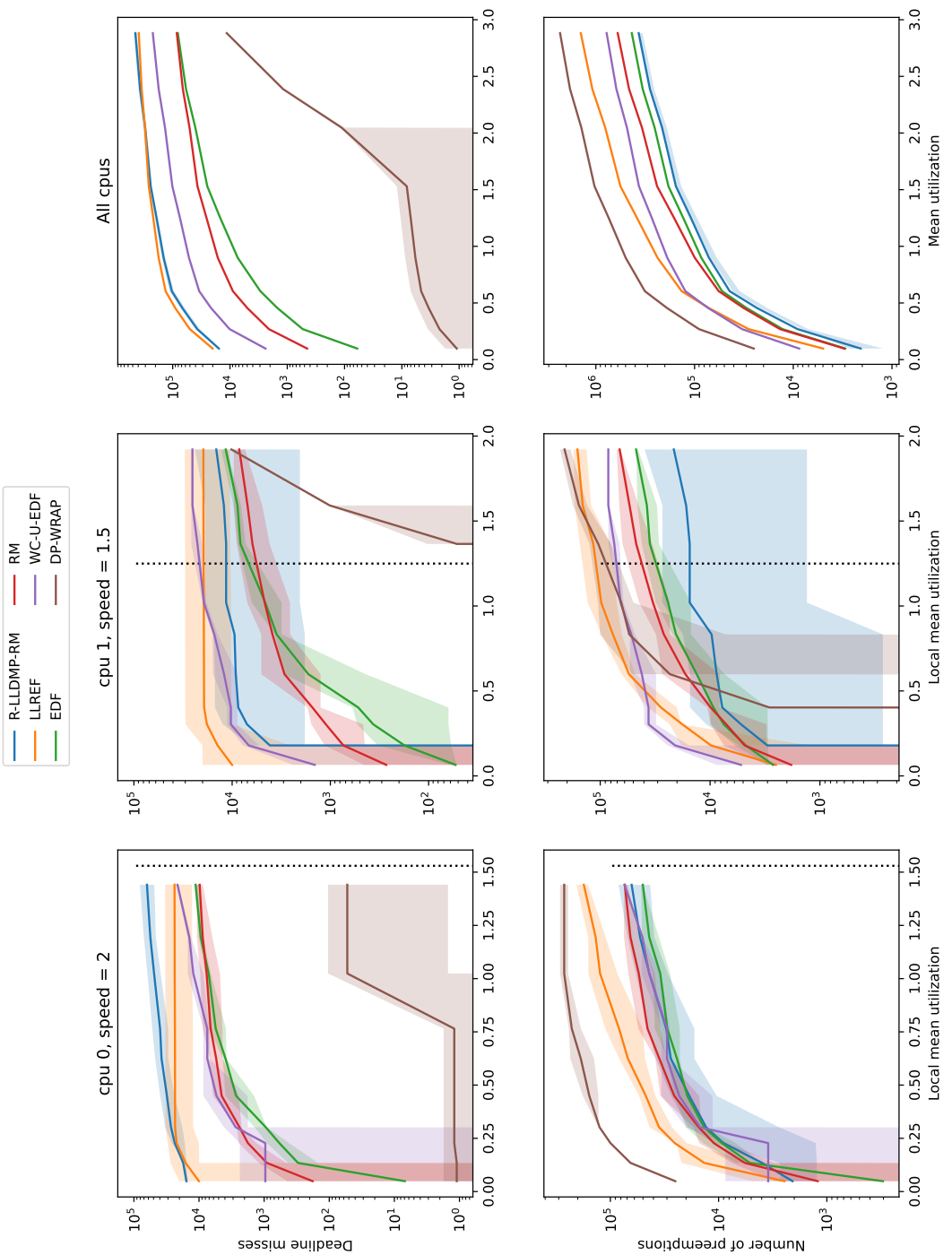


Figure 6.8: Deadline misses and preemptions of 100 schedules using RM-DMP-First-Fit with LLREF, DP-WRAP, U-EDF, R-EDF, Global RM and Global EDF in the periodic case. The first row represents the number of deadline misses, the second row the number of preemptions. Each column represents a processor and the last column the accumulation of all processors.The dotted line represents deadline misses when the maximal local utilization exceeds 1. In solid line the average value and the colored area represent the values between the 25%-percentile and the 75%-percentile.

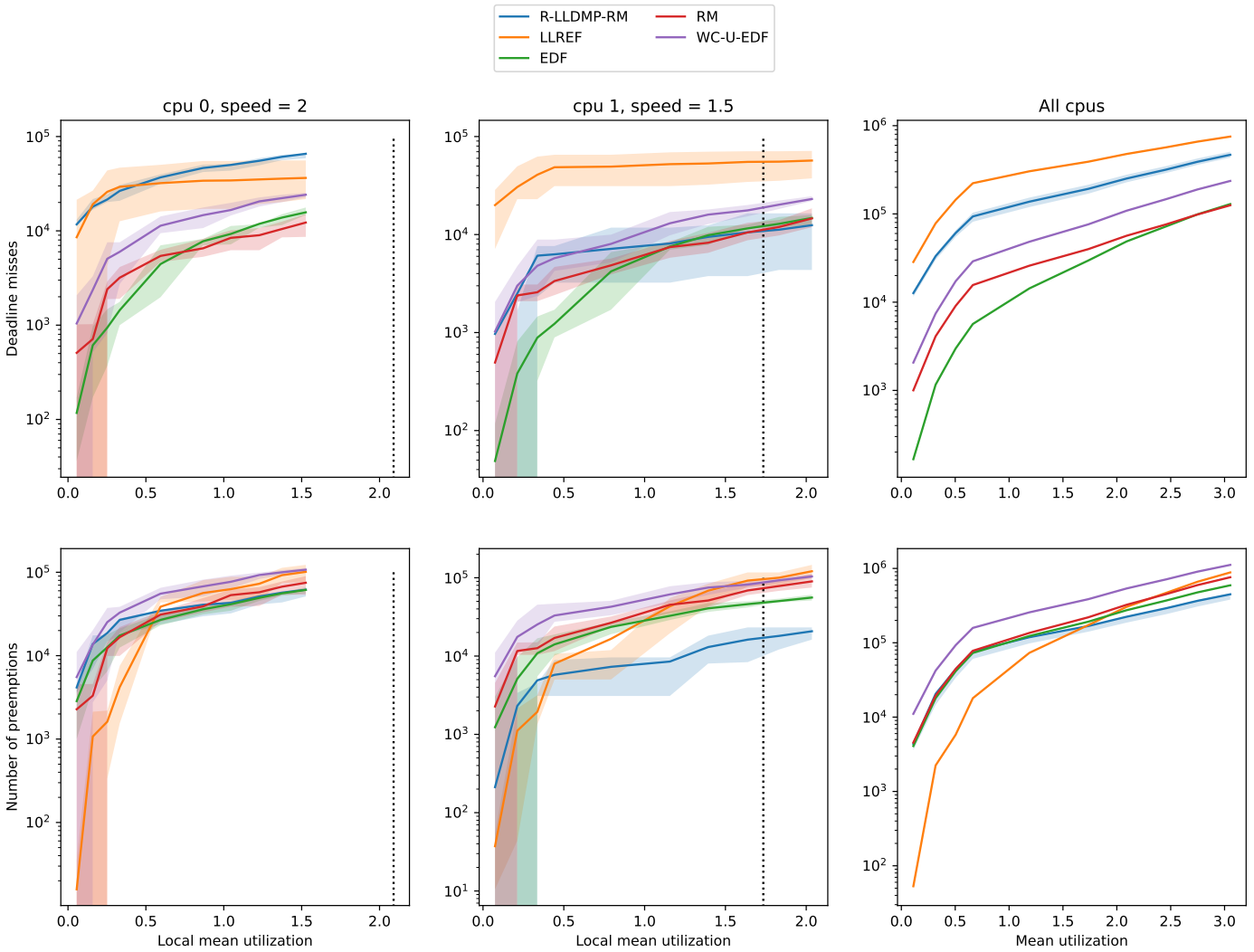


Figure 6.9: Deadline misses and preemptions of 100 schedules using RM-DMP-First-Fit with LLREF,U-EDF, R-EDF, Global RM and Global EDF in the stationary case. The first row represents the number of deadline misses, the second row the number of preemptions. Each column represents a processor and the last column the accumulation of all processors. The dotted line represents deadline misses when the maximal local utilization exceeds 1. In solid line the average value and the colored area represent the values between the 25%-percentile and the 75%-percentile.

6.5 Potential extension to unrelated heterogeneous multiprocessor systems

Time demand analysis in the form presented in Eq. (4.8) is flexible and can easily adapt to speeds changing over time. Indeed, with the more general relation

$$\hat{R}_{i,j} = \inf \left\{ t > 0 : \hat{\beta}_{i-1}(A_{i,j}) + C_{i,j} + W^{\kappa}_{i-1}(t + A_{i,j}) - W^{\kappa}_{i-1}(t) \leqslant s_{\kappa}(t) \right\} \quad (6.22)$$

where s_κ is a non-linear function providing the speed of a processor $\kappa \in \Pi$ over time. In that case, Brownian motion have many properties of reaching times that can be used to quantify such response time. It could also be modeled with Brownian motion evolving in random environments [Karandikar and Kulkarni, 1995]. One can also use s_κ as a stochastic process, and Branching Brownian motions seem to be the appropriate object to use in that case.

7 | Perspectives

Contents

In the scheduling theory, the inference of *scheduling knowledge* [Shaw et al., 1992] is a natural step into the application of the parametric method provided in this paper. The parametric estimation of new parameters, such as new virtual deadlines, may improve the scheduling decisions, see [Aytug et al., 1994] for a review in this field. Based on this thesis, we identify three possible directions to be explored in the short-mid-long term with the scheduling knowledge notion.

7.1 Short-term

A current trend in real-time systems is the application of statistical learning methods in order to find optimal schedules [Plassart, 2020, Mao et al., 2019, ul Islam and Lin, 2015, Lee et al., 1997, Nakasuka and Yoshida, 1992, Kadaba et al., 1991, Lee et al., 1997, Ahmad and Dhodhi, 1996, Gupta et al., 2010, Padmajothi et al., 2022].

As the worst-case reasoning is sustainable, methods that can reduce pessimism are still possible. We introduce the notion of *scheduling knowledge* [Shaw et al., 1992, Powner and Walburn, 1990, Yih and Thesen, 1991, Piramuthu et al., 1993, Mao et al., 2019] as a way to reduce the pessimism using an estimated knowledge

of finished jobs. In this section we use the parametric estimation of backlogs proposed in Chapter 5 to improve the scheduling decisions, see [Aytug et al., 1994] for a review in this field.

An *a priori* distribution is the distribution of the parameters of a given variable. For the response time R_i of the task τ_i running on a processor κ, *i.e.*,considering the allocation $a = (\tau_i, \kappa)$ we see in Chapter 5 that a suited approximation of the distribution of response time takes the form

$$h_i(r) = \int \psi\left(r; \frac{\theta}{1-\bar{u}_i(\kappa)}, \frac{\theta^2}{\bar{v}_i^2(\kappa)}\right)(\mu_{i-1}(\kappa)P_a)(\theta)d\theta \tag{7.1}$$

where we recognize the parameter θ to be the backlog referred to in Section 3.2, and $\psi(r; \frac{\theta}{1-u}, \frac{\theta^2}{v^2}) = \frac{d}{dt}\Psi(\theta, u, v, t)$ is the inverse Gaussian probability function of mean $\frac{\theta}{1-u}$ and shape $\frac{\theta^2}{v^2}$. The distribution $\mu_{i-1}(\kappa) * F_i(\cdot \times s(\kappa))$ is the *a priori* distribution of the response times of τ_i. An estimation of this distribution proposed in Chapter 5

$$\hat{h}_i(r; \boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \sum_{j=1}^{k_i} \psi\left(r; \frac{\theta_{i,j}}{1-\bar{u}_i(\kappa)}, \frac{\theta_{i,j}^2}{\bar{v}_i^2(\kappa)}\right)\pi_{i,j} \tag{7.2}$$

Definition 7.1. *Let μ be a probability function and the shrink operator $\mu \to \mu^+$ defined by*

$$\mu^+(x) = \mu(x)\mathbf{1}_{x>0} + \mathbf{1}_{x=0}\int_{-\infty}^{x} \mu(y)dy \tag{7.3}$$

The shrink operator, used in several studies [Díaz et al., 2002, Kim and Shin, 1996], is used to describe the transition of the backlog distribution after an interval of time Δ. For example for a variable X of distribution μ and scalar $\Delta \geqslant 0$, $(\mu * \delta_{-\Delta})^+$ is the distribution of $(X - \Delta)^+ = \max(0, X - \Delta)$.

Lemma 7.1. *After an interval of time Δ with no allocation of level k or higher, the backlog distribution μ of level k is $(\mu * \delta_{-\Delta})^+$*

Proof. This is a consequence of the work-conserving property of processors. If no higher priority jobs arrives, then it must have executed some work of level k during the interval Δ. If the workload is less than Δ, then it is at zero. □

Proposition 7.1 (Local backlog)**.** *Let $t > 0$ and consider a processor κ at the instant $t > 0$. Let $k \in \{1, \dots, n\}$ and let $w_i(\kappa)$ be the workload already executed on the active job of the task $\tau_i \in \Gamma_k^+(\kappa)$ on the processor κ. The current - i.e., at the instant $t > 0$ - local backlog of level k in the processor κ is*

$$\left(B_k^{max}(\kappa) - \sum_{\tau_i \in \Gamma_k^+(\kappa)} s(\kappa) w_i(\kappa) \right)^+ \tag{7.4}$$

with distribution function $\left(\mu_k(\kappa) * \delta_{-\sum_{\tau_i \in \Gamma_k^+(\kappa)} s(\kappa) w_i(\kappa)} \right)^+$.

Proof. With Lemma 7.1, we should conclude by induction that the current backlog of level k is

$$(\dots((\mu_k(\kappa) * \delta_{-s(\kappa)w_{i_1}})^+ * \delta_{-s(\kappa)w_{i_2}})^+ \cdots * \delta_{-s(\kappa)w_{i_p}})^+$$

However we know that $s(\kappa) \sum_{k=1}^p w_{i_k}$ is necessarily lower than the workload of their associated jobs, otherwise it would not belong to $\Gamma_i^+(\kappa)$. □

Now let us suppose we measure a response time R_i of the lastly executed task τ_i on κ and suppose the last taken action is a. Estimating its *a priori* parameter

$$\hat{\theta} = \text{argmax}_{\theta, \pi \in \{\theta_{i,j}, \pi_{i,j}\}_j} \psi \left(R_i; \frac{\theta}{1 - \bar{u}_{i-1}(\kappa)}, \frac{\theta^2}{\bar{v}_{i-1}^2(\kappa)} \right) \pi \tag{7.5}$$

provides the previous backlog of τ_i. Hence, having the estimator $\hat{\theta}$ provides an estimator of the current distribution of the backlog of level i

$$\left(\delta_{\hat{\theta}} P_a * \delta_{-\sum_{\tau_k \in \Gamma_i^+(\kappa)} s(\kappa) w_k(\kappa)} \right)^+$$

In [Plassart, 2020, Gaujal et al., 2020a], the authors use statistical learning methods in order to find optimal speeds for a low-energy-oriented scheduling for Dynamic Voltage and Frequency Scaling (DVFS). A Markov Decision Process is built in order to adapt the processor speed (which is the action of the process) between

lists of potential instances of programs to run, in which the most energy efficient are chosen by the scheduler. The proposed method is similar to that approach.

7.2 Mid-term

Exploiting the data flow between sensors and tasks to make better predictions is not new [Melani et al., 2013]. However, it can be combined with the potential extension explained in Section 7.1. We discuss here without providing proofs how this information can be inferred into the estimation of execution times distributions. We also introduce the formalization making this estimation well defined in the context of stationary real-time systems.

Let $\tau_k \in \Gamma$. We look at variables $C_k(\cdot, i), i \in I$. Let the projection operator ρ_Q be the projection of Ω_0 to Q, such that for $\omega_0 = (q_0, i_0) \in \Omega_0$, we have $\rho_Q(\omega_0) = q_0$. We can write

$$C_k(\omega_0) = (C_k(\cdot, i_0) \circ \rho_Q)(\omega_0)$$

More generally, let us define $\mathbf{P}^*$ the conditional probability over the inputs space I, *i.e.*,

$$\mathbf{P}^* = \mathbf{P}(\cdot \mid I)$$

where execution times $C_k(\cdot, i_0)$ are well defined. This new probability $\mathbf{P}^*$ is such that $(Q, \mathbf{P}^*)$ is a probability space, depending only on hardware uncertainties. Inferring the data of sensors and other sources could be a new source of scheduling knowledge.

7.3 Long term

The multiprocessor scheduling problem of stationary real-time systems stays an open problem, and deterministic methods appear to fail in providing a simple and efficient

solution. More generally, limit theorems in stochastic analysis offer a large panel of possibilities. Conditional probabilities are a powerful object that has not been explored yet in the real-time systems analysis. In the previous chapter, we give hints on how those conditional probabilities could be used in order to potentially improve DMP-driven bin packing. The relation between the mixed-criticality model and the probabilistic approach has been discussed often in this thesis, and is more explicit as the probabilistic approach provides answers that deterministic approaches cannot provide. It also allows to define and generalize deterministic results. Finally, we open the possibility to use machine learning with a parametric estimation method, as well as the possibility to define a new type of real-time scheduling yet to be built.

Bibliography

[Abdeddaïm and Maxim, 2017] Abdeddaïm, Y. and Maxim, D. (2017). Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, pages 596–601. IEEE.

[Abella et al., 2014] Abella, J., Hardy, D., Puaut, I., Quinones, E., and Cazorla, F. J. (2014). On the comparison of deterministic and probabilistic WCET estimation techniques. In *2014 26th Euromicro Conference on Real-Time Systems*, pages 266–275. IEEE.

[Abeni et al., 2017] Abeni, L., Fontanelli, D., Palopoli, L., and Frías, B. V. (2017). A markovian model for the computation time of real-time applications. In *2017 IEEE international instrumentation and measurement technology conference (I2MTC)*, pages 1–6. IEEE.

[Abundo, 2016] Abundo, M. (2016). On the excursions of drifted brownian motion and the successive passage times of brownian motion. *Physica A: Statistical Mechanics and its Applications*, 457:176–182.

[Ahmad and Dhodhi, 1996] Ahmad, I. and Dhodhi, M. K. (1996). Multiprocessor scheduling in a genetic paradigm. *Parallel Computing*, 22(3):395–406.

[Aitken, 1926] Aitken, A. (1926). Iii.—a series formula for the roots of algebraic and transcendental equations. *Royal Society of Edinburgh*, 45(1):14–22.

[Altmeyer et al., 2015a] Altmeyer, S., Cucu-Grosjean, L., and Davis, R. I. (2015a). Static probabilistic timing analysis for real-time systems using random replacement caches. *Real-Time Systems*, 51(1):77–123.

[Altmeyer et al., 2015b] Altmeyer, S., Lisper, B., Maiza, C., Reineke, J., and Rochange, C. (2015b). WCET and Mixed-Criticality: What does Confidence in WCET Estimations Depend Uponl. In Cazorla, F. J., editor, *15th International Workshop on Worst-Case Execution Time Analysis (WCET 2015)*, volume 47 of *OpenAccess Series in Informatics (OASIcs)*, pages 65–74, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Anderson et al., 2008] Anderson, J. H., Bud, V., and Devi, U. C. (2008). An edf-based restricted-migration scheduling algorithm for multiprocessor soft real-time systems. *Real-Time Systems*, 38(2):85–131.

[Anderson et al., 2016] Anderson, J. H., Erickson, J. P., Devi, U. C., and Casses, B. N. (2016). Optimal semi-partitioned scheduling in soft real-time systems. *Journal of Signal Processing Systems*, 84(1):3–23.

[Andersson et al., 2003] Andersson, B., Abdelzaher, T., and Jonsson, J. (2003). Partitioned aperiodic scheduling on multiprocessors. In *International Parallel and Distributed Processing Symposium*, pages 16–pp. IEEE.

[Andersson et al., 2001] Andersson, B., Baruah, S., and Jonsson, J. (2001). Static-priority scheduling on multiprocessors. In *22nd IEEE Real-Time Systems Symposium (RTSS 2001)(Cat. No. 01PR1420)*, pages 193–202. IEEE.

[Andersson and Jonsson, 2000] Andersson, B. and Jonsson, J. (2000). Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition. In *Seventh International Conference on Real-Time Computing Systems and Applications*, pages 337–346. IEEE.

[Andersson and Jonsson, 2002] Andersson, B. and Jonsson, J. (2002). Preemptive multiprocessor scheduling anomalies. *Proceedings 16th International Parallel and Distributed Processing Symposium*, pages 12 – 19.

[Andersson and Tovar, 2006] Andersson, B. and Tovar, E. (2006). Multiprocessor scheduling with few preemptions. In *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*, pages 322–334. IEEE.

[Axer et al., 2014] Axer, P., Ernst, R., Falk, H., Girault, A., Grund, D., Guan, N., Jonsson, B., Marwedel, P., Reineke, J., Rochange, C., Sebastian, M., Hanxleden, R. V., Wilhelm, R., and Yi, W. (2014). Building timing predictable embedded systems. *ACM Trans. Embed. Comput. Syst.*, 13(4).

[Aytug et al., 1994] Aytug, H., Bhattacharyya, S., Koehler, G. J., and Snowdon, J. L. (1994). A review of machine learning in scheduling. *IEEE Transactions on Engineering Management*, 41(2):165–171.

[Baccelli and Brémaud, 2013] Baccelli, F. and Brémaud, P. (2013). *Elements of queueing theory: Palm Martingale calculus and stochastic recurrences*, volume 26. Springer Science & Business Media.

[Baruah and Baker, 2008] Baruah, S. and Baker, T. (2008). Schedulability analysis of global edf. *Real-Time Systems*, 38(3):223–235.

[Baruah et al., 2012] Baruah, S., Bonifaci, V., DAngelo, G., Li, H., Marchetti-Spaccamela, A., van der Ster, S., and Stougie, L. (2012). The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In *2012 24th Euromicro Conference on Real-Time Systems*, pages 145–154.

[Baruah and Burns, 2006] Baruah, S. and Burns, A. (2006). Sustainable scheduling analysis. In *2006 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, pages 159–168. IEEE.

[Baruah and Fisher, 2005] Baruah, S. and Fisher, N. (2005). The partitioned multiprocessor scheduling of sporadic task systems. In *26th IEEE International Real-Time Systems Symposium (RTSS'05)*, pages 9–pp. IEEE.

[Baruah and Goossens, 2008] Baruah, S. and Goossens, J. (2008). The edf scheduling of sporadic task systems on uniform multiprocessors. In *2008 Real-Time Systems Symposium*, pages 367–374. IEEE.

[Baruah et al., 2011] Baruah, S. K., Burns, A., and Davis, R. I. (2011). Response-time analysis for mixed criticality systems. In *2011 IEEE 32nd Real-Time Systems Symposium*, pages 34–43. IEEE.

[Baruah and Carpenter, 2005] Baruah, S. K. and Carpenter, J. (2005). Multiprocessor fixed-priority scheduling with restricted interprocessor migrations. *Journal of Embedded Computing*, 1(2):169–178.

[Baruah et al., 1993] Baruah, S. K., Cohen, N. K., Plaxton, C. G., and Varvel, D. A. (1993). Proportionate progress: A notion of fairness in resource allocation. In *twenty-fifth annual ACM symposium on Theory of computing*, pages 345–354.

[Baruah and Goossens, 2003] Baruah, S. K. and Goossens, J. (2003). Rate-monotonic scheduling on uniform multiprocessors. *IEEE transactions on computers*, 52(7):966–970.

[Basrak, 2011] Basrak, B. (2011). Fisher-tippett theorem. In *International Encyclopedia of Statistical Science*, pages 525–526. Springer.

[Bastoni et al., 2011] Bastoni, A., Brandenburg, B. B., and Anderson, J. H. (2011). Is semi-partitioned scheduling practical? In *2011 23rd Euromicro Conference on Real-Time Systems*, pages 125–135. IEEE.

[Bellman, 1957] Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

[Bertogna and Cirinei, 2007] Bertogna, M. and Cirinei, M. (2007). Response-time analysis for globally scheduled symmetric multiprocessor platforms. In *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 149–160. IEEE.

[Bertogna et al., 2008] Bertogna, M., Cirinei, M., and Lipari, G. (2008). Schedulability analysis of global scheduling algorithms on multiprocessor platforms. *IEEE Transactions on parallel and distributed systems*, 20(4):553–566.

[Bini and Buttazzo, 2005] Bini, E. and Buttazzo, G. C. (2005). Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1):129–154.

[Bini et al., 2003] Bini, E., Buttazzo, G. C., and Buttazzo, G. M. (2003). Rate monotonic analysis: the hyperbolic bound. *IEEE Transactions on Computers*, 52(7):933–942.

[Blitzstein and Hwang, 2019] Blitzstein, J. K. and Hwang, J. (2019). *Introduction to probability*. Crc Press.

[Bouveyron et al., 2019] Bouveyron, C., Celeux, G., Murphy, T. B., and Raftery, A. E. (2019). *Model-Based Clustering and Classification for Data Science: With Applications in R*, volume 50. Cambridge University Press.

[Bozhko et al., 2021] Bozhko, S., von der Brüggen, G., and Brandenburg, B. B. (2021). Monte carlo response-time analysis. In *Real-Time Systems Symposium*.

[Brandenburg and Gül, 2016] Brandenburg, B. B. and Gül, M. (2016). Global scheduling not required: Simple, near-optimal multiprocessor real-time scheduling with semi-partitioned reservations. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 99–110. IEEE.

[Burns, 2014] Burns, A. (2014). System mode changes-general and criticality-based. In *Proc. of 2nd Workshop on Mixed Criticality Systems (WMC)*, pages 3–8.

[Burns and Baruah, 2008] Burns, A. and Baruah, S. (2008). Sustainability in real-time scheduling. *Journal of Computing Science and Engineering*, 2(1):74–97.

[Burns and Davis, 2017] Burns, A. and Davis, R. I. (2017). A survey of research into mixed criticality systems. *ACM Computing Surveys (CSUR)*, 50(6):1–37.

[Carpenter et al., 2004] Carpenter, J., Funk, S. H., Holman, P., Srinivasan, A., Anderson, J. H., and Baruah, S. K. (2004). A categorization of real-time multiprocessor scheduling problems and algorithms.

[Cazorla et al., 2013] Cazorla, F. J., Vardanega, T., Quiñones, E., and Abella, J. (2013). Upper-bounding program execution time with extreme value theory. In *13th International Workshop on Worst-Case Execution Time Analysis*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Chen and Yao, 2001] Chen, H. and Yao, D. D. (2001). *Fundamentals of queueing networks: Performance, asymptotics, and optimization*, volume 4. Springer.

[Chen and Chen, 2017] Chen, K.-H. and Chen, J.-J. (2017). Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–8. IEEE.

[Chen et al., 2022] Chen, K.-H., Günzel, M., von der Brüggen, G., and Chen, J.-J. (2022). Critical instant for probabilistic timing guarantees: Refuted and revisited. In *IEEE 43rd Real-Time Systems Symposium. IEEE*.

[Chen et al., 2018] Chen, K.-H., Von Der Brüggen, G., and Chen, J.-J. (2018). Analysis of deadline miss rates for uniprocessor fixed-priority scheduling. In *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 168–178. IEEE.

[Cheramy, 2014] Cheramy, M. (2014). Simso. https://maximecheramy.github.io/simso-web.

[Chéramy et al., 2014] Chéramy, M., Hladik, P.-E., and Déplanche, A.-M. (2014). Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms. In *The 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, WATERS.

[Cucu and Goossens, 2007] Cucu, L. and Goossens, J. (2007). Feasibility intervals for multiprocessor fixed-priority scheduling of arbitrary deadline periodic systems. In *2007 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE.

[Cucu-Grosjean et al., 2012] Cucu-Grosjean, L., Santinelli, L., Houston, M., Lo, C., Vardanega, T., Kosmidis, L., Abella, J., Mezzetti, E., Quiñones, E., and Cazorla, F. J. (2012). Measurement-based probabilistic timing analysis for multi-path programs. In *2012 24th euromicro conference on real-time systems*, pages 91–101. IEEE.

[Dasgupta and Raftery, 1998] Dasgupta, A. and Raftery, A. E. (1998). Detecting features in spatial point processes with clutter via model-based clustering. *Journal of the American statistical Association*, 93(441):294–302.

[Davis and Burns, 2011] Davis, R. I. and Burns, A. (2011). A survey of hard real-time scheduling for multiprocessor systems. *ACM computing surveys (CSUR)*, 43(4):1–44.

[Davis and Cucu-Grosjean, 2019] Davis, R. I. and Cucu-Grosjean, L. (2019). A survey of probabilistic timing analysis techniques for real-time systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–60.

[Davis et al., 2016] Davis, R. I., Cucu-Grosjean, L., Bertogna, M., and Burns, A. (2016). A review of priority assignment in real-time systems. *Journal of systems architecture*, 65:64–82.

[Davis et al., 2013] Davis, R. I., Whitham, J., and Maxim, D. (2013). Static probabilistic timing analysis for multicore processors with shared cache. In *Real-Time Scheduling Open Problems Seminar (RTSOPS)*, pages 3–5.

[de Barros Vasconcelos and Lima, 2022] de Barros Vasconcelos, J. and Lima, G. (2022). Possible risks with EVT-based timing analysis: an experimental study on a multi-core platform. In *2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 1–8. IEEE.

[Díaz et al., 2002] Díaz, J. L., García, D. F., Kim, K., Lee, C.-G., Bello, L. L., López, J. M., Min, S. L., and Mirabella, O. (2002). Stochastic analysis of periodic real-time systems. In *23rd IEEE Real-Time Systems Symposium*, pages 289–300. IEEE.

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

[Dixit et al., 2021] Dixit, A., Kumar Chidambaram, R., and Allam, Z. (2021). Safety and risk analysis of autonomous vehicles using computer vision and neural networks. *Vehicles*, 3(3):595–617.

[Doytchinov et al., 2001] Doytchinov, B., Lehoczky, J., and Shreve, S. (2001). Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *The Annals of Applied Probability*, 11(2):332–378.

[Draskovic et al., 2021] Draskovic, S., Ahmed, R., Huang, P., and Thiele, L. (2021). Schedulability of probabilistic mixed-criticality systems. *Real-Time Systems*, 57(4):397–442.

[Edgar and Burns, 2001] Edgar, S. and Burns, A. (2001). Statistical analysis of WCET for scheduling. In *22nd IEEE Real-Time Systems Symposium (RTSS 2001)(Cat. No. 01PR1420)*, pages 215–224. IEEE.

[Esper et al., 2015] Esper, A., Nelissen, G., Nélis, V., and Tovar, E. (2015). How realistic is the mixed-criticality real-time system model? In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, pages 139–148.

[Fan and Quan, 2012] Fan, M. and Quan, G. (2012). Harmonic semi-partitioned scheduling for fixed-priority real-time tasks on multi-core platform. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 503–508. IEEE.

[Fernandez et al., 2014] Fernandez, G., Abella, J., Quiñones, E., Rochange, C., Vardanega, T., and Cazorla, F. J. (2014). Contention in multicore hardware shared resources: Understanding of the state of the art. In *14th International Workshop on Worst-Case Execution Time Analysis*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Folks and Chhikara, 1978] Folks, J. L. and Chhikara, R. S. (1978). The inverse gaussian distribution and its statistical application—a review. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(3):263–275.

[Fraley and Raftery, 2002] Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631.

[Friebe et al., 2020] Friebe, A., Papadopoulos, A. V., and Nolte, T. (2020). Identification and validation of markov models with continuous emission distributions for execution times. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–10. IEEE.

[Funk and Nanadur, 2009] Funk, S. and Nanadur, V. (2009). Lre-tl: An optimal multiprocessor scheduling algorithm for sporadic task sets. In *17th International Conference on Real-Time and Network Systems*, pages 159–168.

[Funk and Meka, 2009] Funk, S. H. and Meka, A. (2009). U-llref: An optimal scheduling algorithm for uniform multiprocessors. In *The 9th Workshop on Models and Algorithms for Planning and Scheduling Problems*, page 262.

[Gaujal et al., 2020a] Gaujal, B., Girault, A., and Plassart, S. (2020a). Discrete and continuous optimal control for energy minimization in real-time systems. In *2020 6th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–8. IEEE.

[Gaujal et al., 2020b] Gaujal, B., Girault, A., and Plassart, S. (2020b). Dynamic speed scaling minimizing expected energy consumption for real-time tasks. *Journal of Scheduling*, 23(5):555–574.

[Gettings et al., 2015] Gettings, O., Quinton, S., and Davis, R. I. (2015). Mixed criticality systems with weakly-hard constraints. In *23rd International Conference on Real Time and Networks Systems*, RTNS '15, pages 237–246, New York, NY, USA. Association for Computing Machinery.

[Gil et al., 2017] Gil, S. J., Bate, I., Lima, G., Santinelli, L., Gogonel, A., and Cucu-Grosjean, L. (2017). Open challenges for probabilistic measurement-based worst-case execution time. *IEEE Embedded Systems Letters*, 9(3):69–72.

[Goossens et al., 2012] Goossens, J., Richard, P., Lindström, M., Lupu, I. I., and Ridouard, F. (2012). Job partitioning strategies for multiprocessor scheduling of real-time periodic tasks with restricted migrations. In *20th International Conference on Real-Time and Network Systems*, pages 141–150.

[Guan and Yi, 2012] Guan, N. and Yi, W. (2012). Fixed-priority multiprocessor scheduling: Critical instant, response time and utilization bound. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 2470–2473. IEEE.

[Gumzej and Halang, 2010] Gumzej, R. and Halang, W. A. (2010). Certification of real-time systems. *Real-time Systems' Quality of Service: Introducing Quality of Service Considerations in the Life-cycle of Real-time Systems*, pages 107–118.

[Guo et al., 2015] Guo, Z., Santinelli, L., and Yang, K. (2015). Edf schedulability analysis on mixed-criticality systems with permitted failure probability. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 187–196. IEEE.

[Guo et al., 2017] Guo, Z., Sruti, S., Ward, B. C., and Baruah, S. (2017). Sustainability in mixed-criticality scheduling. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 24–33.

[Gupta et al., 2010] Gupta, S., Kumar, V., and Agarwal, G. (2010). Task scheduling in multiprocessor system using genetic algorithm. In *2010 Second international conference on machine learning and computing*, pages 267–271. IEEE.

[Ha and Liu, 1994] Ha, R. and Liu, J. W. (1994). Validating timing constraints in multiprocessor and distributed real-time systems. In *14th international conference on distributed computing systems*, pages 162–171. IEEE.

[Hansen et al., 2009] Hansen, J., Hissam, S., and Moreno, G. A. (2009). Statistical-based wcet estimation and validation. In *9th international workshop on worst-case execution time analysis (WCET'09)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[Harel and Pnueli, 1984] Harel, D. and Pnueli, A. (1984). On the development of reactive systems. In Apt, K. R., editor, *Logics and Models of Concurrent Systems - Conference proceedings, Colle-sur-Loup (near Nice), France, 8-19 October 1984*, volume 13 of *NATO ASI Series*, pages 477–498. Springer.

[Hobbs et al., 2019] Hobbs, C., Tong, Z., and Anderson, J. H. (2019). Optimal soft real-time semi-partitioned scheduling made simple (and dynamic). In *27th International Conference on Real-Time Networks and Systems*, pages 112–122.

[Hong and Leung, 1992] Hong, K. and Leung, J. (1992). On-line scheduling of real-time tasks. *IEEE Transactions on Computers*, 41(10):1326–1331.

[Huang et al., 2015] Huang, J., Carmeli, B., and Mandelbaum, A. (2015). Control of patient flow in emergency departments, or multiclass queues with deadlines and feedback. *Operations Research*, 63(4):892–908.

[Itô, 1972] Itô, K. (1972). Poisson point processes attached to markov processes. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability (Univ. California, Berkeley, Calif., 1970/1971)*, volume 3, pages 225–239.

[Jalle et al., 2014] Jalle, J., Kosmidis, L., Abella, J., Quiñones, E., and Cazorla, F. J. (2014). Bus designs for time-probabilistic multicore processors. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE.

[Janssen and Manca, 2006] Janssen, J. and Manca, R. (2006). Renewal theory. *Applied Semi-Markov Processes*, pages 45–104.

[Jeanblanc et al., 2009] Jeanblanc, M., Yor, M., and Chesney, M. (2009). *Mathematical methods for financial markets*. Springer Science & Business Media.

[Joseph and Pandya, 1986] Joseph, M. and Pandya, P. (1986). Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395.

[Kadaba et al., 1991] Kadaba, N., Nygard, K. E., and Juell, P. L. (1991). Integration of adaptive machine learning and knowledge-based systems for routing and scheduling applications. *Expert Systems with Applications*, 2(1):15–27.

[Karandikar and Kulkarni, 1995] Karandikar, R. L. and Kulkarni, V. G. (1995). Second-order fluid flow models: reflected brownian motion in a random environment. *Operations Research*, 43(1):77–88.

[Kato and Yamasaki, 2009] Kato, S. and Yamasaki, N. (2009). Semi-partitioned fixed-priority scheduling on multiprocessors. In *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 23–32. IEEE.

[Kendall, 1953] Kendall, D. G. (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354.

[Khazen et al., 2022] Khazen, M. W. E., Zagalo, K., Clarke, H., Mezouak, M., Abdeddaïm, Y., Bar-Hen, A., Amor, S. B., Bennour, R., Gogonel, A., Kougblenou, K., Sorel, Y., and Cucu-Grosjean, L. (2022). Work in progress: Kdbench – towards open source benchmarks for measurement-based multicore WCET estimators. In *IEEE RTAS*.

[Kim and Shin, 1996] Kim, J. and Shin, K. G. (1996). Execution time analysis of communicating tasks in distributed systems. *IEEE Transactions on Computers*, 45(5):572–579.

[Kim et al., 2005] Kim, K., Diaz, J. L., Bello, L. 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 Transactions on Computers*, 54(11):1460–1466.

[Koenig and Howard, 2004] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE.

[Lawrance, 1973] Lawrance, A. J. (1973). Dependency of intervals between events in superposition processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 35(2):306–315.

[Le Gall, 2016] Le Gall, J.-F. (2016). *Brownian motion, martingales, and stochastic calculus*. Springer.

[Lee et al., 1997] Lee, C.-Y., Piramuthu, S., and Tsai, Y.-K. (1997). Job shop scheduling with a genetic algorithm and machine learning. *International Journal of production research*, 35(4):1171–1191.

[Lehoczky, 1990] Lehoczky, J. P. (1990). Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *11th Real-Time Systems Symposium*, pages 201–209. IEEE.

[Lehoczky, 1996] Lehoczky, J. P. (1996). Real-time queueing theory. In *17th IEEE Real-Time Systems Symposium*, pages 186–195. IEEE.

[Lehoczky, 1997a] Lehoczky, J. P. (1997a). Real-time queueing network theory. In *Real-Time Systems Symposium*, pages 58–67. IEEE.

[Lehoczky, 1997b] Lehoczky, J. P. (1997b). Using real-time queueing theory to control lateness in real-time systems. *ACM SIGMETRICS Performance Evaluation Review*, 25(1):158–168.

[Levin et al., 2010] Levin, G., Funk, S., Sadowski, C., Pye, I., and Brandt, S. (2010). Dp-fair: A simple model for understanding optimal multiprocessor scheduling. In *2010 22nd Euromicro Conference on Real-Time Systems*, pages 3–13. IEEE.

[Lima and Bate, 2017] Lima, G. and Bate, I. (2017). Valid application of EVT in timing analysis by randomising execution time measurements. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 187–198. IEEE.

[Lima et al., 2016] Lima, G., Dias, D., and Barros, E. (2016). Extreme value theory for estimating task execution time bounds: A careful look. In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 200–211. IEEE.

[Lindley, 1952] Lindley, D. V. (1952). The theory of queues with a single server. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 48, pages 277–289. Cambridge University Press.

[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.

[Liu et al., 2013] Liu, M., Behnam, M., and Nolte, T. (2013). An EVT-based worst-case response time analysis of complex real-time systems. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 249–258. IEEE.

[López et al., 2008] López, J. M., Díaz, J. L., Entrialgo, J., and García, D. (2008). Stochastic analysis of real-time systems under preemptive priority-driven scheduling. *Real-Time Systems*, 40(2):180.

[Lopez et al., 2004] Lopez, J. M., Díaz, J. L., and Garcia, D. F. (2004). Minimum and maximum utilization bounds for multiprocessor rate monotonic scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 15(7):642–653.

[Loynes, 1962] Loynes, R. M. (1962). The stability of a queue with non-independent inter-arrival and service times. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 58, pages 497–520. Cambridge University Press.

[Lu et al., 2011] Lu, Y., Nolte, T., Bate, I., and Cucu-Grosjean, L. (2011). A new way about using statistical analysis of worst-case execution times. *ACM SIGBED Review*, 8(3):11–14.

[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 *2012 IEEE 33rd Real-Time Systems Symposium*, pages 351–362. IEEE.

[Maiza et al., 2019] Maiza, C., Rihani, H., Rivas, J. M., Goossens, J., Altmeyer, S., and Davis, R. I. (2019). A survey of timing verification techniques for multi-core real-time systems. *ACM Comput. Surv.*, 52(3).

[Manolache et al., 2001] Manolache, S., Eles, P., and Peng, Z. (2001). Memory and time-efficient schedulability analysis of task sets with stochastic execution time. In *13th EUROMICRO conference on Real-time Systems*, pages 19–26. IEEE.

[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 *2002 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '02, pages 699–706, New York, NY, USA. ACM.

[Mao et al., 2019] Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., and Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In *ACM special interest group on data communication*, pages 270–288. ACM special interest group on data communication.

[Markovic et al., 2022] Markovic, F., Nolte, T., and Papadopoulos, A. V. (2022). Analytical approximations in probabilistic analysis of real-time systems. In *43rd IEEE Real-Time Systems Symposium (RTSS)*.

[Marković et al., 2021] Marković, F., Papadopoulos, A. V., and Nolte, T. (2021). On the Convolution Efficiency for Probabilistic Analysis of Real-Time Systems. In *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Maxim et al., 2017a] Maxim, C., Gogonel, A., Asavoae, I., Asavoae, M., and Cucu-Grosjean, L. (2017a). Reproducibility and representativity: mandatory properties for the compositionality of measurement-based WCET estimation approaches. *ACM SIGBED Review*, 14(3):24–31.

[Maxim and Cucu-Grosjean, 2013] Maxim, D. and Cucu-Grosjean, L. (2013). Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 224–235. IEEE.

[Maxim et al., 2017b] Maxim, D., Davis, R. I., Cucu-Grosjean, L., and Easwaran, A. (2017b). Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling. In *25th International Conference on Real-Time Networks and Systems*, pages 237–246.

[Maxim et al., 2012] Maxim, D., Houston, M., Santinelli, L., Bernat, G., Davis, R. I., and Cucu-Grosjean, L. (2012). Re-sampling for statistical timing analysis of real-time systems. In *20th International Conference on Real-Time and Network Systems*, RTNS '12, pages 111–120, New York, NY, USA. ACM.

[McLachlan et al., 2019] McLachlan, G. J., Lee, S. X., and Rathnayake, S. I. (2019). Finite mixture models. *Annual review of statistics and its application*, 6:355–378.

[Meier et al., 2015] Meier, L., Honegger, D., and Pollefeys, M. (2015). Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240.

[Melani et al., 2013] Melani, A., Noulard, E., and Santinelli, L. (2013). Learning from probabilities: Dependences within real-time systems. In *2013 IEEE 18th*

Conference on Emerging Technologies & Factory Automation (ETFA), pages 1–8. IEEE.

[Metzger et al., 2019] Metzger, F., Hoßfeld, T., Bauer, A., Kounev, S., and Heegaard, P. E. (2019). Modeling of aggregated iot traffic and its application to an iot cloud. *IEEE*, 107(4):679–694.

[Milutinovic et al., 2015] Milutinovic, S., Abella, J., Hardy, D., Quinones, E., Puaut, I., and Cazorla, F. J. (2015). Speeding up static probabilistic timing analysis. In *International Conference on Architecture of Computing Systems*, pages 236–247. Springer.

[Molini et al., 2011] Molini, A., Talkner, P., Katul, G. G., and Porporato, A. (2011). First passage time statistics of brownian motion with purely time dependent drift and diffusion. *Physica A: Statistical Mechanics and its Applications*, 390(11):1841–1852.

[Murgolo, 1988] Murgolo, F. D. (1988). Anomalous behavior in bin packing algorithms. *Discrete Applied Mathematics*, 21(3):229–243.

[Nakasuka and Yoshida, 1992] Nakasuka, S. and Yoshida, T. (1992). Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool. *The International Journal of Production Research*, 30(2):411–431.

[Ntaryamira, 2021] Ntaryamira, E. (2021). *A generalized asynchronous method preserving the data quality of real-time embedded systems : Case of the PX4-RT autopilot.* Theses, Sorbonne Université.

[Pack, 1977] Pack, C. D. (1977). The output of a d/m/1 queue. *SIAM Journal on Applied Mathematics*, 32(3):571–587.

[Padmajothi et al., 2022] Padmajothi, V., Iqbal, J. M., and Ponnusamy, V. (2022). Load-aware intelligent multiprocessor scheduler for time-critical cyber-physical system applications. *Computers & Electrical Engineering*, 97:107613.

[Palmer et al., 2011] Palmer, E. M., Horowitz, T. S., Torralba, A., and Wolfe, J. M. (2011). What are the shapes of response time distributions in visual search? *Journal of experimental psychology: human perception and performance*, 37(1):58.

[Palopoli et al., 2015] Palopoli, L., Fontanelli, D., Abeni, L., and Frias, B. V. (2015). An analytical solution for probabilistic guarantees of reservation based soft real-time systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(3):640–653.

[Palopoli et al., 2012] Palopoli, L., Fontanelli, D., Manica, N., and Abeni, L. (2012). An analytical bound for probabilistic deadlines. In *2012 24th Euromicro Conference on Real-Time Systems*, pages 179–188. IEEE.

[Phavorin and Richard, 2015] Phavorin, G. and Richard, P. (2015). Cacherelated preemption delays and real-time scheduling: A survey for uniprocessor systems. *Laboratory Comput. Sci. Autom. Syst.*

[Piramuthu et al., 1993] Piramuthu, S., Raman, N., Shaw, M. J., and Chan Park, S. (1993). Integration of simulation modeling and inductive learning in an adaptive decision support system. *Decision Support Systems*, 9(1):127–142. Model Management Systems.

[Plassart, 2020] Plassart, S. (2020). *Online optimization in dynamic real-time systems*. PhD thesis, Université Grenoble Alpes [2020-....].

[Powner and Walburn, 1990] Powner, E. and Walburn, D. (1990). A knowledge based scheduler. In *1991 First International Conference on Expert Planning Systems*, pages 82–87. IET.

[Punzo, 2019] Punzo, A. (2019). A new look at the inverse gaussian distribution with applications to insurance and economic data. *Journal of Applied Statistics*, 46(7):1260–1287.

[Quinton et al., 2012] Quinton, S., Ernst, R., Bertrand, D., and Yomsi, P. M. (2012). Challenges and new trends in probabilistic timing analysis. In *2012*

Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 810–815. IEEE.

[Raftery, 1995] Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological methodology*, pages 111–163.

[Real and Crespo, 2004] Real, J. and Crespo, A. (2004). Mode change protocols for real-time systems: A survey and a new proposal. *Real-time systems*, 26(2):161–197.

[Reilly, 2020] Reilly, A. (2020). Multi-core processors are the key to unlocking aviation's future.

[Santinelli et al., 2014] Santinelli, L., Morio, J., Dufour, G., and Jacquemart, D. (2014). On the sustainability of the extreme value theory for WCET estimation. In *14th International Workshop on Worst-Case Execution Time Analysis*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Schwarz, 1978] Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, pages 461–464.

[Schwarz, 2001] Schwarz, W. (2001). The ex-wald distribution as a descriptive model of response times. *Behavior Research Methods, Instruments, & Computers*, 33(4):457–469.

[Seshadri, 2012] Seshadri, V. (2012). *The inverse Gaussian distribution: statistical theory and applications*, volume 137. Springer Science & Business Media.

[Sethuraman and Squillante, 1999] Sethuraman, J. and Squillante, M. S. (1999). Optimal stochastic scheduling in multiclass parallel queues. In *1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 93–102.

[Shaw et al., 1992] Shaw, M. J., Park, S., and Raman, N. (1992). Intelligent scheduling with machine learning capabilities: The induction of scheduling knowledge. *IIE Transactions*, 24(2):156–168.

[Sigman, 2006] Sigman, K. (2006). Stationary marked point processes. In *Springer Handbook of Engineering Statistics*, pages 137–152. Springer.

[Sigman, 2009] Sigman, K. (2009). Introduction to renewal theory ii. In *Introduction to Renewal Theory*.

[Sparaggis and Towsley, 1994] Sparaggis, P. D. and Towsley, D. (1994). Optimal routing and scheduling of customers with deadlines. *Probability in the Engineering and Informational Sciences*, 8(1):33–49.

[Srinivasan and Anderson, 2005] Srinivasan, A. and Anderson, J. H. (2005). Fair scheduling of dynamic task systems on multiprocessors. *Journal of Systems and Software*, 77(1):67–80.

[Stirzaker and Grimmett, 1992] Stirzaker, D. and Grimmett, G. (1992). Probability and random processes. *Probability and random processes*.

[Torab and Kamen, 2001] Torab, P. and Kamen, E. W. (2001). On approximate renewal models for the superposition of renewal processes. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240)*, volume 9, pages 2901–2906. IEEE.

[Tweedie, 1957] Tweedie, M. C. (1957). Statistical properties of inverse gaussian distributions. i. *The Annals of Mathematical Statistics*, 28(2):362–377.

[ul Islam and Lin, 2015] ul Islam, F. M. M. and Lin, M. (2015). Hybrid dvfs scheduling for real-time systems based on reinforcement learning. *IEEE Systems Journal*, 11(2):931–940.

[Vestal, 2007] Vestal, S. (2007). Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *28th IEEE Real-Time Systems Symposium (RTSS 2007)*, pages 239–243.

[Villalba Frias, 2018] Villalba Frias, B. (2018). *Bringing Probabilistic Real-Time Guarantees to the Real World*. PhD thesis, University of Trento.

[von der Brüggen et al., 2022] von der Brüggen, G., Bozhko, S., Günzel, M., Chen, K.-H., Chen, J.-J., and Brandenburg, B. B. (2022). Probabilistic real-time scheduling and its possible link to mixed-criticality systems. RTSS 2022.

[von der Brüggen et al., 2018] von der Brüggen, G., Piatkowski, N., Chen, K.-H., Chen, J.-J., and Morik, K. (2018). Efficiently Approximating the Probability of Deadline Misses in Real-Time Systems. In Altmeyer, S., editor, *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, volume 106 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:22, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[von der Brüggen et al., 2021] von der Brüggen, G., Piatkowski, N., Chen, K.-H., Chen, J.-J., Morik, K., and Brandenburg, B. B. (2021). Efficiently approximating the worst-case deadline failure probability under edf. In *2021 IEEE Real-Time Systems Symposium (RTSS)*, pages 214–226. IEEE.

[Wald, 1944] Wald, A. (1944). On Cumulative Sums of Random Variables. *The Annals of Mathematical Statistics*, 15(3):283 – 296.

[Wartel et al., 2013] Wartel, F., Kosmidis, L., Lo, C., Triquet, B., Quinones, E., Abella, J., Gogonel, A., Baldovin, A., Mezzetti, E., Cucu, L., et al. (2013). Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 241–248. IEEE.

[Wilhelm et al., 2008] Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D. B., Bernat, G., Ferdinand, C., Heckmann, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P. P., Staschulat, J., and Stenström, P. (2008). The worst-case execution-time problem - overview of methods and survey of tools. *ACM Trans. Embed. Comput. Syst.*, 7(3):36:1–36:53.

[Yan and Zhang, 2008] Yan, J. and Zhang, W. (2008). WCET analysis for multi-core processors with shared l2 instruction caches. In *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 80–89. IEEE.

[Ye and Chen, 2014] Ye, Z.-S. and Chen, N. (2014). The inverse gaussian process as a degradation model. *Technometrics*, 56(3):302–311.

[Yih and Thesen, 1991] Yih, Y. and Thesen, A. (1991). Semi-markov decision models for real-time scheduling. *The International Journal of Production Research*, 29(11):2331–2346.

[Zagalo, 2022] Zagalo, K. (2022). Hypoexponential. https://github.com/kevinzagalo/hypoexponential.

[Zagalo et al., 2022a] Zagalo, K., Abdeddaïm, Y., Bar-Hen, A., and Cucu-Grosjean, L. (2022a). Response time stochastic analysis for fixed-priority stable real-time systems. *IEEE Transactions on Computers*, pages 1–12.

[Zagalo and Auvray, 2022] Zagalo, K. and Auvray, M.-A. (2022). Simso with probabilistic execution times. https://github.com/kevinzagalo/simso.

[Zagalo et al., 2020] Zagalo, K., Cucu-Grosjean, L., and Bar-Hen, A. (2020). Identification of execution modes for real-time systems using cluster analysis. In *25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 1173–1176.

[Zagalo and Verbytska, 2022] Zagalo, K. and Verbytska, O. (2022). rinversegaussian. https://github.com/kevinzagalo/rInverseGaussian.

[Zagalo et al., 2022b] Zagalo, K., Verbytska, O., Cucu-Grosjean, L., and Bar-Hen, A. (2022b). Response Times Parametric Estimation of Real-Time Systems. https://hal.inria.fr/hal-03839408.