Federal University of Amazonas – UFAM

Institute of Computing – ICOMP

Graduate Program in Informatics – PPGI

Scheduling Hard Real-Time Tasks in Heterogeneous Multiprocessor Platforms subject to Energy and Temperature Constraints

Eduardo Bezerra Valentin

Manaus – AM

September, 2017

Eduardo Bezerra Valentin

Scheduling Hard Real-Time Tasks in Heterogeneous Multiprocessor Platforms subject to Energy and Temperature Constraints

A thesis submitted to the Graduate Program in Informatics of the Institute of Computing of the Federal University of Amazonas in partial fulfillment of requirements for the degree of Doctor of Sciences. Concentration area: Embedded Systems and Software Engineering

Supervisor: Raimundo Barreto, D.Sc.
Co-Supervisor: Rosiane de Freitas, D.Sc.

Manaus – AM

September, 2017

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Valentin, Eduardo Bezerra
V156s Scheduling Hard Real-Time Tasks in Heterogeneous Multiprocessor Platforms subject to Energy and Temperature Constraints / Eduardo Bezerra Valentin. 2017
154 f.: il. color; 31 cm.

Orientador: Raimundo Barreto
Coorientadora: Rosiane de Freitas
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. scheduling. 2. hard real-time. 3. integer linear programming. 4. heterogeneous systems. 5. energy constraints. I. Barreto, Raimundo II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Scheduling Hard Real-Time Tasks in Heterogeneous Multiprocessor Platforms subject to Energy and Temperature Constraints"

EDUARDO BEZERRA VALENTIN

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Raimundo da Silva Barreto - PRESIDENTE

Prof. José Reginaldo Hughes Carvalho - MEMBRO INTERNO

Prof. Lucas Carvalho Cordeiro - MEMBRO INTERNO

Prof. Rômulo Silva de Oliveira - MEMBRO EXTERNO

Prof. Carlos Renato Belo Azevedo - MEMBRO EXTERNO

Manaus, 29 de Setembro de 2017

To all computer engineers who
might be using a scope to measure voltage wave form.

Acknowledgments

> "And let steadfastness have its full effect, that you may be perfect and complete, lacking in nothing."
>
> *James 1:4*

This has been a difficult, but joyful, journey which I have had God's help to this very day. I know that on every obstacle that I thought too hard to cross, He has given me strength and perseverance. Therefore, I thank God, for blessing me with the patience, the intelligence, and the ability to complete this work. My life belongs to Him.

I want to say thank you to my wife, Belisa Magalhães, Bela, for the true love. She was the one to first believe and trigger the motivation in me to start this journey. Thanks to Bela for the discussions about research challenges and researcher's life and role. Thanks for listening about my research, even when the subject was not of interest. For sure, without her support, patience, and care I would not completed this work. To my kids, Ed and Estela, thanks for giving up the time with Dad, so he could work. Many thanks to my family, in special to my mother, Fátima, for her love, care, attention, and understanding; to my father, Edson (*in memorian*) and my brothers, Fábio and Milton (*in memorian*), for their valuable contribution to my education.

Many thanks to my advisor, professor Raimundo Barreto. One would say I could have done this work anywhere I pleased, and I chose to follow an example of steadfastness. Thanks for the understanding, support, guidance, focus, invested time, and believing in my work. Thanks for sharing with me a little of the challenges of conducting successful research in the Amazon region of Brazil. Perhaps, one day, I too will be an eagle.

I wish many thanks to my co-advisor, professor Rosiane de Freitas. Thanks for the patience to walk me through the world of combinatorial optimization. Thanks for always having insights and showing me different perspectives of the same problem.

Thanks to all colleagues from GISE research group. Thanks to all friends, staff members, and professors at ICOMP/UFAM. Thanks for the partial financial support provided by SOBRAPO, FAPEAM, and CAPES.

"Computers may be thought of as engines for transforming free energy into waste heat and mathematical work"

Charles H. Bennett, 1981

Abstract

The power wall is a barrier to improvement in the processor design process due to the power consumption of components. The production of energy optimum systems demands knowledge of different disciplines. The usage of heterogeneous multicore platforms is appealing for recent applications, e.g., hard real-time systems. The motivation is the potential reduced energy consumption offered by such platforms. Hard real-time systems are present in life critical environments. Reducing the energy consumption on such systems is an onerous process. Scheduling becomes particularly challenging to improve system utilization and minimize system energy consumption and peak temperature on such platforms, specially subject to hard real-time constraints.

Therefore, we propose a study to effectively answer the pertinent research question: "*How to offer users timing correctness and guarantees of hard real-time systems executed on heterogeneous multicore systems with energy and temperature constraints?*". Finding optimal solutions for such question has still several open research questions.

The main aim of this thesis is to propose an *energy optimization method for hard real-time system on heterogeneous multicore platform demonstrating that it is possible to timely compute timing correctness and guarantees using a sufficient and necessary condition; accounting for energy, temperature, preemption, precedence, shared resources constraints, and architectural interference.* The proposal is a two fold approach. First, we investigate the process of finding the optimal task to core and frequency to task processes by means of applying exact schedulability tests for heterogeneous multicore platforms. Second, the outcome of the optimization analysis shall be used as reference to the on-line scheduler. We believe that we have achieved the main objective of this research by combining: (a) schedulability analysis from hard real-time systems, (b) representative mathematical formulations, based on integer linear programming, covering modern processors technological characteristics and using a classical combinatorial mathematical formulation (Multilevel Generalized Assignment Problem), and (c) robust exact implicit enumeration algorithmic strategies from combinatorial optimization, such as branch-and-cut and branch-and-price.

The systematic literature review in the research subject reveals that the field has open questions to be answered. For instance, to the knowledge of the author only five works in the state-of-the-art literature deal with the problem by providing optimal solutions. Typically, the existing approaches focus on either heuristics or approximation algorithms. Also, only one work has a proposal to evaluate the schedulability in this scenario with an exact test. The typical formulation in the specialized literature is a 0/1 integer linear programming model which considers a continuous processor frequency domain and determines a single operating frequency per processor. One of the hypotheses tested in this research is: *stronger feasibility analysis offers tighter bounds for the problem.* We believe

that this can be observed, for example, in the results produced by solvers for fixed priority schedulers, by means of an analysis based on a comparative study. By applying less accurate schedulability tests, such as utilization based, the solvers take longer to converge to optimal solutions, when compared to solvers that apply exact schedulability tests based on response time analysis. Another hypothesis tested in this research is: *practical instances of the problem are timely solvable to optimal.* We have experimented, by means of a comparative study, on finding feasible solutions for workload for fixed priority schedulers with up to 50 tasks distributed on four processors with seven different available frequencies. On independent hard real-time tasks scheduled using EDF policy, we found optimal distribution of up to 90 tasks on four processors with seven different available frequencies. In both cases, the solutions were found within 30 min of execution time. Similarly, on dependent tasks workload, we have optimally distributed 22 tasks, from an automotive control hard real-time application, on four processors with seven different available frequencies, with two shared resources and 23 precedence constraints within 1.5 h. We consider a few hours in the design phase a price worth paying in this context.

Key-words: scheduling, hard real-time, integer linear programming, MGAP, energy constraints, heterogeneous systems.

List of Figures

List of Tables

List of abbreviations and acronyms

CNC	Computer Numerical Control.
DCT	Discrete Co-sine Transform.
DC-DC	Direct Current to Direct Current converter.
DVFS	Dynamic Voltage and Frequency Scaling.
DVS	Dynamic Voltage Scaling.
DPM	Dynamic Power Management.
EDF	Earliest Deadline First.
EOSS	Exact and Optimal Scheduling Strategy.
FFT	Fast Fourier Transform.
FPGA	Field-Programmable Gate Array.
HMP	Heterogeneous Multiple Processors.
ILP	Integer Linear Programming.
LCM	Least Common Multiple.
LMC	Linear Motor Control.
MILP	Mixed Integer Linear Programming.
MPSoC	Multiprocessor System-On-Chip.
PU	Processing Unit.
RT	Real-Time.
SoC	System-On-Chip.
VLSI	Very-Large-Scale Integration.
WCEC	Worst Case Execution Cycles.
WCET	Worst Case Execution Time.

List of symbols

D_i	Deadline of τ_i.
Lp	DVFS Switching Latency.
p_i	Fixed Priority of τ_i.
A_i	Interference due to Architectural Latency Suffered by τ_i.
B_i	Interference due to Shared Resources Suffered by τ_i.
$\mathcal{I}_i$	Interference Suffered by τ_i.
T_i	Period of Execution of τ_i.
τ_i	Real-Time Task i.
J_i	Release Jitter Indicating the Worst Release Time of τ_i.
R_i	Response Time of τ_i.
$hp(i)$	Set of High Priority Tasks of τ_i.
$\mathcal{M}$	Task Model.
U^j_{total}	Total Utilization of Processor j.
u_i	Utilization of τ_i, Defined as $\frac{C_i(f)}{T_i}$.
$WCEC_i$	Worst-Case Execution Cycle
$C_i(f)$	Worst Execution Time, as Function of Frequency f of τ_i.

List of Algorithms

Contents

1 Introduction

The pursuit of outstanding performance of computers is a major concern for computer scientists. The evolution in academia and industry towards high performance computing has led us to the creation of processors that are faster than their predecessors. Nevertheless, the well known "power wall" phenomena limits such performance increments (Venkatachalam and Franz, 2005).

The power wall is a barrier to improvement in the processor design process due to the power consumption of components. Power consumption has become the primary influence in overall microprocessor design complexity due to ideal geometric scaling and non-ideal electrical scaling. It is no longer viable to simply increase clock speeds of existing designs (Cochran, 2013). Power consumption is a major aspect that limits the performance of computers in different sides of the computing spectrum. The pursuit of energy efficiency is useful for mobile devices to improve operating duration and also helpful for server systems to reduce power bills (Chen and Kuo, 2007; Chen et al., 2009; Chen and Thiele, 2008, 2009, 2011).

In large data centers, power consumption is a major concern due to the increasing expense in room cooling systems and mainly due to the expensive power bills. For instance, according to Eric Schmidt, CEO of Google, "What matters most to the computer designers at Google is not speed but, power, low power, because data centers can consume as much electricity as a city" (Markoff and Lohr, 2002). Also, as per report present in Google's web site "The Big Picture - Google Greener" (Google, 2013), Google's power bill in 2012 was 3,324,818 MW h. Google continuously used in 2011 electricity in the scale of hundred of thousands of MW, which can power 200,000 houses.

On the other side of the computing spectrum, the battery life duration is a concern, thus equipping mobile devices with powerful processors decreases their autonomy even further (Venkatachalam and Franz, 2005). The usage of powerful processors may lead to disastrous situations; unless designers apply careful constructions across all phases of product conception – i.e., mechanical, electric, and system software development process. For instance, silicon devices may reach temperatures as high as 125 °C, which heat may spread towards the device enclosure / cover, causing hazards to users.

Nowadays, multicore platforms have become the *de-facto* solution to cope with the rapid increase of system complexity, reliability, and energy consumption (He and Mueller, 2012a). Modern computing systems often adopt multiple processing elements to enhance computing capability or to reduce the power consumption, especially for embedded systems (Chen and Thiele, 2008). Moore's law is no longer sustained by increasing clock

frequencies, but instead by adding extra cores in multiprocessors. The performance per watt ratio is a key metric, as higher clock ratios also demand higher supply voltages. Added to symmetric multicore processors, platforms with co-processing units, pipelined computing, and heterogeneous multicores gain popularity in architecture designs (Awan and Petters, 2013; Chen and Thiele, 2008).

Modern multicore processors for the embedded market are often heterogeneous in nature (Awan and Petters, 2013). For instance, on system-on-a-chip (SoC) platforms, a field-programmable gate array (FPGA) might appear to provide flexibility to execute tasks/jobs in hardware for acceleration. Some helper devices aim to reduce the workload on the processor for the enhancement of special functions. Discrete co-sine transform (DCT), or Fast Fourier transform (FFT) (Chen and Thiele, 2008), are examples of offloading processors' workload. Multimedia platforms often contain one processor and one or more co-processors; such as those for video codec functionality (Chen et al., 2008).

Some applications require real-time requirements. Heterogeneous processors are of great interest for system designers due to the expected energy consumption reduction. Consequently, practitioners consider multiple heterogeneous processors for different applications with hard deadline constraints. Nevertheless, developing software with timing constraints for multiple heterogeneous processors is a complex task. Scheduling becomes especially hard to deal with particularly under low power or temperature constraints.

Therefore, it is beneficial to build knowledge on efficient methods and techniques to produce software with timing guarantees for architectures with multiple heterogeneous processors, under energy and temperature constraints. Such methods and techniques are essential for practitioners and researchers. Such effective methods and techniques support reducing power bills, improve system reliability, and increase the efficient usage of energy, assisting to reduce environmental impacts.

1.1 Context

The context of the present research is the software development for hard real-time heterogeneous multicore systems subject to severe energy and temperature constraints. The objective is to reduce energy consumption and to control the temperature of such systems by means of conscious application of energy and temperature management. The processing model of interest is multiple and parallel heterogeneous processors.

This research aims to aid architects, designers, developers and researchers of embedded systems across the process of specification and deployment of products. The embedded system considered has severe constraints, such as timing, temperature, and energy restrictions.

We conducted a systematic literature review (Valentin and Barreto, 2016) and found the following examples of such embedded systems:

- Unmanned Aerial Vehicles;
- Linear motor control (LMC) systems;
- Computer Numerical Control (CNC) machines;
- X-ray control system;
- Control system in the automotive area (airbag control unit, anti-lock braking system, and electronic stability control system);
- Distributed computing under severe constraints, e.g., target monitoring and tracking, military, environmental remote monitoring.

1.2 Motivation

Design and development processes of embedded systems are peculiarly complex due to its unusual characteristics. The system designer must balance severe constraints, such as size, weight, cost, energy consumption, time-to-marketing, reliability, and others even more specific to the target environment. Excessive heating, vibration, lightning, corrosion, water, fire, power source variation are some examples (Barreto, 2005). Energy management and temperature control, nevertheless, are crucial characteristics on embedded systems, specially those designed to be mobile.

Achieving successful energy-aware application development requires specialized labor force. In general, system designers lack this skill. Thus, aiding the design and modeling process of hard real-time applications subject to energy constraints on multiple heterogeneous platforms is the motivation of this research.

1.3 Problem statement

Users expect timing correctness and guarantees of hard real-time systems (Burns and Wellings, 1997). In the specialized literature, there are schedulability tests to aid system designers in the task of providing such guarantees (Lehoczky et al., 1989; Liu and Layland, 1973; Sha et al., 1990). However, the existing methods (Lehoczky et al., 1989; Liu and Layland, 1973; Sha et al., 1990) lack accounting for the peculiarities of heterogeneous multicore systems with energy and temperature constraints. Additionally, the existing tests discard the overhead imposed by the main power management techniques. He and Mueller (2012a) introduce a sufficient schedulability test for heterogeneous system, but

it is only a sufficient condition. Valentin and Barreto (2010) propose a schedulability analysis using DVFS, but their solution applies only to uni-processed systems.

The addressed problem is: "*How to offer users timing correctness and guarantees of hard real-time systems executed on heterogeneous multicore systems with energy and temperature constraints?*"

1.4 Objectives

The main aim of this thesis is to propose an *energy optimization method for hard real-time system on heterogeneous multicore platform demonstrating that it is possible to timely compute timing correctness and guarantees using a sufficient and necessary condition; accounting for temperature, preemption, precedence, shared resources constraints, and architectural interference.*

The specific objectives are (1) Exact Schedulability Analysis, and (2) Task to Core Assignment considering Architectural Interference. The following items detail each specific objective:

1. To define a schedulability analysis for hard real-time scheduling on heterogeneous multicore systems with sufficient and necessary conditions.

2. To deliver an off-line assignment algorithm to distribute hard real-time tasks among heterogeneous cores of a multicore system; assigning a CPU frequency to each task on active scenarios; using a sufficient and necessary schedulability analysis considering *energy, temperature, preemption, precedence, mutual exclusion, and architectural interference.*

1.5 Existing Research

We have performed a systematic literature review (Kitchenham and Charters, 2007) of scheduling hard real-time tasks on multiple heterogeneous processors under energy constraints, such as low power and temperature control. Using a repeatable, unbiased, and systematic approach (Basili et al., 1994a,b; Biolchini et al., 2007), we have identified, presented, classified, and criticized all 29 existing works on the subject. During the process, we analyzed hundreds of works found in relevant digital libraries (Valentin and Barreto, 2016). We concluded that, even though there has been an increasing interest in the subject over the past years, there are still open questions to address.

Some topics have only been slightly explored. For instance, only five works address the problem of offering optimal solutions, although most works consider small- or medium-

size instances. Most works are also interested in solutions produced off-line. Only five works consider on-line approaches.

The most challenging aspect is exact schedulability tests. We found only one work addressing the analysis of task response times, whereas the majority consider utilization-based analysis, which is a sufficient only condition.

Similarly, only one work explores task migration in this subject. Furthermore, only one work studies mutual exclusion constraints, and only four works consider precedence constraints in their task models. The most explored workload type is periodic or aperiodic tasks. The least explored workload type is the combination of soft and hard real-time tasks.

DVFS is the most used technique for addressing power consumption issues. Only a minority of the works explore DPM, for instance. Similarly, we emphasize the need for studying the thermal topic on this subject, as we found only five works addressing temperature control. Most works have interest in full heterogeneous architectures. However, heterogeneous clusters are becoming increasingly studied.

Most works investigated for the purposes of this review focus on dynamic power reduction. Additionally, most works consider the dynamic power as a convex power function. We highlight that the static power consumption is non-negligible in current modern processors. Moreover, advanced power management techniques, such as Adaptive Voltage Scaling (AVS), and the thermal behavior of processors change the increasing convex power consumption function (Alahmad and Gopalakrishnan, 2011). Therefore, we highlight the need to explore different power models, apart from optimizing the dynamic power using a convex power function.

We found only three benchmarks in use on the subject: E3S (EEMBC, 2014), TGFF (Dick et al., 1998), and UUniFast (Bini and Buttazzo, 2004). The lack of standardization in empirical experimentation design complicates the process of comparing existing solutions. Comparing the results of expected energy consumption reduction is particularly challenging due to the lack of standardization. Consequently, we highlight the need for benchmarks and reference instances. Furthermore, it is crucial to have studies reporting fair comparisons between existing works under controlled environment experimentation. Thus, we also highlight a need for standardized empirical experimentation to improve the level of scientific evidence reported in this subject. Reporting research limitations, costs, and empirical results is a major concern. Only two works highlight the expected costs for applying their solutions.

Table 1 presents a summary of the usage of techniques which relates each work with all techniques found in this study to model, solve, analyze, and experiment the problem of interest. Table 1 is a reference of current modeling assumptions and modeling limitations and highlights under which circumstances each primary work can be best

applied. Whenever the modeling assumption or technique applies to a primary work, the column is marked with a "Y" symbol. For example, in the first row, Yu and Prasanna (2003) use a discrete set of frequencies for their processor clock, in specialized processing units, to allocate a periodic workload off-line considering low power constraints and deriving approximated solutions. However, Yu and Prasanna (2003) overlook the thermal problem, and their solution is applicable when the power consumption is a convex and increasing function of frequency, neglecting the static power commonly present in modern processors.

Table 1 – Summary of technique usage per paper

	Clock		Processor			Soluc.		Alloc.		Workload			Schedulability Test					Energy Management					Problem Model			Power Control			Solution Method							
REF	Continuous	Discrete	Hetero. Clusters	Spec. Architecture	Perf. Variation	Exact	Approximated	Offline	Online	Periodic	Mixed	Migration	Preemption	Precedence	Mutual Exclusion	Response Time	Utilization	Low Power	Thermal	Convex Dyn. Power	Dyn. Power	Dyn. + Static Power	ILP	QPP	NLP	DVFS	DPM	Load Balance	Heuristic	Branch-And-Bound	Protocol	Schedulability Test	Thermal PID	Evolutionary Based	Dyn. Prog.	Linear Relaxation
(Zhang et al., 2015)	Y				Y		Y	Y		Y			Y				Y	Y		Y	Y		Y										Y		
(Yu and Prasanna, 2003)		Y		Y			Y	Y		Y			Y				Y	Y		Y	Y		Y			Y									Y
(Yang et al., 2009)	Y			Y			Y	Y		Y							Y	Y		Y		Y				Y								Y	
(Yang et al., 2012)		Y		Y			Y	Y		Y				Y					Y	Y		Y				Y			Y						
(Terzopoulos and Karatza, 2013)		Y	Y				Y		Y	Y								Y		Y	Y					Y			Y						
(Saha et al., 2012)		Y	Y				Y	Y		Y			Y				Y	Y	Y	Y		Y	Y			Y							Y			
(Prescilla and Selvakumar, 2013)	Y			Y			Y	Y		Y							Y	Y		Y	Y					Y			Y				Y		
(Min-Allah et al., 2012)		Y			Y		Y	Y		Y			Y			Y		Y		Y	Y					Y					Y				
(Goossens et al., 2008)			Y				Y	Y		Y		Y					Y	Y				Y					Y	Y					Y		
(Kim et al., 2008)		Y		Y			Y		Y		Y							Y		Y	Y					Y			Y						
(Kim et al., 2005)		Y		Y			Y		Y		Y							Y		Y	Y					Y			Y						
(Prasanna and Yu, 2002)		Y		Y			Y	Y		Y			Y				Y	Y		Y	Y		Y			Y									Y
(Hung et al., 2006)		Y			Y		Y	Y		Y							Y	Y		Y		Y				Y									
(Hettiarachchi et al., 2013)				Y			Y	Y		Y							Y		Y			Y					Y					Y			
(He and Mueller, 2012b)		Y	Y				Y	Y	Y	Y			Y				Y	Y		Y		Y				Y		Y					Y		
(He and Mueller, 2012a)		Y	Y			Y				Y							Y	Y		Y	Y					Y	Y				Y				
(Chen et al., 2009)	Y			Y			Y	Y		Y							Y	Y				Y	Y			Y			Y						
(Chen and Thiele, 2011)	Y			Y			Y	Y		Y							Y	Y		Y	Y		Y			Y	Y		Y						
(Chen and Thiele, 2009)	Y			Y			Y	Y		Y							Y	Y		Y	Y		Y			Y	Y		Y						
(Chen et al., 2008)	Y			Y		Y		Y	Y	Y			Y	Y	Y		Y	Y		Y	Y				Y	Y	Y				Y				
(Chen and Thiele, 2008)	Y			Y		Y	Y	Y		Y							Y	Y		Y	Y					Y	Y		Y						
(Chen and Kuo, 2007)					Y		Y			Y										Y	Y					Y			Y						
(Chantem et al., 2011)		Y		Y		Y	Y	Y		Y				Y				Y	Y	Y		Y	Y			Y			Y						
(Barrefors et al., 2014)		Y		Y			Y	Y		Y							Y	Y	Y	Y		Y	Y			Y		Y		Y					
(Awan and Petters, 2013)				Y			Y	Y			Y						Y	Y		Y	Y						Y		Y						
(Alahmad and Gopalakrishnan, 2011)		Y		Y		Y	Y	Y		Y							Y	Y				Y	Y			Y	Y		Y						
(Valentin et al., 2015b)		Y	Y			Y		Y		Y			Y	Y	Y	Y		Y		Y		Y				Y					Y				
(Chu et al., 2009)	Y			Y		Y	Y	Y		Y							Y	Y		Y		Y			Y				Y						
(Chen et al., 2011)				Y			Y	Y		Y			Y				Y	Y		Y	Y					Y				Y					

1.6 Hypotheses

The following hypotheses motivate the present research:

1. Stronger feasibility analysis offers tighter bounds for the problem (stated at Section 1.3).

2. Practical instances of the problem are timely solvable to optimal (stated at Section 1.3).

Most works found in specialized literature focus on solving the problem approximately, avoiding optimal solutions. Besides, the authors use utilization bounds, which is a non-exact test for hard real-time systems, causing even further sub-optimal solutions. Accurate power and temperature models are crucial items to determine optimal and realistic solutions. Applying biased models may lead to unrealistic results. Most of the existing works use simplistic power models, ignoring important relations between temperature and power, for instance.

1.7 Proposed Approach

This section introduces the methods and materials needed on this research. The proposed method serves as an artifact to test the hypotheses presented in Section 1.6; i.e., applying exact schedulability tests, feasibility of optimal solution searching on practical instances, usage of accurate power, energy, and temperature processor models.

Section 1.7.1 introduces the overall proposed architecture. Section 1.7.2 presents the proposed schedulability analysis. Section 1.7.3 presents an overview of the mathematical formulations used in this method. Section 1.7.4 describes the computational techniques of resolution to solve the mathematical formulations and to find the optimal tasks to processors and frequency to tasks assignment. Section 1.7.5 discusses the online system execution explaining the real-time scheduling.

1.7.1 Overall Architecture

The problem of power aware real-time scheduling on heterogeneous multicore platforms has solutions in the literature delivering a two steps approach. In the first step, the off-line procedure assesses the workload partitioning among the processors accounting for schedulability analysis. In the second step, the operating system schedules the real workload according to the determined scheduling policy and the calculated assignment. The proposed method, *Exact and Optimal Scheduling Strategy* (EOSS), considers also a combined solution, performing off-line workload analysis benefiting the online scheduling.

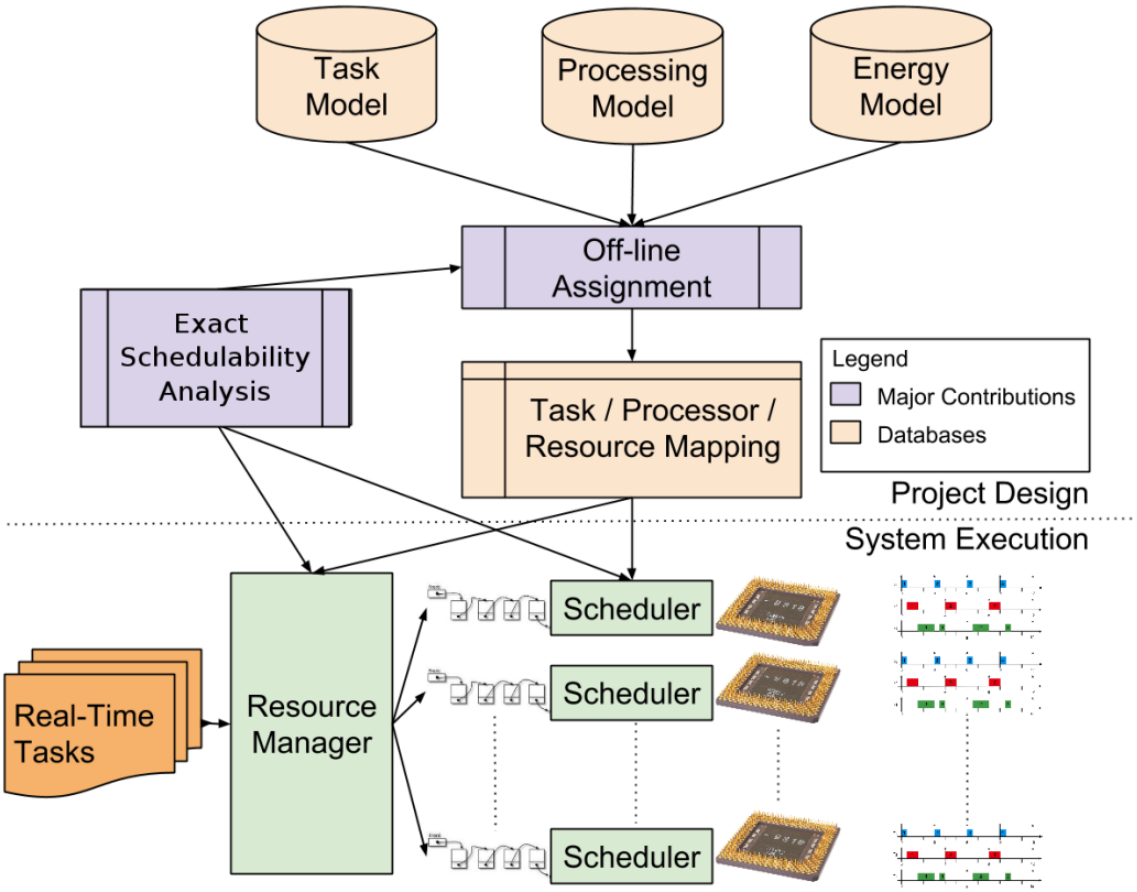


Figure 1 – Overall EOSS architecture

Figure 1 depicts the proposed architecture. During project design time, the analyst delimits and models the system using three databases; which are inputs for EOSS. The first database represents the Task Model and contains items with characteristics of each tasks known to compose the system workload. The second database contains a Processing Model which consists of details of each processing unit. The third database contains an Energy Model that describes the energy consumption relations within the system.

The off-line analysis starts by determining the optimal assignment between tasks to processors and frequency to tasks. The assignment process adopts an Integer Linear Programming (ILP) model. The databases, provided by the analyst, feed the ILP model. The assignment process considers an exact schedulability analysis to produce a workload split that respects the system schedulability. The optimal assignment is then used in the online system execution.

The online system execution consists of two main entities: the resource manager and the real-time scheduler. The resource manager receives all tasks that arrive. Once a task arrives, the resource manager identifies if the task is part of the known workload. When the task is within the known workload, the resource manager looks up the optimal

solution, finds the task to processor assignment, and adds the task to the processor's task queue. At last, the task scheduling passes to local scheduler. The local scheduler is in charge of the corresponding processor. The scheduler of each processor then determines the schedule according to the scheduling policy.

1.7.2 Exact Schedulability Analysis based on Architectural Interference

The proposed exact schedulability analysis provides a necessary and sufficient condition when applied. The schedulability analysis establishes each task response time. The task response time decomposition considers the classical constraints present in the real-time specialized literature; such as jitter (setup time), precedence constraints, task interference due to preemption of high priority tasks, and task interference due to shared resources utilization by low priority tasks. The proposed exact schedulability analysis is innovative because considers interference caused by inherited architectural usage; such as inter-cluster interference (He and Mueller, 2012a).

1.7.3 Mathematical Formulations

The assignment strategy consists in finding the optimal hard real-time workload split among the existing heterogeneous cores. Finding the optimal solution for this problem is NP-Hard (Chen et al., 2009; Chen and Thiele, 2009). Even its decision version is difficult to solve and it is part of NP-Complete (Garey and Johnson, 1979). We adopt mathematical formulations based on 0/1 ILP models.

1.7.4 Computational Technique of Resolution

We use a general branch-and-bound based exact implicit enumeration method combined with schedulability tests to conduct the process of finding optimal solutions. The algorithm's input is the processing model $\mathcal{H}$, the desired task model $\mathcal{M}$, and a possible upper bound ub. The algorithm outputs the optimal distribution of hard real-time tasks among the processors that consumes the least power among the possible assignments, informing as well in which frequency each task may be executed, and the total system estimated energy.

The algorithm starts by denoting the set L of active problem nodes to contain only the initial Integer Linear Problem. When an upper bound ub is known, the objective function value v^* and the optimal solution x^* are set to match ub objective function value and solution structure, otherwise, they are set to $+\infty$ and to $NULL$, respectively. The algorithm iteratively evaluates each element of the set L. Each problem node is initially tested against the schedulability test that fits for the problem scheduling policy. In cases where the schedulability test accepts the node, then a regular branch-and-bound method is

followed. The linear relaxation of the node is then computed and solved. The problem node is then partitioned and new restricted problem nodes are derived and incorporated into L. The iterative process repeats until the set L is empty. This branch-and-bound algorithmic strategy may be specialized into two different classes: branch-and-cut or branch-and-price.

1.7.5 Resource Manager and Online Scheduler

The system execution operates by following a partitioned scheduling strategy. When hard real-time tasks arrive the partitioned scheduling follows the output produced by the assignment phase of Section 1.7.4. In this aspect, the system execution can leverage the investment in finding an optimal solution, while being flexible to adapt to system changes that may occur in run time. Thus, a hard real-time task scheduling algorithm utilizes the optimal solution found as baseline. The differentiation during the system execution is that the partitioned scheduling takes advantage of the optimal solution generated by the off-line phase, explained in Section 1.7.4.

1.8 Outline

The rest of this text has the following structure. For readers consideration, Chapter 2 contains a list of terms and common definitions used in this text. Chapters 3 and 4 conduct the reader throughout the findings and outcomes encountered on the distribution of independent hard real-time task using branch-and-cut and branch-and-price algorithmic strategies, respectively. The details of applying the method while distributing dependent hard real-time tasks are explained in Chapter 5. Chapter 6 describes details of schedulability analysis for cluster based multicore platforms. Chapter 7 presents a summary of the main techniques which offer guarantees and timing correctness for hard real-time systems executed on energy and temperature constrained heterogeneous multicore systems. Chapter 8 ends this text with the conclusions and future work.

2 Theoretical Preliminaries

In this chapter, we include a list of terms used in this text. Section 2.1 covers real-time systems. Section 2.2 details concepts on schedulability analyses. Section 2.3 presents concepts used in multicore systems. Section 2.4 discusses power and energy concepts and Section 2.5 briefly introduces concepts on integer linear programming.

2.1 Real-Time Systems

For embedded real-time (RT) systems, it is imperative to respect timing constraints posed by the environment (Awan and Petters, 2013). Embedded real-time systems must complete the tasks before their deadlines to maintain the system stability (Chen and Kuo, 2007).

There are situations in which missing critical data, or delivering late in time, is catastrophic. When missing deadlines leads to catastrophic situations, the embedded real-time system is known as hard real-time system, whereas when missing deadlines leads to performance degradation, the embedded real-time system is known as soft real-time.

A periodic task τ_j (Liu, 2000) is an infinite sequence of task instances, referred to as jobs, where each job of a task comes in a regular period T_j (Chen and Kuo, 2007). Figure 2 exemplifies the execution of a single real-time task whose period is 20 ms. The hyper-period of the task set $\mathcal{M}$, denoted by L, is the minimum positive number L, so that $\frac{L}{T_j}$ is an integer for every task τ_j in $\mathcal{M}$. For example, L is the least common multiple (LCM) of the periods of tasks in $\mathcal{M}$ when the periods of tasks are all integers (Chen and Kuo, 2007; Chen and Thiele, 2011).

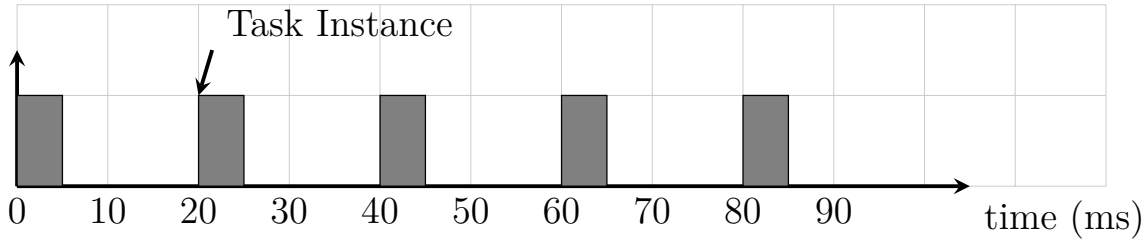


Figure 2 – Example of periodic task execution. Each instance comes in a regular period of 20 ms

The online scheduling of real-time tasks is usually a two fold process (Burns and Wellings, 1997). The first phase consists of a schedulability test or schedulability analysis. A schedulability analysis aims to determine if the set of tasks is schedulable; i.e., the scheduled tasks respect their deadlines. The schedulability analysis provides conditions

Table 2 – Schedulability Analyses for one core

Test	Complexity	Accuracy	
		Sufficient	Exact
Liu and Layland	$O(n)$	Yes	No
Bini Hyperbolic	$O(n)$	Yes	No
R-bound	$O(n)$	Yes	No
Lehoczky[1]	$O(n^2r)$	Yes	Yes
Audsley	$O(n^2r)$	Yes	Yes

1: r is the ratio between the largest period and smallest period: $r = \frac{max(T_i)}{min(T_i)}$.

Source: (AlEnawy and Aydin, 2005)

based on the task model and the criteria defined by the scheduling algorithm. The second phase is the scheduling itself. The Earliest Deadline First (EDF) policy is an optimal uni-processor scheduling policy for independent real-time tasks (Chen and Thiele, 2009, 2011). Similarly, Rate Monotonic is an optimal uni-processor scheduling policy for fixed priority and independent real-time tasks (Farines et al., 2000).

A task model $\mathcal{M}$ is a set composed by n tasks τ_j. A task $\tau_j \in \mathcal{M}$, with $j \leq n$, has the properties: worst-case execution cycle $WCEC_j$; worst-case execution time $C_j(f)$, which is a function of frequency f, thus $C_j(f) = \frac{WCEC_j}{f}$; period of execution T_j; and deadline D_j. A task τ_j also has the following properties, specific to fixed priority policies: fixed priority p_j; set of high priority tasks $hp(j)$ representing the tasks τ_p with priority higher than the priority of τ_j.

2.2 Schedulability Analysis

In the specialized literature, schedulability test or schedulability analysis is a procedure to determine if a real-time task model is schedulable, considering the task model characteristics and the scheduling policy (Lehoczky et al., 1989; Liu and Layland, 1973; Sha et al., 1990). The schedulability analyses differ in computational complexity and accuracy. For example, Table 2 summarizes some tests along with their computational complexity and accuracy for fixed priority real-time tasks on one processor as function of the number of tasks, as reported by AlEnawy and Aydin (2005). We classify the strategies into two categories: utilization based and response time analysis.

2.2.1 Utilization based

Utilization based schedulability analyses are simplistic and fast approximate tests. These analyses use the information of task set or individual task utilization. The task τ_j utilization is $u_j = \frac{C_j(f)}{T_j}$. There are multiple utilization based analyses and bounds. In this section, we highlight the most cited in the literature.

Liu and Layland (1973) propose the following utilization based schedulability test: $U_{total} \leq U_{bound}(n)$, where $U_{total} = \sum_{j=1}^{n} u_j$ and $U_{bound}(n)$ depend on the scheduling policy.

For EDF applied on independent real-time tasks on uni-processor systems, Liu and Layland schedulability test is: $U_{total} \leq 1$, which means EDF achieves 100 % of CPU utilization. For independent real-time tasks, Liu and Layland's utilization based test is an exact condition, i.e., *sufficient* and *necessary*.

For fixed priority real-time tasks, Liu and Layland test is $U_{total} \leq n(2^{\frac{1}{n}} - 1)$. The analyses in this category provide *sufficient* conditions of schedulability: the task set is schedulable if it satisfies the test condition, but if the task set fails to satisfy the test condition, it may still be a schedulable task set.

Bini et al. (2001) provide a hyperbolic based bound stronger than Liu and Layland. Bini Hyperbolic test is designed for large periodic task sets, when the exact analysis is prohibitive due to long execution times. The bound is: $\prod_{j=1}^{n}(1 + u_j) \leq 2$, where u_j is the utilization of task τ_j.

Lauzac et al. (1998) present a utilization bound function of the amount of tasks n and a ration r: $U_{bound}(n, r) = (n-1)(r^{\frac{1}{(n-1)}} - 1) + \frac{2}{r} - 1$, where r is the ratio between the largest period and smallest period: $r = \frac{max(T_j)}{min(T_j)}$. The R-bound is designed for admission control as it has low computational complexity. R-bound yields up to 96 % of processor utilization on large number of tasks.

2.2.2 Response time analysis

Response time analyses are computationally more complex than utilization based analyses. But they are more accurate and provide *exact* conditions, i.e., *sufficient* and *necessary*. The conditions use the information of each task's worst-case execution times and periods. The response time analysis uses the concept of critical instant phasing (Lehoczky et al., 1989).

Lehoczky et al. (1989) present an exact schedulability analysis based on tasks periods and priorities. The condition is:

$$R_j(t) \leq t \leq D_j, \qquad \forall 1 \leq j \leq n \tag{2.1}$$

where:

$$R_j(t) = C_j + \sum_{p \in hp(j)} \left\lceil \frac{t}{T_p} \right\rceil \times C_p, \text{ for } 0 < t < T_j \tag{2.2}$$

Audsley et al. (1993) extend the schedulability analysis proposed by Lehoczki and solve the release jitter problem as well as condense the delay due to semaphore usage. The schedulability analysis is then:

$$R_j \leq D_j, \qquad \forall 1 \leq j \leq n \tag{2.3}$$

where:

$$R_j = I_j + J_j \tag{2.4}$$

$$I_j^{n+1} = C_j + B_j + \sum_{p \in hp(p)} \left\lceil \frac{I_j^n + J_p}{T_p} \right\rceil \times C_p \tag{2.5}$$

and the delay B_j caused by low priority tasks accessing shared resources in the same processors using Priority Ceiling Protocol can be estimated as $B_j = max_{jk}\{D_{jk} | (p_j < p_i) \wedge (C(S_k) \geq p_i)\}$, and $C(S_k)$ is the ceiling priority of the shared resource S_k.

Precedence constraints can be represented by including the maximum response time of the predecessors tasks in the J_j component of the task τ_j. The precedence overhead of a task τ_j may be computed using a topological sorting algorithm applied on a representation of the set of tasks using a precedence graph.

It is worth to highlight that Lehoczky's test and Audsley extensions are *exact* conditions, i.e., *sufficient* and *necessary*. Furthermore, any task model containing tasks with computation equal to deadline is unschedulable, regardless the processor utilization (Audsley et al., 1993).

2.3 Multicore systems

Multicore systems contain multiple processors (processing units). They have been widely adopted. A designer can take advantage of the particular processing units' properties to increase the flexibility of the system (Chen and Thiele, 2009, 2011). An example of such systems is the Multiprocessor System-on-Chip (MPSoC) platforms. The processing units may be heterogeneous or homogeneous.

The DPM/DVS capable chip multi-core processor platforms fall into three categories: full-chip platforms, per-core platforms, and cluster-based platforms (He and Mueller, 2012a).

2.3.1 Full Chip platforms

On the full-chip platforms, the entire processor chip shares the same power supply net. Therefore all cores can only operate at a common frequency at the same time. However, they shut down independently (e.g. Intel CoreTM2 Quad). Switching off some cores is independent of other cores and only requires a minimal amount of transistors. The DVS applies on the whole chip and the DPM applies on the individual cores (He and Mueller, 2012a). Figure 3 illustrates a full chip platform with four processors.

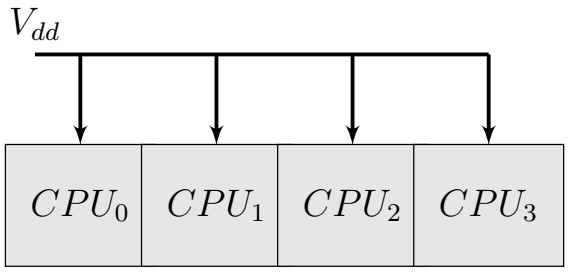


Figure 3 – Full Chip platform: In this example, all four processors, CPU_0, CPU_1, CPU_2, and CPU_3, in the same chip share the same power source, V_{dd}

2.3.2 Per Core platforms

On the per core platforms, it is possible to control the frequency and voltage levels on each existing core separately. With the introduction of Frequency/Voltage Island technique and on-chip voltage regulator, per-core platforms (e.g. AMD PhenomTM Quad-Core) get more and more attention (He and Mueller, 2012a). Figure 4 has an illustration of a per core platform.

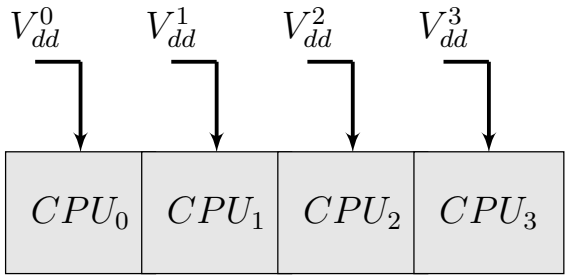


Figure 4 – Per Core platform: CPU_0, CPU_1, CPU_2, and CPU_3 have their own power source, V_{dd}^0, V_{dd}^1, V_{dd}^2, V_{dd}^3, and V_{dd}^4, respectively

2.3.3 Cluster-Based Multi-Core platforms

The cluster-based multi-core platforms are a generalized form of the full-chip and per-core platforms and provides the best compromise. Designers may group the cores of a processor chip into clusters. The cores from the same cluster behave like in the full-chip

platforms and the cores from different clusters behave like in the per-core platforms (He and Mueller, 2012a). Figure 5 illustrates a cluster based platform with four processors grouped into two clusters.

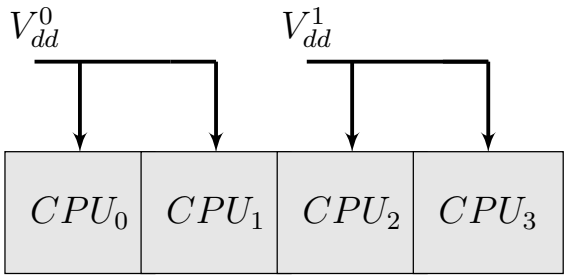


Figure 5 – Cluster Based Multi-Core platform: the power lines group the cores in the chip. The power source V_{dd}^0 powers CPU_0 and CPU_1; whereas the power source V_{dd}^1 powers CPU_2 and CPU_3

2.3.4 Considerations on Response Time Analysis for Multicore Systems

The task influence I_j in multiple processors may be calculated as $I_j^{n+1} = C_j + B_j^r + B_j + \sum_{p \in hp(j)} \left\lceil \frac{I_j^n + J_p + B_p^r}{T_p} \right\rceil \times C_p$. Also, when precedence constraints occur across different processors, this imposes an additional messaging cost that may be incorporated in the emitting task to perform inter-processor communication. When the Multiprocessor Priority Ceiling Protocol is in place to avoid priority inversion issues, the remote blocking delay B_j^r is an upper bound for the blocking time suffered by task τ_j from other tasks in a different processor.

2.4 Power and energy concepts

In the literature, power and energy concepts usually appear interchangeable, even though they are different concepts. Power and energy are commonly defined as the work that a system performs. Energy is the total amount of work a system performs over a period of time, while power is the rate at which the system performs that work (Venkatachalam and Franz, 2005). Formally:

$$P = \frac{W}{T} \tag{2.6}$$

$$E = P \times T \tag{2.7}$$

where P is power, E is energy, T is a specific time interval, and W is the total work performed in that interval. *Joules* is the unit to measure Energy. *Watts* is the unit to measure power.

It is important to differentiate these two concepts, because they play different roles in different use cases. For instance, halving the clock speed of a mobile device's processor

may reduce the dissipated power. However, if this causes executing the workload for as long as twice the original expected processing time, the total energy consumption is the same. In systems where temperature is a major issue, reducing instantaneous power dissipation may be still worthwhile, as it can help reducing temperature.

The power model used in state-of-the-art works assumes two different parts: dynamic (active) power and static (leakage) power. Dynamic power consumption varies with the frequency of the processor. Static power consumption is usually a constant (Awan and Petters, 2013; Chen et al., 2009; Chen and Thiele, 2008, 2011; Jejurikar et al., 2004).

The Dynamic Power Management (DPM) and the Dynamic Voltage and Frequency Scaling (DVFS) are two well-established system-level techniques to adjust the trade-off between the system performance and power consumption during runtime (He and Mueller, 2012a). The following Sections define them.

2.4.1 Dynamic Power Consumption

A modern processor operates at different supply voltages by adopting the dynamic voltage scaling (DVS) technique (Chen and Kuo, 2007; Qu, 2001). DVS is possible due to the advanced technology of VLSI circuit designs. The basic idea of DVS is to slow down the active components by lowering the operating speed and voltage (He and Mueller, 2012a). The P-states define different performance states when the processor core is active and mainly differ in the operating speed/voltage and power consumption (He and Mueller, 2012a).

The dynamic voltage scaling (DVS) technique balances the dynamic energy consumption and the performance of a system. Different supply voltages lead to different execution frequencies (Chen and Thiele, 2011; He and Mueller, 2012a). The dynamic power consumption is usually a convex and increasing function of frequency. Figure 6 illustrates an example of a convex and increasing function. The convex and increasing relation motivates to execute at as low frequency as possible (Chen and Thiele, 2011; He and Mueller, 2012a).

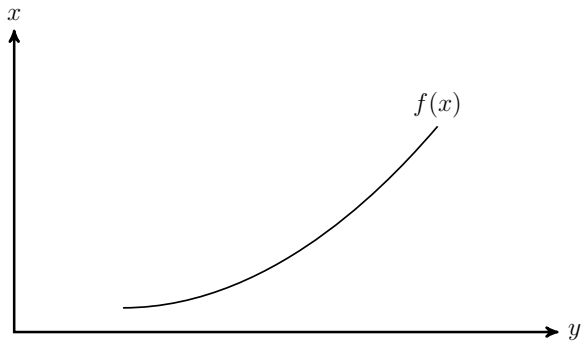


Figure 6 – Example of convex function

Moreover, the overhead in frequency switching is significant. Some worst-case overhead may reach up to hundreds of microseconds in delay because of the changing of the supply voltage of the DC-DC converter or that of the system clock by the phase-locked loop (Zhu and Mueller, 2005).

2.4.2 Static Power Consumption

In general, the main idea behind DPM is to switch the components to a sleep or low power state when they are idle. The switching process from the active state to the sleep state usually consumes both time and energy. Hereby, the mindless switching cannot always ensure the energy saving. The careless switching might even jeopardize the task deadline in a hard real-time system (He and Mueller, 2012a).

One feature often available is multiple sleep states with varying transition cost for entering and leaving target sleep states (Awan and Petters, 2013). The C-states describe the different power states including one active state and multiple low power (sleep) states with different sleep depth (He and Mueller, 2012a). Benini et al. (2000) introduce the break even time concept to capture the required idle time to at least compensate the wasted energy and time during the switching.

The static power consumption has become a non-negligible portion of the overall energy consumption of the system (Awan and Petters, 2013). Unfortunately, in nano-meter manufacturing, leakage current contributes significantly to the static power consumption of the system, while the static power consumption is comparable to the dynamic power dissipation (Chen et al., 2009; Chen and Thiele, 2009, 2011; Jejurikar et al., 2004). For systems with non-negligible leakage power, i.e., the leakage power plays an important role for power consumption when the supply voltage is close to the threshold voltage (Chen and Kuo, 2007).

2.4.3 Energy Aware Real-Time Scheduling and Task Allocation

In what is known as partitioned approach, the actual energy optimization problem of a real-time system on multi-core platforms has three parts: partitioning the tasks, assigning the CPU frequency to the tasks, and developing a scheduling algorithm on each core (He and Mueller, 2012a).

Energy-efficient scheduling for hard real-time tasks on DVS processors aims to minimize the energy consumption. At the same time, all real-time tasks must finish in time (Chen and Kuo, 2007). To minimize the energy consumption of a task for a specified system, we optimize the execution path of the task to reduce the effective switch capacitance (Lee et al., 2003), or slow down the execution speed of the task. Designers may perform the former optimization during the compilation of the task in off-line fashion.

The task scheduler and dispatcher do the latter optimization.

Traditional task assignment algorithms aim to reduce the dynamic power consumption of the system. Therefore, the process of assigning tasks considers the minimization of dynamic power as the major contributor to an optimal distribution. The static power consumption, however, is typically not considered (Awan and Petters, 2013).

For energy-efficient scheduling of periodic real-time tasks, as shown by Aydin et al. (2001b) and Zhu (2006), an optimal solution is to execute at a constant frequency. The utilization is either 100 % or at the minimum/critical frequency with utilization less than 100 %. The critical frequency f^j_{crit} on the Processing Unit (PU) is the available frequency with the minimum energy consumption for execution on the PU (Zhu, 2006).

In general, energy efficiency and timing guarantee are conflicting objectives, i.e., techniques that reduce the energy consumption of the system will usually pay the price of longer execution time, and vice-versa (Chen et al., 2013).

An Inter-Core Preemption of a task τ_i is a preemption of its execution due to core speed change. The arrival or the completion of another task τ_j may cause the core speed change. The condition is τ_i and τ_j to be in the same cluster but on different cores (He and Mueller, 2012a).

2.5 Integer Linear Programming

Linear Programming, LP for short, concerns the problem of maximizing or minimizing a linear functional over a polyhedron. Integer Linear Programming (ILP) investigates linear programming problems in which the variables are restricted to integers ($\mathcal{Z}$). One form of the ILP is: given rational matrix A, and rational vectors b and c, determine $max\{cx|Ax \leq b; x\ is\ integer\}$ (Schrijver, 1998).

2.5.1 Canonical Form for ILPs

An integer linear program in canonical form is expressed as:

$$\text{Minimize } cx \tag{2.8a}$$

$$\text{subject to (s.t.): } Ax \leq b \tag{2.8b}$$

$$x \in \mathcal{Z}^n \tag{2.8c}$$

where the function 2.8a is the objective function, the set of inequalities 2.8b is known as constraints, and the variable x is known as decision variable.

2.5.2 Variations of ILPs

Zero-one linear programming considers only problems in which the decision variable are restricted to be either 0 or 1 ($x \in \{0, 1\}$).

A Linear Relaxation (LR) is a version of an ILP in which the integrality constraint of the decision variable is removed, or relaxed ($1 \leq x \leq 1$).

2.5.3 Exact Algorithms

There are a variety of algorithms that can solve integer linear programs exactly. One class of algorithms are variants of *branch-and-bound* method. A branch-and-bound algorithm is a sophisticated method to solve NP-Hard combinatorial optimization problems, which are based on a mathematical formulation to enumerate promising solutions, avoiding, therefore, the complete generation of all possible solutions for the problem, but still making sure to eventually converge to the optimal solution, or at least the best possible solution.

In a branch-and-bound algorithm, the set of candidate solutions is thought of as tree with the full set at the root. The algorithm explores branches of the tree. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm.

A *branch-and-cut* is a branch-and-bound with cut generation strategies applied to tighten the linear programming relaxations. The use of cutting planes to solve ILPs was introduced by Ralph E. Gomory, being the Gomory's cut a common cutting plane strategy. Gomory's cut method consists of adding linear constraints to the original ILP based on the solution of a linear relaxation of the ILP. The linear constraints are constructed based on the fractional parts of the linear relaxation solution. The new ILP is solved iteratively until an integer solution is found.

A *branch-and-price* method is a hybrid of branch-and-bound and column generation methods. In a branch-and-price, at each node of the search tree, columns may be added to the linear relaxation. The algorithm starts from a reformulation of the original ILP known as Master Problem. The reformulation may be found using methods such as the Dantzig-Wolfe decomposition. A Restricted Master Problem is a subset of the Master Problem containing a smaller number of columns to be solved. To check for optimality, a subproblem, known as Pricing Problem, is solved to determine which columns enter the basis to reduce the objective function. If cutting planes are used to tighten the linear relaxation, the method is known as branch-price-cut.

2.5.4 Heuristic Methods

Given that integer linear programming is applied in many NP-hard problems, several problem instances are intractable. Therefore, heuristics are an alternative typically less expensive computationally.

A metaheuristic is a procedure or heuristic designed to find good enough solutions to an optimization problem. Several techniques exist and typically mimic natural processes, such as swarm algorithms, bee colonies, ant colonies, hill climbing, simulated annealing, and evolutionary algorithms. Evolutionary algorithm is a generic population based metaheuristic that is inspired by biological evolution, such as reproduction, mutation, and selection.

An algorithm has an approximation constant β if the objective function of its derived feasible solution is at most β times of the optimal objective for any input instance (Chen and Thiele, 2009, 2011).

A fully polynomial-time approximation scheme (FPTAS) is an $(1+\epsilon)$-approximation algorithm with polynomial-time complexity by treating $\frac{1}{\epsilon}$ as input (Chen and Thiele, 2008). FPTAS algorithms always answers an $(1+\epsilon)$-approximation in polynomial-time.

2.6 Chapter Summary

In this chapter, we have presented concepts related to real-time systems, schedulability analysis, multi-core systems, power management, and integer linear programming. We use all these different concepts in the next chapters.

Real-time systems are highly dependent on delivering correct results at the right time. We are particularly interested in hard real-time systems, because missing a deadline in hard real-time systems leads to catastrophic situations. We have also reviewed the most used algorithms to determine if a real-time system is scheduled or not, by means of utilization based and response time based schedulability analyses.

The modern processing model is typically parallel and heterogeneous in nature. We have revised how the DVFS power management technique is applied on multicore systems, classifying them into full chip, per core, and cluster based platforms. We also considered the major strategies to reduce processor dynamic and static power consumptions.

Lastly, we have covered a brief overview of integer linear programming. We have reviewed the canonical form of an ILP and we have discussed about computational techniques of ILP resolution, including exact, heuristic based, and approximation algorithms.

3 Distribution of Independent Hard Real-time Tasks Among Heterogeneous Cores

A classical mathematical model that resembles modern heterogeneous multicore platforms is the Multilevel Generalized Assignment Problem (MGAP), even though it was originally conceived in the manufacturing context. The MGAP consists of minimizing the assignment cost of a set of jobs to machines, each having associated therewith a capacity constraint. Each machine can perform a job with different performance states that entail different costs and amount of resources required. The MGAP is originally in the context of large manufacturing systems as a more general variant of the well-known Generalized Assignment Problem (GAP) (Glover et al., 1979). In this chapter, we correlate MGAP model with the problem of assigning frequencies and distributing hard real-time tasks on heterogeneous processors minimizing energy consumption.

Modern processors may be seen as machines with several performance states due to Dynamic Voltage and Frequency Scaling (DVFS) technique. DVFS is a well established power reduction strategy and it has already been a research topic for decades. The premises are the variation of processors' workloads and the quadratic relationship between energy consumption and voltage (Burd and Brodersen, 1996). The energy consumption depends on dynamic and idle energy (Venkatachalam and Franz, 2005): $E_{system} = E_{dyn} + E_{idle}$, where E_{idle} is the energy consumption while the system is idle and accounts for leakage, E_{dyn} is the energy consumption in active use cases. The dynamic energy consumption E_{dyn} is estimated using: $E_{dyn} = C_l \times N_{cycle} \times V_{dd}^2$, where E is energy, C_l is circuitry capacitance, N_{cycle} is number of cycles, and V_{dd} is the voltage. Although DVFS yields meaningful energy consumption reduction, its usage requires care, especially when considering timing constraints.

The implementation of DVFS-capable chips creates three types of platforms: full-chip, per-core, and cluster-based (He and Mueller, 2012a). The categories differ depending on the Dynamic Phased Lock Loop (DPLL) network across the circuit and on the voltage delivery distribution. In full-chip platforms, the design allows changing the clock, and voltage, of all cores at once. The per-core platform, in contrast, implements a DPLL network to achieve frequency (and voltage) manipulation granularity on each individual core. Cluster-based architecture is a generalization of full-chip and per-core platforms. In cluster-based platforms, clusters group the cores, where each cluster acts as a full-chip platform. But clusters are independent of each other having their own clock and voltage network. The cluster-based platforms allow changing clock and voltage of each cluster independently, but the change affects all cores within the cluster. In this chapter, we are

only considering per-core multicore platforms.

Therefore, the problem we are addressing in this chapter is: *how to find optimal hard real-time tasks distribution among heterogeneous processors respecting timing constraints and targeting low power consumption?* The contributions of this work are: (i) comprehensive and representative mathematical formulations that (ii) accounts characteristics of different hard real-time scheduling policies and that (iii) delivers optimal hard real-time task allocation and optimal frequency to task assignment, (iv) with system energy consumption minimization, but still (v) using the advantage of a classical combinatorial optimization model: MGAP. The results we present on this research question focus on solving the problem optimally on practical instances sizes. Effective methods support reducing power bills, improve system reliability, and increase the efficient usage of energy; last but not least, assisting to reduce environmental impacts.

The organization of this chapter is as follows. The processor model and task model are defined in Section 3.1. Formulations for different scheduling policies and a model growth analysis are discussed in Section 3.2. We describe the implementation of solvers and of a evolutionary algorithm in Section 3.3. Computational experiments are detailed in Section 3.4. We compare our results with existing literature in Section 3.5. Section 3.6 closes this chapter with final comments.

3.1 System Models

In this section we present the system models. Section 3.1.1 describes the processor model we consider. We describe the real-time task model in Section 3.1.2.

3.1.1 Processor Model

The processor model resembles a Multi-Processor System-On-Chip (MPSoC) architecture, such as Exynos 5 Octa (Samsung Electronics Co.Ltd., 2014). The system is composed by a set, $\mathcal{H}$, of m processors, $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$. Each core may operate on l different performance states, $1 \leq k \leq l$. The set of frequencies of one core is not necessarily the same of other cores. Also, a task may have different code size and execution time for different processors, due to instruction set differences. The frequency of performance state k on the processor i is F_{ik} and the power consumption is P_{ik}. The idle power of processor i is $P_{idle,i}$.

Our proposal can be used with no extra effort on other architectures. Even though we focus on per-core heterogeneous platforms in our experiments, we have exercised on multiple heterogeneous clusters (Valentin et al., 2015b). The models discussed here may be applicable to full-chip and cluster-based platforms as long as the intrinsic architec-

tural interference is accounted and we recommend the interested reader to consider more sophisticated schedulability tests (Valentin et al., 2015b).

3.1.2 Task Model

In the remaining sections of this chapter we adopt the following notation. A task model $\mathcal{M}$ is a set composed by n tasks τ_j. A task $\tau_j \in \mathcal{M}$, with $j \leq n$, has the properties: worst-case execution cycle $WCEC_j$; worst-case execution time $C_j(f)$, which is a function of frequency f, thus $C_j(f) = \frac{WCEC_j}{f}$; period of execution T_j; deadline D_j, we consider scheduling policies in which $D_j = T_j$. A task τ_j also has the following properties, specific to fixed priority policies: fixed priority p_j; set of high priority tasks $hp(j)$ representing the tasks τ_p with priority higher than the priority of τ_j. The response time R_j is dependent not only on task set characteristics, but also on the target platform, and on the task allocation and frequency distribution that have been selected for the workload. Switching frequency and voltage has an intrinsic required overhead Lp which, in per-core platforms, may be accounted as an addition in the execution time C_i of each task. The DVFS overhead Lp has to be accounted twice, one for entry another for exit (He et al., 2012; Valentin et al., 2015b).

3.2 Mathematical Formulations for Different Scheduling Policies

In the following, we present the proposed MGAP based formulations, applicable and refined for Earliest Deadline First (EDF) (Liu and Layland, 1973) and Rate Monotonic (RM) (Lehoczky et al., 1989; Liu and Layland, 1973) scheduling policies. The MGAP based formulations are more realistic to this problem because they represent the available set of frequencies per processor (Valentin et al., 2016b). However, representing the constraints of scheduling policies requires modifying and extending such formulations.

3.2.1 Theoretical Basis and Reference: The MGAP Model

Considering the problem characteristics and the set of frequencies of each processor as machine performance states, we propose to use the MGAP integer programming mathematical formulation. The classical MGAP formulation is in Equation 3.1.

$$\text{Minimize} \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l} c_{ijk} x_{ijk} \right\} \tag{3.1a}$$

$$\text{s.t.:} \sum_{i=1}^{m} \sum_{k=1}^{l} x_{ijk} = 1, \; j \in \{1, \ldots, n\} \tag{3.1b}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{l} a_{ijk}x_{ijk} \leq \theta_i,\ i \in \{1,\ldots,m\} \tag{3.1c}$$

$$x_{ijk} \in \{0,1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{3.1d}$$

where the tri-dimensional decision variable x_{ijk} represents the distribution and assignment, i.e. when $x_{ijk} = 1$ the task τ_j executes in the processor i at performance state k, or frequency F_{ik}, when $x_{ijk} = 0$, the task τ_j is distributed somewhere else. A distribution is a partitioned approach in which each processor i executes a local scheduler responsible for a partition of the real-time task workload and migration is not allowed. The objective function 3.1a minimizes the system energy consumed by processors. The tri-dimensional matrix c_{ijk} represents the energy consumed by task τ_j while executing on processor i at performance state k. The set of constraints 3.1b models the allocation of task τ_j to a single processor. The set of limits 3.1c represents each processor utilization constraint. The tri-dimensional matrix a_{ijk} represents utilization used by task τ_j while executing on processor i at performance state k. The variable θ_i[1] represents the maximum possible total utilization of processor i, according to the scheduling policy used. We present in the following sections different bounds for θ_i. We note that $a_{ijk'} < a_{ijk''} \iff c_{ijk'} > c_{ijk''}$.

The above problem can be described using the three field notation for theoretic scheduling problems $\alpha|\beta|\gamma$ (Brucker, 2010). The machines environment is unrelated parallel machines ($\alpha = R$) because the matrix a_{ijk} depends on the task and machine relation. The job characteristics (β) are deadline (D_j) and preemption ($pmnt$). Also, the jobs have arbitrary execution time (see C_j in Section 3.1.2). The optimally criteria (γ) is unspecified because we minimize the overall energy consumption. Thus, the scheduling theory notation is $R|Dj; pmnt|\sum f_i$.

3.2.2 Estimating System Energy in the Objective Function

In the following formulations, unless specified, we are using an objective function that minimizes energy consumption, accounting dynamic and idle energy, over the time window represented by the hyperperiod of the real-time tasks, i.e., the Least Common Multiple (LCM) of tasks periods. We extend the objective functions presented by Valentin et al. (2016b) by improving the idle energy estimation. Equation 3.2 has the objective function.

[1] In the classical MGAP formulation, θ_i is known as b_i. We decided to rename it to avoid confusion with real-time task blocking time B_i.

$$\text{Minimize } \Psi(x) = \sum_{i=1}^{m} \left(E_{dyn,i}(x) + E_{idle,i}(x)\right) \tag{3.2a}$$

$$E_{dyn,i}(x) = \sum_{j=1}^{n} \sum_{k=1}^{l} \left(\left(\frac{LCM}{T_j} \right) C_l WCEC_{ij} V_{dd,ik}^2 x_{ijk} \right) \tag{3.2b}$$

$$E_{idle,i}(x) = P_{idle,i} LCM \left(1 - \sum_{j=1}^{n} \sum_{k=1}^{l} \frac{WCEC_{i,j}}{F_{ik} T_j} x_{ijk} \right) \tag{3.2c}$$

where $E_{dyn,i}$ is the energy consumption when processor i is active, $E_{idle,i}$ is the energy consumption when processor i is idle, $\frac{WCEC_{ij}}{F_{ik}T_j}$ represents the task τ_j utilization, u_{ijk}, while executing in processor i at frequency F_{ik} of performance state k, C_l is the circuit capacitance constant, and $V_{dd,ik}$ is the voltage level to achieve frequency F_{ik}.

The term $\left(\frac{LCM}{T_j}\right) C_l WCEC_{ij} V_{dd,ik}^2 x_{ijk}$ represents the dynamic energy associated with the instances of execution of task j within the LCM. Each processor idle energy, within the LCM time window, is computed for its estimated idle time in the term $P_{idle,i} LCM \left(1 - \sum_{j=1}^{n} \sum_{k=1}^{l} \frac{WCEC_{i,j}}{F_{ik}T_j} x_{ijk}\right)$.

The objective function represented in Equation 3.2 may still be seen as a MGAP formulation (Valentin et al., 2017). Note that, without loss of generality, when we take the term $P_{idle,i} LCM$ out of the sum, leaving the term $m P_{idle,i} LCM$ to be added to the final objective function value, we have $c_{ijk} = \left[\left(\frac{LCM}{T_j}\right) C_l WCEC_{ij} V_{dd,ik}^2 - P_{idle,i} LCM \left(\frac{WCEC_{i,j}}{F_{ik}T_j}\right)\right]$.

3.2.3 Models for EDF

EDF is a dynamic priority based on-line scheduler in which earliest deadlines are first scheduled. Liu and Layland (1973) propose a utilization based schedulability test for uniprocessor systems. The test is: $\sum_{j=1}^{n} C_j(f)/T_j \leq 1$. We are considering implicit deadlines for EDF ($D_j = T_j$), in the same way as the original Liu and Layland work.

Utilization based schedulability analyses are fast approximate tests that use the information of the task set total utilization. For EDF, with implicit deadlines and no jitters, the analyses are *sufficient* and *necessary* conditions. The utilization based schedulability analysis for RM, however, is only a *sufficient* condition, i.e the task set is schedulable if it satisfies the test, but if the task set fails to satisfy the test, it may still be schedulable.

The EDF scheduler is known to be an optimal online scheduler that can achieve 100% of CPU utilization for independent real-time tasks. Taking advantage of the EDF's utilization bound is common practice on the specialized literature (Alahmad and Gopalakrishnan, 2011; Awan and Petters, 2013; Chen et al., 2011; Chen and Thiele, 2011; Goossens

et al., 2008; He and Mueller, 2012a,b; Prescilla and Selvakumar, 2013; Valentin and Barreto, 2010; Yang et al., 2009; Yu and Prasanna, 2003). We present mathematical formulations for real-time task allocation taking into account the utilization bound for EDF as follows.

3.2.3.1 MGAP Formulation with Utilization Bound for EDF

Valentin et al. (2016b) propose to use the Multi-level Generalized Assignment Problem (MGAP) as a more suitable formulation for representing the hard real-time task assignment problem. Equation 3.3 (VAL1), similarly to Valentin et al. (2016b), has a MGAP formulation for EDF, with extensions in the objective function.

$$\text{Minimize } \Psi(x) \tag{3.3a}$$

$$\text{s.t.: } \sum_{i=1}^{m}\sum_{k=1}^{l} x_{ijk} = 1,\ j \in \{1,\ldots,n\} \tag{3.3b}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{l} \frac{WCEC_{ij}}{F_{ik}T_j} x_{ijk} \leq 1,\ i \in \{1,\ldots,m\} \tag{3.3c}$$

$$x_{ijk} \in \{0,1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{3.3d}$$

3.2.3.2 Barrefor's Formulation with Utilization and Frequency Bound for EDF

Barrefors et al. (2014) use the EDF's utilization bound in an integer programming mathematical formulation to distribute hard real-time tasks on heterogeneous processors. Their formulation expects as valid CPU frequencies only those whose power consumption can be sustained for long period of time without the need to throttle due to temperature constraints. Equation 3.4 (BARREFORS) lists their formulation.

$$\text{Minimize } \left\{ P \sum_{i=1}^{m} \Phi_i(\tilde{f}_i^{max}) x_i \right\} \tag{3.4a}$$

$$\text{s.t.: } \sum_{i=1}^{m} \alpha_i \tilde{f}_i^{max} x_i \geq U_{tot} \tag{3.4b}$$

$$x_i \in \{0,1\}, 1 \leq i \leq m \tag{3.4c}$$

where the decision variable x_i determines if processor i is turned on and in use ($x_i = 1$) or not ($x_i = 0$), α_i is the performance coefficient of processor i, m is the number of processors, $\tilde{f}_i^{max}$ is processor i's maximum frequency satisfying the temperature constraint, and $\Phi_i(f)$ is the power consumption of processor i when executed at a frequency f. BARREFORS' formulation determines only which processors to turn on to process the tasks. In a second phase, BARREFORS' approach distributes the tasks and determines which frequency each processor executes based on a worst-fit decreasing algorithm (Barrefors et al., 2014).

3.2.3.3 MGAP Formulation with Utilization and Frequency Bound for EDF

Although the MGAP formulation for EDF expressed in Equation 3.3 already covers most problem characteristics, results on specialized literature suggest that on an optimal allocation, each CPU executes at a fixed frequency during the entire system lifetime (Aydin et al., 2001b). However, the state-of-the-art results assume ideal CPUs which can be programmed essentially to any frequency from a continuous frequency domain, even though practical and modern CPUs still use a discrete and finite set of frequencies. Therefore, we propose extending Equation 3.3 to still accommodate the single frequency assumption, but considering a discrete set of frequencies. We accomplish this constraint by establishing a minimum bound on the average of CPU frequencies assigned to each task, because on MGAP based formulations, each task may receive a different frequency (Valentin et al., 2017). Equation 3.5 (VAL2) describes the formulation.

$$\text{Minimize } \Psi(x) \tag{3.5a}$$

$$\text{s.t.: } \sum_{i=1}^{m} \sum_{k=1}^{l} x_{ijk} = 1,\ j \in \{1, \ldots, n\} \tag{3.5b}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{l} \frac{WCEC_{ij}}{F_{ik} T_j} x_{ijk} \leq 1,\ i \in \{1, \ldots, m\} \tag{3.5c}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{l} \frac{F_{ik}}{n^*} x_{ijk} \leq \tilde{f}_i^{max} \sum_{j=1}^{n} \sum_{k=1}^{l} \frac{WCEC_{ij}}{F_{ik} T_j} x_{ijk},\ i \in \{1, \ldots, m\} \tag{3.5d}$$

$$x_{ijk} \in \{0, 1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{3.5e}$$

where $\tilde{f}_i^{max}$ is processor i's maximum frequency satisfying the temperature constraint and n_i^* is the number of tasks assigned to processor i for a given allocation configuration. According to Aydin et al. (2001b), on a system with normalized continuous frequencies

available (from 0 to 1, being 1 the maximum frequency), the optimal CPU frequency assignment is equal to its maximum allowed frequency multiplied by the total workload utilization. The Constraint 3.5d maps this finding from the state-of-the-art by computing the average frequency on the assigned workload and restricting it to the total assigned workload utilization multiplied by the CPU maximum frequency.

3.2.4 MGAP Formulation with Utilization Bound for RM

RM is a fixed priority based on-line scheduler in which task priorities decrease with larger periods. We also assume implicit deadlines ($T_j = D_j$) for RM. Liu and Layland test for n tasks for RM is $\sum_{j=1}^{n} C_j(f)/T_j \leq n(2^{\frac{1}{n}} - 1)$. Therefore, we propose the MGAP formulation for RM (VAL3) listed in Equation 3.6 by extending Valentin et al. (2016b).

$$\text{Minimize } \Psi(x) \tag{3.6a}$$

$$\text{s.t.: } \sum_{i=1}^{m} \sum_{k=1}^{l} x_{ijk} = 1,\ j \in \{1, \ldots, n\} \tag{3.6b}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{l} \frac{WCEC_{ij}}{F_{ik} T_j} x_{ijk} \leq n_i^* (2^{\frac{1}{n_i^*}} - 1),\ i \in \{1, \ldots, m\} \tag{3.6c}$$

$$x_{ijk} \in \{0, 1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{3.6d}$$

where n_i^* is the number of tasks assigned to processor i for a given allocation.

3.2.5 MGAP Formulation with Response Time Bound

Lehoczky et al. (1989) present an exact schedulability analysis based on tasks periods and priorities: $R_j \leq D_j, \forall\ 1 \leq j \leq n$, where R_j is computed using the iterative equation $R_j^{n+1} = C_j + \sum_{p \in hp(j)} \left\lceil \frac{R_j^n}{T_p} \right\rceil \times C_p$. Response time are computationally expensive but provide *exact* conditions, i.e., *sufficient* and *necessary*. The test uses task's WCEC, periods, and the concept of critical instant phasing (Lehoczky et al., 1989).

Considering the schedulability test proposed by Lehoczky, we propose the MGAP formulation using tasks response times (VAL4) as seen in Equation 3.7, based on the formulation of Valentin et al. (2016b). This formulation applies each task deadline as a constraint against their response time in the linear programming. The response time of each task vary depending on the workload distribution and the frequency assignment of the configuration because a change on the value of x_{ijk} may result on a different computation time (C_i).

$$\text{Minimize } \Psi(x) \tag{3.7a}$$

$$\text{s.t.: } \sum_{i=1}^{m} \sum_{k=1}^{l} x_{ijk} = 1,\ j \in \{1, \ldots, n\} \tag{3.7b}$$

$$R_j^* \leq D_j,\ j \in \{1, \ldots, n\} \tag{3.7c}$$

$$x_{ijk} \in \{0, 1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{3.7d}$$

where R_j^* is the response time of tasks τ_j for a given allocation configuration. Equation 3.7 is applicable for RM scheduling policy ($D_j = T_j$).

3.2.6 Analysis on Formulations

The MGAP based mathematical formulations are similar in the sense of growth. They have a tri-indexed decision variable. Also, the number of variables increases with the number of processors, number of tasks, and the number of performance states. They share similar growth of number of constraints, which increases with the number of processors and tasks, with the exception of VAL4, whose constrains increase quadratically with the number of tasks. Also, among the MGAP based formulations, the VAL2 has an additional set of constraints. Table 3 summarizes characteristics of these formulations.

Table 3 – Analysis of mathematical formulations for this problem

Formulation	Number of Constraints	Number of Variables	Types of Variables	Type
MGAP	$O(m \times n)$	$O(m \times n \times l)$	tri	MILP
VAL1	$O(m \times n)$	$O(m \times n \times l)$	tri	MILP
VAL2	$O(m \times n)$	$O(m \times n \times l)$	tri	MILP
VAL3	$O(m \times n)$	$O(m \times n \times l)$	tri	NILP
VAL4	$O(n^2)$	$O(m \times n \times l)$	tri	NILP

The BARREFORS formulation is simpler when compared to MGAP formulations. BARREFORS has a uni-dimensional decision variable that grows only with the number of processors. Similarly, the number of constraints grows with the number of processors.

The constraints of VAL4 and VAL3, however, are harder to implement. Each θ_i in these formulations is dependent on the decision variable, i.e. on the assignment and task distribution. Because the exponents of the decision variables are different than one, these are Non-Linear Programming (NILP) formulations. MGAP, VAL1, BARREFORS, and VAL2 are Mixed Integer Linear Programming (MILP) formulations.

3.3 Computational Techniques of Resolution

In this section, we explain the algorithmic strategy developed for each mathematical formulation of Section 3.2. In Section 3.3.1, we explain an evolutionary algorithm which produces an initial solution that can be used by the exact algorithm for finding optimal solutions, described in Section 3.3.2.

3.3.1 Approximation by means of Evolutionary Algorithm (EA)

We wrote an evolutionary algorithm (EA), based on genetic algorithm, for each mathematical model (Valentin, 2009). We follow a similar approach as existing in the literature for other formulations on this problem (Goossens et al., 2008). The algorithm's input is the processing model $\mathcal{H}$ and the desired task model $\mathcal{M}$ (see Section 3.1). In our EA implementation, a solution is a chromosome that is a sequence of 0's and 1's and each gene represents one of the elements of the tri-dimensional decision variable of the mathematical model. The algorithm can be simplified into two steps: (i) *Initialization* with random-generated individuals and (ii) *Generations* composed by individuals selected in tournaments and by the evolutionary operators of elitism and crossover. Algorithm 1. illustrates the overall process of our EA strategy and we describe the pieces of the EA as follows.

In the *Initialization*, we random-generate individuals. Random-generating individuals does not guarantee their feasibility, i.e. the generated individual may be infeasible. The process of validating or transforming individuals into feasible solution is onerous. Even then, we maintain all generations composed by feasible individuals only. We random-generate a large number of individuals, 5000, to start with a high diversity. If none of them is a feasible solution, we return the empty set $\emptyset$. If we find less than 50 feasible individuals, then we return the one with highest fitness. But when we find 50 feasible individuals, we repeat the following steps for a maximum of 100 generations, or 10 generations with same best fitness, and return the individual with best fitness. We perform the *Elitism* operator by always including the individual with best fitness in the next generation. We execute *Selection* by means of a tournament in the current population. Only 5 individuals, randomly selected, participate in the tournament. The winner of the tournament is the individual with best fitness among those participating of it. We also insert in the next generation the result of a *Crossover* between winners of two tournaments. The crossover operation between individuals I_1 and I_2 is done by means of selecting a pivot gene p. The genes lower than p are copied from I_1, the remaining genes are copied from I_2. When resulting individual is not feasible, we return I_1, if $fitness(I_1) > fitness(I_2)$, or I_2 otherwise. We define the *Fitness function* to be: $1/E(individual)$, where the function $E(individual)$ is the estimated energy consumption for the individual in consideration. The function E is computed using the same energy estimation as in the objective functions of the integer

programming mathematical formulations (see Section 3.2.2).

3.3.2 Finding Optimal Solutions

We use a general branch-and-cut method combined with schedulability tests to conduct the process of finding optimal solutions. A branch-and-cut is a branch-and-bound with cut generation strategies. The algorithm's input is the processing model $\mathcal{H}$, the desired task model $\mathcal{M}$, and a possible upper bound ub, with objective function value and the solution structure found by the EA. The algorithm outputs the optimal distribution of hard real-time tasks among the processors that consumes less power among the possible assignments, informing as well in which frequency each tasks may be executed, and the total system estimated energy. The general solving strategy is listed in Algorithm 2.

The algorithm starts by denoting the set L of active problem nodes to contain only the initial Integer Linear Problem. When the EA returns a feasible solution, the upper bound v^* and the optimal solution x^* are set to match the output of the EA, otherwise they are set to $+\inf$ and to $NULL$, respectively. The algorithm iteratively evaluates each element of the set L. Each problem node is initially tested against the schedulability test that fits for the problem scheduling policy. In the case the schedulability test accepts the node, then a regular branch-and-cut is followed. The linear relaxation of the node is then computed and solved. When the linear relaxation is feasible, a procedure of generation of cutting planes is performed and followed by a fathoming and pruning process. The problem node is then partitioned and new restricted problem nodes are derived and incorporated into L. The iterative process repeats until the set L is empty.

3.4 Computational Experience

In this section we detail the computational experiments and analyse the generated results. The environment description is outlined in Section 3.4.1. Section 3.4.2 describes the workload used and the target platform considered in these experiments. Section 3.4.3 analyses the parameters of the EA. After that, we present two main scenarios of experiments. We first experiment in Section 3.4.4 the comparison between EDF and RM based formulations. In Section 3.4.5, we present further computational experiments targeting only EDF based models because EDF is rather commonly used in the literature.

3.4.1 Experiment Environment

The aim of this computational experiment is to evaluate the model in terms of performance, objective function, and execution time. The performance of each solver is determined by the amount of valid solutions it is able to find for a given input. The evaluation of solver's objective function aims to understand the ability to reduce energy

Algorithm 1 Evolutionary Algorithm (EA) for Independent Tasks

```
1: Input: H, M
2: Output: best feasible solution ub found
3: /* Random-generate 5000 individuals
4: * (feasible and infeasible),
5: * to start with high diversity.
6: */
7: i ← initialization(5000);
8: /* Select 50 feasible individuals. */
9: p ← feasible(i, 50);
10: if |p|= 0 then
11:    return ∅;
12: end if
13: b ← prev ← best_individual(p);
14: if |p|< 50 then
15:    return b;
16: end if
17: g ← e ← 1;
18: while (g++ ≤ 100)and(e ≤ 10) do
19:    /* Evolve population. */
20:    tournament(p);
21:    selection(p);
22:    elitism(p);
23:    ub ← best_individual(p);
24:    if ub == prev then
25:       e++;
26:    end if
27:    prev ← ub;
28: end while
29: return (ub.structure, ub.val);
```

Algorithm 2 Branch-and-Cut (B&C) for Independent Tasks

```
1: Input: H, M, ub
2: Output: optimal solution (x*, v*)
3: v* ← ub.val; x* ← ub.structure;
4: L ← ILP^0;
5: while L != ∅ do
6:    n ← remove_node(L);
7:    if !schedulability(n) then
8:       continue;
9:    end if
10:   lp ← relaxation(n)
11:   (x, v) ← solve(lp);
12:   if x = infeasible then
13:      continue;
14:   end if
15:   p ← cut_planes(x, v);
16:   if p! = ∅ then
17:      add(p, lp);
18:      goto 7;
19:   end if
20:   if v ≥ v* then
21:      continue;
22:   end if
23:   if x is integer then
24:      v* ← v;x* ← x;
25:      continue;
26:   end if
27:   partition(lp);
28: end while
29: return (x*, v*);
```

consumption for a given input. Finally, the execution time is measured to compare the impact on project design phase.

We estimate the execution time of each solver using system's Real-Time Clock (RTC). The machine used to perform the simulation experiments has an AMD FX ™-9370, 8 cores executed at 1.4 GHz. We use Debian GNU/Linux 8.2 with 32 GB of DDR3 memory executed at 1.33 GHz.

We wrote an implementation of the previous models using C++ (Valentin, 2009). The generic branch-and-cut method is implemented in CPLEX Concert (IBM, 2016), version 12.3, and we conduct their branch-and-bound strategy strictly so we have the opportunity to perform the schedulability tests. In the implementation, we configure Concert to sustain reproducibility. We have limited the amount of threads to one (`IloCplex ::`

`Threads = 1`). The workable memory is set to 1 GB (`IloCplex :: WorkMem = 1024`). The solver tree is restricted to 2 GB (`IloCplex :: TreLim = 2048`). We also configured Concert to be deterministic (`IloCplex :: Param :: Parallel = 1`).

3.4.2 Workload and target platform considerations

We use random-generated task models. We vary number of tasks in each random-generated models between 10 and 90. We vary the total system utilization between 10% and 90%. The **total system utilization** is estimated using the highest frequency that does not cause a thermal issue, i.e. **using the same strategy adopted by BARREFORS** (Barrefors et al., 2014), but **final total system utilization** is determined by the solution given by the solver. A task may have different amount of cycles for each processor, representing their difference in instruction set. We chose to have tasks periods uniformly distributed between three sets: large period, medium period, and short period. When a task has a large period, its period is chosen from the set $\{100, 250, 750\}$ milliseconds. Similarly, a medium period is selected from the set $\{5, 10, 50\}$ milliseconds, and a short period is selected from the set $\{100, 250, 750\}$ microseconds.

Our experiment considers a target platform of four processors. The target platform is similar to Exynos platform (Samsung Electronics Co.Ltd., 2014). There are two high performance processors that can operate at frequencies from 600 MHz to 1.7 GHz. Also, there are two low power consumption processors that can operate at frequencies from 200 MHz to 1.3 GHz. The idle power consumption is 260 mW. The DVFS switching overhead Lp is 3000 cycles. The energy consumption is estimated for the duration of the LCM of tasks' periods, as in the objective function of each model.

The data of all executions on each instance is available for download[2] in a repository and we present the summary of them in the following sections.

3.4.3 Analysis on EA Parameters

We have tuned the EA algorithm based on an analysis of five of its parameters: number of generations, size of population, number of individuals in the tournament, the use of elitism, and percentage of mutation. We considered the CPU time needed to solve an instance with 30 tasks and 50 % of estimated target CPU utilization. In Figure 7 we present some graphics in which the left column shows the average CPU time and in the right column we present the average solution energy consumption, for each analysed EA parameter. We plot only observations that could be collected within an execution of less than one minute of CPU time.

[2] Available upon request to authors.

As we can observe in Figure 7, as expected, the execution time of the EA increases with the number of generations used, but we have noticed almost no change in the energy consumption. Similar pattern is seen for the number of individuals participating in the tournaments. We see an improvement in the energy consumption when the size of the population is higher than 20, but increasing the size of the population also increases the EA execution time. We have decided to set the parameters *population* and *generation* to 50 and the parameter *tournament* to 5, to avoid increasing the EA execution time, but still finding solutions with lower energy consumption. We have noticed that when we enable *mutation*, specially with a rate higher than 7 %, the execution time of the EA increases considerably, reaching more than 1 min in this analysis, and therefore, we decided to disable *mutation*. We have not noticed any major difference in the convergence time when enabling or disabling *elitism* for this particular analysis, but we decided to keep it enabled to avoid loosing promising solutions found across generations.

3.4.4 Experiment with Different Scheduling Policies

In this section we compare the models VAL1, VAL3, and VAL4. We explain our results of combining the evolutionary algorithm with solvers to reach optimal solutions. In this section, we vary number of tasks in each random-generated models between $5, 10, 15$, and 20. We vary the total system utilization between $10\%, 20\%, 30\%, 40\%$, and 50%, of total system utilization if all processors are kept at the maximum speed, to achieve a fair comparison between all scheduling policies, avoiding crossing their theoretical limits.

The search for optimal solutions using the solvers of each model found, in some cases, only feasible, but not optimal, solutions. In several cases, we reached the time limit of 30 min. We can also confirm this observation by looking at the difference between the lower and upper bounds at the end of execution of each solver (GAP). Table 4 lists the average of the executions of each combination of number tasks and total system utilization (U_{total}). We call special attention to the VAL3 results in these experiments. For example, for 20 tasks and 40 % of total system utilization, the average GAP was 40.36 %, and the VAL3 solver always reached the time limit of 30 min.

We, then, executed the Evolutionary Algorithm (EA) on every instance with feasible solutions. The EA produces a feasible solution, typically non-optimal. However, the solution provisioned by the EA can be used as an upper bound to solvers in the search for optimal solutions. Therefore, **to boost each solver's resolution process, we start them considering not only an upper bound for the objective function but also the solution structure based on the best feasible solution found by the EAs**.

The results of boosting VAL4 are listed in Table 5, which groups the execution of each instance per number of tasks and total system utilization. Using the EA's solution structure and upper bound in the VAL4 solver, reduced, for example, the execution time

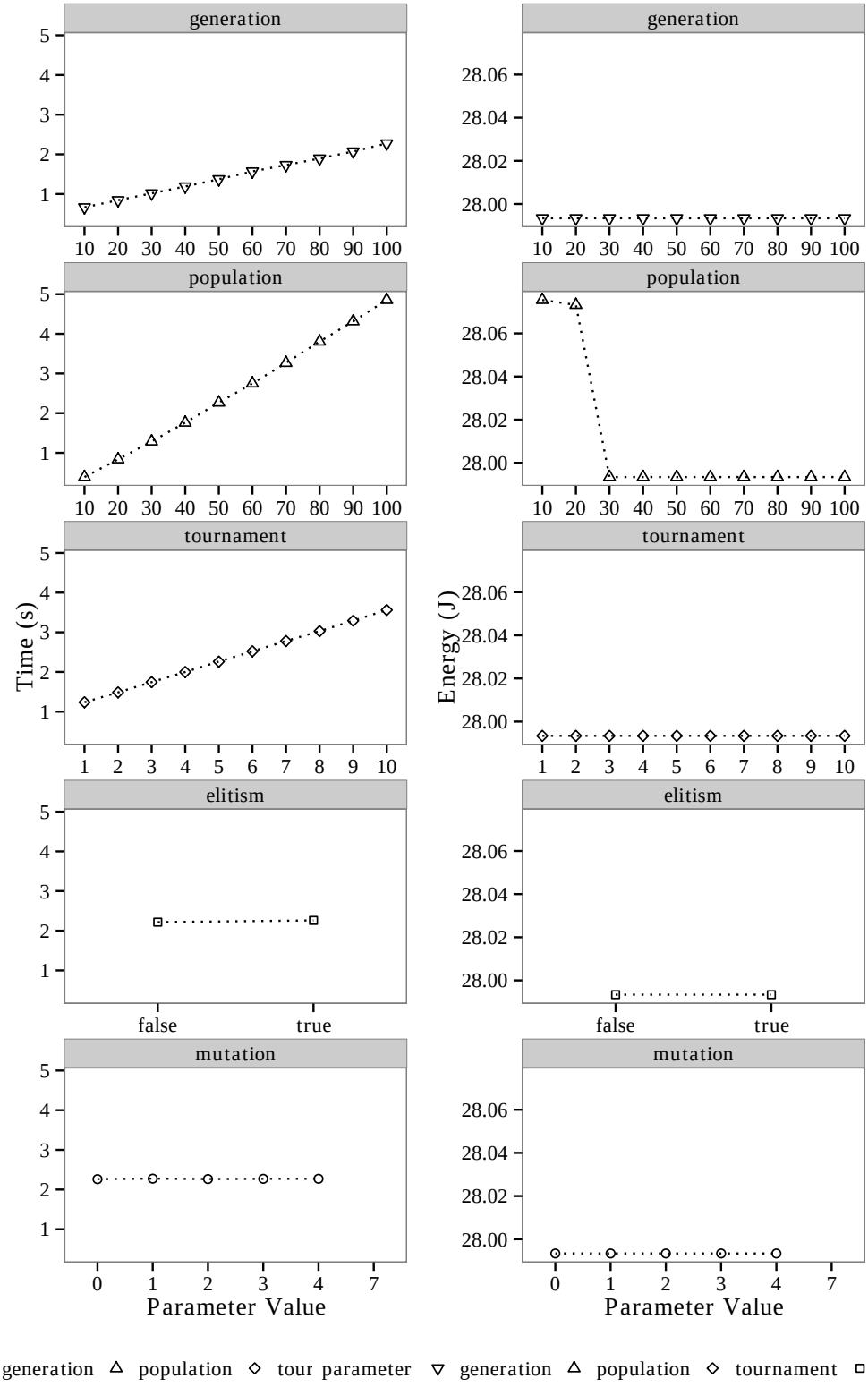


Figure 7 – Analysis of the influence of EA parameters on the EA execution time and on the quality of the objective function (Energy). Parameters: number of generations (*generation*), size of population (*population*), number of individuals in the tournament (*tournament*), the use of elitism (*elitism*), and percentage of mutation (*mutation*).

from 30 min to less than 5 min in the case of 20 tasks and 50 % of total system utilization, while still reducing the total system energy consumption.

For these instances, we observed a higher improvement while applying the EA's solution structure and upper bound to the solver VAL3. Table 6 lists the results for the boosted VAL3 using the EA's solution. In Table 6, we also group the execution of each instance per number of tasks and total system utilization, and present the average on each column. In all cases in which the GAP for VAL3 was significant, using the EA's solution structure and upper bound made the VAL3 solver to reach a GAP of either 0.00 or 0.01. Also, taking into account both the EA's solution structure and the upper bound, they reduced the gap from 40.36 % to 31.07 % considering 20 tasks and 40 % of total system utilization, in this case, reducing the total system energy.

The improvement for VAL1 is less apparent as this solver already had a small GAP. However, in some cases we still see an improvement in execution time. The results of using EA's solution structure and upper bound in the solver VAL1 are listed in Table 7.

3.4.5 Experiment on Formulations for EDF

As seen in Section 3.4.4, the computation experience for EDF policy has advantages compared to other policies. Therefore, in this section we compare the models based on EDF: VAL1, BARREFORS, and VAL2.

Figure 8 illustrates the mean energy consumption of solutions provided by each solver. The energy consumption rises when the number of tasks increases or when the total system utilization increases. In this experiment, the solvers VAL1 and VAL2 have same energy curve as they converge to same optimal configuration. The solutions produced by BARREFORS algorithm consistently have a higher and increasing energy consumption, as compared to the MGAP based formulations. The difference is expected due to the heuristic step to distribute the tasks in the final step of BARREFORS algorithm. We also highlight that, as seen in the graphics of Figure 8, the algorithm BARREFORS does not produce feasible solution for all cases, specially on configuration with utilization at 80% or higher.

Figure 9 illustrates the final utilization of solutions provided by each solver. Similarly to the energy curves, the MGAP based solvers have very similar curves of utilization of their optimal solution. The optimal solutions of VAL1 and VAL2 quickly converge to almost the EDF's theoretical limit: 99.99%. BARREFORS converges to the utilization estimated using the maximum allowed frequency, because this estimation is used by its algorithm to determine which processors to power on.

Figure 10 illustrates the results of execution time of each solver. In this experi-

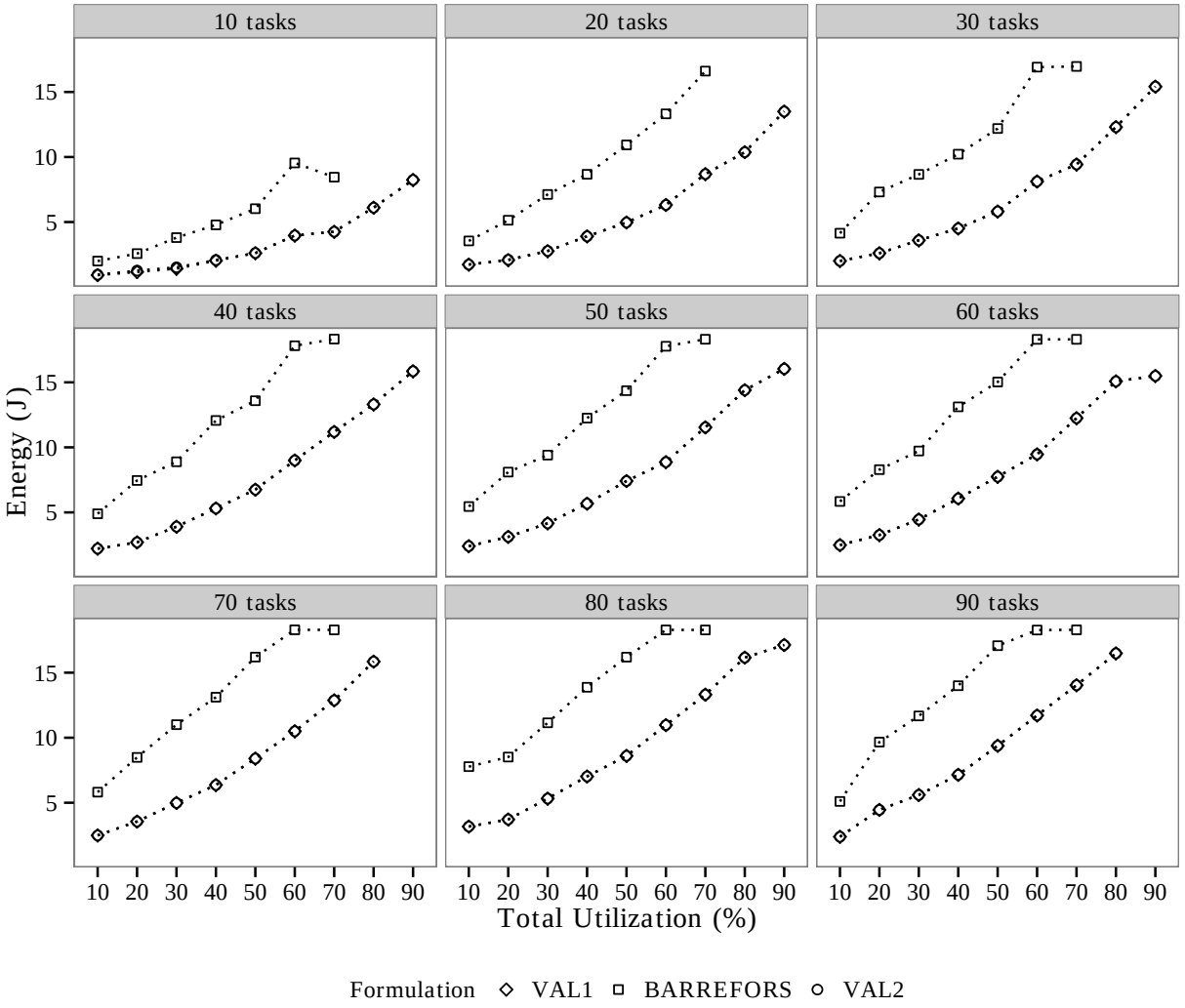


Figure 8 – System energy consumption of hard real-time allocations for BARREFORS, VAL1, and VAL2. BARREFORS produces configurations with higher energy consumption as compared to the MGAP based formulations. The VAL1 and VAL2 have same energy curve.

ment, we limited the execution time of each run to 30 minutes for practical reasons. The execution time of all solvers increases when the number of tasks increases or when the total system utilization is high. BARREFORS is the algorithm with the fastest execution time in this experiment because it has a heuristic step. The VAL1 and VAL2 have similar times, being VAL2 faster in most cases due to its additional restriction. For the evaluated instances (2500), VAL2 finds the optimal solution faster than VAL1 in 58.1 % of the cases.

3.5 Discussion of Results

The interaction between different schedulability analyses and integer programming mathematical formulation have influence on the performance of solvers while searching for

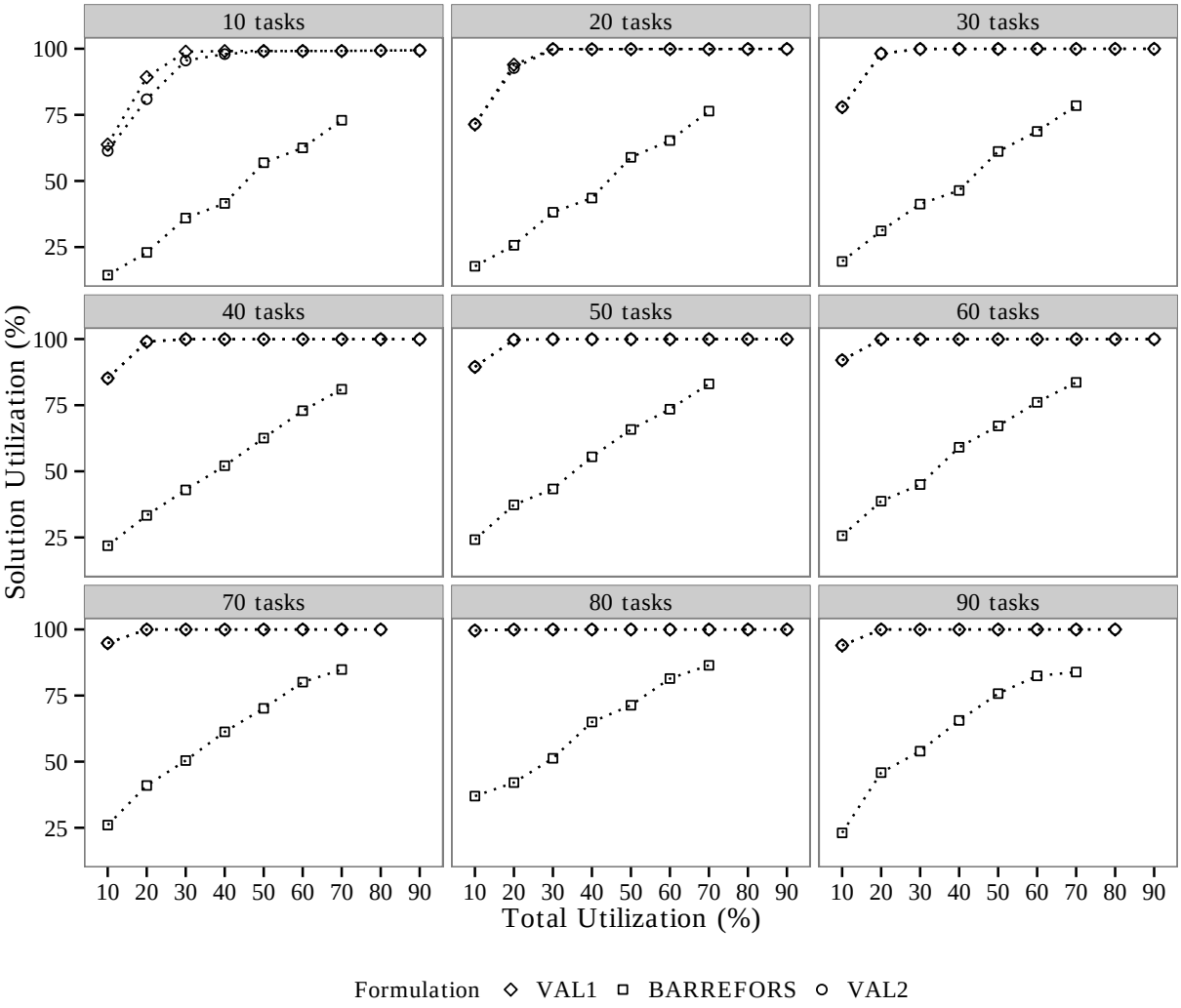


Figure 9 – System total utilization of the solution produced by each solver. VAL1 and VAL2 quickly converges to 100%. BARREFORS converges to the utilization using maximum processor frequency.

the optimal solution. Providing upper bounds and initial feasible solutions may accelerate the search process. We also observe that, while targeting optimal solutions, using strict constraints may reduce overall system energy consumption (Valentin et al., 2015b).

In the literature, there are strategies to determine hard real-time task distribution in heterogeneous platforms. Their approaches typically focus on either heuristics or approximation algorithms (Chen and Thiele, 2011). There are also models proposed to cover optimal solutions that minimize the energy consumption of hard real-time systems with multiple heterogeneous processors. Even though GLPK (Free Software Foundation, 2012) or CPLEX (IBM, 2016) can be used to deriving optimal solutions from their formulations, to the best of our present knowledge, this is the first work to report computational experiments on the search for optimal solution for this problem.

The typical formulation in the specialized literature is a 0/1 integer linear program-

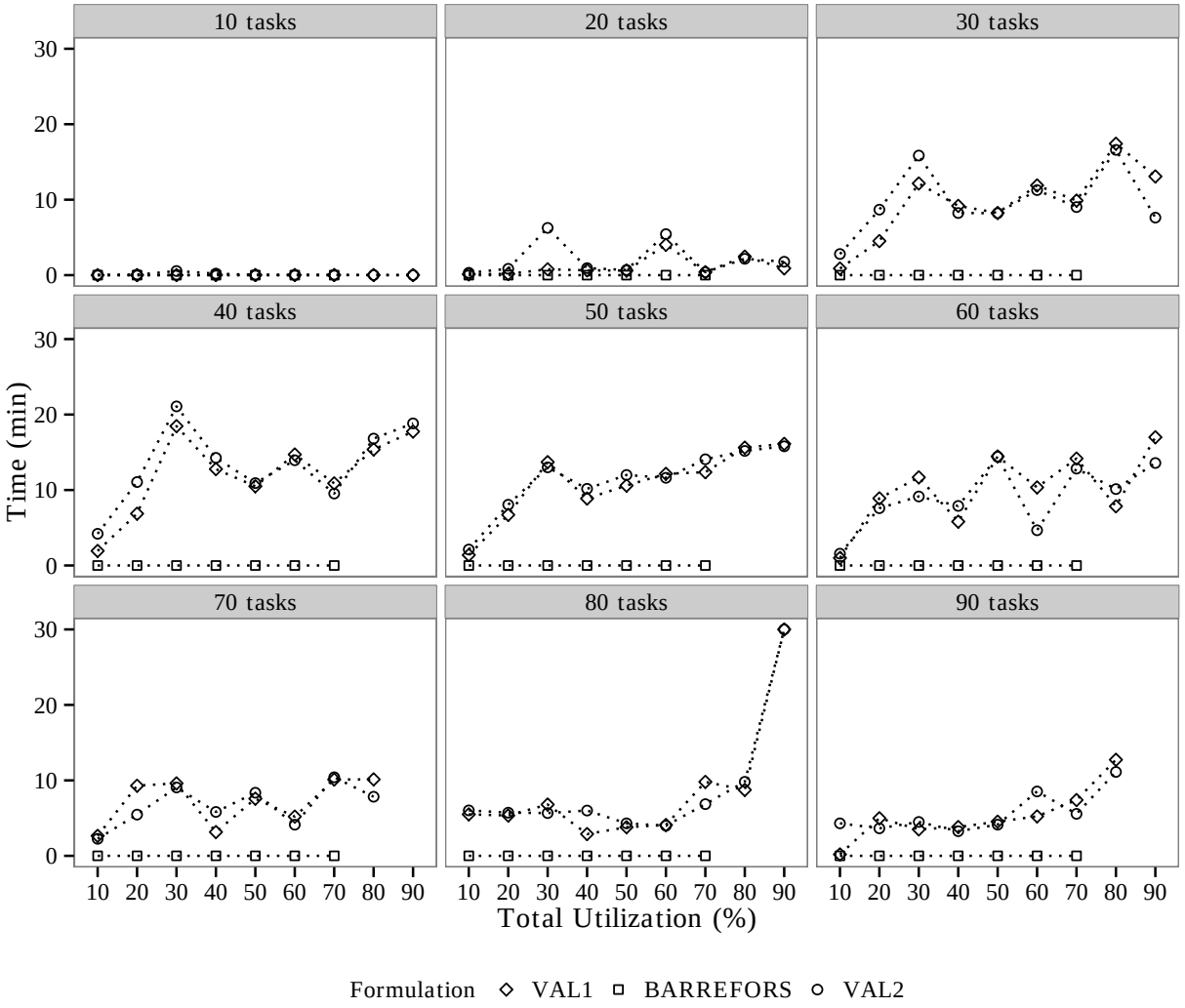


Figure 10 – Execution time of each solver. BARREFORS is the fastest in this experiment. The VAL1 and VAL2 have similar times, being VAL2 faster in most cases.

ming model which considers a continuous processor frequency domain and determines a single operating frequency per processor (Alahmad and Gopalakrishnan, 2011; Awan and Petters, 2013; Chen et al., 2011; Chen and Thiele, 2011; He and Mueller, 2012a). However, using the MGAP model is a more suitable fit to this problem because practical processors still use a discrete set of frequencies (Valentin et al., 2016b).

The adoption of DVFS is common in optimization procedures, such as task allocation, and frequency to task assignment. The aim is to find optimal energy-aware scheduling on heterogeneous platforms while considering individual task deadlines (Chen and Thiele, 2011). Overall system energy reduction is due to workload split and to frequency minimization to meet tasks deadlines. Adoption of the well-known utilization based schedulability analyses is common (Alahmad and Gopalakrishnan, 2011; Awan and Petters, 2013; Chen et al., 2011; Chen and Thiele, 2011; Goossens et al., 2008; He and Mueller, 2012a,b; Prescilla and Selvakumar, 2013; Valentin and Barreto, 2010; Yang et al., 2009; Yu and

Prasanna, 2003). The simplification on the model and on the solving process as using utilization constraints produces formulations similar to the multiple knapsack problem. We confirm these efforts with VAL1 solver, one of the fastest in our experiments. However, our work also propose formulations for different scheduling polices for practical processors (Valentin et al., 2017).

For the classic MGAP there are extreme fast algorithms. MGAP problem instances are solvable up to hundreds of machines with tens of speed levels, to map hundreds of tasks. For example, Osorio and Laguna (2003) proposed a Branch-and-Cut algorithm in 2003 and were able to solve instances up to 60 tasks, 30 machines, and two speed levels. Ceselli and Righini (2006), in 2006, proposed a Branch-and-Price strategy, solving up to 400 tasks, 80 machines and five levels. Another Branch-and-Cut algorithm proposed by Avella et al. (2013) can solve 200 tasks, 30 machines and five levels, for specific problem instances. We contribute with the present work with a new application of the MGAP model (Valentin et al., 2017).

In this chapter, we compared one algorithm based on continuous processor frequency domain (BARREFORS) with algorithms based on MGAP formulation (VAL1 and VAL2), considering a discrete set of frequencies. We also observe that, while targeting optimal solutions for this problem, using strict constraints may reduce overall system energy consumption. The MGAP based algorithms produces system configurations with lower power consumption, but with the penalty of solver execution time, when compared to BARREFORS. Our proposed extended MGAP based formulation (VAL2) has same energy curve as the state-of-the-art MGAP formulation (VAL1) for this problem, but also has a faster solver, as seen in most cases of our experiment (Valentin et al., 2017).

3.6 Chapter Summary

In this chapter we assess the problem of how to find optimal hard real-time tasks distribution among heterogeneous processors respecting timing constraints and minimizing power consumption. Our study focuses on optimal solutions after reviewing the existing models.

We proposed the usage of a well-known classical combinatorial optimization problem, MGAP, to represent the characteristics and constraints of the problem. We first used MGAP to model systems using EDF ($D_j = T_j$). We also extended the proposed MGAP to derive extra models considering utilization bounds for RM ($D_j = T_j$) and response time analysis for RM ($D_j = T_j$). The implementation of such models delivers optimal hard real-time task allocation and optimal frequency to task assignment minimizing system energy consumption. Based on our experimental results we recommend for fixed priority policies the usage of MGAP response time based model. The MGAP response time based

model, implemented for RM in our experiments, finds configurations with the same energy consumption as the literature EDF representation, but with an execution time penalty.

We proposed to extend the state-of-the-art MGAP formulation by adding an extra constraint based on the problem characteristic. The additional constraint mimics the optimal configuration which exists only on processors that can operate at any frequency, in which each CPU executes at a fixed and ideal frequency (Aydin et al., 2001b). We accomplish this constraint by establishing a minimum bound on the average of each task assigned CPU frequency, but we still consider a discrete set of CPU frequencies. Experimental results show that the MGAP based algorithms produce system configurations with lower power consumption, but with the penalty of solver execution time, when compared to BARREFORS. Our proposed MGAP based formulation (VAL2) has same energy curve as the state-of-the-art MGAP formulation (VAL1) but executes faster, as seen in most cases of our experiment.

We recognize that targeting optimal solutions for this problem is not a simple task, specially considering the fast growth of its formulations. However, the optimal is still reachable in some instances, as shown in our computational experiment, by applying well-known combinatorial optimization techniques. For example, we have exercised the usage of an EA to find upper bounds and initial feasible solution structure, boosting the search for the optimal while executing solvers.

Table 4 – Comparative results of each solver: VAL1, VAL4, and VAL3.

Tasks	U_{total}	VAL4			VAL3			VAL1		
		Best Sol. (J)	Time (μs)	GAP (%)	Best Sol (J).	Time (μs)	GAP (%)	Best Sol (J).	Time (μs)	GAP (%)
5	10	0.633	1.21E+05	0.00	0.664	9.39E+05	0.00	0.627	3.39E+04	0.00
10	10	0.961	2.15E+05	0.00	1.03	1.8E+09	4.49	0.96	1.85E+05	0.00
15	10	1.66	2.56E+06	0.01	1.89	1.8E+09	11.72	1.66	1.7E+06	0.01
20	10	1.73	9.02E+07	0.01	2.12	1.8E+09	18.03	1.73	6.94E+06	0.01
5	20	0.346	3.93E+04	0.00	0.347	4.55E+05	0.00	0.346	3.8E+04	0.00
10	20	0.887	9.09E+05	0.00	0.957	1.8E+09	9.64	0.885	2.58E+05	0.00
15	20	1.72	1.49E+06	0.01	2.26	1.8E+09	23.67	1.72	6.54E+05	0.01
20	20	2.09	5.04E+07	0.01	3.02	1.8E+09	30.15	2.09	7.74E+06	0.01
5	30	0.311	5E+04	0.00	0.318	4.5E+05	0.00	0.31	5.92E+04	0.00
10	30	1.61	2.65E+07	0.01	2.06	1.8E+09	20.82	1.61	3.22E+05	0.00
15	30	1.95	2.1E+08	0.01	3.04	1.8E+09	36.65	1.95	5.28E+06	0.01
20	30	3.05	1.47E+09	0.12	5.45	1.8E+09	44.00	3.04	1.21E+07	0.01
5	40	0.844	8.55E+04	0.00	0.892	2.87E+06	0.00	0.843	5.47E+04	0.00
10	40	1.11	2.21E+06	0.01	1.46	1.8E+09	24.00	1.1	1.27E+06	0.01
15	40	1.92	2.82E+08	0.01	2.99	1.8E+09	36.08	1.92	2.85E+07	0.01
20	40	4.11	1.8E+09	0.63	6.84	1.8E+09	40.36	4.09	4.42E+08	0.01
5	50	0.97	3.52E+04	0.00	1.01	8.7E+05	0.00	0.97	3.95E+04	0.00
10	50	0.712	1.08E+07	0.01	1.08	1.8E+09	33.93	0.706	7.28E+05	0.01
15	50	4.71	2.76E+08	0.01	7.38	1.8E+09	36.62	4.7	7.91E+06	0.01
20	50	4.96	1.5E+09	0.00	9.89	1.8E+09	38.62	4.97	3.23E+08	0.01

Table 5 – Results of using EA's solution structure and upper bound to boost the VAL4 solver.

Tasks	U_{total}	VAL4			EA VAL4		VAL4 + EA VAL4		
		Best Sol. (J)	Time (μs)	GAP (%)	Best Sol (J).	Time (μs)	Best Sol (J).	Time (μs)	GAP (%)
5	10	0.633	1.21E+05	0.00	1.03	3.48E+05	0.633	7.87E+04	0.00
10	10	0.961	2.15E+05	0.00	2.99	7.59E+05	0.961	1.8E+05	0.00
15	10	1.66	2.56E+06	0.01	7.12	1.24E+06	1.66	2.3E+06	0.01
20	10	1.73	9.02E+07	0.01	9.5	1.8E+06	1.73	8.29E+07	0.01
5	20	0.346	3.93E+04	0.00	0.536	3.61E+05	0.346	3.29E+04	0.00
10	20	0.887	9.09E+05	0.00	2.46	8.19E+05	0.887	7.45E+05	0.00
15	20	1.72	1.49E+06	0.01	6.68	1.16E+06	1.72	1.38E+06	0.01
20	20	2.09	5.04E+07	0.01	10.3	1.86E+06	2.09	6.36E+07	0.01
5	30	0.311	5E+04	0.00	0.462	3.99E+05	0.311	4.72E+04	0.00
10	30	1.61	2.65E+07	0.01	3.84	9.4E+05	1.61	2.22E+07	0.00
15	30	1.95	2.1E+08	0.01	6.16	1.48E+06	1.95	1.4E+08	0.01
20	30	3.05	1.47E+09	0.12	12.5	1.87E+06	3.05	1.48E+09	0.12
5	40	0.844	8.55E+04	0.00	1.09	4.91E+05	0.844	5.67E+04	0.00
10	40	1.11	2.21E+06	0.01	2.19	1.23E+06	1.11	2.5E+06	0.01
15	40	1.92	2.82E+08	0.01	5.03	2.31E+06	1.92	2.3E+08	0.01
20	40	4.11	1.8E+09	0.63	13.6	1.5E+06	4.1	1.8E+09	0.44
5	50	0.97	3.52E+04	0.00	1.18	9.12E+05	0.97	3.08E+04	0.00
10	50	0.712	1.08E+07	0.01	1.3	4.24E+06	0.712	3.21E+07	0.01
15	50	4.71	2.76E+08	0.01	10.5	1.85E+07	4.71	2.5E+08	0.01
20	50	4.96	1.8E+09	0.00	4.97	1.29E+07	4.96	2.9E+08	0.00

Table 6 – Results of using EA's solution structure and upper bound to boost the VAL3 solver.

Tasks	U_{total}	VAL3			EA VAL3		VAL3 + EA VAL3		
		Best Sol. (J)	Time (μs)	GAP (%)	Best Sol (J).	Time (μs)	Best Sol (J).	Time (μs)	GAP (%)
5	10	0.664	9.39E+05	0.00	1.04	3.76E+05	0.664	6.56E+05	0.00
10	10	1.03	1.8E+09	4.49	3	6.41E+05	1.03	1.8E+09	4.10
15	10	1.89	1.8E+09	11.72	7.21	1.01E+06	1.93	1.8E+09	13.50
20	10	2.12	1.8E+09	18.03	9.57	1.32E+06	2.13	1.8E+09	18.55
5	20	0.347	4.55E+05	0.00	0.552	3.74E+05	0.347	5.62E+04	0.00
10	20	0.957	1.8E+09	9.64	2.48	6.53E+05	0.955	1.8E+09	8.18
15	20	2.26	1.8E+09	23.67	6.72	1.06E+06	2.25	1.8E+09	23.13
20	20	3.02	1.8E+09	30.15	10.4	1.33E+06	2.99	1.8E+09	29.78
5	30	0.318	4.5E+05	0.00	0.481	3.48E+05	0.318	2.35E+05	0.00
10	30	2.06	1.8E+09	20.82	3.98	9.7E+05	1.86	1.8E+09	19.05
15	30	3.04	1.8E+09	36.65	6.37	1.18E+06	2.89	1.8E+09	31.57
20	30	5.45	1.8E+09	44.00	12.7	1.72E+06	4.69	1.8E+09	35.34
5	40	0.892	2.87E+06	0.00	1.14	5.36E+05	0.892	1.62E+06	0.00
10	40	1.46	1.8E+09	24.00	2.29	2.41E+06	1.49	1.8E+09	25.09
15	40	2.99	1.8E+09	36.08	5.27	8.32E+06	2.78	1.8E+09	30.84
20	40	6.84	1.8E+09	40.36	13.9	6.45E+06	5.92	1.8E+09	31.07
5	50	1.01	8.7E+05	0.00	1.27	2.53E+06	1.01	6.15E+05	0.00
10	50	1.08	1.8E+09	33.93	1.39	3.32E+07	1.01	1.8E+09	29.91
15	50	7.38	1.8E+09	36.62	11.9	8.42E+07	7.3	1.8E+09	35.95
20	50	9.89	1.8E+09	38.62	13.5	9.42E+06	8.61	5.8E+09	35.3

Table 7 – Results of using EA's solution structure and upper bound to boost the VAL1 solver.

Tasks	U_{total}	VAL1			EA VAL1		VAL1 + EA VAL1		
		Best Sol. (J)	Time (μs)	GAP (%)	Best Sol (J).	Time (μs)	Best Sol (J).	Time (μs)	GAP (%)
5	10	0.627	3.39E+04	0.00	1.02	3.32E+05	0.627	2.53E+04	0.00
10	10	0.96	1.85E+05	0.00	2.99	5.79E+05	0.96	1.18E+05	0.00
15	10	1.66	1.7E+06	0.01	7.12	9.18E+05	1.66	1.13E+06	0.01
20	10	1.73	6.94E+06	0.01	9.5	1.17E+06	1.73	5.89E+06	0.01
5	20	0.346	3.8E+04	0.00	0.536	3.42E+05	0.346	2.47E+04	0.00
10	20	0.885	2.58E+05	0.00	2.41	5.71E+05	0.885	1.49E+05	0.00
15	20	1.72	6.54E+05	0.01	6.68	9.19E+05	1.72	3.85E+05	0.01
20	20	2.09	7.74E+06	0.01	10.3	1.13E+06	2.09	6.65E+06	0.01
5	30	0.31	5.92E+04	0.00	0.462	3.19E+05	0.31	3.3E+04	0.00
10	30	1.61	3.22E+05	0.00	3.83	8.12E+05	1.61	3.21E+05	0.00
15	30	1.95	5.28E+06	0.01	6.18	9.09E+05	1.95	4.7E+06	0.01
20	30	3.04	1.21E+07	0.01	12.5	1.07E+06	3.04	1.05E+07	0.01
5	40	0.843	5.47E+04	0.00	1.09	3.19E+05	0.843	3.5E+04	0.00
10	40	1.1	1.27E+06	0.01	2.19	8.82E+05	1.1	1.54E+06	0.01
15	40	1.92	2.85E+07	0.01	5.03	9.39E+05	1.92	3.16E+07	0.01
20	40	4.09	4.42E+08	0.01	13.6	1.71E+06	4.09	4.93E+08	0.01
5	50	0.97	3.95E+04	0.00	1.18	5E+05	0.97	2.62E+04	0.00
10	50	0.706	7.28E+05	0.01	1.25	2.18E+06	0.706	5.11E+05	0.01
15	50	4.7	7.91E+06	0.01	10.5	1.61E+07	4.7	8.81E+06	0.01
20	50	4.97	3.23E+08	0.01	12.8	1.91E+07	4.97	3.22E+08	0.01

4 A Branch-and-Price Algorithm to Distribute Independent Hard Real-Time Tasks

Modern processors may be seen as machines with several performance states due to Dynamic Voltage and Frequency Scaling (DVFS) technique. DVFS is a well established power reduction strategy and it has already been a research topic for decades. The premises are the variation of processors' workloads and the quadratic relationship between energy consumption and voltage (Burd and Brodersen, 1996). The energy consumption depends on dynamic and idle energy (Venkatachalam and Franz, 2005): $E_{system} = E_{dyn} + E_{idle}$, where E_{idle} is the energy consumption while the system is idle and accounts for leakage, E_{dyn} is the energy consumption in active use cases. The dynamic energy consumption E_{dyn} is estimated using: $E_{dyn} = C_l \times N_{cycle} \times V_{dd}^2$, where E is the energy, C_l is the circuitry capacitance, N_{cycle} is the number of cycles, and V_{dd} is the voltage. Although DVFS yields meaningful energy consumption reduction, its usage requires care, especially when considering timing constraints.

The problem we are addressing in this chapter is: *how to find optimal hard real-time tasks distribution among heterogeneous processors respecting timing constraints and targeting low power consumption?* The contributions of this work are: (i) comprehensive and representative mathematical formulations that (ii) accounts characteristics of the Earliest Deadline First (EDF) hard real-time scheduling policy and that (iii) delivers optimal hard real-time task allocation and optimal frequency to task assignment, (iv) with system energy consumption minimization, but still (v) using the advantage of a classical combinatorial optimization model: MGAP. The results we present on this research question focus on solving the problem optimally on practical instances sizes. We use, in this chapter, a branch-and-price approach as the main computational technique of resolution for this problem.

The organization of this chapter is as follows. We first review the system models in Section 4.1. We describe how to model the problem using a column generation approach in Section 4.2. We also review the state-of-the-art method in Section 4.3. We describe the results of computational experiments in Section 4.4. We discuss the results in Section 4.5. At last, we present a summary of this chapter in Section 4.6.

4.1 System Models

In this section we present the system models. Section 4.1.1 describes the processor model we consider. We show the real-time task model in Section 4.1.2.

4.1.1 Processor Model

The processor model resembles a Multi-Processor System-On-Chip (MPSoC) architecture, such as Exynos 5 Octa (Samsung Electronics Co.Ltd., 2014). The system is composed by a set, $\mathcal{H}$, of m processors, $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$. Each core may operate on l different performance states, $1 \leq k \leq l$. The set of frequencies of one core is not necessarily the same of other cores. Also, a task may have different code size and execution time for different processors, due to instruction set differences. The frequency of performance state k on the processor i is F_{ik} and the power consumption is P_{ik}. The idle power of processor i is $P_{idle,i}$.

Our proposal can be used with no extra effort on other architectures. Even though we focus on per-core heterogeneous platforms in our experiments, we have exercised on multiple heterogeneous clusters (Valentin et al., 2015b). The models discussed here may be applicable to full-chip and cluster-based platforms as long as the intrinsic architectural interference is accounted, including the DVFS related overhead, and we recommend the interested reader to consider more sophisticated schedulability tests (Valentin et al., 2015b).

4.1.2 Task Model

In the remaining sections of this chapter we adopt the following notation. A task model $\mathcal{M}$ is a set composed by n tasks τ_j. A task $\tau_j \in \mathcal{M}$, with $j \leq n$, has the properties: worst-case execution cycle $WCEC_j$; worst-case execution time $C_j(f)$, which is a function of frequency f, thus $C_j(f) = \frac{WCEC_j}{f}$; period of execution T_j; deadline D_j, we consider scheduling policies in which $D_j = T_j$. A task τ_j also has the following properties, specific to fixed priority policies: fixed priority p_j; set of high priority tasks $hp(j)$ representing the tasks τ_p with priority higher than the priority of τ_j. The response time R_j is dependent not only on task set characteristics, but also on the target platform, and on the task allocation and frequency distribution that have been selected for the workload.

4.2 Columns Generation Algorithm

The models for this problem have the characteristic of having more variables than constraints. The growth of variables makes the problem a good fit for a Branch-Cut-and-Price (BCP) algorithm, where upper bounds on the optimal value are computed by column generation and cut planes are generated to accelerated the derivation of integer solutions.

4.2.1 Original Mathematical Formulation

EDF is a dynamic priority based on-line scheduler in which earliest deadlines are first scheduled. Liu and Layland (1973) propose a utilization based schedulability test for uniprocessor systems. The test is: $\sum_{j=1}^{n} C_j(f)/T_j \leq 1$. We are considering implicit deadlines for EDF ($D_j = T_j$), in the same way as the original Liu and Layland work.

The EDF scheduler is known to be an optimal online scheduler that can achieve 100% of CPU utilization for independent real-time tasks. Taking advantage of the EDF's utilization bound is common practice on the specialized literature (Alahmad and Gopalakrishnan, 2011; Awan and Petters, 2013; Chen et al., 2011; Chen and Thiele, 2011; Goossens et al., 2008; He and Mueller, 2012a,b; Prescilla and Selvakumar, 2013; Valentin and Barreto, 2010; Yang et al., 2009; Yu and Prasanna, 2003). We present mathematical formulations for real-time task allocation taking into account the utilization bound for EDF as follows.

Valentin et al. (2016b) propose to use the Multi-level Generalized Assignment Problem (MGAP) as a more suitable formulation for representing the hard real-time task assignment problem. Equation 4.1 has the original problem formulation.

$$\text{Minimize} \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l} c_{ijk} x_{ijk} \right\} \tag{4.1a}$$

$$\text{s.t.:} \sum_{i=1}^{m} \sum_{k=1}^{l} x_{ijk} = 1,\ j \in \{1, \ldots, n\} \tag{4.1b}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{l} a_{ijk} x_{ijk} \leq \theta_i,\ i \in \{1, \ldots, m\} \tag{4.1c}$$

$$x_{ijk} \in \{0, 1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{4.1d}$$

4.2.2 Reformulation for the Master Problem

We present here an alternative formulation of the MGAP, which is viable for a branch-and-price approach. Let a *duty* d for processor i be an assignment of tasks to processor i, that is the vector $y_i^d = (y_{i11}^d, ..., y_{inl}^d)$, where each component y_{ijk}^d is 1 when task j is assigned to processor i at level k, 0 otherwise.

Let D_i be the set of all feasible duties for processor i, i.e. vectors $(y_{i11}^d, ..., y_{inl}^d)$ such

that:

$$\left(\sum_{j=1}^{n} \sum_{k=1}^{l} a_{ijk} y_{ijk}^{d} \leq \theta_i \right) \tag{4.2a}$$

$$\sum_{k=1}^{l} y_{ijk}^{d} \leq 1,\ j \in \{1, \ldots, n\} \tag{4.2b}$$

$$y_{ijk}^{d} \in \{0,1\}, 1 \leq j \leq n, 1 \leq k \leq l \tag{4.2c}$$

Let z_i^d indicate if a duty d is selected to processor i. The exp reformulation of MGAP is (Ceselli and Righini, 2006):

$$\text{Minimize} \left\{ \sum_{i=1}^{m} \sum_{d \in D_i} \left(\sum_{j=1}^{n} \sum_{k=1}^{l} c_{ijk} y_{ijk}^{d} \right) z_i^d \right\} \tag{4.3a}$$

$$\text{s.t.:} \sum_{i=1}^{m} \sum_{d \in D_i} \left(\sum_{k=1}^{l} y_{ijk}^{d} \right) z_i^d = 1,\ j \in \{1, \ldots, n\} \tag{4.3b}$$

$$\sum_{d \in D_i} z_i^d = 1,\ 1 \leq i \leq m \tag{4.3c}$$

$$z_i^d \in \{0,1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{4.3d}$$

This reformulation, as required by the Dantzig-Wolfe decomposition method, relies on the fact that a non-empty, bounded convex polyhedron can be represented as a convex combination of its extreme points and a linear combination of extreme rays.

4.2.3 The Pricing Problem

Let $\lambda \in R_+^m$ and $\mu \in R_+^n$ be the vectors of non-negative dual variables corresponding to constraints 4.3b and 4.3c, respectively. The reduced cost of duty d for processor i is:

$$r_i^{-d} = \sum_{j=1}^{n} \sum_{k=1}^{l} c_{ijk} y_{ijk}^{d} - \sum_{j=1}^{n} \lambda_j (\sum_{k=1}^{l} y_{ijk}^{d}) + \mu_i \tag{4.4a}$$

Therefore, To find columns with negative reduced cost we must solve the following pricing problem for each processor:

$$\text{Minimize} \left\{ r_i^{-d} = \sum_{j=1}^{n} \sum_{k=1}^{l} \left(c_{ijk} - \lambda_i\right) y_{ijk}^{d} + \mu_i \right\} \tag{4.5a}$$

$$\sum_{j\,=1}^{n} \sum_{k\,=1}^{l} a_{ijk} y_{ijk}^{d} \leq \theta_i \tag{4.5b}$$

$$\sum_{k\,=1}^{l} y_{ijk}^{d} \leq 1,\ j \in \{1, \ldots, n\} \tag{4.5c}$$

$$y_{ijk}^{d} \in \{0, 1\}, 1 \leq j \leq n, 1 \leq k \leq l \tag{4.5d}$$

that is a multiple choice knapsack problem (MCKP).

4.2.4 Dynamic Programming for the Pricing Problem

The pricing problem is solvable with the dynamic programming described in Equation 4.6a.

$$\begin{aligned} K(w, n) = \text{Min:} \left\{ K(w - w_{jk}, j - 1) + c_{jk}, K(w, j - 1) \right\}, 1 \leq k \leq l \\ K(0, j) = 0, \forall 1 \leq j \leq n \\ K(w, 0) = 0, \forall 1 \leq j \leq W \end{aligned} \tag{4.6a}$$

4.2.5 Computational Technique of Resolution

In this section, we explain the algorithmic strategy developed for solving the problem using a column generation approach. In Section 4.2.5.1, we explain the branch-and-price algorithm and in Section 4.2.5.2, we present the dynamic programming algorithm for the pricing problem.

4.2.5.1 Branch-And-Price Algorithm

We follow a general method for branch-and-price.The overall idea is to attempt to tighten the linear relaxation of each node of the branch-and-bound algorithm. Also, because there are many columns to be solved in the Master problem, a set of columns are left out of the linear relaxation, as many of their associated variables have value of zero in an optimal solution. To check for optimally, the pricing problem is invoked to identify columns to enter the basis. Branching occurs when no columns price out to enter the basis and the linear relaxation solution does not satisfy integrality conditions.

Algorithm 3 Branch-and-Price (B&P) for Independent Tasks

```
1: Input: H, M, ub
2: Output: optimal solution (x*, v*)
3: v* ← ub.val; x* ← ub.structure;
4: L ← ILP^0;
5: while L != ∅ do
6:    n ← remove_node(L);
7:    lp ← relaxation(n)
8:    (x, v) ← solve(lp);
9:    if x = infeasible then
10:      continue;
11:   end if
12:   p ← dual_bounds(x, v);
13:   solve_pricing(p, subproblems);
14:   added ← FALSE;
15:   for all column ∈ columns do
16:      if column have positive reduced costs then
17:         add(column, lp);
18:         added ← TRUE;
19:      end if
20:   end for
21:   if added then
22:      goto 7;
23:   end if
24:   if v ≥ v* then
25:      continue;
26:   end if
27:   if x is integer then
28:      v* ← v;x* ← x;
29:      continue;
30:   end if
31:   partition(lp);
32: end while
33: return (x*, v*);
```

The algorithm starts by defining a Restricted Master Problem (RMP), which is a version of the Master Problem with a subset of the Master Problem's columns. The linear relaxation of the RMP is then solved and the dual bounds are determined. The pricing sub-problems are solved by passing the dual bounds. When columns with positive reduced costs are found after solving the pricing problems, such columns are added to the RMP and the process is repeated. When no columns are found with positive reduced cost, the solution for the RMP is checked for integrality. When the solution is not integral, a branching is performed. When the solution is integral, a solution is found and the process stops. The Algorithm 3 illustrates the process.

4.2.5.2 Dynamic Programming Pricing Algorithm

We have developed a dynamic programming algorithmic solving strategy for the pricing problem. The dynamic programming strategy is similar to a knapsack dynamic programming algorithms, but specific for the multiple choice knapsack problem version.

The algorithm maintains a $(n \times 1000)$ matrix f of reduced costs and indexes to compose the final solution. The matrix is composed of n rows, representing each task, and 1000 columns, representing the utilization multiplied by a factor of 1000.

The matrix first column and first row are initialized to 0 (reduced cost) and INT_MAX (index). The remaining rows and columns are initialized according to the Equation 4.6a. A simple walk back algorithm is used to determine the final solution, when available, starting from the indexes n and 1000, deacreasing to indexes 1 and 1. The Algorithm 4 illustrates the process.

4.3 Barrefor's Formulation with Utilization and Frequency Bound for EDF

Barrefors et al. (2014) use the EDF's utilization bound in an integer programming mathematical formulation to distribute hard real-time tasks on heterogeneous processors. For comparison purposes, we use Barrefors et al. (2014) as a reference from the state-of-the-art. Their formulation expects as valid CPU frequencies only those whose power consumption can be sustained for long period of time without the need to throttle due to temperature constraints. Equation 4.7 (BARREFORS) lists their formulation.

$$\text{Minimize} \left\{ P \sum_{i=1}^{m} \Phi_i(\tilde{f}_i^{max}) x_i \right\} \tag{4.7a}$$

$$\text{s.t.:} \sum_{i=1}^{m} \alpha_i \tilde{f}_i^{max} x_i \geq U_{tot} \tag{4.7b}$$

$$x_i \in \{0, 1\}, 1 \leq i \leq m \tag{4.7c}$$

where the decision variable x_i determines if processor i is turned on and in use $(x_i = 1)$ or not $(x_i = 0)$, α_i is the performance coefficient of processor i, m is the number of processors, $\tilde{f}_i^{max}$ is processor i's maximum frequency satisfying the temperature constraint, and $\Phi_i(f)$ is the power consumption of processor i when executed at a frequency f. BARREFORS' formulation determines only which processors to turn on to process the tasks. In a second

Algorithm 4 Pricing Dynamic Programming (PDP)

```
1: Input: p, w, n, l, r
2: Output: optimal solution x
3: c ← 1000;
4: for all 0 ≤ i ≤ n do
5:    f[i][0].f = 0;
6:    f[i][0].imin = INT_MAX;
7: end for
8: for all 0 ≤ j ≤ c do
9:    f[0][j].f = 0;
10:   f[0][j].imin = INT_MAX;
11: end for
12: for all 1 ≤ i ≤ n do
13:    for all 1 ≤ j ≤ c do
14:       f[i][j].f = f[i − 1][j].f;
15:       f[i][j].imin = f[i − 1][j].imin;
16:       for all 0 ≤ k ≤ l do
17:          imin ← (i − 1) ∗ l + k;
18:          if j ≥ w[imin] then
19:             if p[imin] + f[i − 1][j − w[imin]].f < f[i][j].f then
20:                f[i][j].f ← p[imin] + f[i − 1][j − w[imin]].f;
21:                f[i][j].imin = imin;
22:             end if
23:          end if
24:       end for
25:    end for
26: end for
27: i ← n;
28: j ← r;
29: while i > 0 and j > 0 do
30:    if f[i][j].imin! = INT_MAX and f[i][j].imin! = f[i−1][j].imin
       then
31:       k ← f[i][j].imin;
32:       x[k] ← 1;
33:       j = j − w[k];
34:    end if
35:    i ← i − 1;
36: end while
37: return x;
```

phase, BARREFORS' approach distributes the tasks and determines which frequency each processor executes based on a worst-fit decreasing algorithm (Barrefors et al., 2014).

4.4 Computational Experience

In this section we detail the computational experiments and analyse the generated results. The environment description is outlined in Section 4.4.1. After that, we present two main scenarios of experiments. We first experiment in Section 4.4.2 comparing the B&P algorithm against state-of-the-art BARREFORS approach. In Section 4.4.3, we present further computational experiments targeting only EDF based models.

4.4.1 Experiment Environment

We use random-generated task models. We vary number of tasks in each random-generated models between 10 and 90. We vary the total system utilization between 10% and 90%. We chose to have tasks periods uniformly distributed between three sets: large period, medium period, and short period. When a task has a large period, its period is chosen from the set $\{100, 250, 750\}$ milliseconds. Similarly, a medium period is normally selected from the set $\{5, 10, 50\}$ milliseconds, and a short period is normally selected from the set $\{100, 250, 750\}$ microseconds.

Our experiment considers a target platform of four processors. The target platform is similar to Exynos platform (Samsung Electronics Co.Ltd., 2014). There are two high performance processors that can operate at frequencies from 600 MHz to 1.7 GHz. Also, there are two low power consumption processors that can operate at frequencies from 200 MHz to 1.3 GHz. The idle power consumption is 260 mW. The energy consumption is estimated for the duration of the LCM of tasks' periods, as in the objective function of each model.

The Branch-Cut-And-Price and the column management algorithm is derived using Firula (Pessoa, 2017). Firula (Framework for Intelligible Robust User-defined Linear-programming Algorithms) is a framework for implementing Branch-Cut-and-Price algorithms where the user needs only to define a mixed integer linear programming model for the master problem and provide a solver for each subproblem. All variables and constraints are identified by a character string in the API. It requires a linear programming solver to run.

4.4.2 Experiments against BARREFORS

In this section, we present the results for comparing the Branch-and-Price (BP) strategy and the method proposed by Barrefors et al. (2014) (BARREFORS). We present the results of the average execution time of each algorithm and the objective function value.

Figure 11 illustrates the mean energy consumption of solutions provided by each

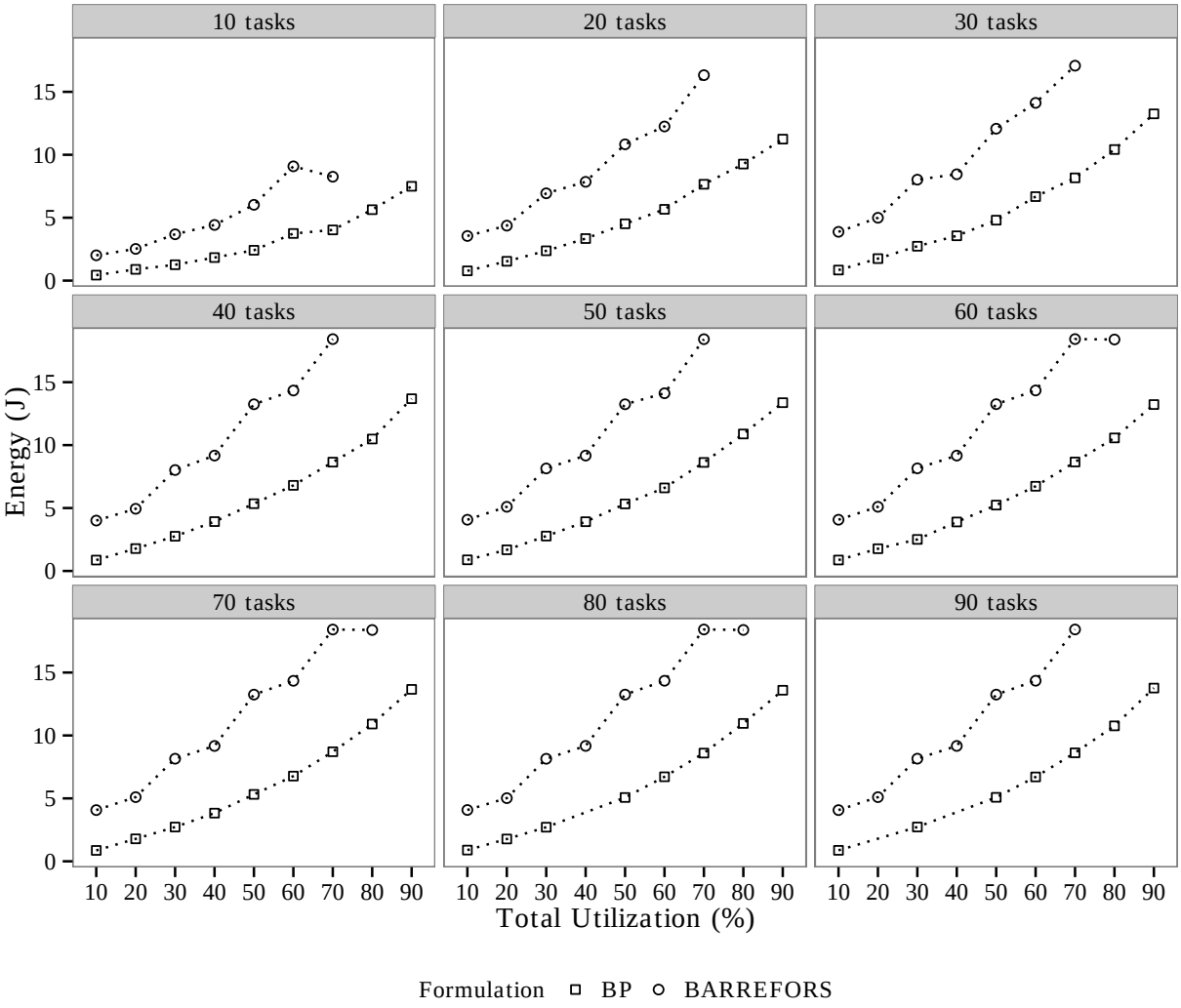


Figure 11 – System energy consumption of hard real-time allocations for BARREFORS and Branch-and-Price (BP) strategies

solver. As observed, the BARREFORS strategy, because it has a heuristic step to distribute the tasks, delivers a solution with higher energy consumption when compared to solutions found by BP.

Figure 12 illustrates the mean final system utilization provided by the final configuration as result of each solver. Similarly to the energy curves, the optimal solutions provided by BP quickly converge to almost the EDF's theoretical limit: 99.99%. BARREFORS converges to the utilization estimated using the maximum allowed frequency, because this estimation is used by its algorithm to determine which processors to power on.

Figure 13 illustrates the mean solver execution time. As observed, once again, because BARREFORS strategy uses a heuristic step to distribute the tasks, BARREFORS has a much lower solver execution time, when compared to BP.

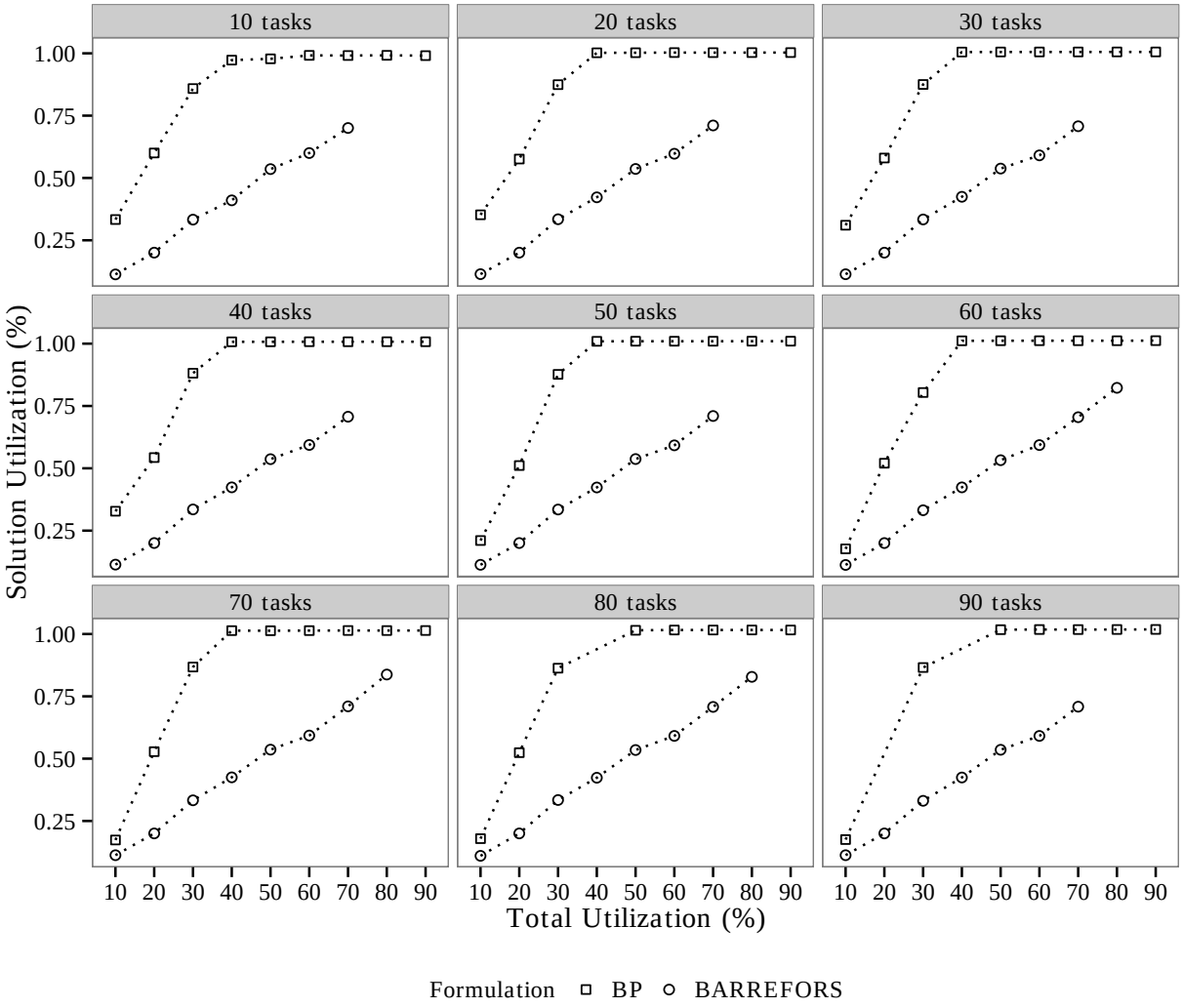


Figure 12 – System final utilization of hard real-time allocations for BARREFORS and Branch-and-Cut (BC) strategies

4.4.3 Experiments against B&C MGAP

In this section, we present the results for comparing the solver using the Branch-and-Price (BP) strategy and the Branch-and-Cut (BC) strategy presented in Chapter 3. We present the results of the average execution time of each algorithm and the objective function value.

Figure 14 illustrates the mean energy consumption of solutions provided by each solver. As observed in the energy curves, both solvers converge to the same optimal solutions.

Figure 15 illustrates the mean final system utilization provided by the final configuration as result of each solver. As observed, both solvers were able to converge to the same optimal value of system utilization.

Figure 16 illustrates the mean solver execution time. As observed, for these in-

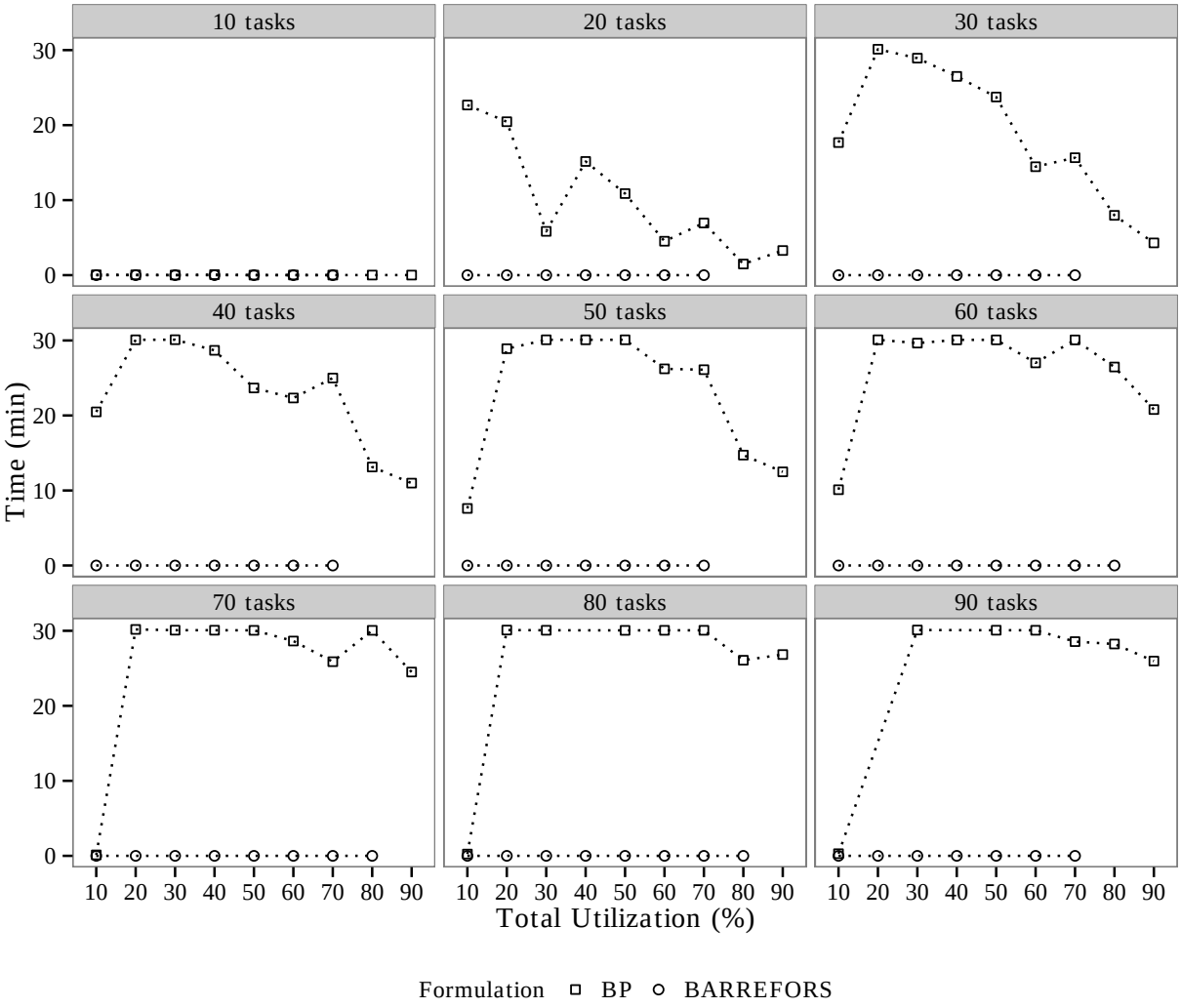


Figure 13 – Solver execution time for BARREFORS and Branch-and-Price (BP)strategies

stances, the BC based implementation still produces a faster solver. However, in some instances, the BP Firula based solver is faster. We observe that in some of the instances, when the system utilization is higher, the Firula based solver may be faster than the BC based solver.

4.5 Discussion of Results

In the literature, there are strategies to determine hard real-time task distribution in heterogeneous platforms, as noted in Chapter 3. To the best of knowledge, this is the first work to present a branch-and-price based resolution method for this problem. Their approaches typically focus on either heuristics or approximation algorithms (Chen and Thiele, 2011). There are also models proposed to cover optimal solutions that minimize the energy consumption of hard real-time systems with multiple heterogeneous processors. Even though GLPK (Free Software Foundation, 2012) or CPLEX (IBM, 2016) can be

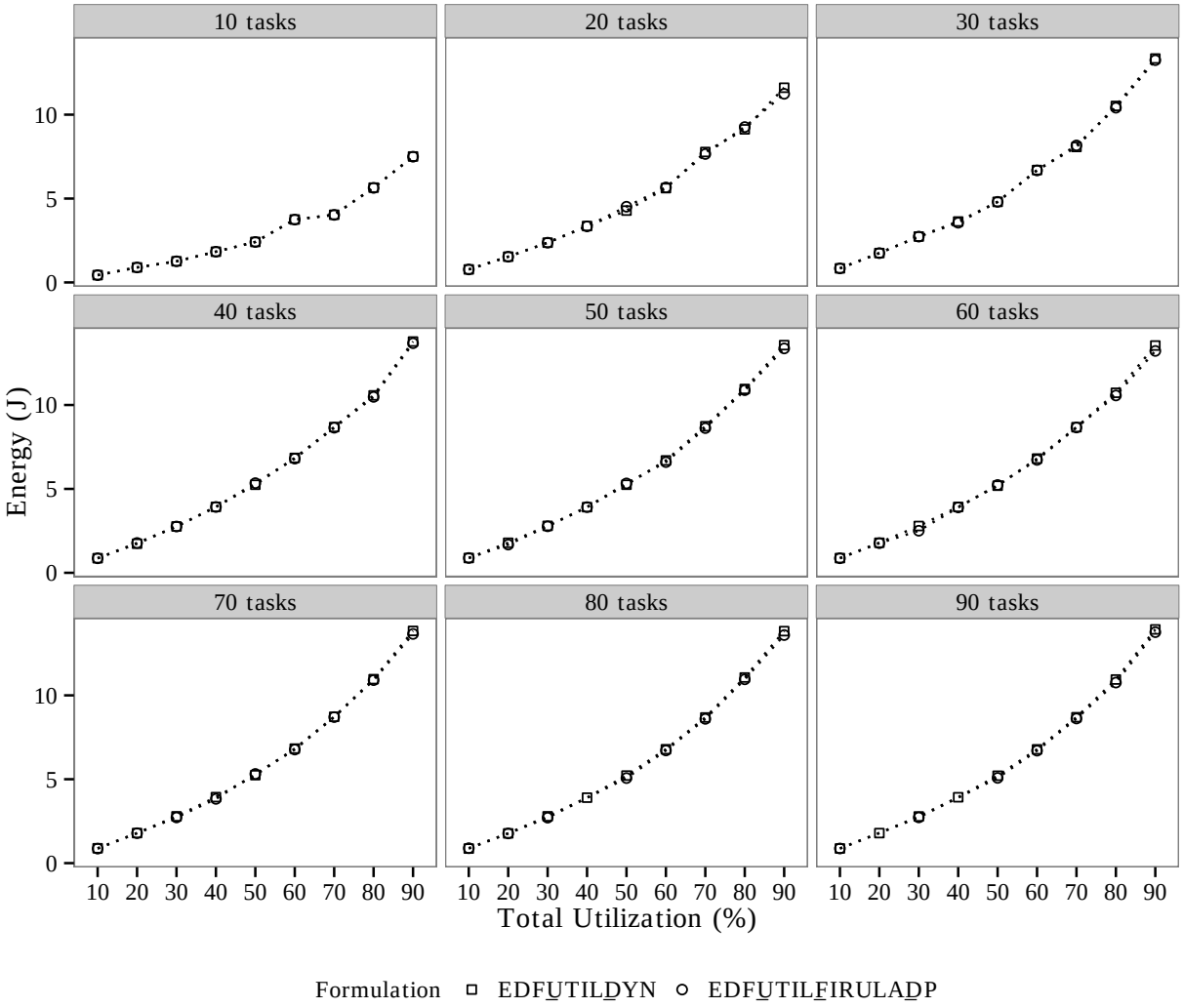


Figure 14 – System energy consumption of hard real-time allocations for Branch-and-Price (BP) and Branch-and-Cut (BC) strategies

used to deriving optimal solutions from their formulations, to the best of our present knowledge, this is the first work to report computational experiments on the search for optimal solution for this problem.

The typical formulation in the specialized literature is a 0/1 integer linear programming model which considers a continuous processor frequency domain and determines a single operating frequency per processor (Alahmad and Gopalakrishnan, 2011; Awan and Petters, 2013; Chen et al., 2011; Chen and Thiele, 2011; He and Mueller, 2012a). However, using the MGAP model is a more suitable fit to this problem because practical processors still use a discrete set of frequencies (Valentin et al., 2016b).

The adoption of DVFS is common in optimization procedures, such as task allocation, and frequency to task assignment. The aim is to find optimal energy-aware scheduling on heterogeneous platforms while considering individual task deadlines (Chen and Thiele,

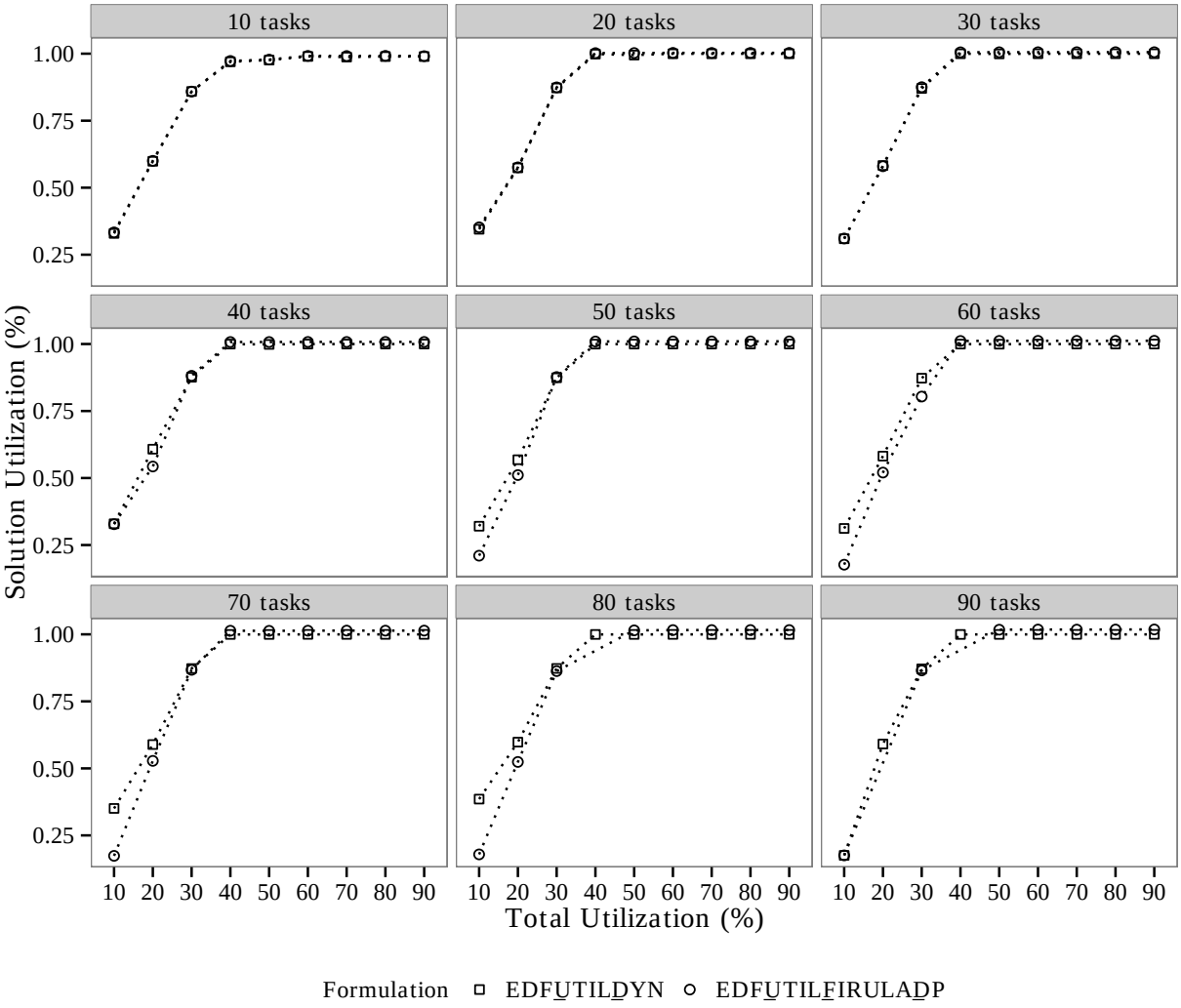


Figure 15 – System final utilization of hard real-time allocations for Branch-and-Price (BP) and Branch-and-Cut (BC) strategies

2011). Overall system energy reduction is due to workload split and to frequency minimization to meet tasks deadlines. Adoption of the well-known utilization based schedulability analyses is common (Alahmad and Gopalakrishnan, 2011; Awan and Petters, 2013; Chen et al., 2011; Chen and Thiele, 2011; Goossens et al., 2008; He and Mueller, 2012a,b; Prescilla and Selvakumar, 2013; Valentin and Barreto, 2010; Yang et al., 2009; Yu and Prasanna, 2003). The simplification on the model and on the solving process as using utilization constraints produces formulations similar to the multiple knapsack problem. However, none of the works targeting optimal solution uses a branch-and-price as the computational technique of resolution, to the best of our knowledge, this is the first work to apply branch-and-price in this context.

For the classic MGAP there are extreme fast algorithms. MGAP problem instances are solvable up to hundreds of machines with tens of speed levels, to map hundreds of tasks. For example, Osorio and Laguna (2003) proposed a Branch-and-Cut algorithm in

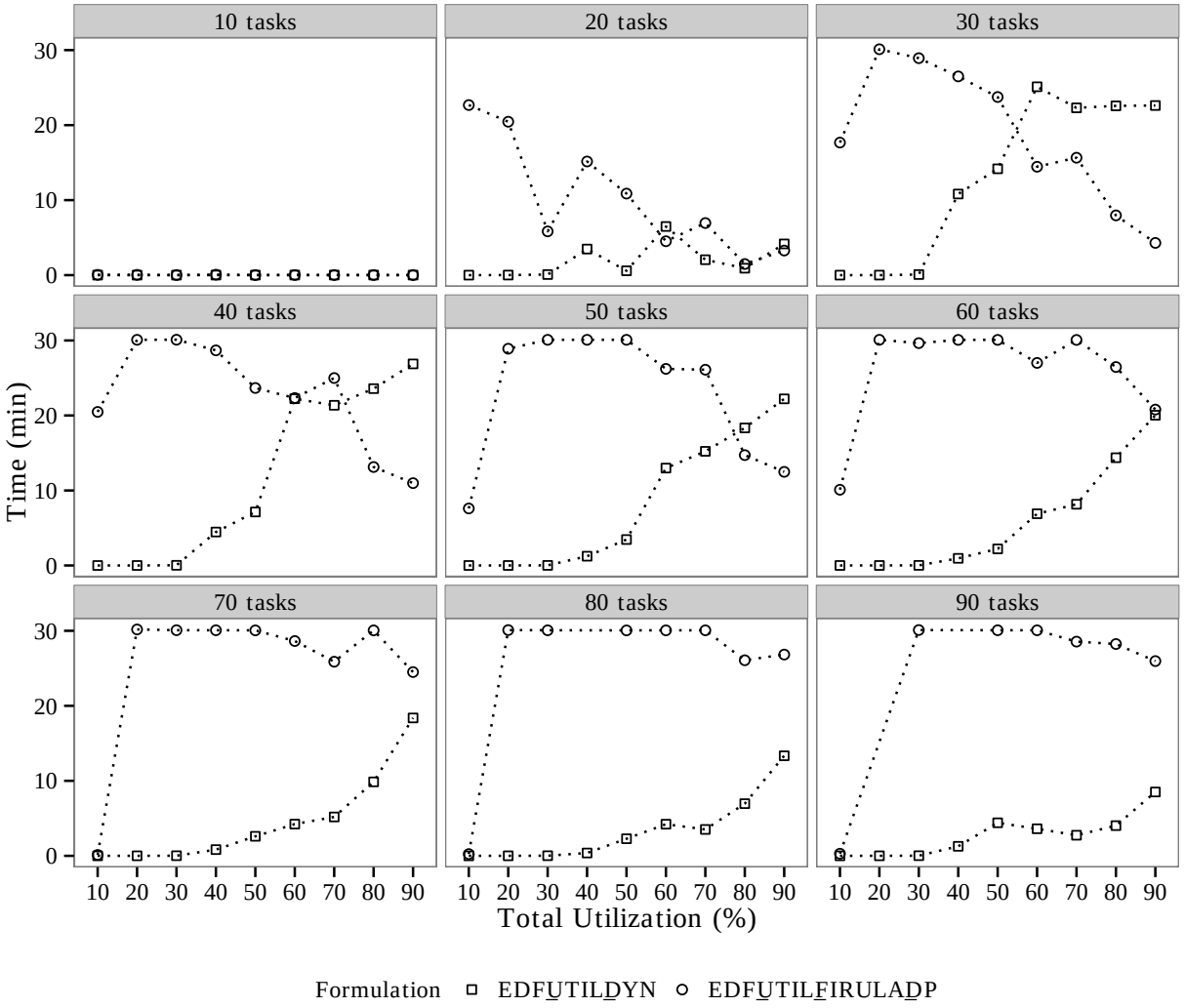


Figure 16 – Solver execution time for Branch-and-Cut (BC) and Branch-and-Price (BP) strategies

2003 and were able to solve instances up to 60 tasks, 30 machines, and two speed levels. Ceselli and Righini (2006), in 2006, proposed a Branch-and-Price strategy, solving up to 400 tasks, 80 machines and five levels. Another Branch-and-Cut algorithm proposed by Avella et al. (2013) can solve 200 tasks, 30 machines and five levels, for specific problem instances. We contribute with the present work with a new application of the MGAP model.

In this chapter, we compared one algorithm based on continuous processor frequency domain (BARREFORS) with a branch-and-price (BP) strategy, considering a discrete set of frequencies. The BP based algorithm produces system configurations with lower power consumption, but with the penalty of solver execution time, when compared to BARREFORS.

4.6 Chapter Summary

In this chapter, we make introduce a novel branch-and-price algorithmic strategy to solve the problem of how to find optimal hard real-time tasks distribution among heterogeneous processors respecting timing constraints and minimizing power consumption. Our study focuses on optimal solutions after reviewing the existing models. We have presented how to derive the master problem by reformulating the original formulation into one which suits the branch-and-price strategy. We also have discussed about the pricing problem, which is a version of the MCKP, and we have introduced a dynamic programming approach to solve the pricing problem.

Experimental results show that the BP based algorithm produces system configurations with lower power consumption, but with the penalty of solver execution time, when compared to the state-of-the-art strategy, BARREFORS. Our proposed BP based solver has same energy and final utilization curves as our branch-and-cut (BC) solver (see Chapter 3), as seen in most in our experiments.

5 Distribution of Dependent Hard Real-Time Tasks

In 2015, an estimated 1949 lives were saved by electronic stability control among passenger vehicle occupants (National Highway Traffic Safety Administration, 2017). But we still need to do better as around 5,615,000 traffic crashes occurred in the same year in the US only, causing 33,561 deaths (National Highway Traffic Safety Administration, 2016). Missing a single constraint may be catastrophic, as can be observed in the investigations performed in the unintended acceleration case of a model of a large automobile company. Investigators obtained and reviewed the source code for the 'sub-CPU', and they uncovered gaps and defects in throttle fail safes: "For one thing, by looking within the real-time operating system, the experts identified unprotected critical variables", says one of the experts (Yoshida, 2013). The existence of such cases calls attention in the industry and in the academia experts for designing robust solutions.

Robust methods, such as combinatorial mathematical formulations, can be of great help in the design phase of safety critical systems to cover for all constraints and optimize them, avoiding infeasible states. In this chapter, we demonstrate how to associate combinatorial optimization mathematical formulations and response time based schedulability analysis to optimally distribute hard real-time workload, considering precedence, preemption, mutual exclusion, timing, temperature, and capacity constraints, avoiding, thus, infeasible system configurations, such as in the unintended acceleration case.

The organization of this chapter is as follows. We present how a typical heterogeneous multi-core platform and how a hard real-time task model look like in Section 5.1 and Section 5.2, respectively. We show how a mathematical formulation can be written associating combinatorial optimization with schedulabitility analysis in Section 5.3. We also explain how such formulation can be solved using a robust branch-and-cut strategy in Section 5.4. We exemplify how such advanced techniques can be applied in a case study of an automotive workload in Section 5.5. Section 5.6 closes this chapter with final comments.

5.1 Processor Model

Semiconductor OEMs already produce specialized system-on-chips for the purpose of supplying hard real-time applications, such as the automotive market. For example, R-Car H3 heterogeneous multi-core platform by Renesas is capable of running a wide range of Advanced Driver Assistance Systems (ADAS) applications including surround

view, front camera, park assist, and sensor fusion on a single architecture (Renesas, 2017).

Practitioners execute applications with hard deadline restrictions on multiple heterogeneous processors due to the expected energy consumption reduction. Nevertheless, developing software with timing constraints for multiple heterogeneous processors is a complex task. Scheduling becomes especially hard to deal with, particularly under low power constraints.

Adopting multiple processing elements to enhance the computing capability and to reduce the power consumption is a common design strategy, especially for embedded systems. Besides, modern multicore processors for the embedded market are often heterogeneous in nature (Awan and Petters, 2013). Therefore, the heterogeneous multicore platforms have become the *de-facto* solution to cope with the rapid increase of system complexity, reliability, and energy consumption (He and Mueller, 2012a).

For this reason, a simple way to create a processing model is to use as reference a Multi-Processor System-On-Chip (MPSoC) architecture, such as R-Car H3 (Renesas, 2017), Samsung Exynos 5 (Samsung Electronics Co.Ltd., 2014), or Texas Instrument TDA3x (Texas Instruments, 2017). We can state then that the system is composed by a set, $\mathcal{H}$, of m processors, $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$. Each core may operate on l different performance states, $1 \leq k \leq l$. The frequency of performance state k on the processor i is F_{ik} and the power consumption is P_{ik}. The set of frequencies of one core is not necessarily the same of other cores. Also, a task may have different code size and execution time for different processors, due to instruction set and performance state differences. The idle power of processor i is $P_{idle,i}$.

5.2 Task Model

A typical hard real-time workload can be represented by a task model of periodic tasks. A task model $\mathcal{M}$ is a set composed by n tasks τ_j. A task $\tau_j \in \mathcal{M}$, with $1 \leq j \leq n$, has the properties: worst-case execution cycle $WCEC_j$; worst-case execution time $C_j(f)$, which is a function of frequency f, thus $C_j(f) = \frac{WCEC_j}{f}$; period of execution T_j; deadline D_j. A task τ_j also has the following properties, specific to fixed priority policies: fixed priority p_j; and set of high priority tasks $hp(j)$ representing the tasks τ_p with a priority higher than the priority of τ_j. The response time R_j is dependent not only on task set characteristics, but also on the target platform, on the task allocation, and on the frequency distribution that have been selected for the workload. A task model can be locally processed in a single processor using a fixed priority based on-line scheduler, such as Deadline Monotonic.

Deadline Monotonic (DM) is a fixed priority based on-line scheduler in which task priorities decrease with larger deadlines. Audsley et al. (1993) extend the schedulability

test proposed by Lehoczky et al. (1989) for DM, considering the release jitter J_j and the local blocking delay B_j due to semaphore usage. The delay B_j caused by low priority tasks accessing shared resources in the same processors using Priority Ceiling Protocol can be estimated as $B_j = max_{jk}\{D_{jk}|(p_j < p_i) \wedge (C(S_k) \geq p_i)\}$, where $C(S_k)$ is the ceiling priority of the shared resource S_k. The schedulability test proposed by Audsley is $R_j \leq D_j, \forall 1 \leq j \leq n$, where $R_j = I_j + J_j$.

The task influence I_j in multiple processors may be calculated as $I_j^{n+1} = C_j + B_j^r + B_j + \sum_{p \in hp(j)} \left\lceil \frac{I_j^n + J_p + B_p^r}{T_p} \right\rceil \times C_p$. Precedence constraints can be represented by including the maximum response time of the predecessors tasks in the J_j component of the task τ_j. Also, when precedence constraints occur across different processors, this imposes an additional messaging cost that may be incorporated in the emitting task to perform inter-processor communication. When the Multiprocessor Priority Ceiling Protocol is in place to avoid priority inversion issues, the remote blocking delay B_j^r is an upper bound for the blocking time suffered by task τ_j from other tasks in a different processor. Response time tests are computationally expensive but provide *exact* conditions, i.e., *sufficient* and *necessary*. The test uses task's WCEC, periods, and the concept of critical instant phasing (Lehoczky et al., 1989).

5.3 Mathematical Formulations for Dependent Tasks

A classical mathematical model that resembles modern heterogeneous multicore platforms is the Multilevel Generalized Assignment Problem (MGAP) (Glover et al., 1979), though it was originally conceived in the manufacturing context. The MGAP consists of minimizing the assignment cost of a set of jobs to machines, each having associated therewith a capacity constraint. Each machine can perform a job with different performance states that entail different costs and amount of resources required. The MGAP is originally in the context of large manufacturing systems as a more general variant of the well-known Generalized Assignment Problem (GAP). In this chapter, we correlate MGAP model with the problem of assigning frequencies and distributing hard real-time tasks on heterogeneous processors, minimizing energy consumption.

Considering the schedulability test proposed by Audsley, we propose the MGAP formulation using tasks response times as seen in Equation 5.1, based on the formulation of Valentin et al. (2016b).

$$\text{Minimize } \Psi(x) \tag{5.1a}$$

$$\text{s.t.: } \sum_{i=1}^{m} \sum_{k=1}^{l} x_{ijk} = 1, \; j \in \{1, \ldots, n\} \tag{5.1b}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{l}\frac{WCEC_{ij}}{F_{ik}T_j}x_{ijk} \leq 1, i \in \{1,\ldots,m\} \tag{5.1c}$$

$$\psi_i \leq \frac{\kappa_i^{max} - \kappa_{amb}}{\rho},\ i \in \{1,\ldots,m\} \tag{5.1d}$$

$$R_j \leq D_j,\ j \in \{1,\ldots,n\} \tag{5.1e}$$

$$x_{ijk} \in \{0,1\}, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l \tag{5.1f}$$

where the tri-indexed decision variable x_{ijk} represents the distribution and assignment, i.e. when $x_{ijk} = 1$ the task τ_j executes in the processor i at performance state k, or frequency F_{ik}; when $x_{ijk} = 0$, the task τ_j is distributed somewhere else. A distribution is a partitioned approach in which each processor i executes a local scheduler responsible for a partition of the real-time task workload and migration is not allowed (see the set of constraints 5.1b). The set of constraints 5.1c represents the maximum system utilization capacity of each processor i. The set of constraints 5.1d represents the temperature limits by creating a linear relation, where κ_{amb} is the ambient temperature, κ_i^{max} is the maximum junction temperature of each processor i, ρ is thermal resistance constant, and ψ_i is power consumption of each processor i. This formulation applies each task deadline as a constraint against their response time in the linear programming (see the set of constraints 5.1e). The matrix R_j is the response time of tasks τ_j for a given allocation configuration. The response time of each task varies depending on the workload distribution and the frequency assignment of the configuration because a change in the value of x_{ijk} may result in a different computation time (C_i). Equation 5.1 is applicable for DM scheduling policy ($D_j \leq T_j$).

We are using an objective function $\Psi(x)$ that minimizes energy consumption, accounting dynamic and idle energy, over the time window represented by the hyperperiod of the real-time tasks, i.e., the Least Common Multiple (LCM) of tasks periods. We extend the objective functions presented by Valentin et al. (2016b) by improving the idle energy estimation. Equation 5.2 has the objective function.

$$\text{Minimize } \Psi(x) = \sum_{i=1}^{m}\left(E_{dyn,i}(x) + E_{idle,i}(x)\right) \tag{5.2a}$$

$$E_{dyn,i}(x) = \sum_{j=1}^{n}\sum_{k=1}^{l}\left(\left(\frac{LCM}{T_j}\right)C_l WCEC_{ij} V_{dd,ik}^2 x_{ijk}\right) \tag{5.2b}$$

$$E_{idle,i}(x) = P_{idle,i} LCM \left(1 - \sum_{j=1}^{n} \sum_{k=1}^{l} \frac{WCEC_{i,j}}{F_{ik} T_j} x_{ijk}\right) \tag{5.2c}$$

where $E_{dyn,i}$ is the energy consumption when processor i is active, $E_{idle,i}$ is the energy consumption when processor i is idle, $\frac{WCEC_{ij}}{F_{ik}T_j}$ represents the task τ_j utilization, u_{ijk}, while executing in processor i at frequency F_{ik} of performance state k, C_l is the circuit capacitance constant, and $V_{dd,ik}$ is the voltage level to achieve frequency F_{ik}.

The term $\left(\frac{LCM}{T_j}\right) C_l WCEC_{ij} V_{dd,ik}^2 x_{ijk}$ represents the dynamic energy associated with the instances of execution of task j within the LCM. Each processor idle energy, within the LCM time window, is computed for its estimated idle time in the term $P_{idle,i} LCM \left(1 - \sum_{j=1}^{n} \sum_{k=1}^{l} \frac{WCEC_{i,j}}{F_{ik}T_j} x_{ijk}\right)$.

The objective function represented in Equation 5.2 may still be seen as a MGAP formulation. Note that, without loss of generality, when we take the term $P_{idle,i} LCM$ out of the sum, leaving the term $mP_{idle,i} LCM$ to be added to the final objective function value, we have $c_{ijk} = \left[\left(\frac{LCM}{T_j}\right) C_l WCEC_{ij} V_{dd,ik}^2 - P_{idle,i} LCM \left(\frac{WCEC_{i,j}}{F_{ik}T_j}\right)\right]$.

5.4 Computational Technique of Resolution

We use a general branch-and-cut exact implicit enumeration method combined with schedulability tests to conduct the process of finding optimal solutions. A branch-and-cut is a branch-and-bound with cut generation strategies. The algorithm's input is the processing model $\mathcal{H}$, the desired task model $\mathcal{M}$, and a possible upper bound ub. The algorithm outputs the optimal distribution of hard real-time tasks among the processors that consumes the least power among the possible assignments, informing as well in which frequency each task may be executed, and the total system estimated energy. The general solving strategy is listed in Algorithm 5.

The algorithm starts by denoting the set L of active problem nodes to contain only the initial Integer Linear Problem. When an upper bound ub is known, v^* and the optimal solution x^* are set to match ub objective function value and solution structure, otherwise, they are set to $+\infty$ and to $NULL$, respectively. The algorithm iteratively evaluates each element of the set L. Each problem node is initially tested against the schedulability test that fits for the problem scheduling policy. In cases where the schedulability test accepts the node, then a regular branch-and-cut is followed. The linear relaxation of the node is then computed and solved. When the linear relaxation is feasible, a procedure of generation of cutting planes is performed and followed by a fathoming and pruning process. The problem node is then partitioned and new restricted problem nodes are

Algorithm 5 Branch-and-Cut (B&C) for Dependent Tasks

```
1: Input: H, M, ub
2: Output: optimal solution (x*, v*)
3: v* ← ub.val; x* ← ub.structure;
4: L ← ILP^0;
5: while L != ∅ do
6:    n ← remove_node(L);
7:    if !schedulability(n) then
8:       continue;
9:    end if
10:   lp ← relaxation(n)
11:   (x, v) ← solve(lp);
12:   if x = infeasible then
13:      continue;
14:   end if
15:   p ← cut_planes(x, v);
16:   if p! = ∅ then
17:      add(p, lp);
18:      goto 11;
19:   end if
20:   if v ≥ v* then
21:      continue;
22:   end if
23:   if x is integer then
24:      v* ← v;x* ← x;
25:      continue;
26:   end if
27:   partition(lp);
28: end while
29: return (x*, v*);
```

derived and incorporated into L. The iterative process repeats until the set L is empty.

5.5 Case Study: a Cruiser and Collision Detector

In this section, we exemplify how to optimally distribute the hard real-time workload of an Advanced Driver Assistance Systems (ADAS) in a target platform based on R-Car H3. Assisting the driver on safety-related operations of a vehicle is vital because maneuvering a vehicle is a complex and time-critical task. The assistance, highly based on sensors, mainly vision based, may provide information to the driver from the surrounding environment and at times may take control of the vehicle to avoiding hazards. The ADAS application we are considering can be seen as simple control application to perform a Cruiser mode with Collision Detection. Table 8 summarizes the task model of this exam-

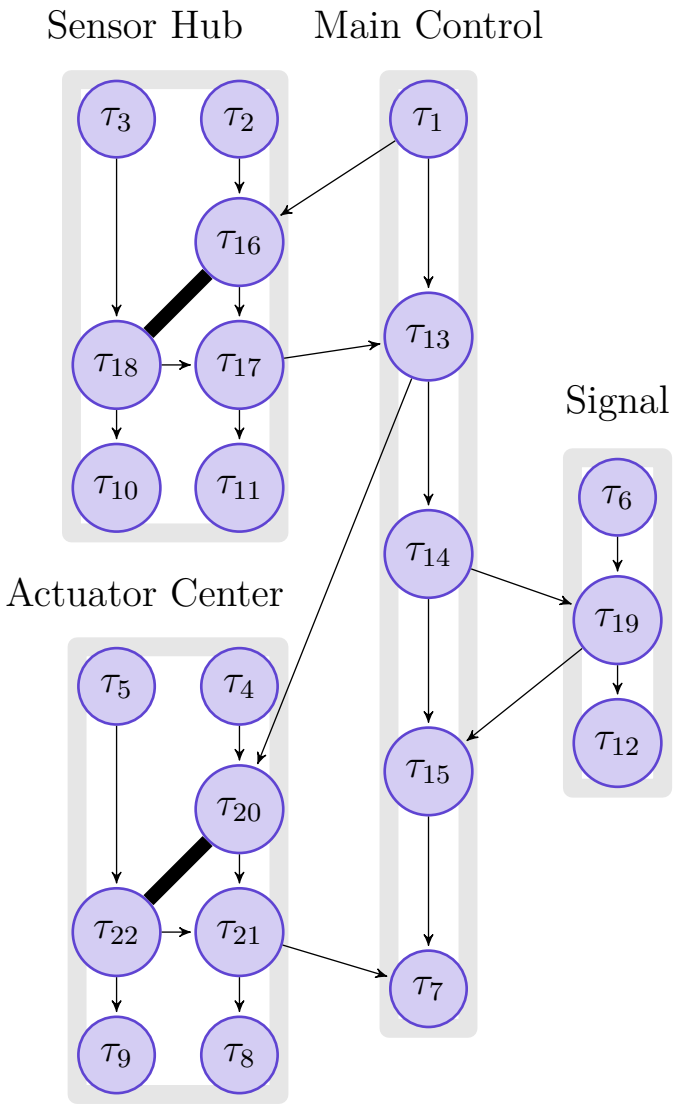


Figure 17 – Precedence Graph of Cruiser with Collision Detection Control. Arrows represent a precedence constraint, for example, τ_1 precedes τ_{13}. Dark thick edges represent mutual exclusion constraint, for example, τ_{16} shares a resource with τ_{18}

ple. We are considering precedence, preemption, mutual exclusion, temperature, capacity, and timing constraint while distributing the workload. We illustrate the precedence constraints (thin arrows) and mutual exclusion constraints (dark thick edges) of this task model in the precedence graph of Figure 17.

The application example we consider, the Cruiser mode with Collision Detection, is composed of four logical activities that communicate among themselves: the Main Control activity, the Sensor Hub activity, the Actuator Center activity, and the Signal activity. The Main Control activity is responsible for managing the overall control system and communicating with the other activities. The Sensor Hub activity monitors speed and detects collisions, the Actuator Center activity is in charge of controlling the speed, and

the Signal activity reports and records any significant event detected in the system.

The Main Control activity always starts by requesting (τ_1) data from the Sensor Hub. Current speed is then sent back to the Main Control (τ_{17}). The Main Control computes any needed speed adjustments to achieve the Cruiser targeted speed and communicates with Actuator Center (τ_{13}) to implement any acceleration (τ_{20}) or brakes (τ_{22}) needed. At the same time, similarly, the Sensor Hub sends collision detection data to the Main Control, which communicates with Actuator Center to perform the necessary breaking. Main Control also sends (τ_{14}) regular reports of the events that happen in the control system to the Signal activity, which is responsible for activating alarms and warnings.

As an example platform, we are considering four processors: two ARM A57's and two ARM A53's. The ARM A57's may operate on seven different frequencies from 500 MHz to 1.9 GHz, and the A53's may operate on seven different frequencies from 400 MHz to 1.2 GHz. The idle power consumption is 50 mW. The circuit capacitance constant C_l is $1 \times 10^{-9}\,\mathrm{W\,V^2\,Hz^{-1}}$. The thermal resistance ρ is $0.11\,^{\circ}\mathrm{C\,W^{-1}}$. In this platform, we are considering the DVFS switching latency as an operation executed within the context switch of tasks with a cost of 30 ms, included in the release jitter J_j of each task. More robust response time analysis considering the switching overhead in clusters and architecture influence (Valentin et al., 2015b) may be also combined with the branch-and-cut algorithm when necessary. We list the platform characteristics in Table 9.

Even though temperature is a constraint left for mechanical engineering, it can play a role while distributing the system workload. The ambient temperature in a vehicle will typically be higher than the regular room temperature (25 °C) because the system is exposed to heat flowing from the mechanical engines, reaching as high as 85 °C. The common silicon junction temperature is 125 °C.

After executing the branch-and-cut optimization algorithm considering the task model of Table 8 and the R-Car H3 based model of Table 9, we obtain the optimal distribution listed in Table 10. The precedence graph with the allocation is also illustrated in Figure 18. For this case study, the optimal energy consumption is 0.1049 J for the duration of the LCM (200 ms) of tasks periods. We initialized the algorithm with the solution structure and an upper bound for the objective function extracted from the logical feasible distribution in which all tasks of each activity are present in the same processor. Utilizing this initial upper bound, the full optimization process took less than 1.65 h to finish and the final optimal solution differs from the logical initial distribution. Even though this case study has a set of 22 tasks, this algorithm has a reasonable performance on task models with up to 50 tasks, finishing in less than 30 minutes with a feasible solution for independent tasks (Valentin et al., 2016a).

As seen in Table 10, the schedulability analysis shows that the computed response time of each task is less than their respective deadline, meeting all timing, precedence, and

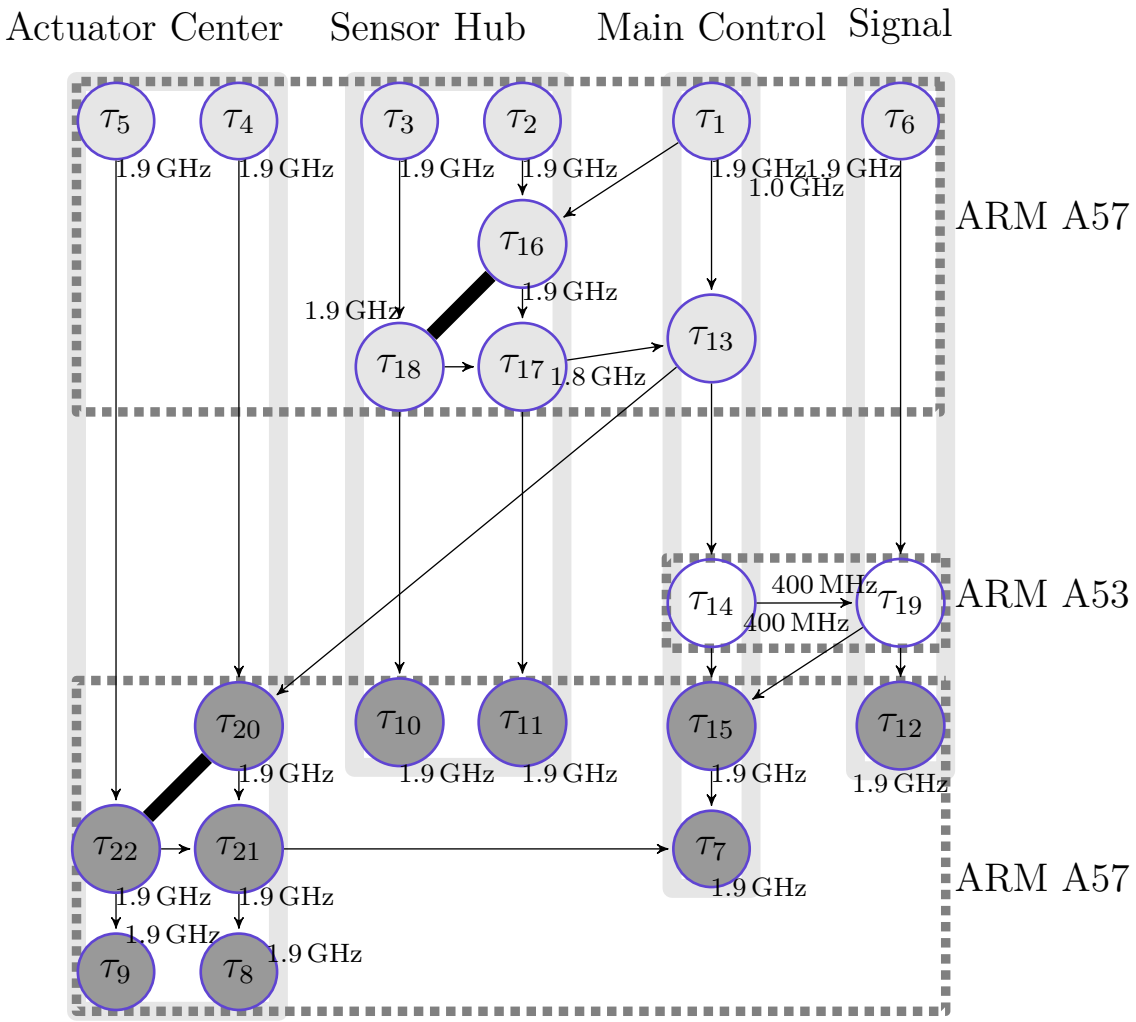


Figure 18 – Precedence Graph and Task Distribution of Cruiser with Collision Detection Control. White nodes are allocated in one ARM A53. Light gray nodes are allocated in one ARM A57. Dark gray nodes are allocated in the other ARM A57. The frequency that each task executes is represented close to each respective node in the graph, for example, τ_{19} executes at 400 MHz

mutual exclusion constraints. It is worth noting that Table 10 includes the inter-processor communication cost of tasks $\tau_1, \tau_4, \tau_5, \tau_6, \tau_{13}, \tau_{14}, \tau_{17}, \tau_{18}$, and τ_{19}. The optimization process converged to an optimal solution in which tasks sharing resources are allocated in the same processor, avoiding remote blocking delays. The optimal configuration for this case study uses only three of the four available processors. The total utilization of the active processors (6.05 %, 16.98 %, and 25.00 %) is well within their respective theoretical values (100 %), safely respecting the capacity constraint. This configuration with low utilization is selected by the algorithm because it consumes the least energy, although it is common practice to design real-time systems with high utilization. Also, the estimated temperature of each ARM processors is less than 87 °C, in the thermal stabilization, giving enough room in the temperature constraint.

We highlight, for example, that the logical distribution setting each application activity to one processor is also feasible. This configuration uses all four processors at their respective maximum frequency. The timing, preemption, precedence, and mutual exclusion constraints are met, given that each task response time is less than their respective deadline. The capacity and temperature constraints are also met. However, this configuration's estimated total system energy is 0.1127 J for the LCM (200 ms) of tasks periods, being at least 7.4 % higher than the optimal.

An intuitive approach would be to target a low power configuration, having all tasks allocated to a single ARM A53 CPU, executing at the lowest frequency of 400 MHz. That, however, is not a feasible configuration, given that the capacity constraint is not met because the total CPU utilization would be 117.5 % and several tasks would not meet their deadlines in this situation.

Another intuitive approach would be to use again the logical distribution of one activity to one processor, but locking the lowest available frequency, as the utilization of each processor is not high. In this configuration, each processor utilization is less than 32 %, but the system is not schedulable because the response time analysis indicates that tasks $\tau_7, \tau_8, \tau_{10}, \tau_{14}, \tau_{15}, \tau_{18}$, and τ_{21} miss their respective deadlines basically due to the accumulated precedence.

5.6 Chapter Summary

In this chapter, we described how to optimally distribute the hard real-time workload into a heterogeneous multicore platform. We used as an example a case study from an automotive system. We applied robust methods to avoid infeasible system configurations. Even though they can be computationally expensive, their usage in design time is still justified, given that they help prevent catastrophic scenarios, such as the unintended acceleration case.

We associated combinatorial optimization mathematical formulations and response time based schedulability analysis to optimally distribute hard real-time workload. We solved the combinatorial problem by using a branch-and-cut algorithm that applies response time analysis while walking through the problem nodes. We showed that all the considered constraints of precedence, preemption, mutual exclusion, timing, temperature, and capacity were met properly in our case study by using the response time analysis with branch-and-cut combined method.

Table 8 – An example of automotive hard real-time task model

τ_i	p_i	T_i(ms)	D_i(ms)	WCEC ($\times 10^3$)			
				0	1	2	3
1	1	200.0	100.0	3000	3000	3000	3000
13	2	200.0	100.0	15000	15000	15000	15000
14	3	200.0	100.0	10000	10000	10000	10000
15	4	200.0	100.0	1000	1000	1000	1000
7	5	200.0	100.0	2000	2000	2000	2000
3	6	200.0	40.0	3000	3000	3000	3000
16	7	200.0	200.0	5000	5000	5000	5000
17	8	200.0	100.0	7000	7000	7000	7000
2	9	200.0	200.0	3000	3000	3000	3000
11	10	200.0	200.0	2000	2000	2000	2000
18	11	200.0	40.0	6000	6000	6000	6000
10	12	200.0	40.0	2000	2000	2000	2000
4	13	200.0	100.0	3000	3000	3000	3000
5	14	200.0	100.0	3000	3000	3000	3000
20	15	200.0	100.0	2000	2000	2000	2000
22	16	200.0	100.0	7000	7000	7000	7000
21	17	200.0	100.0	1000	1000	1000	1000
9	18	200.0	100.0	2000	2000	2000	2000
8	19	200.0	100.0	2000	2000	2000	2000
6	20	200.0	200.0	3000	3000	3000	3000
19	21	200.0	200.0	10000	10000	10000	10000
12	22	200.0	200.0	2000	2000	2000	2000

Table 9 – Architecture characteristics of a typical Automotive platform

CPU	C_l(WV^2/Hz)	κ_i^{max}(°C)	ρ(°C/W)	Voltages (V)	Frequencies (GHz)
0	1e-09	125	0.11	0.94 0.86 0.86 0.78 0.77 0.77 0.77	1.9 1.8 1.7 1.6 1.5 1.0 0.5
1	1e-09	125	0.11	0.94 0.86 0.86 0.78 0.77 0.77 0.77	1.9 1.8 1.7 1.6 1.5 1.0 0.5
2	1e-09	125	0.11	0.82 0.82 0.7825 0.7575 0.7075 0.6825 0.6575	1.2 1.1 1.0 0.9 0.8 0.7 0.4
3	1e-09	125	0.11	0.82 0.82 0.7825 0.7575 0.7075 0.6825 0.6575	1.2 1.1 1.0 0.9 0.8 0.7 0.4

Table 10 – Optimal workload distribution result of the optimization process

Processor: 0	Utilization: 6.05 %	Temperature: 86.85 °C				
τ_i	WCEC ($\times 10^3$)	Frequency(GHz)	Computation(ms)	T_i(ms)	D_i(ms)	R_i(ms)
15	1000	1.9	0.526	200.0	100.0	97.331
20	2000	1.9	1.053	200.0	100.0	48.684
22	7000	1.9	3.684	200.0	100.0	13.219
21	1000	1.9	0.526	200.0	100.0	49.797
7	2000	1.9	1.053	200.0	100.0	98.443
8	2000	1.9	1.053	200.0	100.0	52.458
9	2000	1.9	1.053	200.0	100.0	18.512
10	2000	1.9	1.053	200.0	40.0	28.513
11	2000	1.9	1.053	200.0	200.0	38.019
12	2000	1.9	1.053	200.0	200.0	108.909
Processor: 1	Utilization: 16.98 %	Temperature: 86.73 °C				
τ_i	WCEC ($\times 10^3$)	Frequency(GHz)	Computation(ms)	T_i(ms)	D_i(ms)	R_i(ms)
1	3000	1.9	1.579	200.0	100.0	1.609
2	3000	1.9	1.579	200.0	200.0	3.188
3	3000	1.9	1.579	200.0	40.0	4.767
4	3001	1.9	1.579	200.0	100.0	6.346
5	3001	1.9	1.579	200.0	100.0	7.926
6	3001	1.9	1.579	200.0	200.0	9.505
16	5000	1.9	2.632	200.0	200.0	12.198
18	6001	1.9	3.158	200.0	40.0	18.483
17	7001	1.9	3.685	200.0	100.0	26.936
13	15002	1.0	15.002	200.0	100.0	46.707
Processor: 2	Utilization: 25 %	Temperature: 85.04 °C				
τ_i	WCEC ($\times 10^3$)	Frequency(GHz)	Computation(ms)	T_i(ms)	D_i(ms)	R_i(ms)
14	10001	0.4	25.003	200.0	100.0	71.739
19	10002	0.4	25.005	200.0	200.0	96.774
Processor: 3	Utilization: 0.0 %	Temperature: 85.005 °C				
τ_i	WCEC ($\times 10^3$)	Frequency(GHz)	Computation(ms)	T_i(ms)	D_i(ms)	R_i(ms)

6 Response Time Schedulability Test for Multicore Platforms

This chapter presents a new multicore platform schedulability analysis that accounts for architecture overhead. We take the cluster-based platform as driving architecture and derive the test based on DVFS switching overhead estimations.

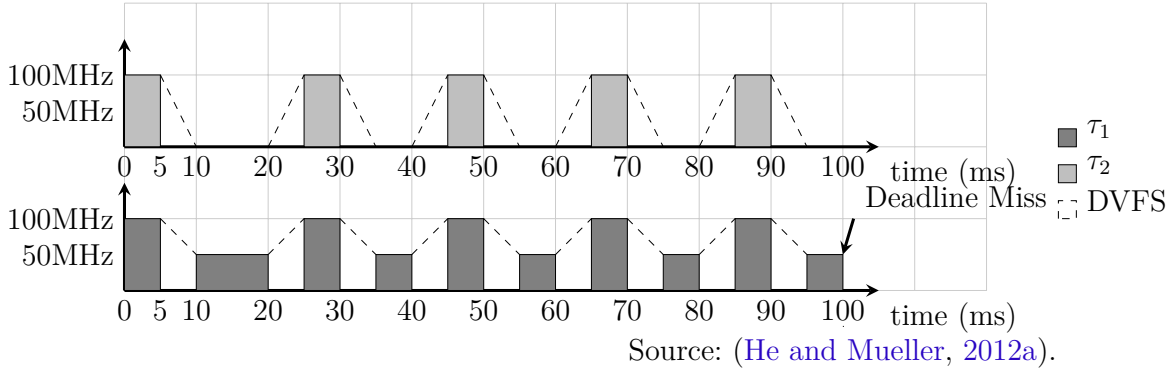
6.1 Motivational Example

A simple example of the problem is the situation of a single cluster composed by two processors executing two tasks. Both processors support 50 MHz and 100 MHz clock frequencies. Let us suppose that the worst-case latency to switch from one clock frequency to another is 5 ms. The optimal allocation determines the first task to run at 50 MHz in the first processor; the second task to run at 100 MHz in the second processor. The task model states that $T_1 = 100$ ms, $C_1 = 90$ ms, $T_2 = 20$ ms, $C_2 = 5$ ms.

The utilization analysis on *each processor* evaluates the system as schedulable. The utilization of τ_1 is $u_1 = \frac{90}{100} = 0.9$. According to Liu and Layland bound, the total utilization on the first processor $U^1_{total} = u_1 = 0.9$, and the utilization bound is $U^1_{bound} = 1(2^{\frac{1}{1}} - 1) = 1$; i.e., $U^1_{total} < U^1_{bound}$. Similarly, $u_2 = \frac{5}{20} = 0.25$, $U^2_{total} = u_2 = 0.25$, and $U^2_{bound} = 1(2^{\frac{1}{1}} - 1) = 1$; i.e., $U^2_{total} < U^2_{bound}$. The original Liu and Layland bound states that this system is schedulable on each core.

Performing the extended test proposed by Audsley, assuming the switching latency is part of the release jitter, we end with the following response time analysis. For τ_1: $I_1 = 90+0+0 = 90$ ms and $R_1 = 90+5ms = 95$ ms ≤ 100 ms. For τ_2: $I_2 = 5+0+0 = 5$ ms and $R_2 = 5 + 5ms = 10$ ms ≤ 20 ms.

The response time analysis on each core evaluates to a schedulable system. However, due to DVFS switching latency, the system is unschedulable. The execution of each task is illustrated on Figure 19. Between instant 0 ms and 5 ms, the two tasks have a speed conflict, and they execute at the 100 MHz to satisfy task τ_2 requirement. At instant 5 ms, τ_2 finishes and τ_1 may return to 50 MHz speed. The change requires 5 ms of switching latency. At 20 ms, τ_2 becomes active again. Activating τ_2 requires switching back to 100 MHz and additional 5 ms of overhead. The cumulative switching overhead leads the system to an unschedulable situation in which τ_1 misses its deadline at instant 100 ms. The problematic present in this example is first observed in the work reported by He and Mueller (2012a) and is known as **Inter-Core Preemption**.



Source: (He and Mueller, 2012a).

Figure 19 – Example of deadline miss in cluster based platforms

6.2 Architecture latency for response time schedulability analysis

We extend the response time schedulability analysis equation by introducing the concept of architecture latency.

Definition 1 *An Inter-Core Preemption of a task τ_i is a preemption of its execution due to core speed change. The arrival or the completion of another task τ_j may cause the core speed change. The condition is τ_i and τ_j are in the same cluster but on different cores (He and Mueller, 2012a).*

Definition 2 *Architecture Latency A_i of a task τ_i is any delay suffered due to hardware latency.*

Therefore, we can rewrite Equation 2.3 as follows:

$$R_i \leq D_i, \qquad \forall 1 \leq i \leq n, \text{ where:} \tag{6.1}$$

$$R_i = I_i + J_i \tag{6.2}$$

$$I_i^{n+1} = C_i + B_i + A_i + \sum_{j \in hp(i)} \left\lceil \frac{I_i^n + J_j}{P_j} \right\rceil \times C_j \tag{6.3}$$

For cluster-based platforms, He and Mueller (2012a) proposed an estimation of switching overhead due to Inter-Core preemption. However, the estimation composes an imprecise schedulability analysis. Thus, for cluster-based platforms, we recommend having the architecture latency defined as the same estimation as in He and Mueller (2012a) and composing a response time schedulability analysis. Therefore, we propose reusing the estimation as in D. He and defining A_i as $A_i = 2 \times Lp + num_i \times Lp$, where Lp is the maximum frequency and voltage switching overhead and the computation of num_i is $num_i = \sum_{\tau_j} \left\lceil \frac{T_i}{T_j} \right\rceil \times 2$, where τ_j satisfy Definition 1 (He and Mueller, 2012a).

Let us reconsider the motivational example. We compute the response time analysis for τ_1 as follows: $I_1 = 90 + 0 + 60 + 0 = 150\,\text{ms}$ and $R_1 = 150 + 0 = 150\,\text{ms} > 100\,\text{ms}$. Therefore, the system is unschedulable.

6.3 Argumentation

Theorem 1 *Given periodic tasks $\tau_1, \tau_2, \ldots, \tau_n$, and a distribution of tasks to cores and frequency assignments on a cluster-based platform, the entire task set is schedulable if and only if $R_i \leq D_i, \forall i \leq n$, where R_i is as in Equation 6.1.*

Proof: We restrict our attention to the critical instant, because it is the worst-case situation, as proven by Liu and Layland (1973). Lehoczky et al. (1989) proved for the situation of perfect preemption, no blocking, no overhead, and jobs are ready at their initiation times. Audsley et al. (1993) demonstrated for situation considering the usage of semaphores and the existence of release jitter. The proof for the situation in which the DVFS switching latency Lp is non-negligible is trivial because A_i is a constant value for a given task model and task to processor and frequency assignment.

Theorem 2 *The time complexity of an algorithm to compute Equation 6.1 for one processor is $O(n^2 \times max(r, p))$; where n is the number of tasks; r is the ratio between the largest period and smallest period: $r = \frac{max(T_i)}{min(T_i)}$; and p is the number of processors per cluster.*

Proof: It follows that the component A_i has a constant influence in the convergence of the iterative Equation 6.1. For this reason, the component A_i is a constant value. Computing A_i has $O(n^2 \times p)$ time complexity, where p is the number of processors per cluster. We can calculate all A_i's before the iteration process. Therefore, we can reduce the time complexity of the iterative Equation 6.1 to the same as Lehoczky and Audsley of Table 2. According to AlEnawy and Aydin (2005), the time complexity of Lehoczky is $O(n^2 \times r)$. Thus, the time complexity of Equation 6.1 is $O(n^2 \times max(r, p))$.

6.4 Empirical Experiments

In this section we present the computational experiments. We discuss experiment design, results, and discussion in Section 6.4.1, Section 6.4.2, and Section 6.4.3, respectively.

6.4.1 Experiment design

The performance evaluation of the proposed schedulability test uses random generated task models by a simple simulator program, *esim*[1]. The simulator also produces random generated task to processor and frequency assignments. The simulator uses the API provided by Akaroa2 (Pawlikowski et al., 1994) to implement random variables, pseudo-random number generators, and statistical analysis and treatment in the course of our quantitative discrete-event stochastic simulation. The random-generated task models and task assignments are then processed by the module with the schedulability analyses and the results are sent to the Akaroa2 back-end for statistical analysis.

The implementation of the schedulability analysis returns one when the test evaluates the random-generated task model and assignment as schedulable; the return value is equal to zero when unschedulable. We estimate the mean value of the difference between the result produced by the response time schedulability analysis presented in Section 6.2 and the other evaluated tests (Audsley et al., 1993; He and Mueller, 2012a; Liu and Layland, 1973). In these experiments, we consider our proposal as reference. The tests agree on the schedulability of the given sample when the difference is equal to zero. The compared test produces a false positive error if the difference is less than zero. The compared test produces a false negative error when the difference is greater than zero.

We also estimate the execution time of each schedulability test using wall clock. The machine used to perform the simulation experiments is an Intel Core i5 executed at 1.7 GHz. The machine has 4 GB of DDR3 memory executed at 1.6 GHz. The operating system is OS X 10.9.2.

In all experiments, we configure Akaroa2 to achieve 99 % of statistical confidence ($1 - \beta = 0.01$) and significance of 0.05 ($\alpha = 0.05$). We use the Schruben algorithm (Pawlikowski, 1990) to remove transient samples. We configure Akaroa2 to apply Spectral statistic analysis (Pawlikowski, 1990) in the steady state simulation phase. The seed is 2^8.

We experiment on two types of platforms. The first has one single cluster composed by XScale processors. We extend the first platform with a secondary cluster to compose the second platform. The secondary cluster contains powerful processors. We adopt the power model listed in Table 11. The first cluster uses only processors of type 1 and the second cluster uses processors of type 2. The switching latency is 450 µs. We also vary each platform to contain 4, 8, and 16 processors per cluster. The energy consumption is estimated during the LCM of tasks' periods. We estimate the system energy consumption

[1] esim is currently available via request to authors.

Table 11 – Processor power model

Processor Type	Idle Power p_{idle} (mW)	Frequency (MHz)	Dynamic Power p_{dyn} (mW)
1	260	624	925
		520	747
		416	570
		312	390
		208	279
2	600	1500	1700
		1200	1100
		800	990
		750	950
		600	900

as: $E_{system} =$

$$\sum_{k=1}^{M}\sum_{j=1}^{K}\sum_{i=1}^{n_{k,j}}\sum_{f=1}^{F}\left\lfloor\frac{LCM}{T_i}\right\rfloor C_l \times N_{cycle,k} \times V_{dd,f}^2 \times x_{kjif}$$
$$+LCM \times \sum_{k=1}^{M}\sum_{j=1}^{K}\left(1-\sum_{i=1}^{n_{k,j}} u_i\right) \times p_{idle,k},$$

where x_{kjif} is a decision variable that evaluates to true if tasks i executes in processor j of cluster k at frequency f, u_i is the task T_i utilization defined as $u_i = \frac{C_i(f)}{P_i}$, and $n_{k,j}$ is the number of tasks to processor j of cluster k.

We use random-generated task models. The task models considered have 30 independent tasks. We choose to disregard precedence and shared resources to perform a fair comparison against utilization based schedulability tests.

We divide the experiments in two groups: A and B. The differences between the groups are the compared schedulability tests, the system utilization, the usage of switching latency Lp, and the generated task models.

The first group, group A, evaluates the difference between the response time analysis for cluster-based platforms against schedulability analyses that neglect the switching latency Lp. We consider Liu and Layland (1973) original utilization bound (LL), the Earliest Deadline First utilization bound (EDF), and Audsley et al. (1993) response time analysis (AUDS). The tasks WCEC is uniformly distributed between 50,000 and 100,000 cycles. The period is uniformly distributed between 500 µs and 20 ms.

The second group, group B, evaluates the difference between the response time analysis for cluster-based platforms against schedulability analyses that incorporate the switching latency Lp. We consider the conservative utilization based test proposed by He and Mueller (2012a) combined with: (a) Liu and Layland (1973) original utilization bound

(DA_HE_LL), and the EDF bound (DA_HE_EDF). The tasks WCEC is uniformly distributed between 50,000 and 100,000 cycles. The period is uniformly distributed between 40 ms and 65 ms. We changed the tasks' period range to affect the system utilization. Thus we evaluate the behavior of the proposal on different utilization. The tasks deadline is less than equal the random-generated period.

6.4.2 Results

In group A, each combination of number of clusters and number of processors requires millions of random-generated task models and task to processor assignments. Table 12 summarizes the number of random generated task models and task assignments, the average system utilization on each experiment combination, and the estimated average energy consumption during the LCM of tasks's periods. The increase in the number of processors reduces their utilization because the amount of tasks is the same on all experiments. The tasks are randomly distributed between the processors and their frequencies. Due to the uniform distribution of tasks, the effect on energy consumption is minimal. We observe energy consumption reduction when we add the second cluster, as the impact on system utilization is higher.

Table 12 – Summary of experiment on group A

Number of Clusters	Processors per Cluster	Number of Samples	Utilization (%)	Energy (J)
1	04	2618483	28.23	4.47E+12
1	08	1231974	14.65	4.47E+12
1	16	899091	7.39	4.47E+12
2	04	1665647	10.40	4.20E+12
2	08	1162773	5.23	4.20E+12
2	16	448869	2.62	4.20E+12

Figure 20 illustrates the results of experiments on group A. The bar graph in Figure 20 contains plots of confidence intervals for the amount of false positive errors produced by the tests that disregard the DVFS switching latency, *Lp*. For the experimentation on group A, all compared tests (LL, EDF, and AUDS) produce false positive errors. For instance, we notice that the LL test produces false positive errors in 83.6 % of the analyzed task sets for the first platform, with single cluster containing four processors. The test AUDS is the closest to our proposed test. The test AUDS is the one with less false positive errors. AUDS shows a low difference when the number of processors is four. The amount of false positives using AUDS test increases when the number of processors is higher. We observe the same behavior while using the other tests, but the increment is lower. The test EDF has the highest score of false positive errors.

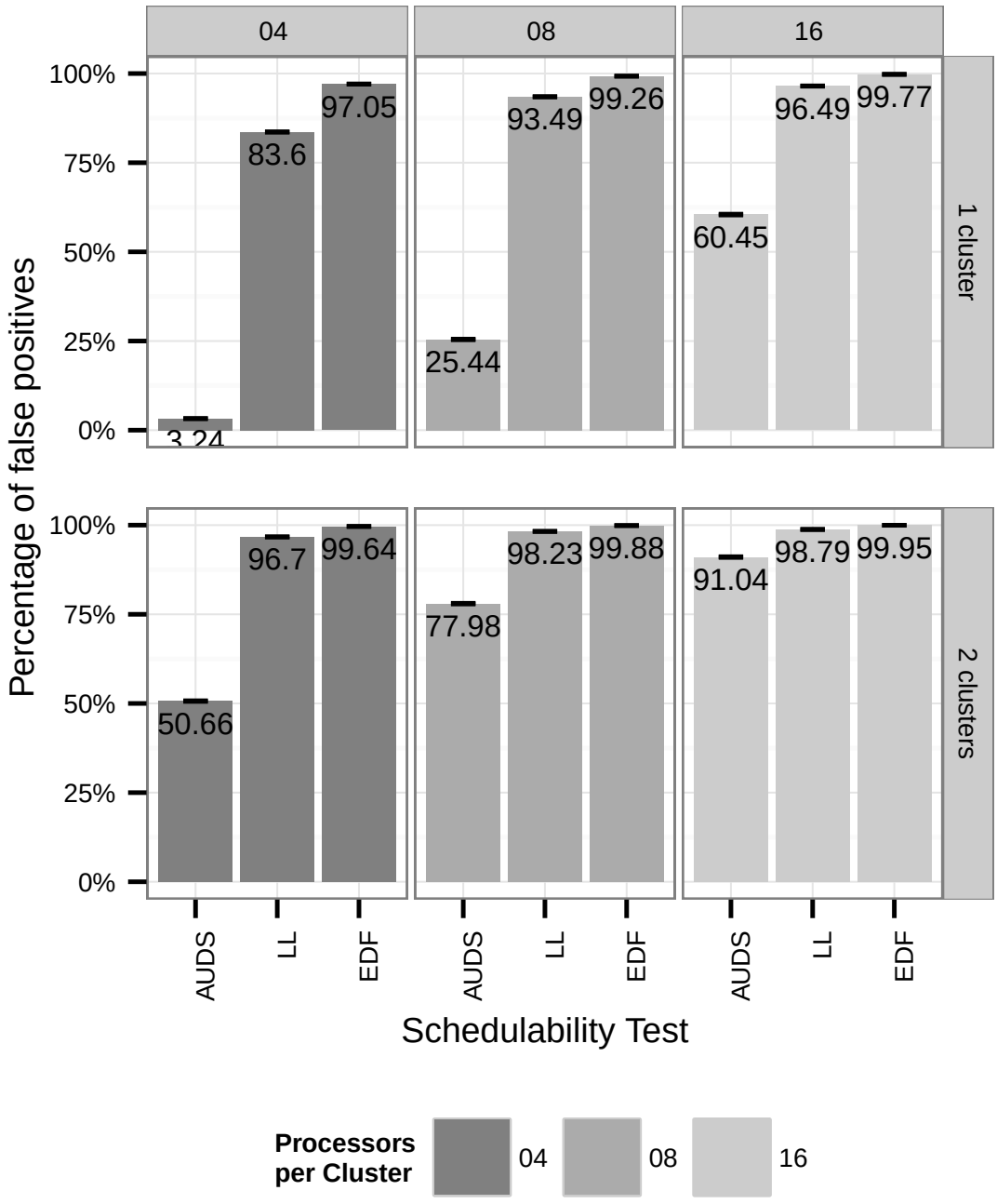


Figure 20 – Group A: Percentage of false positive errors of schedulability tests that neglect DVFS switching latency. On the top of the figure, we present the results for the first platform (single cluster). On the bottom of the figure, we present the results for the second platform (two clusters). The number of processors is per each cluster

Figure 21 illustrates the mean execution time of schedulability tests evaluated in group A. The bar graph in Figure 21 contains plots of confidence intervals for the mean value of the execution time of each test. The y-axis is in logarithmic scale to facilitate

the visualization. TIME_RTA represents the mean execution time of the response time schedulability analysis presented in Section 6.2. The time execution of TIME_RTA is 1000 times slower than the schedulability analyses that disregard the architecture latency. Also, it grows faster if the number of processors per cluster increases.

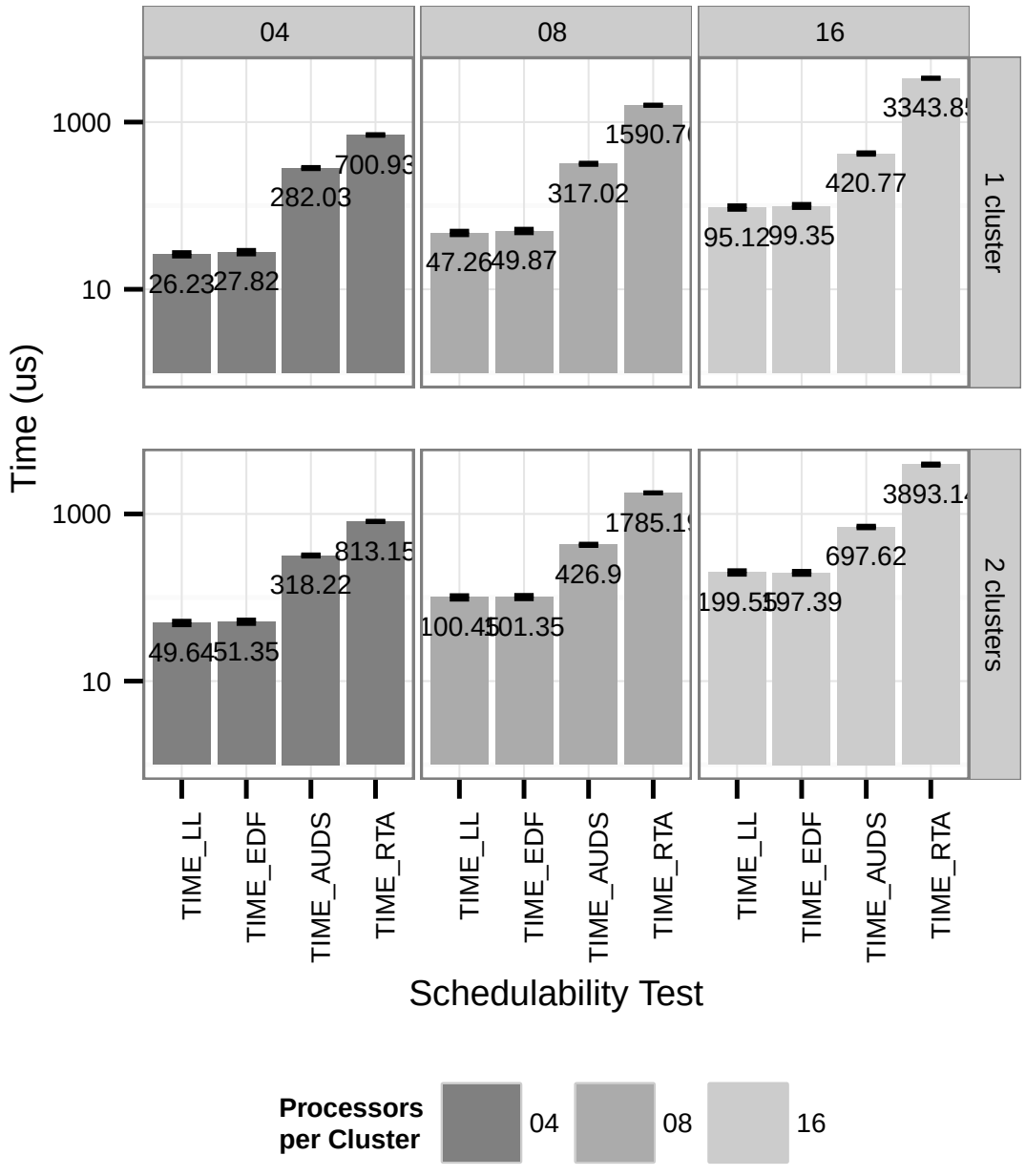


Figure 21 – Group A: Execution time of schedulability tests disregarding DVFS switching latency. On the top of the figure, we present the results for the first platform (single cluster). On the bottom of the figure are the results for the second platform (two clusters). The number of processors is per each cluster

In group B, each combination of number of clusters and number of processors

required millions of random-generated task models and task to processor assignments. Table 13 presents for group B the number of random generated task models and task assignments, the average system utilization on each experiment combination, and the estimated average energy consumption during the LCM of tasks' periods. In this group, we observe a higher energy consumption, compared to group A. The cause is the longer tasks' periods which increases significantly the LCM used to estimate the energy. We observe a lower energy consumption when we add the second cluster.

Table 13 – Summary of experiment on group B

Number of Clusters	Processors per Cluster	Number of Samples	Utilization (%)	Energy (J)
1	04	115971	3.05	1.52E+31
1	08	105567	1.52	1.52E+31
1	16	178449	0.76	1.51E+31
2	04	152106	1.08	1.43E+31
2	08	60600	0.54	1.43E+31
2	16	48480	0.27	1.43E+31

Figure 22 illustrates the results of empirical experiments on group B. The bar graph in Figure 22 contains plots of confidence intervals for the amount of false negative errors produced by the tests that account the DVFS switching latency, *Lp*. For the experimentation on group B, all compared tests (DA_HE_LL and DA_HE_EDF) produce false negative errors. For instance, we notice that the DA_HE_LL test produces false negative errors in 11.23 % of the analyzed task sets for the first platform, with single cluster containing eight processors. Although both compared tests consider *Lp*, they are still imprecise. They generate false negative errors because they are only sufficient conditions. The results of both tests are similar. The results differ for the scenario of two clusters with 16 processors. The test DA_HE_LL is the one with less false negative errors. The test DA_HE_EDF is the one with the highest score of false negative errors.

Figure 23 represents a comparison of the execution time of the schedulability analyses considering the architecture latency. The bar graph in Figure 23 contains plots of confidence intervals for the mean value of the execution time of each test. TIME_RTA is still the slowest on these experiments. However, the difference is smaller because the execution time of all schedulability analyses in this group grow with the number of processors per cluster.

6.4.3 Discussion

The empirical experiments highlight the need for a response time schedulability analysis that accounts for architecture latency. Our proposed response time based analysis is the first to consider such latency.

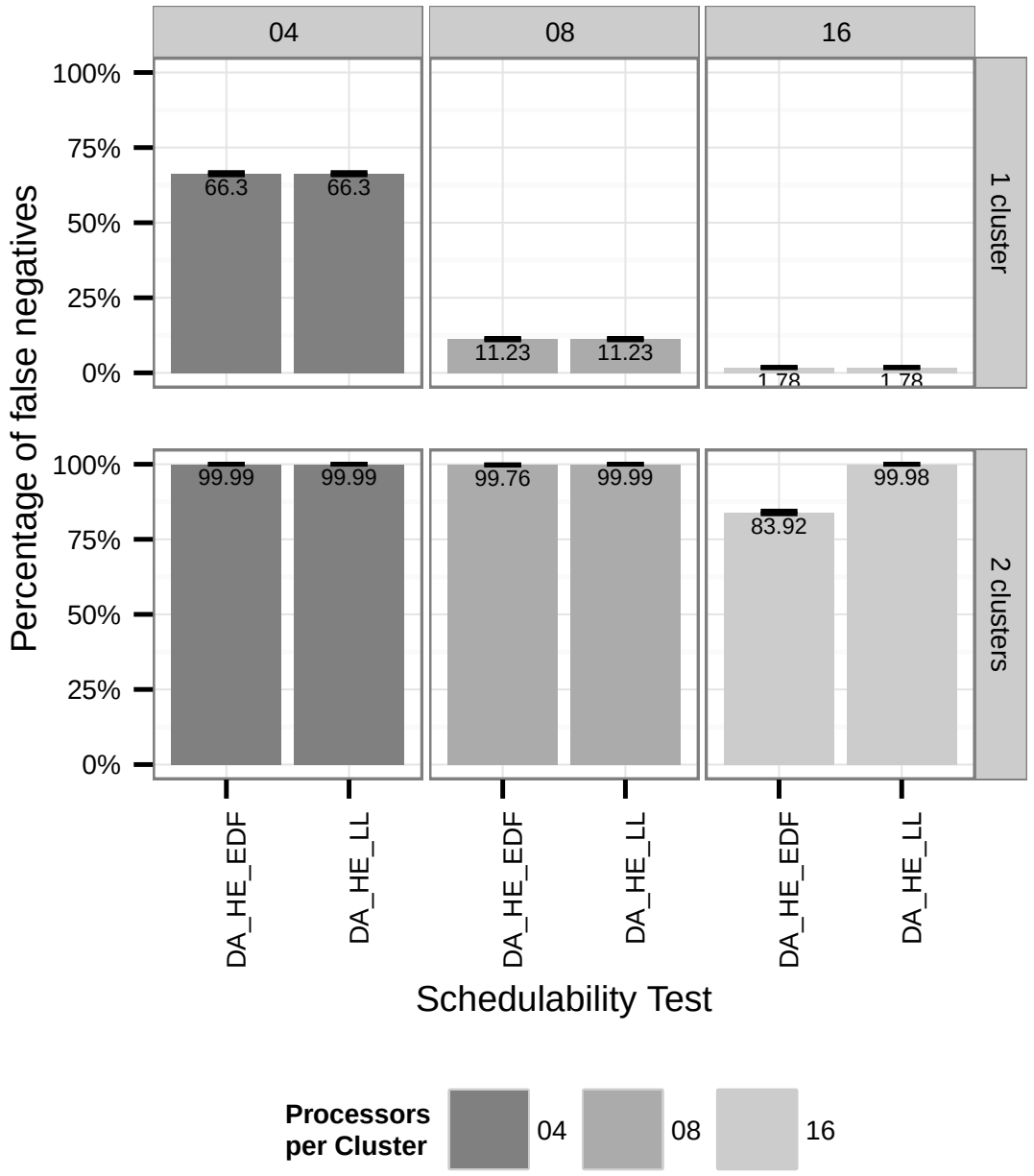


Figure 22 – Group B: Percentage of false negative errors of schedulability tests that account for DVFS switching latency. On the top of the figure, we present the results for the first platform (single cluster). On the bottom of the figure, we present the results for the second platform (two clusters). The number of processors is per each cluster

Results suggest that original utilization based and response time based analyses (Audsley et al., 1993; Liu and Layland, 1973) lack the information of the DVFS switching latency. Therefore, they are prone to commit false positive errors. In the literature exist only utilization based schedulability analysis accounting the DVFS switching latency (He

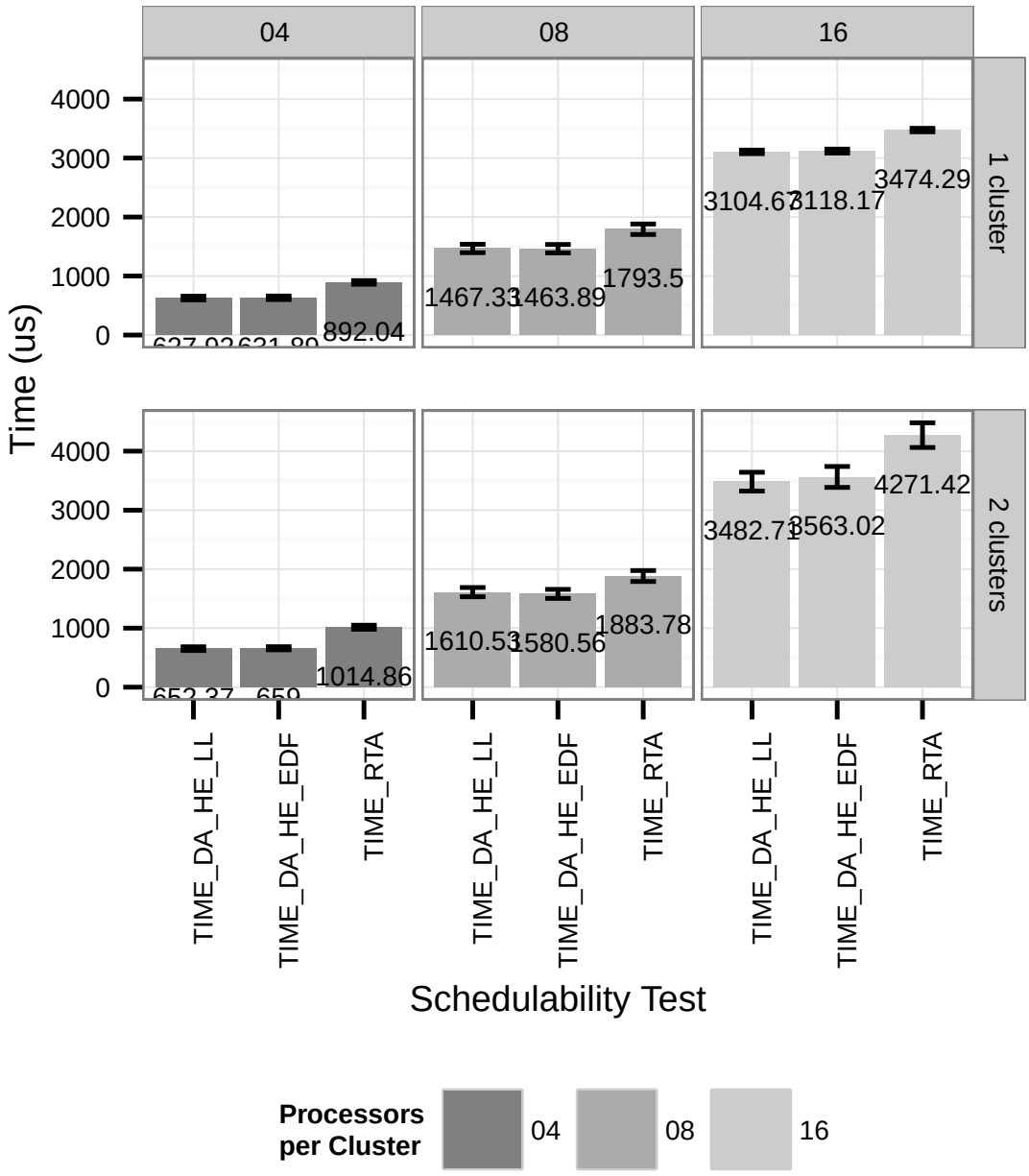


Figure 23 – Group B: Execution time of schedulability tests considering DVFS switching latency. The results for the first platform (single cluster) are on the top of the figure. On the bottom of the figure are the results for the second platform (two clusters). The number of processors is per each cluster

and Mueller, 2012a). Even though it accounts for the Lp it is still an imprecise test. Therefore, it commits false negatives errors.

On the other hand, it is important to highlight that the proposed response time schedulability analysis is more expensive than the existing tests. The time complexity grows with the number of tasks and the number of processors. However, it is still worth

using it because the execution time difference is small when compared to sufficient tests considering architecture latency.

Both errors are problematic. False positives can lead systems to unschedulable situations. False negatives may leave room for extra optimization, when allocating tasks to processors and assigning frequencies, for instance.

6.5 Chapter Summary

In this chapter, we assess the problem of providing response time schedulability test for cluster-based platforms. We spot, with examples and experiments, situations in which the existing tests in the literature might produce false positive and false negative errors. Combining the observations of existing works (Audsley et al., 1993; He and Mueller, 2012a) we derive a novel response time schedulability analysis for modern cluster-based platforms.

Empirical experiments suggest that existing tests may lead hard real-time systems to unschedulable situations. The existing schedulability analyses lack the information of DVFS switching latency and thus produce false positive errors, as shown by empirical experiments. A higher number of processors increases the amount of false positive errors. The existing sufficient test proposed by He and Mueller (2012a) considers the Lp but also produces false negative errors.

Therefore, whenever possible, we suggest using a response time schedulability analysis, such as the proposed response time based test. The proposed schedulability analysis combines the switching latency estimation in a response time test. Therefore, we avoid false positive, and false negatives errors.

7 Related Work

This chapter discusses each of the 29 works identified by a systematic literature review (Valentin and Barreto, 2016) as existing research related to the present thesis. First, Section 7.1 discusses the works that combine existing solutions of uni-processors applied to multicore environment. Then, Section 7.2 discusses works about temperature control. Finally, Section 7.3 discusses works regarding innovative schedulability analysis.

7.1 Single Processor Schedulability Analysis combined with Task Allocation

This section discusses the works that combine existing solutions of uni-processors applied to multicore environment. Section 7.1.1 describes each work and presents an analysis of their findings. Section 7.1.2 ends the section with a discussion about how the present research extends the state of the art.

7.1.1 Techniques and Methods applying Single Processor Solutions combined with Task Allocation

Chen et al. (2013) make an observation that deadline partitioning may be too conservative for the problem of allocation of real-time tasks with minimal energy consumption on pipelined computing systems. Therefore, authors present an algorithm based on the Pay Burst Only Once principle. Authors also make a fair comparison with existing solutions based on deadline partitioning. However, this work disregards parallel computing in the pipeline.

Chen et al. (2011) propose a meta-heuristic strategy based on Ant Colony Optimization (ACO) which solves the problem of determining an assignment of a set of periodic tasks to a set of heterogeneous processors without deadline violations and reducing energy consumption. The ACO strategy is innovative because it implements a local search instead of a heuristic in the artificial ants, with bounded search depth. For this reason, authors claim their strategy is competitive when compared against a Genetic Algorithm (GA) and a Fully Polynomial-Time Approximation Scheme (FPTAS) approximation algorithm by Baruah. Experimental results show that ACO algorithm outperforms GA and Baruah methods; furthermore, achieves an average of 15.8 % energy saving over its prototype. However, authors oversight a comparison to state of art strategy, such as the FPTAS proposed by Chen et al. (2009) or the heuristic proposed by Yu and Prasanna (2003), for instance. The proposed ACO is also limited to processors which execute one instruction

per cycle. Energy minimization is a secondary objective. The work disregards common energy savings strategies, such as DVS, or DPM, and neglect static power dissipation.

Chen et al. (2009) tackle the problem of minimizing the energy consumption of periodic real-time tasks on platforms with heterogeneous Processing Units (PU). Authors propose four FPTAS algorithms. Two consist of a proposed relaxation of integer linear programming model. The other two extend the original algorithms by using graph theory to solve a restricted version of the original problem. In the restricted version, a constant restricts the number of allocated PU. The authors present the problem hardness and prove that the solutions of these algorithms are an $(m + 1)$-approximation of optimal solution. Authors neglect existing state of art algorithms, because theirs are the first to address the problem.

Chen and Thiele (2009) propose a FPTAS algorithm with $(1+lnn)$-approximation for the problem of Energy efficient task partitioning and platform synthesis with heterogeneous PU types. The solution reuses the Minimum Average Energy First Heuristic (MAEF). The target system has multiple PUs of M different types. Each PU schedules a partition of the periodic real-time task set using EDF based scheduler. The power model considers load dependent (DVFS) and load independent (non DVFS) energy consumption and static and dynamic power consumption. Authors model their problem using ILP and relaxation techniques. The authors use the weak duality theory to derive the approximation for their algorithm. The authors compare the results of their algorithm against a synthetic baseline they created. The work neglects state of art algorithms.

Chen and Thiele (2008) consider the problem of allocating hard real-time tasks on two heterogeneous processing units with the objective to achieve minimal energy consumption. The authors propose three solution for the problem of minimization of energy consumption of periodic real-time tasks on platforms with Two Heterogeneous PEs. The first solution is a greedy heuristic based on task architectural characteristics and energy consumption derivative. The other two solutions employ dynamic programming, in which one is an exact solution, and the other is a FPTAS approximation. Limitations in this work include: it is applicable only to dual core systems, the task deadline is equal to period and processors use only one frequency, once determined the best frequency for the assigned workload.

Chen and Thiele (2011) propose a FPTAS algorithm with $(1+lnn)$-approximation for the problem of energy efficient task partitioning and platform synthesis with Heterogeneous PU types. The solution reuses the Minimum Average Energy First (MAEF) heuristic. The target system has multiple PUs of M different types. Each PU schedules a partition of the periodic real-time task set using EDF based scheduler. The power model considers load dependent (DVFS) and independent (non DVFS) energy consumption. Their model accounts for static and dynamic power consumption. Their models use ILP

and relaxation techniques. The authors use the weak duality theory to derive the approximation for their algorithm. They solve a restrict version of the problem using the Restricted Minimum Average Energy First (R-MAEF) algorithm. A constant A_j limits the number of allocated PUs in the restricted problem version. The authors compare the results of their algorithm against a synthetic baseline they created. The work neglects state of art algorithms.

Chen et al. (2008) develop a novel priority ceiling protocol named MFP-PCP. MFP-PCP aims systems with one processor and multiple co-processors. The protocol locks frequencies and makes frequency inheritance in co-processors to avoid priority inversion problem. The protocol is highly based on PCP. The difference resides on task scheduling and the frequency switching sections. The proposed protocol is unsuitable for multiple heterogeneous processors. The sub-task to co-processor assignment is a requirement and expected as input. The authors focus the performance evaluation only against FP-PCP protocol. The comparison method is hard to follow and difficult to reproduce.

Goossens et al. (2008) address the energy-aware real-time scheduling problem upon heterogeneous platforms. Two MPSoCs compose their environment. One is a high performance processor. Another is low performance, but consumes low power. The authors propose applying four well known heuristics: random, FFDD, GA, and SA; to split the workloads between the two platforms. Authors claim that sophisticated heuristics, such as GA and SA, are crucial to have. Simplistic heuristics, such as random and FFDD, perform poorly when the amount of tasks increases. Empirical experiments compare their solution against to running the full workload in a high performance MPSoC. Average energy consumption reduction is within 20 % and 40 %.

Alahmad and Gopalakrishnan (2011) propose three possible solutions for the problem of energy minimization via task allocation of a real-time system executed in a heterogeneous multicore architecture. They consider only periodic, implicit-deadline real-time tasks. The authors propose a dynamic programming approach based on Woeginger formulation. A FPTAS and a greedy heuristics are also proposed for the same problem. The authors analyze their algorithms' complexity. The experiments report a performance evaluation to compare the approximated solutions. The comparison neglects state of art algorithms, though.

The main focus of the work done by Awan and Petters (2013) is the provisioning of realist energy model for heterogenous processing elements with respect to static power consumption. The authors derive two heuristics (LLED and MM) for hard real-time allocation and dynamic power consumption minimization. The power models account for multiple sleep states, energy, and latency transitioning overhead during sleep state switching. The heuristics take into consideration the task energy density functions. The heuristic for static power consumption minimization applies also race-to-idle scheduling policies.

The authors, however, neglect performance state transitioning (DVFS). Also the overhead of entering sleep states and leaving sleep states is the same. The authors present also a fair comparison with state of art heuristics applicable to the same scenario via extensive simulations based on SPARTS. Results suggest that their algorithm outperforms First Fit on different utilization scenarios.

Hung et al. (2006) propose multiple approximation algorithms for the minimization problem of energy consumption and the maximization problem of energy saving in migrating task executions from the DVS PE to the non-DVS PE. The authors propose formulations for different NP-Complete problems: MII, SID, MDI, and SDD. MII stands for Energy-Minimization on an Ideal Processor and a Workload-Independent PE (MII) problem. SID stands for Energy-Saving on An Ideal Processor and A Workload-Dependent PE (SID) problem. MDI and SDD are their respective versions on a non-ideal DVS processor. The studied system has two processors, one with DVS, another without DVS. The proposed formulations and solutions utilize the knapsack problem theory. Authors propose greedy heuristics and dynamic programming solutions. Authors analyze the respective time complexity and approximation constants. The authors perform extensive simulations to evaluate the proposed heuristics. The baseline for comparison is the execution of the workload fully in the DVS PE. The non-DVS PE is only an energy reduction strategy because the system is always capable of feasibly scheduling the workload using only the DVS PE. The authors perform a poor comparison as they only evaluate their proposed solutions. The experiments lack comparison to any existing strategy. The usage of a convex power function is the base for some of the proofs. This assumption limits the usage of their solutions on real use cases / processors.

Kim et al. (2005) propose seven heuristics for dynamic mapping of tasks onto wireless devices participating in an *ad hoc* grid, or heterogeneous computing environment. The heuristics are OLB, FG, Switching, Min-Min, Sufferage, Random, and Originator. Heuristics resemble classical scheduling methods, such as Min-Min, Greedy, switching, OLB. The empirical experiments lack comparison to existing or classical methods. According to the simulation results, the heuristics of best performance, which are capable of processing the most amount of tasks within their deadlines, are the batch modes Min-Min and Sufferage, followed by the immediate mode Switching.

Kim et al. (2008) propose seven heuristics for dynamic mapping of tasks onto wireless devices participating in an ad hoc grid, or heterogeneous computing environment: OLB, MEG, ME-ME, CRME, ME-MC, Random, and Originator. Heuristics resemble classical scheduling methods, such as Min-Min, Greedy, switching, OLB. The empirical experiments lack comparison to existing or classical methods. According to the simulation results, the heuristics of best performance, which are capable of processing the most amount of tasks within their deadlines, are the batch modes ME-ME and CRME, followed

by the immediate mode ME-MC.

Prescilla and Selvakumar (2013) propose two algorithms: Novel BPSO and Modified BPSO. They use the concept of the meta-heuristic know as Particle Swarm Optimization (PSO). The problem is assignment of a set of independent periodic task to a heterogeneous multiprocessor without exceeding the utilization bound with minimum energy consumption. The PSO is a population based search algorithm. The algorithm simulates the social behavior of birds, bees, or a school of fishes. Novel BPSO and Modified BPSO essentially propose different way of interpreting and implementing the particle velocity. Using their proposal requires characterization of each task for number of cycles per processor. A training phase is also mandatory for their algorithms. The work oversights static power consumption and tasks' response time bounds. Energy minimization is a secondary optimization objective. Authors did not comment all their results; specially those related to energy consumption. They perform a comparison against a state of art strategy based on Ant Colony Optimization, proposed by Chen et al. (2011). Results suggest that modified particle swarm optimization assigns a greater number of tasks for problem instances with consistent utilization matrix, when compared to ACO. But Modified BPSO gives slightly lower performance for inconsistent utilization matrix compared to ACO. This is because BPSO has weak local search ability.

Yang et al. (2009) propose a FPTAS based on Dynamic Programming to partition real-time tasks on a platform with heterogeneous processing elements (processors). The target is to minimize energy consumption. Authors assume that the energy consumption with higher workload is larger than that with lower workload. The proposed scheme is a FPTAS when the number of processors is a constant. System designers can specify a parameter for trading the quality, energy consumption reduction, of the derived solution to the running time of the algorithm. Authors compare the performance against the state of the art approach (Huang et al., 2007). Empirical experiments show that the derived solutions improve 10 % - 15 % when static/leakage power consumption is negligible. Also, their solution improves 30 % - 60 % when static power consumption is non-negligible. The authors oversight exact schedulability analysis. The presented performance evaluation considers only ideal processors and frame based real-time tasks.

Prasanna and Yu (2002) study the problem of static allocation of a set of independent tasks onto a real-time system consisting of heterogeneous processing elements, each enabled with discrete Dynamic Voltage Scaling. The allocation problem is first formulated as an extended Generalized Assignment Problem. A linearization heuristic (LR-heuristic) is then extended for solving the problem. Experiments show that when the real-time constraints are tight, the LR-heuristic achieves 15 % off the optimal energy consumption for small size problems, while the performance of a classic greedy heuristic is around 90 % off the optimal. A relative performance improvement of up-to 40 % over the classic greedy

heuristic is also observed for large size problems. Authors base their models on a sufficient condition for optimal scheduling for EDF. They report a lack of exact conditions. The power model used is an increasing convex function. Authors minimize only dynamic power. Authors neglect static power. Their work also discards the DPM technique due to its complex applicability on real-time systems.

Yu and Prasanna (2003) study the problem of statically allocating a set of independent real-time tasks to a system consisting of heterogeneous processing elements, each enabled with discrete Dynamic Voltage Scaling. The goal is to minimize the overall energy dissipation of the system without violating the real-time requirements of the tasks. The problem is first formulated as an extended Generalized Assignment Problem. A linearization heuristic (LR-heuristic) is then extended to solve the problem. Experiments show that when the average utilization of the system is high, the LR-heuristic achieves 15 % off the optimal energy dissipation for small size problems, while the performance of a classic greedy heuristic is around 90 % off the optimal. A relative performance improvement of up-to 40 % over the classic greedy heuristic is also observed for large size problems. Authors perform an analytic performance comparison between the LR-heuristic and the greedy heuristic. Authors base their models on a sufficient condition for optimal scheduling for EDF. They report a lack of exact conditions. The power model used is an increasing convex function. Authors minimize only dynamic power. Authors neglect static power. Their work also discards the DPM technique due to its complex applicability on real-time systems.

He and Mueller (2012b) propose a heuristic based on simulated annealing (SA) to find a solution for task partition and frequency assignment so that all tasks are schedulable. The SA minimizes the system energy consumption over one hyper period on a heterogeneous platform. The SA collaborates with a task penalty based heuristic to improve SA convergence. The algorithm has also an online version. Each step of the offline strategy is a unit in the online version executed on scheduling points. The online version computes the required task data on each scheduling point. Upon task arrival time, the system updates a heap and the system Speed Inheritance Protocol (SIP) data. Based on the task heap and the SIP data the system determines each task frequency. The system then executes one iteration of the SA On each LCM of all tasks period. This algorithm may converge to optimal solution if the system is unchangeable for multiple executions of the LCM. The SA initial assignment uses a solution based on the (LA+LTF+FF), but authors neglect the required computation time to resolve the initial assignment. The power model restricts dynamic power to a convex function. Besides, there are platforms that have complex C-states, with required sequencing or power domain dependencies. The authors, in their experimentation, evaluate only a version of their model considering homogeneous platforms. The empirical experiments neglect other existing state of art strategies. Experiments evaluate only the (LA+LTF+FF) and the generalized SA.

7.1.2 Discussion

As discussed in Section 7.1.1, the majority of the existing works in state-of-the-art combines solutions for single processor with task allocation and frequency assignment processes. We intend to take into account the latency inherited of the natural behavior of heterogeneous platforms. Therefore, our research brings innovative results because it uses findings of single processors combined with characteristics found in multiple heterogeneous platforms.

Besides, we concentrate our focus on modern platforms in which the dynamic power consumption is non-convex (Alahmad and Gopalakrishnan, 2011). Most of the existing works take advantage of increasing convex power consumption in the task allocation and frequency assignment processes. The fastest algorithms to optimize energy optimization is polynomial in time for single processors (Aydin et al., 2001a,b; Sha et al., 1990; Shin et al., 2000) when the power consumption is a convex function. However, the findings are scarce when considering non-convex power functions in the energy optimization process.

7.2 Temperature Control

This section discusses works about temperature control. Section 7.2.1 describes each work and presents an analysis of their findings. Section 7.2.2 ends the Chapter with a discussion about how the present research extends the state-of-the-art.

7.2.1 Temperature Control Techniques on Heterogeneous Systems

Chantem et al. (2011) address the problem of allocating and scheduling hard real-time tasks on heterogeneous MPSoC while minimizing the peak temperature. Authors argument that existing solutions that reduce peak power or peak energy are sub-optimal for addressing peak temperature issues. The reasoning is because these two approaches neglect thermal temporal and spatial evolution. Therefore, such approaches lead to sub-optimal solutions. Authors propose then a MILP formulation for the problem. The MILP covers peak temperature minimization subject to assignment, deadlines, precedence, and overlap constraints. Because MILP formulations have a limitation of small problem instances, authors also propose two heuristics, SSAB and TAB. The heuristics address the problem with steady state thermal simulations and transient thermal simulations, respectively. Authors also propose a delay insertion technique to avoid system reaching peak temperature while using TAB. Experimental results show a peak temperature reduction of 10.09 °C on average and up to 30.75 °C for embedded processors when compared to energy minimization. Their approach reduces peak temperature of 8.98 °C on average and up to 23.25 °C for high power density chips when compared to peak power minimization. The phased steady-state analysis based heuristic finds an optimal solution in multiple

contexts with a maximum deviation of 3.40 °C from optimality. The heuristic achieves a temperature reduction of 10.94 °C on average when compared to previous work. The transient analysis based heuristic models and exploits the transient thermal effects of short tasks to further improve upon the existing solution by 0.67 °C in the best case. Finally, the work shows that incorporating the concept of delay insertion into the proposed heuristic framework results in an additional peak temperature reduction of up to 11.92 °C. However, authors oversight DVFS, the most common power reduction technique. Also they disregard leakage power reduction, using DPM for instance. They also consider P_{dyn} a constant, while when using recent processors this assumption falls apart, due to variability caused by DVFS or AVS.

Hettiarachchi et al. (2013) propose a control-theoretic framework to ensure hard real-time deadlines on a multiprocessor platform in a dynamic thermal environment. The method uses real-time performance modes to permit the system to adapt to changing conditions. Also, authors show how the system designer can use their framework to allocate asymmetric processing resources upon a multicore CPU and still maintain thermal constraints. Authors develop analysis for determining what modes the system can support for a given external thermal condition. The system design extends the derivation of thermal-resiliency (originally proposed for uni-processor systems) to multicore systems and determines the limitations of external thermal stress that any hard real-time performance mode can withstand. Simulations and physical testbed results show that the algorithm predicts how a system will gracefully and predictably degrade under external thermal stress. Experiments suggest that their method is capable of sustaining reference temperature. However, authors neglect comparison of their strategy against any other existing similar solutions, claiming theirs is the first proposed solution. The proposed solution requires fine tuning a duty cycle period (active and inactive periods). The fine tuning step is a time consuming task for system designers. Their proposed method neglects the power variation per operating point. The solution also restricts the duty cycle setup to a divisor of the sampling interval.

Saha et al. (2012) propose HyWGA, a GA combined with Min-core Worst-fit (MW) based heuristic to generate a thermal-constrained energy-aware partitioning of periodic hard real-time tasks in heterogeneous multi-core multiprocessor systems. The authors formulate the problem using an ILP. The evaluation of HyWGA uses two proposed thermal models: Heat Independent Thermal (HIT) Model and Head-Dependent Thermal (HDT) Model. Experimental results show HyWGA approach is most effective in minimizing the total energy consumption. Empirical experiments compare HyWGA against to the MW heuristic. The comparison suggests that HyWGA can minimize the total energy consumption by up to 11 % and 21 % when using the HIT and HDT models, respectively. However, their experiments lack comparison to any existing state of art solution. The experiments include only the combination of their solutions. Authors disregard exact schedulability

analysis. Also, in their experiments, the ambient temperature is low, 0 °C.

Yang et al. (2012) propose a heuristic to map and schedule tasks onto a set of heterogeneous processing units. Authors model leakage as function of temperature and supply voltage. The scenario is MPSoC platforms. The algorithm is Heuristic Temperature-Aware DVS Scheduling (HTADS). HTADS algorithm takes the task graph and system architecture as the input, and outputs the optimal schedule with the minimal peak temperature. HTADS maps tasks to the fastest processors with maximum voltage level firstly, and then uses the Critical Path Scheduling Algorithm (CPSA) to get the schedule and peak temperature under current assignment. The algorithm determines the deadline to scale down the voltage of the task with the peak temperature repeatedly. The target is to minimize the peak temperature of MPSoC. Algorithm CPSA takes the tasks assignment as input, computes the task priority according critical path, and then schedules the tasks under the priority constrain. The output of CPSA is the optimal scheduling and deadline. The authors consider a task set which deadline and period are common to all tasks, frame of task. The authors neglect feasibility analyses, such as processor utilization bounds or tasks response time, in their proposed solution. The reasoning may be because CPSA already generates an offline scale. The temperature models neglect the ambient temperature. Their solution disregards minimization of overall power consumption.

7.2.2 Discussion

The temperature control is a challenging subject, specially considering heterogeneous platforms. Modern platforms have higher leakage power consumption. In our research, we plan to explore the relation of temperature and leakage power found in modern platforms.

Most of the existing works neglect the relation between temperature and leakage power in the energy optimization process. Therefore, our research brings innovative results while exploring this relation and modeling modern platforms.

7.3 Schedulability Analysis For Multicore Systems

This section discusses works regarding innovative schedulability analysis. Section 7.3.1 describes each work and presents an analysis of their findings. Section 7.3.2 ends the section with a discussion about how the present research extends the state-of-the-art.

7.3.1 Schedulability Analysis Techniques

The work done by He and Mueller (2012a) present a novel schedulability analysis for clustered heterogeneous systems. The authors revisit the utilization based schedu-

lability analysis. The tests are aware of delays and overheads imposed by the usage of DVFS and DPM. Their work takes advantage of Advanced Configuration and Power Interface (ACPI) common definitions of P-states (DVFS) and S-states (DPM). The authors change the existing utilization bounds and also present proofs for the correctness of their proposed schedulability tests. They limit the amount of available sleep states to a constant, although it may vary per power domain, in practice. Also they simplify the P-state switching overhead variability considering only the worst case. In practice, the switching overhead might be strongly bound to the voltage and frequency delta. They also consider a convex power function.

Min-Allah et al. (2012) propose an exact schedulability analysis for multicore platforms executing hard real-time periodic tasks under RM/DM scheduling policies. Authors extend the classical schedulability analysis that considers only the first feasible speed (FFS). Authors compare FFS against their proposal to search for the minimal speed instead (LFS). Results suggest that their approach increases speed much slower than FFS when the system load increases. Authors also propose a task shifting heuristic to balance the load across cores, given a partitioned workload. The high load tasks are shift to light loaded cores. The heuristic determines that the complete system must execute on same frequency (avg frequency across minimal required frequency of all cores). The simulation results suggest that their heuristic is capable of evenly distribute the workload. The processor model reported by Min-Allah is applicable to heterogeneous platforms, however, it is limiting as they fix the amount of discrete frequencies and the step between each frequency, which may cause severe rounding problems. Running the system at same frequency may be under-utilization of the DVS technology.

7.3.2 Discussion

Schedulability analyses for platforms with multiple heterogeneous processors are scarce in the state-of-the-art works. Most works apply the well established single processor schedulability analyses after performing a task allocation process. Our research intends to propose innovative schedulability analyses targeting finding better solutions during the energy optimization process.

The majority of the state-of-the-art works apply utilization based schedulability tests. The utilization based tests are sufficient only conditions and therefore may discard system configurations that could improve further the energy consumption. For this reason, we are proposing in the present research an exact schedulability analysis for multiple heterogeneous processor platforms, grouped as clusters.

Two works consider the problematic of multiple heterogeneous processors in the process of providing schedulability conditions. However, He and Mueller (2012a) provide a sufficient only condition. Min-Allah et al. (2012) propose exact schedulability analysis

applicable to heterogeneous platforms, however, it is limiting as they fix the amount of discrete frequencies and the step between each frequency, which may cause severe rounding problems.

7.4 Notes about related literature reviews

The systematic literature review (Valentin and Barreto, 2016) returns also multiple works reporting literature reviews. However, it is difficult to determine if the reviewed works apply to this study. The reasoning is because the review reports summarized research. The current study avoids searching recursively through the reference list and focus only on results out of the search string. This Chapter, nevertheless, attempts to comment the literature review reports returned during this study.

Only the work by Chen and Kuo (2007) has enough details to extract relevant techniques.The authors review the following primary works. Luo and Jha (2002) propose list-scheduling-based heuristics for the scheduling of real-time tasks with precedence constraints in heterogeneous distributed systems. Schmitz et al. (2002) originally propose genetic list-scheduling algorithms. Kirovski and Potkonjaka (1997) explore for the first time the synthesis problem for energy-efficient task scheduling of periodic hard real-time tasks. Processors have cost constraints. Also, processors might have different costs. H.-R.Hsu et al. (2006) show that the problem is NP-hard in a strong sense. Besides, they also prove that any polynomial-time approximation algorithm exists with a constant approximation ratio by providing a L-reduction. Chen and Kuo (2006) propose a 1.5-approximation algorithm with constant violations and a 2-approximation algorithm when there is only one processor type with continuously available speeds and provide extensions to ideal and non-ideal processor types with multiple processor types.

Sheikh et al. (2012a) review the recent works for energy and performance aware scheduling on distributed systems. Singh *et al.* surveys the emerging trends on task mapping on multi/many-core systems (Singh et al., 2013). Beloglazov et al. (2011) present a taxonomy and survey of energy-efficient data centers. The work by Virlet et al. (2011) characterizes resource allocation heuristics for heterogeneous computing systems.

Venkatachalam and Franz (2005) summarize and review multiple power reduction and control techniques. Davis and Burns (2011) survey hard real-time scheduling algorithms and schedulability analysis techniques for homogeneous multiprocessor systems. Kong et al. (2012) review the recent scientific works related to thermal control. J. Kong also includes basic concepts about control theory and thermal constraints. Sheikh et al. (2012b) also review thermal related works for multicore platforms.

7.5 Chapter Summary

In this chapter, we have reviewed the existing research related to the present thesis. We performed a systematic literature review (Valentin and Barreto, 2016) and found 29 works. We have discussed them into three major categories: schedulability analysis combined with task allocation, temperature control, and schedulability analysis for multicore systems.

The majority of the existing works combines solutions for single processor with task allocation and frequency assignment processes. Our research brings innovative results compared to the state-of-the-art because it uses findings of single processors combined with characteristics found in multiple heterogeneous platforms. Besides, we concentrate our focus on modern platforms in which the dynamic power consumption can also be non-convex.

The temperature control is a challenging subject, specially considering heterogeneous platforms. Modern platforms have higher leakage power consumption. Most of the existing works neglect the relation between temperature and leakage power in the energy optimization process.

Schedulability analyses for platforms with multiple heterogeneous processors are scarce in the state-of-the-art works. Most works apply the well established single processor schedulability analyses after performing a task allocation process. Two works consider the problematic of multiple heterogeneous processors in the process of providing schedulability conditions. However, He and Mueller (2012a) provide a sufficient only condition. Min-Allah et al. (2012) propose exact schedulability analysis applicable to heterogeneous platforms, however, it is limiting as they fix the amount of discrete frequencies and the step between each frequency, which may cause severe rounding problems.

8 Final Remarks

Achieving successful energy-aware application development requires specialized labor force. In general, system designers, architects, and engineers lack this skill because it requires specialization in different fields, as typically it needs to be balanced between severe constraints, such as size, weight, cost, time-to-marketing, reliability, and others even more specific to the target environment, such as heating, vibration, lightning, corrosion, water, fire, power source variation (Barreto, 2005). The challenge is even harder when hard real-time constraints are mandatory to be covered. Thus, aiding the design and modeling process of hard real-time applications subject to energy and temperature constraints on multiple heterogeneous platforms is the motivation of this research. The lack of existing exact methods in the subject is also an additional motivation to extend the knowledge in this subject.

The results of a systematic literature review (Valentin and Barreto, 2016) highlight that, even though there is an increasing interest in the subject in the last years, there are still open questions to be addressed. The subject is of interest to different research groups, across diverse industry contexts. There are topics that have been slightly explored. For instance, only five works address the problem of offering optimal solutions, although most works consider small or medium instance sizes. Most works are also interested in solutions produced on-line; just five works consider off-line approaches. The most challenging aspect is exact schedulability tests. We found only one work dealing with tasks' response time analysis, while the majority consider utilization based analysis, which is known to be a sufficient only condition in the presence of inter-task dependence, such as precedence and mutual exclusion. Similarly, we emphasize the need of exploration of the thermal topic on this subject, as we found only five works dealing with temperature control.

Therefore, based on the outcome of the systematic literature review, the research question of this thesis was "*How to offer users timing correctness and guarantees of hard real-time systems executed on heterogeneous multicore systems with energy and temperature constraints?*" This thesis aims to solve optimally the problem of scheduling hard real-time tasks in heterogeneous multicore platforms subject to energy and temperature constraints. Hard real-time systems are present in life critical environments. In such scenarios, correctness depends on integrity of results and on the time when they are produced. Thus, reducing the energy consumption on such systems becomes specially challenging.

8.1 Revisiting Objectives and Hypotheses

The main aim of this thesis was to propose an *energy optimization method for hard real-time system on heterogeneous multicore platform demonstrating that it is possible to timely compute timing correctness and guarantees using a sufficient and necessary condition; accounting for temperature, preemption, precedence, shared resources constraints, and architectural interference*. We achieved the main objective of this research by combining: (a) schedulability analysis from hard real-time systems, (b) representative mathematical formulations covering modern processors technological characteristics, and (c) robust exact implicit enumeration algorithmic strategies from combinatorial optimization.

One of the secondary objectives of this research was *to define a schedulability analysis for hard real-time scheduling on heterogeneous multicore systems with sufficient and necessary conditions.* We achieved this secondary objective by applying the correct schedulability analysis depending on the workload. On the one hand, existing utilization based schedulability analysis are effective for workloads composed by independent periodic hard real-time tasks, mainly because the test can be incorporated in different ILP formulations, allowing to reuse several existing robust computational techniques of resolution. On the other hand, response time based schedulability analyses are effective for workloads with dependent hard real-time tasks (see Chapter 5), covering for preemption, precedence, mutual exclusion, and architectural influence (Valentin et al., 2015b) (see Chapter 6).

In fact, one of the hypotheses tested in this research was: *stronger feasibility analysis offers tighter bounds for the problem*. This can be observed, for example, in the results produced by solvers for fixed priority schedulers. By applying less accurate schedulability tests, such as utilization based, the solvers take longer to converge to optimal solutions, when compared to solvers that apply exact schedulability tests based on response time analysis (Valentin et al., 2017, 2016b).

The other secondary objective of this research was *to deliver an off-line assignment algorithm to distribute hard real-time tasks among heterogeneous cores of a multicore system; assigning a CPU frequency to each task on active scenarios; using a sufficient and necessary schedulability analysis considering energy, temperature, preemption, precedence, mutual exclusion, and architectural interference*. We achieved this secondary objective, once again, by applying the correct mathematical formulations, representing modern processors characteristics, for example, using a discrete set of frequencies instead of a continuous frequency domain (Valentin et al., 2017, 2016b), and by using robust computational techniques of ILP resolution, such as branch-and-cut (see Chapter 3) and branch-and-price (see Chapter 4).

Another hypothesis tested in this research was: *practical instances of the problem*

are timely solvable to optimal. We have experimented on finding feasible solutions for workload for fixed priority schedulers with up to 50 tasks distributed on four processors with seven different available frequencies. On independent hard real-time tasks scheduled using EDF policy, we found optimal distribution of up to 90 tasks on four processors with seven different available frequencies. In both cases, the solutions were found within 30 min of execution time (Valentin et al., 2016a, 2017). Similarly, on dependent tasks workload, we have optimally distributed 22 tasks, from an automotive control hard real-time application, on four processors with seven different available frequencies, with two shared resources and 23 precedence constraints within 1.5 h. We consider a few hours in the design phase a price worth paying in this context.

8.2 Future Work

Even though we consider that we have achieved the objectives of this research and tested the supporting hypotheses, we also recognize that further improvement in the developed methods are necessary. In this section, we discuss possible extensions of the present research as well as future work.

Branch-Cut-Price. As future work, we will investigate different algorithms for MGAP applied on the problem of assigning hard real-time tasks among heterogeneous processors with different performance state. We believe that combining the response time analysis with the existing algorithms may produce faster solvers for this problem. One possibility to explore is the combination of branch-and-cut with branch-and-price.

Exploring further response time analysis. Another interesting subject is to find relaxations of response time schedulability analysis to derive bounds and apply them in the branch-and-bound algorithms. Might be worth checking if such bounds can be used in a novel cut generation strategy.

Branch-and-price. In the branch-and-price based algorithmic strategies, as future work, we will investigate different algorithms for MCKP applied on the pricing problem. We will also considerer different branching strategies. We will also investigate how to perform a local search strategy for finding primal bounds.

Dependent workload. We will evaluate combining response time analysis in a Branch-Cut-Price algorithm as future work.

Migration. We envision considering migration by performing sensibility analysis to determine other feasible and optimal configurations to allow for dynamic configuration switching.

DPM and C-States. Modern processors are able to enter different C-states when idle. It is worth considering if it is possible to model and deliver optimal distributions considering DPM technological characteristics.

8.3 List of Publications

This section presents a list of works published as outcome of the present thesis. We also list works published in collaboration with other authors, in related topics of this research.

8.3.1 Published Papers

We have published the following works as outcome of this thesis:

1. Valentin, E., de Freitas, R., and Barreto, R. (2017). Towards optimal solutions for the low power hard real-time task allocation on multiple heterogeneous processors. *Science of Computer Programming*, pages –

2. **(Best Paper Award of the Real-Time Track)** Valentin, E., de Freitas, R., and Barreto, R. (2016a). Reaching optimum solutions for the low power hard real-time task allocation on multiple heterogeneous processors problem. In *VI Brazilian Symposium on Computing Systems Engineering (SBESC'2016)*, pages 128–135

3. Valentin, E. B., de Freitas, R., and Barreto, R. (2016b). Applying MGAP Modeling to the Hard Real-time Task Allocation on Multiple Heterogeneous Processors Problem. *Procedia Computer Science*, 80:1135 – 1146. International Conference on Computational Science (ICCS'2016), 6-8 June 2016, San Diego, California, USA

4. Valentin, E. and Barreto, R. (2016). Energy constraints for scheduling tasks on heterogeneous hard real-time systems: A systematic literature review. GISE Lab Technical Report TR-2016-01, Federal University of Amazonas, Manaus, AM - Brazil. URL: http://gise.icomp.ufam.edu.br/reports/TR-2016-01.pdf

5. Valentin, E., Salvatierra, M., de Freitas, R., and Barreto, R. (2015b). Response time schedulability analysis for hard real-time systems accounting DVFS latency on heterogeneous cluster-based platform. In *25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS'2015)*, pages 1–8

6. **(Ranked as First Place by SOBRAPO among the Brazilian Works)** Valentin, E. (2014). On the frequency assignment to hard-real time tasks of multicore systems. In *Proceedings of the XVIII Escuela LatinoAmericana de Verano en Investigación de Operaciones (ELAVIO'2014)*, Areia – PB, Brazil

Additionally, we have also co-authored the following works in subjects related to the present research during the course of this project:

1. Gonçalves, R. S., Pinheiro, D. Q., Valentin, E. B., Oliveira, H. A. B. F., and Barreto, R. S. (2016). Real-time tasks and voltage/frequency controller collaboration on low power energy operational systems. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 47–54

2. Bentes, L., Rocha, H., Valentin, E., and Barreto, R. (2016). Jfortes: Java formal unit test generation. In *2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 16–23

As outcome of an extension project to mentor and nurture agile process within a research environment, we have published the following work:

1. Valentin, E., Carvalho, J. R. H., and Barreto, R. (2015a). Rapid improvement of students' soft-skills based on an agile-process approach. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–9

8.3.2 Book Chapters

We have submitted the following book chapter proposal which got accepted and it is under a peer-review process.

1. **(accepted proposal)** Valentin, E., de Freitas, R., and Barreto, R. (2018). Designing distributed real-time systems to process complex control workload in the energy industry. In *Collective and Computational Intelligence Applications in Energy INDUSTRY*, pages xxx–yyy. Springer

References

Alahmad, B. N. and Gopalakrishnan, S. (2011). Energy efficient task partitioning and real-time scheduling on heterogeneous multiprocessor platforms with QoS requirements. *Sust. Computing: Inform. and Systems*, 1(4):314–328.

AlEnawy, T. and Aydin, H. (2005). Energy-aware task allocation for rate monotonic scheduling. In *Real Time and Embedded Technology and Applications Symposium (RTAS'2005)*, pages 213–223.

Audsley, N., Burns, A., Richardson, M., Tindell, K., and Wellings, A. J. (1993). Applying new scheduling theory to static priority pre-emptive scheduling. *Software Eng. Journal*, 8(5):284–292.

Avella, P., Boccia, M., and Vasilyev, I. (2013). A branch-and-cut algorithm for the multilevel generalized assignment problem. *Access, IEEE*, 1:475–479.

Awan, M. A. and Petters, S. M. (2013). Energy-aware partitioning of tasks onto a heterogeneous multi-core platform. In *Real-Time Technology and Applications - Proceedings*, pages 205–214, Philadelphia, PA.

Aydin, H., Melhem, R., Mosse, D., and Mejia-Alvarez, P. (2001a). Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Real-Time Systems, 13th Euromicro Conference on, 2001.*, pages 225–232.

Aydin, H., Melhem, R., Mossé, D., and Mejía-Alvarez, P. (2001b). Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium.*, page 95–105.

Barrefors, B., Lu, Y., Saha, S., and Deogun, J. S. (2014). A novel thermal-constrained energy-aware partitioning algorithm for heterogeneous multiprocessor real-time systems. In *IEEE 33rd International Performance Computing and Communications Conference (IPCCC'2014)*, pages 1–8.

Barreto, R. (2005). *A Time Petri Net-Based Methodology for Embedded Hard Real-Time Software Synthesis*. PhD thesis, Cin/UFPE.

Basili, V., Caldiera, G., and Rombach, H. D. (1994a). Goal Question Metric Paradigm. Technical report, Encyclopedia of Software Engineering, New York.

Basili, V., Caldiera, G., and Rombach, H. D. (1994b). The Experience Factory. Technical report, Encyclopedia of Software Engineering, New York.

Beloglazov, A., Buyya, R., Lee, Y. C., and Zomaya, A. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82:47 – 111.

Benini, L., Bogliolo, A., and Micheli, G. D. (2000). A survey of design techniques for system-level dynamic power management. In *IEEE Trans. Very Large Scale Integr. Syst.*, volume 8, page 299–316.

Bentes, L., Rocha, H., Valentin, E., and Barreto, R. (2016). Jfortes: Java formal unit test generation. In *2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 16–23.

Bini, E. and Buttazzo, G. (2004). Biasing effects in schedulability measures. In *the 16th Euromicro Conference on Real-Time Systems*, page 196–203. IEEE Computer Society.

Bini, E., Buttazzo, G., and Buttazzo, G. (2001). A hyperbolic bound for the rate monotonic algorithm. In *Real-Time Systems, 13th Euromicro Conference on, 2001.*, pages 59–66.

Biolchini, J., Mian, P. G., Natali, A. C. C., Conte, T. U., and Travassos, G. H. (2007). Scientific research ontology to support systematic review in software engineering. *Adv. Eng. Inform.*, 21(2):133–151.

Brucker, P. (2010). *Scheduling Algorithms*. Springer Publishing Company, Incorporated, 5th edition.

Burd, T. D. and Brodersen, R. W. (1996). Processor design for portable systems. *Journal of VLSI signal processing systems for signal, image and video technology*, 13(2-3):203–221.

Burns, A. and Wellings, A. (1997). *Real-Time Systems and Programming Languages*. Addison-Wesley, second edition.

Ceselli, A. and Righini, G. (2006). A branch-and-price algorithm for the multilevel generalized assignment problem. *Operations Research*, pages 1172–1184.

Chantem, T., Hu, X., and Dick, R. (2011). Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(10):1884–1897.

Chen, G., Huang, K., Buckl, C., and Knoll, A. (2013). Energy optimization with worst-case deadline guarantee for pipelined multiprocessor systems. In *Proceedings -Design, Automation and Test in Europe, DATE*, pages 45–50, Grenoble.

Chen, H., Cheng, A. M. K., and Kuo, Y.-W. (2011). Assigning real-time tasks to heterogeneous processors by applying ant colony optimization. *Journal of Parallel and Dist. Computing*, 71(1):132–142.

Chen, J.-J. and Kuo, C.-F. (2007). Energy-Efficient Scheduling for Real-Time Systems on Dynamic Voltage Scaling (DVS) Platforms. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'2007)*, pages 28–38, Daegu. IEEE.

Chen, J.-J. and Kuo, T.-W. (2006). Allocation cost minimization for periodic hard real-time tasks in energy-constrained dvs systems. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, page 255–260.

Chen, J.-J., Schranzhofer, A., and Thiele, L. (2009). Energy minimization for periodic real-time tasks on heterogeneous processing units. In *IEEE International Symposium on Parallel Distributed Processing (IPDPS'2009)*, pages 1–12.

Chen, J.-J. and Thiele, L. (2008). Energy-efficient task partition for periodic real-time tasks on platforms with dual processing elements. In *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, pages 161–168, Melbourne, VIC.

Chen, J.-J. and Thiele, L. (2009). Task Partitioning and Platform Synthesis for Energy Efficiency. *2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 393–402.

Chen, J.-J. and Thiele, L. (2011). Platform synthesis and partitioning of real-time tasks for energy efficiency. *Journal of Systems Architecture*, 57(6):573–583.

Chen, Y.-S., Chang, L.-P., and Kuo, T.-W. (2008). Multiprocessor frequency locking for real-time task synchronization. In *Proceedings of the ACM Symposium on Applied Computing*, pages 289–293, Fortaleza, Ceara.

Chu, E. T. H., Huang, T. Y., and Tsai, Y. C. (2009). An Optimal Solution for the Heterogeneous Multiprocessor Single-Level Voltage-Setup Problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(11):1705–1718.

Cochran, R. (2013). *Techniques for Adaptive Power and Thermal Sensing and Management of Multi-core Processors*. Thesis, Brown University.

Davis, R. I. and Burns, A. (2011). A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys*, 43(4):1–44.

Dick, R. P., Rhodes, D. L., and Wolf, W. (1998). TGFF: Task graphs for free. In *Int. Workshop Hardw./Softw. Co-Des.*, page 97–101.

EEMBC (2014). The embedded microprocessor benchmark consortium.

Farines, J. M., Fraga, J. S., and Oliveira, R. S. (2000). *Sistemas de Tempo Real*. Universidade Federal de Santa Catarina, Departamento de Automação e Sistemas.

Free Software Foundation (2012). Gnu linear programming kit.

Garey, M. R. and Johnson, D. S. (1979). *Computer and Intractability: a Guide to the Theory of the NP-Completeness.* W. H. Freeman and Company.

Glover, F., Hultz, J., and Klingman, D. (1979). Improved computer-based planning techniques. Part II. *Interfaces*, 9(4):12–20.

Gonçalves, R. S., Pinheiro, D. Q., Valentin, E. B., Oliveira, H. A. B. F., and Barreto, R. S. (2016). Real-time tasks and voltage/frequency controller collaboration on low power energy operational systems. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 47–54.

Google, I. (2013). The big picture - google greener.

Goossens, J., Milojevic, D., and Nélis, V. (2008). Power-aware real-time scheduling upon dual cpu type multiprocessor platforms. In *Proceedings of the 12th International Conference on Principles of Distributed Systems (OPODIS '08)*, pages 388–407, Berlin, Heidelberg. Springer-Verlag.

H.-R.Hsu, Chen, J.-J., and T.-W.Kuo (2006). Multiprocessor synthesis for periodic hard real-time tasks under a given energy constraint. In *ACM/IEEE Conference of Design, Automation, and Test in Europe (DATE).*

He, C., Zhu, X., Guo, H., Qiu, D., and Jiang, J. (2012). Rolling-horizon scheduling for energy constrained distributed real-time embedded systems. *Journal of Systems and Software*, 85(4):780–794.

He, D. and Mueller, W. (2012a). Enhanced schedulability analysis of hard real-time systems on power manageable multi-core platforms. In *Proceedings of the 14th IEEE International Conference on HPCC - 9th IEEE ICESS*, pages 1748–1753, Liverpool.

He, D. and Mueller, W. (2012b). A heuristic energy-aware approach for hard real-time systems on multi-core platforms. In *Proceedings of the 2012 15th Euromicro Conference on Digital System Design (DSD '12)*, pages 288–295, Washington, DC, USA. IEEE Computer Society.

Hettiarachchi, P. M., Fisher, N., and Wang, L. Y. (2013). Achieving Thermal-Resiliency for Multicore Hard-Real-Time Systems. In *2013 25th Euromicro Conference on Real-Time Systems*, pages 37–46, Paris. IEEE.

Huang, T.-Y., Tsai, Y.-C., and Chu, E. T.-H. (2007). A Near-optimal Solution for the Heterogeneous Multi-processor Single-level Voltage Setup Problem. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–10, Long Beach, CA. IEEE.

Hung, C.-M., Chen, J.-J., and Kuo, T.-W. (2006). Energy-efficient real-time task scheduling for a DVS system with a non-DVS processing element. In *Proceedings - Real-Time Systems Symposium*, pages 303–312, Rio de Janeiro.

IBM (2016). Ilog cplex. http://www.ilog.com/products/cplex/.

Jejurikar, R., Pereira, C., and Gupta, R. (2004). Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the Design Automation Conference*, pages 275–280.

Kim, J.-K., Siegel, H. c., Maciejewski, A., and Eigenmann, R. (2005). Dynamic mapping in energy constrained heterogeneous computing systems. In *Proceedings - 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'2005)*, volume 2005, page 64a, Denver, CO.

Kim, J.-k., Siegel, H. J., Maciejewski, A. A., Eigenmann, R., and Member, S. (2008). Dynamic Resource Management in Energy Constrained Heterogeneous Computing Systems Using Voltage Scaling. *IEEE Transactions on Parallel and Distributed Systems*, pages 1445–1457.

Kirovski, D. and Potkonjaka, M. (1997). System-level synthesis of low-power hard real-time systems. In *Proceedings of the 34th ACM/IEEE Conference on Design Automation Conference*, page 697–702.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, School of Computer Science and Mathematics Keele University and Department of Computer Science University of Durham.

Kong, J., Chung, S. W., and Skadron, K. (2012). Recent thermal management techniques for microprocessors. *ACM Computing Surveys*, 44(3):1–42.

Lauzac, S., Melhem, R., and Mosse, D. (1998). An efficient rms admission control and its application to multiprocessor scheduling. In *Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International and Symposium on Parallel and Distributed Processing 1998*, pages 511–518.

Lee, C., Lee, J. K., Hwang, T., and Tsai, S.-C. (2003). Compiler optimization on vliw instruction scheduling for low power. In *ACM Transactions on Design Automation of Electronic Systems (TODAES).*, volume 8, page 252 – 268.

Lehoczky, J., Sha, L., and Ding, Y. (1989). The rate monotonic scheduling algorithm: exact characterization and average case behavior. *Proceedings of the IEEE Real Time Systems Symposium*, pages 166–171.

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, J. W. (2000). *Real-Time Systems*. Prentice Hall, Englewood, Cliffs, NJ.

Luo, J. and Jha, N. (2002). Static and dynamic variable voltage scheduling algorithms for realtime heterogeneous distributed embedded systems. In *15th International Conference on VLSI Design (VLSID'02)*, page 719–726.

Markoff, J. and Lohr, S. (2002). Intel's huge bet turns iffy,.

Min-Allah, N., Hussain, H., Khan, S. U., and Zomaya, A. Y. (2012). Power efficient rate monotonic scheduling for multi-core systems. *Journal of Parallel and Distributed Computing*, 72(1):48–57.

National Highway Traffic Safety Administration (2016). Quick facts 2015. Technical report, U.S. Department of Transportation.

National Highway Traffic Safety Administration (2017). Estimating lives saved by electronic stability control, 2011–2015. Technical report, U.S. Department of Transportation.

Osorio, M. A. and Laguna, M. (2003). Logic cuts for multilevel generalized assignment problems. *European Journal of Operational Research*, 151(1):238 – 246.

Pawlikowski, K. (1990). Steady-state simulation of queueing processes: Survey of problems and solutions. *ACM Comput. Surv.*, 22(2):123–170.

Pawlikowski, K., Yau, V. W. C., and McNickle, D. (1994). Distributed stochastic discrete-event simulation in parallel time streams. In *Winter simulation conference (WSC '94)*, pages 723–730. Society for Computer Simulation International.

Pessoa, A. (2017). Sourceforge - firula. [visited in February-2017].

Prasanna, V. and Yu, Y. (2002). Power-aware resource allocation for independent tasks in heterogeneous real-time systems. In *Ninth International Conference on Parallel and Distributed Systems, 2002. Proceedings.*, pages 341–348.

Prescilla, K. and Selvakumar, A. I. (2013). Modified binary particle swarm optimization algorithm application to real-time task assignment in heterogeneous multiprocessor. *Microprocess. Microsyst.*, 37(6-7):583–589.

Qu, G. (2001). What is the limit of energy saving by dynamic voltage scaling? In *IEEE/ACM International Conference on Computer Aided Design*, page 560–563.

Renesas (2017). R-Car H3. [visited in May, 2017]. URL: https://www.renesas.com/en-us/solutions/automotive/products/rcar-h3.html#spec.

Saha, S., Lu, Y., and Deogun, J. S. (2012). Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems. In *Proceedings - 18th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2012 - 2nd Workshop on Cyber-Physical Systems, Networks, and Applications, CPSNA*, pages 41–50, Seoul.

Samsung Electronics Co.Ltd. (2014). Samsung exynos.

Schmitz, M., Al-Hashimi, B., and Eles, P. (2002). Energy-efficient mapping and scheduling for dvs enabled distributed embedded systems. In *DATE '02: Proceedings of the conference on Design, automation and test in Europe)*, page 514.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. Wiley.

Sha, L., Rajkumar, R., and Lehoczky, J. P. (1990). Priority inheritance protocols: An approach to real-time synchronization. *IEEE Trans. Comput.*, 39(9):1175–1185.

Sheikh, H., Tan, H., Ahmad, I., Ranka, S., and Bv, P. (2012a). Energy- and performance-aware scheduling of tasks on parallel and distributed systems. *ACM Journal on Emerging Technologies in Computing Systems*, 8(4).

Sheikh, H. F., Ahmad, I., Wang, Z., and Ranka, S. (2012b). An overview and classification of thermal-aware scheduling techniques for multi-core processing systems. *Sustainable Computing: Informatics and Systems*, 2(3):151–169.

Shin, Y., Choi, K., and Sakurai, T. (2000). Power optimization of real-time embedded systems on variable speed processors. In *Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on*, pages 365–368.

Singh, A., Shafique, M., Kumar, A., and Henkel, J. (2013). Mapping on multi/many-core systems: Survey of current and emerging trends. In *Proceedings - Design Automation Conference*, Austin, TX.

Terzopoulos, G. and Karatza, H. D. (2013). Dynamic voltage scaling scheduling on power-aware clusters under power constraints. In *Proceedings - IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 72–78, Delft. IEEE Computer Society.

Texas Instruments (2017). TDA3x SoC Processors for Advanced Driver Assist Systems (ADAS) Technical Brief.

Valentin, E. (2009). Github - hydra. [visited in February-2016].

Valentin, E. (2014). On the frequency assignment to hard-real time tasks of multicore systems. In *Proceedings of the XVIII Escuela LatinoAmericana de Verano en Investigación de Operaciones (ELAVIO'2014)*, Areia – PB, Brazil.

Valentin, E. and Barreto, R. (2010). smartenum: A branch-and-bound algorithm for optimum frequency set establishment in real-time dvfs. *Workshop on Real-Time Systems*, pages 27–38.

Valentin, E. and Barreto, R. (2016). Energy constraints for scheduling tasks on heterogeneous hard real-time systems: A systematic literature review. GISE Lab Technical Report TR-2016-01, Federal University of Amazonas, Manaus, AM - Brazil. URL: http://gise.icomp.ufam.edu.br/reports/TR-2016-01.pdf.

Valentin, E., Carvalho, J. R. H., and Barreto, R. (2015a). Rapid improvement of students' soft-skills based on an agile-process approach. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–9.

Valentin, E., de Freitas, R., and Barreto, R. (2016a). Reaching optimum solutions for the low power hard real-time task allocation on multiple heterogeneous processors problem. In *VI Brazilian Symposium on Computing Systems Engineering (SBESC'2016)*, pages 128–135.

Valentin, E., de Freitas, R., and Barreto, R. (2017). Towards optimal solutions for the low power hard real-time task allocation on multiple heterogeneous processors. *Science of Computer Programming*, pages –.

Valentin, E., de Freitas, R., and Barreto, R. (2018). Designing distributed real-time systems to process complex control workload in the energy industry. In *Collective and Computational Intelligence Applications in Energy INDUSTRY*, pages xxx–yyy. Springer.

Valentin, E., Salvatierra, M., de Freitas, R., and Barreto, R. (2015b). Response time schedulability analysis for hard real-time systems accounting DVFS latency on heterogeneous cluster-based platform. In *25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS'2015)*, pages 1–8.

Valentin, E. B., de Freitas, R., and Barreto, R. (2016b). Applying MGAP Modeling to the Hard Real-time Task Allocation on Multiple Heterogeneous Processors Problem. *Procedia Computer Science*, 80:1135 – 1146. International Conference on Computational Science (ICCS'2016), 6-8 June 2016, San Diego, California, USA.

Venkatachalam, V. and Franz, M. (2005). Power reduction techniques for microprocessor systems. *ACM Comput. Surv.*, 37(3):195–237.

Virlet, B., Zhou, X., Giacalone, J.-P., Kuhn, B., Garzarán, M., and Padua, D. (2011). Scheduling of stream-based real-time applications for heterogeneous systems. In *Proceedings of the ACM SIGPLAN Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 1–10, Chicago, IL.

Yang, C. Y., Chen, J. J., Kuo, T. W., and Thiele, L. (2009). An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems. In *2009 Design, Automation Test in Europe Conference Exhibition*, pages 694–699.

Yang, Z., Xu, C., Zhou, X., and Liu, Y. (2012). Heuristic temperature-aware DVS scheduling algorithm for MPSoC. *International Journal of Advancements in Computing Technology*, 4(1):67–76.

Yoshida, J. (2013). Toyota Case: Single Bit Flip That Killed. [visited in May, 2017]. URL: http://www.eetimes.com/document.asp?doc_id=1319903&page_number=1.

Yu, Y. and Prasanna, V. K. (2003). Resource allocation for independent real-time tasks in heterogeneous systems for energy minimization. *Journal of Information Science and Eng.*, 19(3):433–449.

Zhang, W., Bai, E., He, H., and Cheng, A. (2015). Solving energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms. *Sensors (Switzerland)*, 15(6):13778–13804.

Zhu, D. (2006). Reliability-aware dynamic energy management in dependable embedded real-time systems. In *IEEE Real-time and Embedded Technology and Applications Symposium*, page 397–407.

Zhu, Y. and Mueller, F. (2005). Feedback edf scheduling exploiting hardware-assisted asynchronous dynamic voltage scaling. In *F. LCTES*.

Annex

ANNEX A – Systematic Literature Review

Producing optimal energy systems requires knowledge of different disciplines. Heterogeneous multi-core platforms are gaining increasing interest due to their low-power applicability. Scheduling becomes particularly challenging for improving system utilization and minimizing system energy consumption and peak temperature on such platforms, particularly those subject to hard real-time constraints. We survey the state-of-the-art works in this field using a systematic literature review approach. This chapter summarizes and organizes recent research results of 29 papers (of a total of 527 searched in five different digital libraries). Because this field is multidisciplinary, we emphasize the classification of the existing literature from the perspective of different areas, e.g. real-time, scheduling, power management, and optimization. We conclude that finding optimal solutions for this scenario still has several open research questions. However, it is still too early to determine which techniques provide the best solutions and coverage for this problem.

A.1 Introduction

A systematic literature review (often referred to as a systematic review) is a method for identifying, evaluating, and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest (Kitchenham and Charters, 2007). Primary studies or primary works represent regular studies, observational studies, and experimental studies. Individual works contributing to a systematic review are primary studies. Secondary studies use the results of primary studies to obtain knowledge and stronger evidence. A systematic review is a form of secondary study. Therefore, the purpose of this systematic review is (1) to survey the state-of-the-art methods and techniques for producing software with timing guarantees for architectures with multiple heterogeneous processors under energy and temperature constraints; and (2) to categorize the existing literature from the perspective of different areas. Such effective methods and techniques support reducing power bills, improve system reliability, and increase the efficient usage of energy, thereby aiding in reducing environmental impacts.

A.1.1 Objective and Scope

The scope of this study is a systematic literature review of the subject of scheduling tasks with low power, energy consumption, and temperature constraints on heterogeneous hard real-time systems. We outline the objective of this study based on the GQM *(goal, question, and metric)* paradigm (Basili et al., 1994a,b) as follows: *analyze* primary studies on scheduling methods, schedulability analysis, and techniques applied to hard real-time

systems running on multiple heterogeneous processors with temperature and energy constraints; *for the purpose of* characterization of existing research; *With respect to their* cost/benefit offered by the existing solutions; *from the point of view of* researcher and practitioner; *in the context of* low-power-aware heterogeneous real-time system design.

Table 14 outlines the objective of this study based on the GQM *(goal, question, and metric)* paradigm (Basili et al., 1994a,b).

Table 14 – Objective of the study, structured as GQM

Analyze	Primary studies on scheduling methods, schedulability analysis, and techniques applied to hard real-time systems running on multiple heterogeneous processors with temperature and energy constraints.
For the purpose of	Characterization of existing research.
With respect to their	Cost/benefit offered by the existing solutions.
From the point of view of	Researcher and Practitioner.
In the context of	Low-power-aware heterogeneous real-time system design.

The aim of this systematic review is to find the maximum number of primary studies using an unbiased search strategy. For transparency to those interested in reproducing our results, we present the research method in Section A.2.

A.1.2 Outline

The remainder of this text is organized as follows. For consideration of the readers, Section 2 presents a list of terms and common definitions used in this text. Section A.2 includes a detailed description of the method employed to construct a comprehensive description of the state-of-the-art solutions for the addressed problem. Section A.3 guides the reader through the findings and outcomes encountered in the state-of-the-art literature. The answers to each guiding question of this systematic review are summarized in Section A.4. In Section A.5, we highlight research trends and open research questions. Section A.6 concludes this text with final remarks and considerations.

A.2 Research Method: Systematic Literature Review Protocol

A systematic literature review was conducted as part of the present work. The objective is to identify methods for addressing the problem of controlling energy constraints on heterogeneous multi-core hard real-time systems. This section presents the guiding protocol of the systematic literature review process.

The protocol defined in this text follows the design specified by Kitchenham and Charters (2007) and Biolchini et al. (2007). We encourage interested readers to consider the technical report (Valentin and Barreto, 2016) to access the full protocol definition, including the search string, all the extracted data, and complete description of each inclusion and exclusion criteria.

A.2.1 Research Questions

This systematic literature review aims to answer the following questions:

Q1 What are the available solutions (treatments/interventions) to offer guarantees of temporal correctness for hard real-time systems executed on multiple heterogeneous processors with low energy and temperature constraints?

Q1.1 What are the costs involved in / imposed by the existing solutions? (non-functional software metrics: overhead, latency, project delays, etc.)

Q1.2 What is the energy consumption reduction provided by each solution?

Q1.3 Are there any side effects expected while applying each solution (collaterals) on other software metrics (overhead, system cooling, etc.)?

Q1.4 What are the expected industry contexts that each solution applies to (reported context: real-time system, servers, telecommunication systems, cellphone terminals, Internet routers, etc.)?

Q1.5 Is the solution sufficient (context of schedulability test results)?

Q1.6 Is the solution necessary (context of schedulability test results)?

This study also extracts additional data for supporting the classifying, ranking, and positioning of each work. We summarize the obtained answers for each research question and the additional extracted data in Section A.3.

A.2.2 Search Process

A.2.2.1 Process for selecting and classifying primary studies (search strategy)

The steps constituting the process for selecting and classifying primary studies are as follows: in Selection Step 01, the researchers use the search string in the selected digital library; in Selection Step 02, the researchers construct a set of valid papers by applying the inclusion and exclusion criteria; the focus is on paper title, abstract, and keywords only; in Selection Step 03, the researchers read the papers included in Step 02 fully and confirms the inclusion and exclusion; in Selection Step 04, the researchers classify the papers included in Step 03 based on its outcome results.

A.2.2.2 Inclusion/Exclusion Criteria

In this study, we include only papers addressing hard real-time constraints on multiple heterogeneous processors; applying schedulability analysis, and (or) task assignment / partitioning / allocation / scheduling as interventions; and whose the direct outcome is guaranteed energy consumption reduction for processor, memories, caches, and system.

The exclusion criteria are designed to avoid including unrelated works. Additionally, we exclude works that are not accessible. Therefore, the exclusion criteria are as follows: *Language* – papers not written in the English language; *Availability* – papers where the full text is not available on the world wide web or by request from the authors; *Different Subject* – papers that clearly cover other topics different from the primary object of this study; *Non-Digital* – papers where the full text is not available in digital format; *No Key* – papers where the title, abstract, or full text do not contain the keywords or its synonyms composing the search string; *Key in Reference* – papers where keywords appear only in references, acknowledgements, authors' biographies, bibliography entries, and appendices; *Different Genre* – event schedules, editorials, keynotes speeches, tutorials, courses, workshops, white-papers, or similar genres are excluded; *Non Real-Time* – papers where real-time systems are not the main type of systems considered or where time constraints are not mentioned; *Non HMP* – papers where multiple heterogeneous processors, or similar systems, are not the main focus or are not even mentioned; *Non Scheduling* – papers that do not mention or do not consider schedulability analysis, task allocation/partitioning/assignment, frequency allocation/assignment or resource allocation; *Homogeneous Cores* – papers covering only homogeneous processors; *Single-Core* – papers covering only systems using a single processor; *Soft Real-Time* – papers covering only soft real-time systems; and *Non Power Management (PM) nor Thermal Management (TM), Non PM/TM* – papers that do not clearly consider low-power, energy management or temperature management.

A.2.2.3 List of (digital) libraries

The searched digital libraries are as follows: (i) Scopus: <http://www.scopus.com>;(ii) IEEE Xplorer: <http://ieeexplore.ieee.org/Xplore/home.jsp>; (iii) Web of Science: <http://apps.webofknowledge.com>; (iv) Science Direct: <http://www.sciencedirect.com>; and (v) Springer: <http://www.springer.com>.

A.2.2.4 Study quality assessment checklists and procedures

The main purpose of this quality assessment is to explicitly determine whether the source of information is reliable. The researchers evaluate whether each study reports repeatable results. The researchers evaluate each primary study according to the criteria presented in Table 15.

Table 15 – Quality Assessment Questionnaire

#	Questions	Criteria
1	What type of study is presented in this work?	*{0.0, 0.2, 0.4, 0.6, 0.8, 1.0}*[a]
2	Were the basic data/studies adequately reported?	*{No = 0.0, Partially = 0.5, Yes = 1.0}*
3	Is the employed evaluation method valid? Can it be reproduced from the data/information reported?	*{No = 0.0, Partially = 0.5, Yes = 1.0}*
4	Is the importance of the results for the knowledge area commented?	*{No = 0.0, Partially = 0.5, Yes = 1.0}*
5	The result interpretations arise in a logical way from the data?	*{No = 0.0, Partially = 0.5, Yes = 1.0}*
6	Are the research design deficiencies clearly reported?	*{No = 0.0, Partially = 0.5, Yes = 1.0}*

[a] – Presentation of a new Product/Tool=0.2, Presentation of something different=0.4, Presentation of something presumably new=0.6, Presentation of something admittedly new=0.8, Presentation of proof=1.0.

A.2.2.5 Data Collection

From Dec 4th, 2013 to Dec 16th, 2013, the researchers performed the first phase of data selection in the preliminary database of works. The data extraction phase occurred from Dec 17th, 2013 to Feb 28th, 2014. In this phase, the researchers read the works in their entirety. The researchers applied the inclusion and exclusion criteria of Step 03 and the classification criteria of Step 04. After including and classifying the works, the researchers extracted the following information: (i) answers to the quality assessment questions, (ii) answers to the research question, (ii) answers to the questions of the researchers' interest, and answers to the ranking questions. The researchers performed another data collection phase on April 03rd, 2016 to update the database with primary works published between 2013 and 2016. The results were processed and merged with the first database during the period of April 2016 to June 2016.

A.2.2.6 Data Analysis

The researchers merged the works returned by the digital libraries into a table. Additionally, the researchers removed duplicate works. The researchers used the inclusion and exclusion criteria present in Step 03 to filter the primary works. A senior researcher

approved and discussed the decisions for including or excluding works whenever a doubt appeared. The resulting database is the primary source of extracted data. We classify, position, and rank each primary work presented in this study using the extracted data. The extracted data also detail the methods and techniques used to model, solve, experiment, and analyze information regarding the problem of interest.

A.3 Results of the Systematic Literature Review

After merging the results of all digital libraries and removing the duplicates, 527 works compose the preliminary database of primary works as result of the selection Step 01. Considering the results of all five digital libraries in the first phase of data collection, there are 468 primary works, including duplicates. Scopus returns 219 works. IEEE Xplorer returns seven works with the first string and eight works with the second string, totaling 15 works. Web of Science returns three works. ScienceDirect returns 127 works using the first string and 104 works using the second string, totaling 231 works. Springer returns zero works. All of these total 468 works, of which 154 are duplicates. The second phase of data collection gathered an extra of 228 primary works; one from Web of Science, 46 from IEEE, 61 from Science Direct, and 153 from Scopus.

Step 02 retrieves the full text; reads the title, keywords, and abstract; performs a scanning read of the full text; and applies the inclusion and exclusion criteria. Step 02 selects 86 works. In addition, Step 02 also includes a phase to read the list of references present in the primary works. The list of references identifies two relevant works. The criteria of Step 02 also select the two manually identified works.

Therefore, a total of 88 selected works compose the database of primary works. The bar graph presented in Figure 24 illustrates the distribution of the applied exclusion criteria. We present the distribution of exclusion criteria as a matter of transparency with respect to the systematic review methodology. Readers interested in repeating the same research methodology, for updating or validating our reported data, may find the same exclusion results when considering the same search period. The exclusion criteria are those of Step 02. More than one criterion classifies a single work. Most excluded works, 169, belong to studies using architectures different from Heterogeneous Multiple Processors (HMP). Most excluded works are also part of studies that disregard power and temperature restrictions, 155. Step 03 excludes 59 from the 88 studies. Step 03 considers the full text of each work. Step 04 only classifies the works selected during Step 03. The final database contains 29 primary works.

The publication vehicles present in the database of primary works are diverse. The *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* is the main publication venue. Three (10 %) of the 29 works were published in

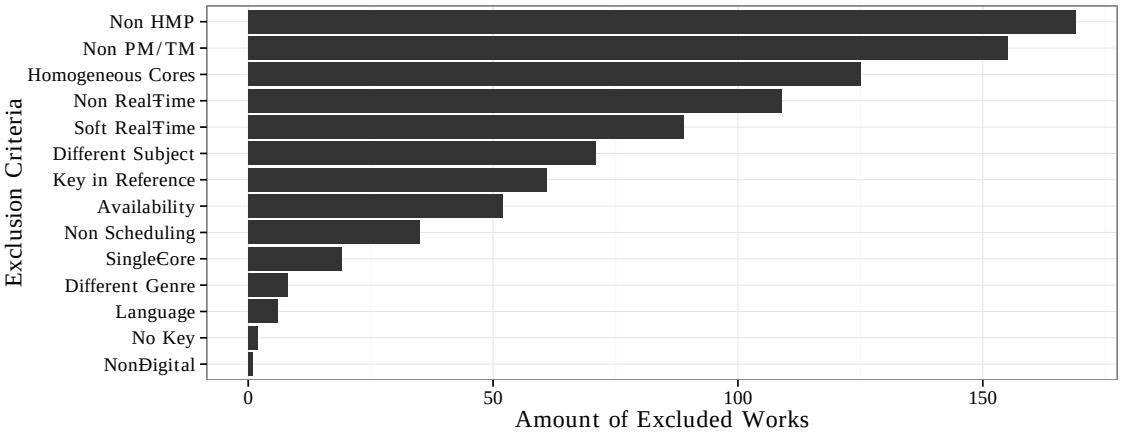


Figure 24 – Distribution of Exclusion Criteria of Step 02

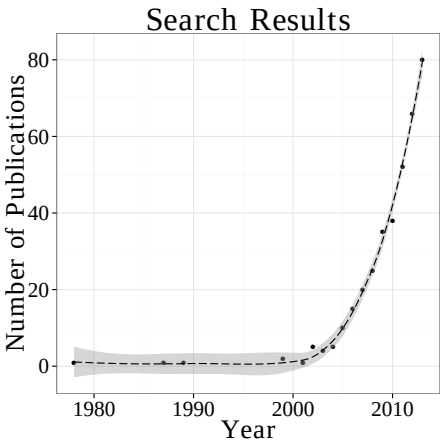
it. Two journals compete for the second place of the most used publication vehicle in this topic: *Microprocessors and Microsystems journal* and *Journal of Parallel and Distributed Computing*. Each of them contains two (6 %) items of the database of primary works.

Figure 25 illustrates the yearly publication evolution of all works during each phase of the systematic literature review. Figure 25a presents the yearly evolution of all 529 considered works. Figure 25b presents the yearly publication evolution of the 88 resulting works after Step 02. Figure 25c illustrates the yearly publication evolution of the 29 works present in the final database.

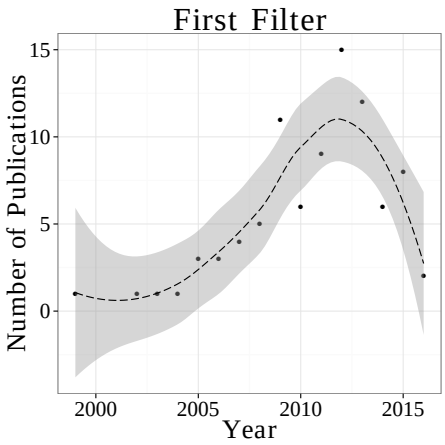
The inclusion criteria in Step 02 classify the 29 items of the database of primary works. There are works that use Multiple Heterogeneous Processors (MPH), i.e., applied in the same device, such as MPSoCs or heterogeneous clusters. But there are also primary works applied on distributed computing systems. A total of 27 (93 %) works address multiple heterogeneous processors. There are two (6 %) works on distributed computing systems.

The authors with multiple publications on this subject are Jian-Jia Chen and Lothar Thiele. Jian-Jia Chen appears in seven publications (24 % of the database of primary works). At the time of his publications, Jian-Jia Chen was with the National Taiwan University, Taiwan, and with Swiss Federal Institute of Technology, Switzerland. Lothar Thiele appears in five publications (17 % of the database of primary works). Da He, Rudolf Eigenmann, Jong-Kook Kim, Tei-Wei Kuo, Anthony Maciejewski, Viktor Prasanna, Howard Jay Siegel, and Wolfgang Mueller are authors that appear in two publications (6 % of the database of primary works).

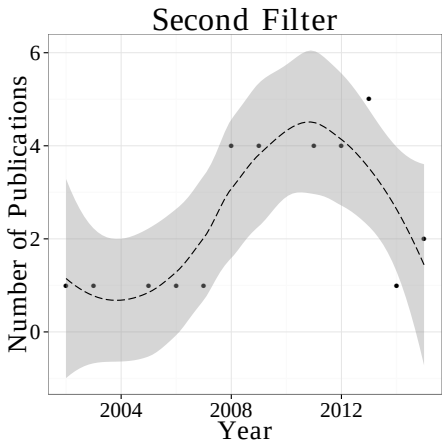
Table 16 lists the works that compose the final database of primary works. The extraction process utilizes all works present in Table 16, as explained in the protocol described in Section A.2.



(a) Preliminary Database.



(b) After First Filter.



(c) Final Database.

Figure 25 – Publication count per year

Table 16 – Final database of primary works

Authors	Title	Year	REF
Yang, Y. *et al.*	Power-aware resource allocation for independent tasks in heterogeneous real-time systems	2002	(Prasanna and Yu, 2002)
Yu, Y. *et al.*	Resource allocation for independent real-time tasks in heterogeneous systems for energy minimization	2003	(Yu and Prasanna, 2003)

continued

Table 16 – Final database of primary works

Authors	Title	Year	REF
Kim, J.-K. *et al.*	Dynamic mapping in energy constrained heterogeneous computing systems	2005	(Kim et al., 2005)
Hung, C.-M. *et al.*	Energy-efficient real-time task scheduling for a DVS system with a non-DVS processing element	2006	(Hung et al., 2006)
Chen, J.-J. *et al.*	Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms	2007	(Chen and Kuo, 2007)
Chen, J.-J. *et al.*	Energy-efficient task partition for periodic real-time tasks on platforms with dual processing elements	2008	(Chen and Thiele, 2008)
Chen, Y.-S. *et al.*	Multiprocessor frequency locking for real-time task synchronization	2008	(Chen et al., 2008)
Goossens, J. *et al.*	Power-aware real-time scheduling upon dual CPU type multiprocessor platforms	2008	(Goossens et al., 2008)
Kim, J.-K. *et al.*	Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling	2008	(Kim et al., 2008)
Chen, J.-J. *et al.*	Energy minimization for periodic real-Time tasks on heterogeneous processing units	2009	(Chen et al., 2009)
Chen, J.-J. *et al.*	Task partitioning and platform synthesis for energy efficiency	2009	(Chen and Thiele, 2009)
Yang, C.-Y. *et al.*	An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems	2009	(Yang et al., 2009)

continued

Table 16 – Final database of primary works

Authors	Title	Year	REF
Chu, E. T. H. *et al.*	An Optimal Solution for the Heterogeneous Multiprocessor Single-Level Voltage-Setup Problem	2009	(Chu et al., 2009)
Alahmad, B. N. *et al.*	Energy efficient task partitioning and real-time scheduling on heterogeneous multiprocessor platforms with QoS requirements	2011	(Alahmad and Gopalakrishnan, 2011)
Chantem, T. *et al.*	Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs	2011	(Chantem et al., 2011)
Chen, H. *et al.*	Assigning real-time tasks to heterogeneous processors by applying ant colony optimization	2011	(Chen et al., 2011)
Chen, J.-J. *et al.*	Platform synthesis and partitioning of real-time tasks for energy efficiency	2011	(Chen and Thiele, 2011)
He, D. *et al.*	Enhanced schedulability analysis of hard real-time systems on power manageable multi-core platforms	2012	(He and Mueller, 2012a)
Min-Allah, N. *et al.*	Power efficient rate monotonic scheduling for multi-core systems	2012	(Min-Allah et al., 2012)
Saha, S. *et al.*	Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems	2012	(Saha et al., 2012)
Yang, Z. *et al.*	Heuristic temperature-aware DVS scheduling algorithm for MPSoC	2012	(Yang et al., 2012)

continued

Table 16 – Final database of primary works

Authors	Title	Year	REF
Awan, M. A. *et al.*	Energy-aware partitioning of tasks onto a heterogeneous multi-core platform	2013	(Awan and Petters, 2013)
Hettiarachchi, P.M. *et al.*	Achieving thermal-resiliency for multicore hard-real-time systems	2013	(Hettiarachchi et al., 2013)
Prescilla, K. *et al.*	Modified Binary Particle Swarm optimization algorithm application to real-time task assignment in heterogeneous multiprocessor	2013	(Prescilla and Selvakumar, 2013)
He, D. *et al.*	A heuristic energy-aware approach for hard real-time systems on multi-core platforms	2013	(He and Mueller, 2012b)
Terzopoulos, G. *et al.*	Dynamic voltage scaling scheduling on power-aware clusters under power constraints	2013	(Terzopoulos and Karatza, 2013)
Barrefors, B. *et al.*	A novel thermal-constrained energy-aware partitioning algorithm for heterogeneous multiprocessor real-time systems	2014	(Barrefors et al., 2014)
Valentin, E. *et al.*	Response time schedulability analysis for hard real-time systems accounting DVFS latency on heterogeneous cluster-based platform	2015	(Valentin et al., 2015b)
Zhang, W. *et al.*	Solving energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms	2015	(Zhang et al., 2015)

A.3.1 Classifications and Metrics of the Existing Literature

This section classifies the papers in different categories: modeling strategies, solution type, maximum instance size found on each work, workload complexity, task model, and complexity of the energy model. Section A.3.1.7 summarizes the coverage of what is present in each category.

We have assigned crosses (+) to each work for all categories. The number of crosses scores is given based on the complexity of each item. For example, a study that considers a discrete set of processor frequencies receives more crosses in this aspect than a study that considers a continuous domain of frequencies, because (i) nowadays the processors in semiconductor market have only discrete frequencies available; and (ii) in the process of finding optimal frequencies assignment, using a discrete set makes the decision problem a Mixed Integer Programming (usually a 0-1 formulation), which is typically harder to solve than and problem formulation that uses continuous domain (Real), which can be solved using Simplex.

A.3.1.1 Overall Model

This section presents a classification based on the complexity of the model presented in the text of the primary works. Table 17 presents a list of primary works with their respective model applicability. The column Total represents the number of crosses.

Table 17 – Classification based on overall model

REF	Frequency Domain Type[1]	Processor Heterogeneity[2]	Total
(Chen and Kuo, 2007)	0	+	1
(Zhang et al., 2015)	+	+	2
(Goossens et al., 2008)	0	++	2
(Awan and Petters, 2013), (Chen et al., 2011; Hettiarachchi et al., 2013)	0	+++	3
(Hung et al., 2006; Min-Allah et al., 2012)	+++	+	4
(Chen et al., 2009; Chen and Thiele, 2008, 2009, 2011; Chen et al., 2008; Chu et al., 2009; Prescilla and Selvakumar, 2013; Yang et al., 2009)	+	+++	4
(He and Mueller, 2012a,b; Saha et al., 2012; Terzopoulos and Karatza, 2013; Valentin et al., 2015b)	+++	++	5

continued

Table 17 – Classification based on overall model

REF	Frequency Domain Type[1]	Processor Heterogeneity[2]	Total
(Alahmad and Gopalakrishnan, 2011; Barrefors et al., 2014; Chantem et al., 2011; Kim et al., 2005, 2008; Prasanna and Yu, 2002; Yang et al., 2012; Yu and Prasanna, 2003)	+++	+++	6

[1] – Continuous = +, Discrete = +++, Not Applicable = 0.
[2] – Full-chip or Per-core based = +, Heterogeneous Clusters = ++, Specialized Architecture = +++.

Most works in this study, 19 out of 29, consider full heterogeneous systems (Specialized Architecture = +++). The frequency domain is continuous in 9 out of 29 texts. Most works, 15 out of 29, consider the discrete frequency domain (Discrete = +++). The works of Alahmad and Gopalakrishnan (2011), Barrefors et al. (2014), Chantem et al. (2011), Kim et al. (2008), Kim et al. (2005), Prasanna and Yu (2002), Yang et al. (2012), and Yu and Prasanna (2003) are the most complex according to these criteria.

A.3.1.2 Solution Type

This section classifies the primary works according to solution type. Table 18 presents a list of primary works classified by solution type in terms of appropriateness in exact or approximated results. The column Total represents the number of crosses.

Table 18 – Classification based on solution type

REF	Exact Solution[1]	Approximated Solution[2]	Total
(Awan and Petters, 2013; Barrefors et al., 2014), (Chen et al., 2011; Chen and Kuo, 2007; Chen et al., 2009; Chen and Thiele, 2009, 2011; Goossens et al., 2008; He and Mueller, 2012b; Hettiarachchi et al., 2013; Hung et al., 2006; Kim et al., 2005, 2008; Min-Allah et al., 2012; Prasanna and Yu, 2002; Prescilla and Selvakumar, 2013; Saha et al., 2012; Terzopoulos and Karatza, 2013; Yang et al., 2009, 2012; Yu and Prasanna, 2003; Zhang et al., 2015)	0	+	1
(Chen et al., 2008; He and Mueller, 2012a; Valentin et al., 2015b)	+++	0	3
(Alahmad and Gopalakrishnan, 2011; Chantem et al., 2011; Chen and Thiele, 2008; Chu et al., 2009)	+++	+	4

[1] – No = 0, Yes = +++ .
[2] – No = 0, Yes = + .

Most works present in this study, 26 out of 29, offer approximated solutions. Only seven works out of 29 offer exact solutions: Chen et al. (2008), He and Mueller (2012a), Valentin et al. (2015b), Alahmad and Gopalakrishnan (2011), Chantem et al. (2011), Chen and Thiele (2008), and Chu et al. (2009). The number of exact solutions suggests a need to investigate how to exactly solve this problem.

A.3.1.3 Instance size

This section exposes the size of the problem instances used during empirical experiments in each primary work. Table 19 presents a list of primary works with their respective largest reported instance size. The column Total is the result of the multiplication between the maximum reported value on each parameter, which includes the number of tasks, number of processors, number of power modes, and number of processor frequencies.

Table 19 – Classification based on the size of the instance

REF	Exact Solution	Tasks*	Processors*	Power Modes*	Frequencies*	Total
(Chen et al., 2011)	No	N/R	N/R	1	1	1
(Chen and Kuo, 2007)	No	N/R	N/R	N/R	N/R	1
(Yang et al., 2012)	No	54	N/R	1	N/R	54
(Yang et al., 2009)	No	14	6	1	1	84
(Hung et al., 2006)	No	10	2	1	5	100
(Goossens et al., 2008)	No	20	10	2	1	400
(Prescilla and Selvakumar, 2013)	No	100	4	1	N/R	400
(Terzopoulos and Karatza, 2013)	No	3	8	2	12	576
(Hettiarachchi et al., 2013)	No	64	8	2	1	1024
(Zhang et al., 2015)	No	140	8	1	1	1120
(Chen and Thiele, 2009)	No	30	20	2	1	1200
(Chen et al., 2009)	No	30	30	2	1	1800
(Chen and Thiele, 2011)	No	40	30	2	1	2400
(He and Mueller, 2012b)	No	20	16	3	5	4800
(Kim et al., 2005, 2008)	No	50	8	1	16	6400
(Prasanna and Yu, 2002; Yu and Prasanna, 2003)	No	100	10	1	8	8000
(Awan and Petters, 2013)	No	500	5	5	1	12500
(Saha et al., 2012)	No	200	64	1	N/R	12800
(Barrefors et al., 2014)	No	450	20	2	1	18000
(Min-Allah et al., 2012)	No	50	12	1	100	60000
(Chen et al., 2008)	Yes	20	N/R	2	1	40
(Chen and Thiele, 2008)	Yes	20	2	2	1	80
(Alahmad and Gopalakrishnan, 2011)	Yes	10	4	2	4	320
(Chantem et al., 2011)	Yes	30	11	1	1	330
(Chu et al., 2009)	Yes	25	30	1	1	750
(Valentin et al., 2015b)	Yes	30	32	2	5	9600
(He and Mueller, 2012a)	Yes	100	16	3	5	24000

* – N/R = Not Reported.

The largest problem instance solved exactly is 100 tasks, 16 cores, three power modes, and five frequencies. The largest problem instance reported with an approximated solution is an experiment with 50 tasks, 12 cores, one power state, and 50 frequencies.

A.3.1.4 Workload considerations

This Section categorizes each primary work based on the workloads applied in empirical experiments. Table 20 presents a list of primary works with their respective coverage and applicability on each workload type. The column Total represents the number of crosses.

Table 20 – Classification based on the type of workloads

REF	Off-line allocation[1]	On-line allocation[2]	Workload Type[3]	Total
(Chen and Kuo, 2007; He and Mueller, 2012a)	0	0	+	1
(Alahmad and Gopalakrishnan, 2011; Barrefors et al., 2014; Chantem et al., 2011), (Chen et al., 2011, 2009; Chen and Thiele, 2008, 2009, 2011; Chu et al., 2009; Goossens et al., 2008; Hettiarachchi et al., 2013; Hung et al., 2006; Min-Allah et al., 2012; Prasanna and Yu, 2002; Prescilla and Selvakumar, 2013; Saha et al., 2012; Valentin et al., 2015b; Yang et al., 2009, 2012; Yu and Prasanna, 2003; Zhang et al., 2015)	+	0	+	2
(Terzopoulos and Karatza, 2013)	0	+++	+	4
(Chen et al., 2008; He and Mueller, 2012b)	+	+++	+	5
(Awan and Petters, 2013)	+	0	++++	5
(Kim et al., 2005, 2008)	0	+++	++++	7

[1] – No = 0, Yes = + .
[2] – No = 0, Yes = +++ .
[3] – Mixed Hard & Soft RT = ++++, Periodic/Sporadic Tasks (P) = +.

Most works, 24 out of 29, present an off-line allocation solution. There are only five works providing on-line based allocation solutions. The most used workload type, 26 out of 29, is Periodic or Sporadic real-time tasks. Only the works conducted by Awan and Petters (2013), Kim et al. (2008), and Kim et al. (2005) consider a mixed workload type,

sporadic or periodic tasks together with best effort tasks.

A.3.1.5 Task model considerations

This section classifies the primary works according to the complexity of task models reported in their text. Table 21 presents a list of primary works with their respective applicability to each task model characteristic. The column Total represents the number of crosses.

Table 21 – Classification based on task model considerations

REF	MA[1]	PA[2]	PC[3]	EC[4]	RTA[5]	UA[6]	Total
(Chen and Kuo, 2007; Hettiarachchi et al., 2013; Kim et al., 2005, 2008; Terzopoulos and Karatza, 2013)	0	0	0	0	0	0	0
(Alahmad and Gopalakrishnan, 2011; Awan and Petters, 2013; Barrefors et al., 2014; Chen et al., 2009; Chen and Thiele, 2008, 2009, 2011; Chu et al., 2009; He and Mueller, 2012a; Hung et al., 2006; Prescilla and Selvakumar, 2013; Yang et al., 2009)	0	0	0	0	0	+	1
(Chantem et al., 2011; Yang et al., 2012)	0	0	+++	0	0	0	3
(Chen et al., 2011; He and Mueller, 2012b; Prasanna and Yu, 2002; Saha et al., 2012; Yu and Prasanna, 2003; Zhang et al., 2015)	0	+++	0	0	0	+	4
(Goossens et al., 2008)	+++	0	0	0	0	+	4
(Min-Allah et al., 2012)	0	+++	0	0	+++	0	6
(Chen et al., 2008)	0	+++	+++	+++	0	+	10
(Valentin et al., 2015b)	0	+++	+++	+++	+++	0	12

[1] – Migration allowed (MA): No = 0, Yes = +++ .
[2] – Preemption allowed (PA): No = 0, Yes = +++ .
[3] – Precedence Constraints (PC): No = 0, Yes = +++ .
[4] – Exclusion Constraints (EC): No = 0, Yes = +++ .
[5] – Response Time Analysis (RTA): No = 0, Yes = +++ .
[6] – Utilization Analysis (UA): No = 0, Yes = + .

Only the work of Goossens et al. (2008) considers task migration. Additionally, only Min-Allah et al. (2012), and Valentin et al. (2015b) take response time analysis into account. Similarly, only Chen et al. (2008), and Valentin et al. (2015b) account for mutual exclusion constraints. Most works, 20 out of 29, consider utilization bound. Only four out of 29 works consider precedence constraints. The work conducted by Valentin et al. (2015b) accounts for preemption, precedence and exclusion constraints, together with response time analysis, which compose the most complex task model according to these criteria.

A.3.1.6 Energy constraints

This section classifies the primary works according to the complexity of the energy models. Table 22 presents a list of primary works with their respective consideration of energy constraints in their models. The column Total represents the number of crosses.

Table 22 – Classification based on energy model considerations

REF	Power-Aware[1]	Thermal-Aware[2]	Power Model[3]	Energy Model[4]	Power Techniques[5]	Total
(Chen and Kuo, 2007)	0	0	+	+	+	3
(Awan and Petters, 2013; Zhang et al., 2015)	+	0	+	+	+	4
(Chen et al., 2011; Kim et al., 2005, 2008; Min-Allah et al., 2012; Prasanna and Yu, 2002; Prescilla and Selvakumar, 2013; Terzopoulos and Karatza, 2013; Yu and Prasanna, 2003)	+	0	+	+	+	4
(Chen and Thiele, 2008, 2009, 2011; Chen et al., 2008; He and Mueller, 2012a)	+	0	+	+	++	5
(Chu et al., 2009)	+	0	+	+++	+	6
(Yang et al., 2012)	0	+	+	+++	+	6
(Hung et al., 2006; Valentin et al., 2015b; Yang et al., 2009)	+	0	+	+++	+	6
(Chantem et al., 2011; Saha et al., 2012)	+	+	+	+++	+	7

continued

Table 22 – Classification based on energy model considerations

REF	Power-Aware[1]	Thermal-Aware[2]	Power Model[3]	Energy Model[4]	Power Techniques[5]	Total
(Hettiarachchi et al., 2013)	0	+	+++	+++	+	8
(He and Mueller, 2012b)	+	0	+	+++	+++	8
(Chen et al., 2009)	+	0	+++	+++	+	8
(Alahmad and Gopalakrishnan, 2011; Goossens et al., 2008)	+	0	+++	+++	++	9
(Barrefors et al., 2014)	+	+	+	+++	+++	9

[1] – No = 0, Yes = + .
[2] – No = 0, Yes = + .
[3] – Convex function = +, Non-Convex Function = +++.
[4] – Dynamic Power = +, Dynamic & Static Power = +++.
[5] – DPM = +, DVFS = +, DVFS & DPM = ++, DVFS & DPM & Load Balance = +++ , Load Balance = +, Load Balance & DPM = ++.

In this set of works, only five works propose solutions applicable to thermal constrained scenarios: Yang et al. (2012), Chantem et al. (2011), Saha et al. (2012), Hettiarachchi et al. (2013), and Barrefors et al. (2014). Most works, 26 out of 29, consider low-power use cases. The most frequent power model, 25 out of 29, is the convex power function. Only four works out of 29 consider non-convex power models. The lack of existing works considering non-convex power models suggests that it is worth exploring different models. Only 7 of these 29 works consider DVFS combined with the DPM technique in their solutions. Alahmad and Gopalakrishnan (2011), Barrefors et al. (2014), and Goossens et al. (2008) contain the most complex power models according to these criteria.

A.3.1.7 Compendious classification

This section summarizes each previous category. We list each primary work according to their results in the classification of overall model, solution type, workload considerations, task model considerations, energy constraints, and instance size. Table 23 presents a list of primary works with their respective coverage for each category.

Table 23 – Compendious classification of primary works

REF	Overall Model	Solution Type	Workload Type	Task Model	Energy	Instance Size
(Chen et al., 2011)	3	1	2	4	4	1
(Chu et al., 2009)	4	4	2	1	6	750
(Valentin et al., 2015b)	5	3	2	12	6	9600
(Alahmad and Gopalakrishnan, 2011)	6	4	2	1	9	320
(Awan and Petters, 2013)	3	1	5	1	4	12500
(Barrefors et al., 2014)	6	1	2	1	9	18000
(Chantem et al., 2011)	6	4	2	3	7	330
(Chen and Kuo, 2007)	1	1	1	0	3	1
(Chen and Thiele, 2008)	4	4	2	1	5	80
(Chen et al., 2008)	4	3	5	10	5	40
(Chen and Thiele, 2009)	4	1	2	1	5	1200
(Chen and Thiele, 2011)	4	1	2	1	5	2400
(Chen et al., 2009)	4	1	2	1	8	1800
(He and Mueller, 2012a)	5	3	1	1	5	24000
(He and Mueller, 2012b)	5	1	5	4	8	4800
(Hettiarachchi et al., 2013)	3	1	2	0	8	1024
(Hung et al., 2006)	4	1	2	1	6	100
(Prasanna and Yu, 2002)	6	1	2	4	4	8000
(Kim et al., 2005)	6	1	7	0	4	6400
(Kim et al., 2008)	6	1	7	0	4	6400
(Goossens et al., 2008)	2	1	2	4	9	400
(Min-Allah et al., 2012)	4	1	2	6	4	60000
(Prescilla and Selvakumar, 2013)	4	1	2	1	4	400
(Saha et al., 2012)	5	1	2	4	7	12800
(Terzopoulos and Karatza, 2013)	5	1	4	0	4	576
(Yang et al., 2012)	6	1	2	3	6	54
(Yang et al., 2009)	4	1	2	1	6	84
(Yu and Prasanna, 2003)	6	1	2	4	4	8000
(Zhang et al., 2015)	2	1	2	4	4	1120

The work of Valentin et al. (2015b) covers most items considered in this classification. Although the work performed by Valentin et al. (2015b) lacks experimentation on large instances, it contains complex power models, task models, and solutions. Chen and Kuo (2007) has the least coverage of characteristics considered in this study.

A.3.2 Statistics on the primary works

This section classifies and presents statistics regarding primary works. The statistics presented in this section provide an overview of the coverage of different aspects, including modeling techniques, solution methods, and benchmarks. We present the statistics related to schedulability tests, problem models, solution methods, empirical experiment design, benchmarks, solution precision, allocation time, workload type, energy constraints, type of power reduction, power model, and power control techniques. The following statistics consider the 29 primary works presented in the final database of this study.

The most frequent schedulability test is the utilization-based condition, whereas the least frequent is the response-time analysis. Figure 26 illustrates the distribution of the schedulability tests across the 29 works.

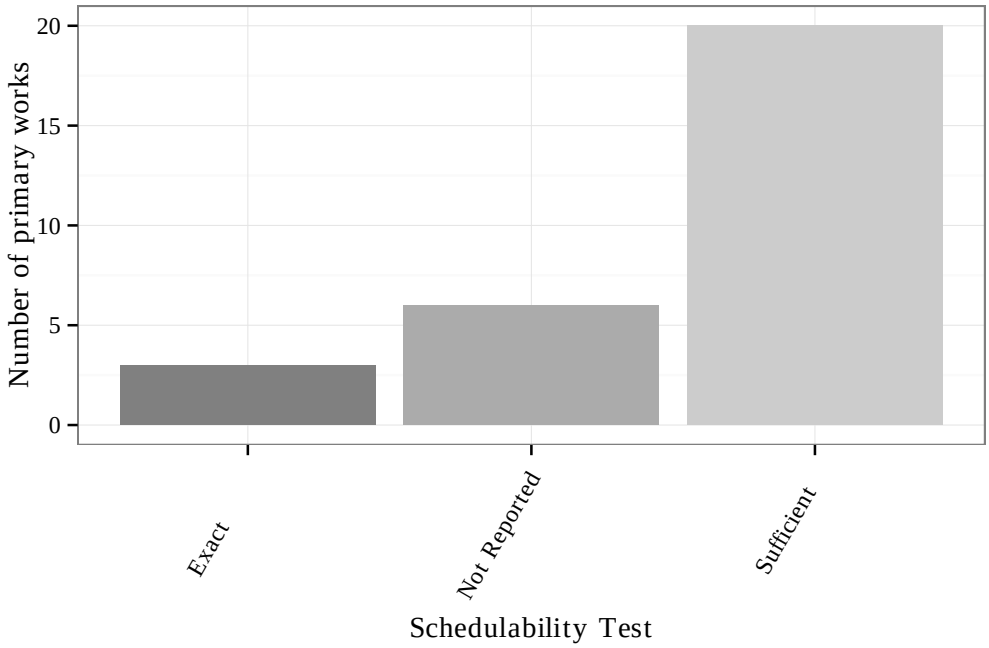


Figure 26 – Distribution of schedulability tests found in this study

Regarding the problem of minimizing the energy consumption of hard real-time systems executed on heterogeneous multicore platforms, the most frequent formal problem formulation is integer linear programming (ILP). However, most works, a total of 15 (51 %), present a non-formal problem description. Figure 27 has a bar graph with the distribution of problem models across the 29 works.

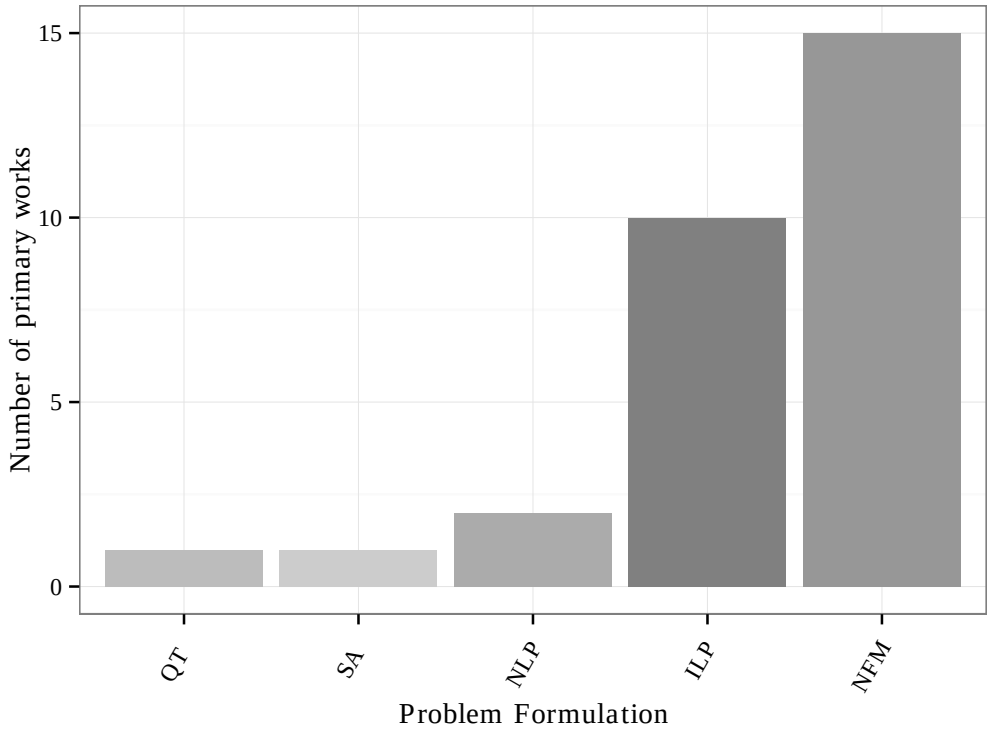


Figure 27 – Distribution of problem formulations found in this study: Integer Linear Programming (ILP), Non-Linear Programming (NLP), Schedulability Analysis (SA), Queue Theory (QT), and No Formal Model (NFM)

Figure 28 presents a bar graph with the distribution of solution methods across the 29 works. The most frequent, 9 (31 %), solution method in this study is heuristics.

Simulated environments, 26 (89 %), are the most frequent empirical experimentation method. Only one work reports the results of experiments conducted using a real-life environment. The tools E3S (EEMBC, 2014), TGFF (Dick et al., 1998), and UUnifast (Bini and Buttazzo, 2004) are benchmarks used in four primary works of this study. The majority of the works, 24 (82 %), disregard benchmark usage.

Figure 29 shows the distribution of power control techniques across the 29 works. The most frequent, 16 (55 %), power control technique is DVFS. DVFS also appears combined with other techniques, such as DPM and load balance.

Figure 30a shows the solution precision presented by the works in this study, in which the most frequent is approximated solutions. Figure 30b shows the type of allocation used by the works in this study, in which the most frequent is off-line allocation. Figure 30c depicts the workload type, in which the most frequent is periodic or sporadic real-time tasks. Figure 30d shows the energy constraints found in this study, in which the most frequent is energy minimization. Figure 30e shows the type of energy minimized, in which the most frequent is dynamic power minimization. Figure 30f illustrates the type

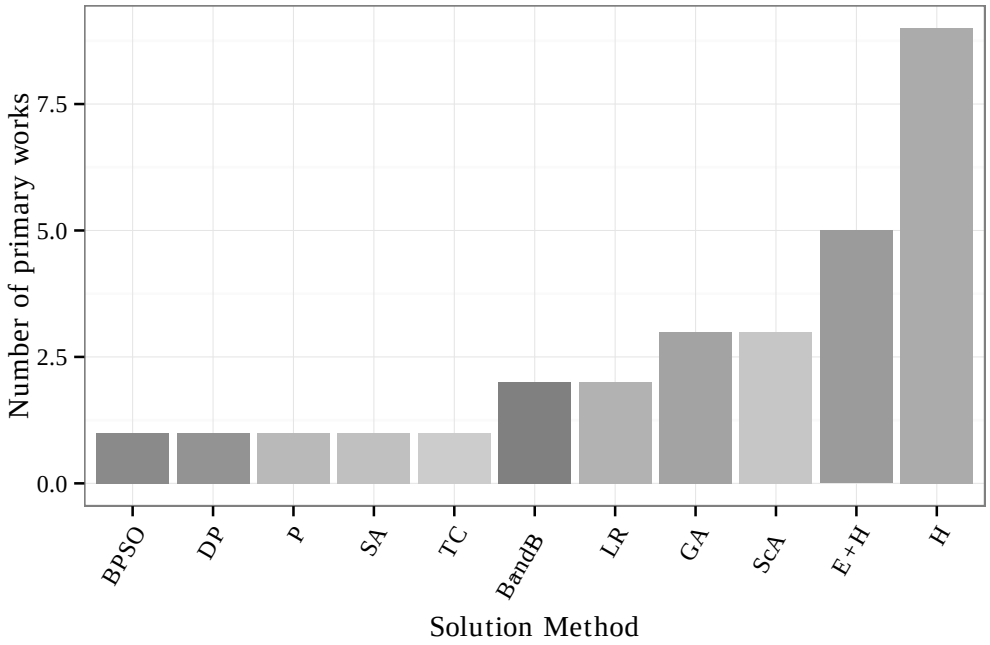


Figure 28 – Distribution of solution methods found in this study: (BPSO) Methaheuristic, Dynamic Programming (DP), Evolutionary Methods (GA), Branch-And-Bound (B-and-B), Exact + Heuristic (E+H), Heuristics(H), Linear Relaxation (LR), Protocol (P), Schedulability Analysis (ScA), Simulated Annealing (SA), Thermal Controller (TC)

of dynamic power model, in which the most frequent is the convex function model.

For the readers' consideration, we present a summary of the usage of techniques in Table 24. Table 24 relates each work with all techniques found in this study to model, solve, analyze, and experiment the problem of interest. Table 24 is a reference of current modeling assumptions and modeling limitations and highlights under which circumstances each primary work can be best applied. Whenever the modeling assumption or technique applies to a primary work, the column is marked with a "Y" symbol. For example, in the first row, Yu and Prasanna (2003) use a discrete set of frequencies for their processor clock, in specialized processing units, to allocate a periodic workload off-line considering low power constraints and deriving approximated solutions. However, Yu and Prasanna (2003) overlook the thermal problem, and their solution is applicable only when the power consumption is a convex and increasing function of frequency, neglecting the static power that is commonly present in modern processors.

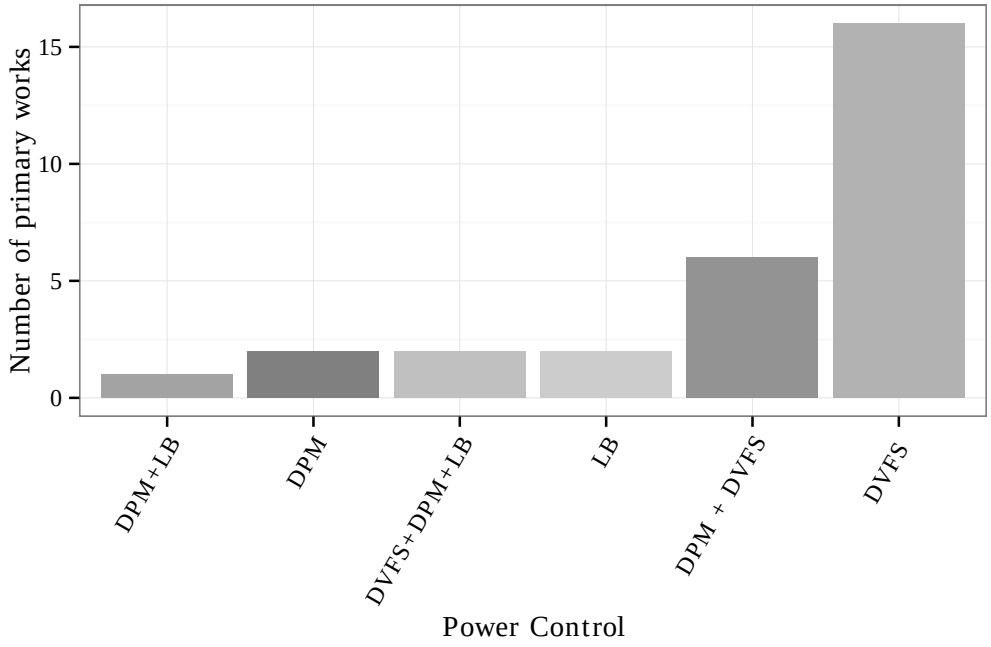
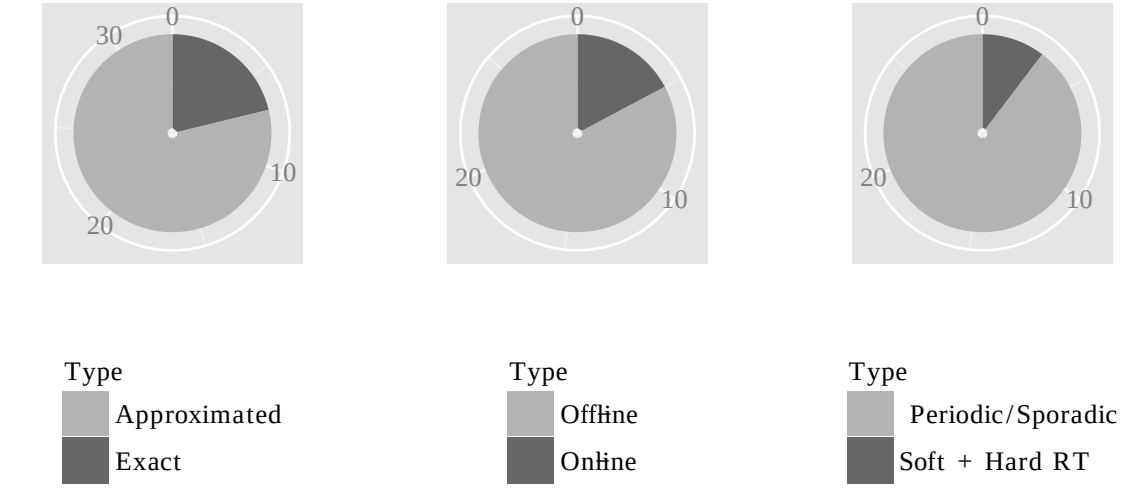
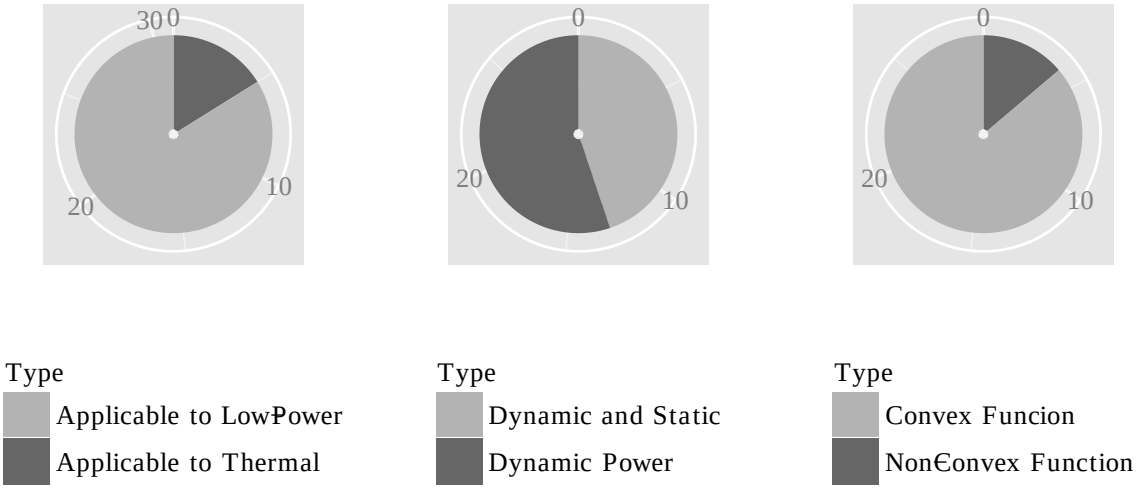


Figure 29 – Distribution of power control techniques found in this study: DVFS, DPM, and Load Balance (LB)



(a) Solution Precision. (b) Allocation Moment. (c) Workload Type.



(d) Energy Constraint. (e) Power Minimized. (f) Power Model.

Figure 30 – Distribution of works with respect to solution precision, allocation moment, workload type, energy constraint, power minimized, and power model. The pie graphs present the number of works (0 - 29) on each category

Table 24 – Summary of technique usage per paper

	Clock		Processor			Soluc.		Alloc.		Workload			Schedulability Test					Energy Management					Problem Model			Power Control			Solution Method								
REF	Continuous	Discrete	Hetero. Clusters	Spec. Architecture	Perf. Variation	Exact	Approximated	Offline	Online	Periodic	Mixed	Migration	Preemption	Precedence	Mutual Exclusion	Response Time	Utilization	Low Power	Thermal	Convex Dyn. Power	Dyn. Power	Dyn. + Static Power	ILP	QPP	NLP	DVFS	DPM	Load Balance	Heuristic	Branch-And-Bound	Protocol	Schedulability Test	Thermal PID	Evolutionary Based	Dyn. Prog.	Linear Relaxation
(Zhang et al., 2015)	Y				Y		Y	Y		Y			Y				Y	Y		Y	Y		Y										Y		
(Yu and Prasanna, 2003)		Y		Y			Y	Y		Y			Y				Y	Y		Y	Y		Y			Y									Y
(Yang et al., 2009)	Y			Y			Y	Y		Y							Y	Y		Y		Y				Y								Y	
(Yang et al., 2012)		Y		Y			Y	Y		Y				Y					Y	Y		Y				Y			Y						
(Terzopoulos and Karatza, 2013)		Y	Y				Y		Y	Y								Y		Y	Y					Y			Y						
(Saha et al., 2012)		Y	Y				Y	Y		Y			Y				Y	Y	Y	Y		Y	Y			Y								Y		
(Prescilla and Selvakumar, 2013)	Y			Y			Y	Y		Y							Y	Y		Y	Y					Y			Y				Y		
(Min-Allah et al., 2012)		Y			Y		Y	Y		Y			Y			Y		Y		Y	Y					Y					Y				
(Goossens et al., 2008)			Y				Y	Y		Y		Y					Y	Y			Y						Y	Y					Y		
(Kim et al., 2008)		Y		Y			Y		Y		Y							Y		Y	Y					Y			Y						
(Kim et al., 2005)		Y		Y			Y		Y		Y							Y		Y	Y					Y			Y						
(Prasanna and Yu, 2002)		Y		Y			Y	Y		Y			Y				Y	Y		Y	Y		Y			Y									Y
(Hung et al., 2006)		Y			Y		Y	Y		Y							Y	Y		Y		Y				Y									
(Hettiarachchi et al., 2013)				Y			Y	Y		Y									Y			Y					Y					Y			
(He and Mueller, 2012b)		Y	Y				Y	Y	Y	Y			Y				Y	Y		Y		Y				Y		Y					Y		
(He and Mueller, 2012a)		Y	Y			Y				Y							Y	Y		Y	Y					Y	Y				Y				
(Chen et al., 2009)	Y			Y			Y	Y		Y							Y	Y				Y	Y			Y			Y						
(Chen and Thiele, 2011)	Y			Y			Y	Y		Y							Y	Y		Y	Y		Y			Y	Y		Y						
(Chen and Thiele, 2009)	Y			Y			Y	Y		Y							Y	Y		Y	Y		Y			Y	Y		Y						
(Chen et al., 2008)	Y			Y		Y		Y	Y	Y			Y	Y	Y		Y	Y		Y	Y				Y	Y	Y				Y				
(Chen and Thiele, 2008)	Y			Y		Y	Y	Y		Y							Y	Y		Y	Y					Y	Y		Y						
(Chen and Kuo, 2007)					Y		Y			Y										Y	Y					Y			Y						
(Chantem et al., 2011)		Y		Y		Y	Y	Y		Y				Y				Y	Y	Y		Y	Y			Y			Y						
(Barrefors et al., 2014)		Y		Y			Y	Y		Y							Y	Y	Y	Y		Y	Y			Y		Y		Y					
(Awan and Petters, 2013)				Y			Y	Y			Y						Y	Y		Y	Y						Y		Y						
(Alahmad and Gopalakrishnan, 2011)		Y		Y		Y	Y	Y		Y							Y	Y				Y	Y			Y	Y		Y						
(Valentin et al., 2015b)		Y	Y			Y		Y		Y			Y	Y	Y	Y		Y		Y		Y				Y						Y			
(Chu et al., 2009)	Y			Y		Y	Y	Y		Y							Y	Y		Y		Y			Y				Y						
(Chen et al., 2011)				Y			Y	Y		Y			Y				Y	Y		Y	Y					Y				Y					

A.4 Answers to the research questions

In this section, we discuss the answers to each of our research questions.

Q1 – *What are the available solutions?*

Table 25 answers this question listing all solutions found (algorithms, heuristics, methods, schedulability tests, techniques, etc). This table also presents a relation between the reference and the proposed techniques and lists each method and technique.

Table 25 – List of all solutions found

Reference	Proposed solution
(Alahmad and Gopalakrishnan, 2011)	ScheduleExact SchedulePruned
(Awan and Petters, 2013)	LLED (Least Loss Energy Density) MM (MaxMin Algorithm) SP (Second Phase of Task Mapping)
(Chantem et al., 2011)	MILP formulation ThermalSched DelayInsertion
(Chen et al., 2013)	Pay Burst Only Once Algorithm (PBOOA)
(Chen et al., 2011)	ACO
(Chen et al., 2009)	S-GREEDY E-GREEDY S-GREEDY-GV E-GREEDY-GV
(Chen and Thiele, 2011)	MAEF (Minimum Average Energy First) R-MAEF (Restricted Minimum Average Energy First)
(Chen and Thiele, 2009)	MAEF (Minimum Average Energy First)
(Chen and Thiele, 2008)	GREEDY Optimal solution by DP FPTAS TRIM
(Luo and Jha, 2002)	List-scheduling Heuristic
(Schmitz et al., 2002)	Genetic list-scheduling
(Kirovski and Potkonjaka, 1997)	Energy-efficient synthesis
(Chen and Kuo, 2006)	Approximations
(Chen et al., 2008)	MFP-PCP
(Goossens et al., 2008)	Random Algorithm (RA) First-Fit Decreasing Density (FFDD)

continued

Table 25 – List of all solutions found

Reference	Proposed solution
	Simulated Annealing (SA)
	Genetic Algorithm (GA)
(He and Mueller, 2012a)	ACPI based heuristic
(Hettiarachchi et al., 2013)	M-TROC
(Hung et al., 2006)	R-GREEDY
	S-GREEDY
	D-GREEDY
	D-E-GREEDY
	D-DP
	D-S-GREEDY
(Kim et al., 2005)	OLB (Opportunistic Load Balancing)
	FG (Fast Greedy)
	Switching
	Min-Min
	Sufferage
	Originator
	Random
(Kim et al., 2008)	OLB (Opportunistic Load Balancing)
	MEG (Minimum Energy Greedy)
	ME-MC (Minimum Energy Minimum Completion Time)
	ME-ME (Minimum Energy Minimum Energy)
	CRME (Contention Resolved Minimum Energy)
	Originator
	Random
(Min-Allah et al., 2012)	Lowest Feasible Speed (LFS)
	Lightest task shifting policy (LTSP)
(Prescilla and Selvakumar, 2013)	Novel BPSO
	Modified BPSO
(Saha et al., 2012)	Hybrid Worst-fit Genetic Algorithm (HyWGA)
(Yang et al., 2009)	MTRIM
(Prasanna and Yu, 2002)	LR-Heuristic
(Yang et al., 2012)	HTDSA
(Yu and Prasanna, 2003)	LR-Heuristic
(He and Mueller, 2012b)	Simulated Annealing (SA)

continued

Table 25 – List of all solutions found

Reference	Proposed solution

Q1.1 *– What are the costs involved in / imposed by the existing solutions?*

Authors avoid reporting the costs of their solutions. Only four works (Chantem et al., 2011; Prescilla and Selvakumar, 2013; Terzopoulos and Karatza, 2013; Valentin et al., 2015b) explicitly report related costs while using their solution. For instance, the Novel BPSO and Modified BPSO require a trained input to obtain good solutions (Prescilla and Selvakumar, 2013). This requires initial engineering effort / time to train the algorithm. Chantem et al. (2011) report only the computational cost required while using a MILP approach and highlight the time required to compute transient thermal simulations. Similarly, additional computational cost is required to use the response time analysis proposed by Valentin et al. (2015b). Terzopoulos and Karatza (2013) also report performance degradation in terms of success rate while using ABDVS.

However, the majority of the works, 25, avoid explicitly reporting costs. Per researchers judgement, however, the most recurring cost found is related to characterization of the system energy consumption. It is required to know before-hand the energy cost of executing each task on each processing unit. The cost representation is either by means of worst-case execution cycles or the worst-case execution time of each task on each processor, for every frequency available. Characterizing such items is a time-consuming procedure, particularly when considering tens, or even hundreds, of processing units.

Q1.2 *– What is the energy consumption reduction provided by each solution?*

Unfortunately, it was not possible to answer this research question with the data reported by the selected works. The main difficulty is the lack of standardized experimentation. The area could achieve better comparative results by means of either benchmarks or standard experiment design, such as using instance databases, clear task generation procedures, and transparent description of simulation strategy and parameters. Another complication is the energy results being typically reported in terms of percent energy reduction. Also, the used references are different among works. The issue with percentage or normalized results is the absence of a reference point.

Q1.3 *– Are there any side effects expected while applying each solution (collaterals) on other software metrics?*

Similar to cost reporting, side effects are missing from existing reports. Authors leave to the readers judgement to find possible side effects.

Q1.4 *– What are the expected industry contexts that each solution applies to?*

The industry contexts reported by the works present in this study are as follows: industrial automation; linear motor control (LMC) systems; CNC machines; X-ray control systems; control systems in the automotive area (ACAD), which consists of an airbag control unit, an anti-lock braking system, and an electronic stability control system; virtual machines scheduled in a cloud computing environment; traffic control (ground or air); aerospace applications; engine control; control of chemical and nuclear power plants; small mobile robots, soccer robots, electronic pet robots, whose tasks are obstacle avoidance, environmental monitoring, localization, multirobot cooperation; *ad hoc* grids in heterogeneous computing environments for wildfire fighting, disaster management, and military situations; mobile computing environments; and distributed embedded systems.

Q1.5 – *Is the solution sufficient?*

As detailed in Section A.3.1.5, most works consider utilization bounds, 20 out of 29. As well-known in the real-time systems literature, the utilization bound is a sufficient schedulability condition. The common use of utilization bound in this subject can be explained by its simplicity. The usage of a utilization bounds simplifies mathematical formulation and the solving process because using such constraints produces models similar to the knapsack problem.

Q1.6 – *Is the solution necessary?*

As detailed in Section A.3.1.5, only Min-Allah et al. (2012) and Valentin et al. (2015b) propose a solution based on response time analysis, and therefore with sufficient and necessary conditions. This result exposes a need for studies on this aspect.

Threats to Validity Systematic reviews may suffer of common threats to validity of the presented findings. We highlight here potential threats on publication bias and identification of studies, as well as the mitigation strategies that we adopted.

Positive findings are more likely to be accepted for publication when compared to negative findings (Kitchenham and Charters, 2007). We see a side effect of the *publication bias* in the answers to the research questions Q1.2 and Q1.3. However, we do consider the bias as limitation to some of the answers and not as a major threat to the study.

Another limiting factor is the research coverage during the process of *identification of studies*. The search strategy utilizes syntax matching with keywords in the search string. Relevant works may not be retrieved, either because they use different nomenclature or because the currently established nomenclature differs from what was in use at the time the work was published. Nevertheless, we carefully built a search string to be as inclusive as possible, adding not only keywords, but also synonyms.

A.5 Open research questions

The result of this systematic review, apart from producing the answers listed in Section A.4, also allows us to highlight open research questions in the subject. Some topics are only briefly addressed in primary works. In this section, we summarize the subjects that require further exploration.

Some topics have only been slightly explored. For instance, only five works address the problem of offering optimal solutions, although most works consider small- or medium-size instances. Most works are also interested in solutions produced off-line. Only five works consider on-line approaches.

The most challenging aspect is exact schedulability tests. We found only one work addressing the analysis of task response times, whereas the majority consider utilization-based analysis, which is a sufficient only condition.

Similarly, only one work explores task migration in this subject. Furthermore, only one work studies mutual exclusion restrictions, and only four works consider precedence constraints in their task models. The most explored workload type is periodic or aperiodic tasks. The least explored workload type is that which combines soft real-time with hard real-time tasks.

DVFS is the most used technique for addressing power consumption issues. Only a minority of the works explore DPM, for instance. Similarly, we emphasize the need for studying the thermal topic on this subject, as we found only five works addressing temperature control. Most works have interest in full heterogeneous architectures. However, heterogeneous clusters are becoming increasingly studied.

Most works investigated for the purposes of this review focus on dynamic power reduction. Additionally, most works consider the dynamic power as a convex power function. We highlight that the static power consumption is non-negligible in current modern processors. Moreover, advanced power management techniques, such as AVS, and the thermal behavior of processors change the increasing convex power consumption function (Alahmad and Gopalakrishnan, 2011). Therefore, we highlight the need to explore different power models, apart from optimizing the dynamic power using a convex power function.

We found only three benchmarks in use on the subject. The lack of standardization in empirical experimentation design and reporting complicates the process of comparing existing solutions. Comparing the results of expected energy consumption reduction is particularly challenging due to the lack of standardization. Consequently, we highlight the need for benchmarks and reference instances. Furthermore, it is crucial to have studies reporting fair comparisons between existing works under controlled environment experimentation. Thus, we also highlight a need for standardized empirical experimentation

to improve the level of scientific evidence reported in this subject. Reporting research limitations, costs, and empirical results is a major concern. Only two works highlight the expected costs for applying their solutions.

A.6 Chapter Summary

We have presented a systematic literature review of scheduling hard real-time tasks on multiple heterogeneous processors under energy constraints, such as low power and temperature control. Using a repeatable, unbiased, and systematic approach, we have identified, presented, classified, and criticized all 29 existing works on the subject. During the process, we analyzed hundreds of works found in relevant digital libraries. We concluded that, even though there has been an increasing interest in the subject over the past years, there are still open questions to address.

The contribution of this chapter is to provide an understanding of the field through a novel systematic review approach. The outcome of the systematic review contributes by summarizing the recent research, evaluating trends, and exposing gaps. The process to identify, extract data, and report the results is transparent and repeatable.

The subject is of interest to different research groups, across diverse industry contexts. We emphasize that the interested industries generally address life-critical situations. Therefore, provisioning correctness is appealing for this research subject. We have also highlighted, by means of this systematic review, different open research questions. We believe that future research shall focus on this lacuna. As a future work, we intend to reassess the systematic literature review to track the evolution of the research in this subject. Another point of our interest as future work is to extend the present research to consider a major focus to be sporadic task models.