Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Doctoral Thesis

Energy-aware scheduling with resource state considerations: Modeling and optimization

Ondřej Benedikt

Fall 2022

Energy-aware scheduling with resource state considerations: Modeling and optimization

by

Ondřej Benedikt

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Doctoral Thesis

Supervisor: doc. Ing. Přemysl Šůcha, Ph.D.
Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics
Submission date: Fall 2022

Ondřej Benedikt: Energy-aware scheduling with resource state considerations: Modeling and optimization, Ph.D. Thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, Fall 2022, Prague.

Declaration

This doctoral thesis is submitted in partial fulfillment of the requirements for the degree of doctor (Ph.D.). The work submitted in this thesis is the result of my own investigation, except where otherwise stated. I declare that I worked out this thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis. Moreover, I declare that it has not already been accepted for any degree and is also not being concurrently submitted for any other degree.

Ondřej Benedikt
Prague, Fall 2022

Abstract

Energy-efficient scheduling has been attracting a considerable amount of attention in recent years. This trend is most likely to continue in the future since energy-efficient scheduling helps to achieve and maintain production sustainability, improve energy efficiency, and reduce energy costs.

This thesis focuses on offline energy-efficient scheduling in environments where the state of the resource changes in time and influences energy consumption. Examples of such resources include industrial furnaces used for steel hardening, manufacturing equipment (e.g., CNC), or even embedded electronic systems on a chip.

We focus especially on the modeling of the resource state and design of energy-aware optimization methods, including the integration of the resource model into the methods. The goal is to show how suitable resource modeling combined with an appropriate optimization method can improve state-of-the-art solutions. To reach that goal, we study three use-case problems representing samples of both theoretical and practical problems involving offline energy-aware scheduling with a complex resource model.

The first problem is motivated by the steel-hardening production line in Škoda Auto. The objective is to analyze and model the hardening furnace and propose an optimization method to minimize energy consumption during the time intervals when the furnace is idle. The second problem involves a general resource that can be characterized by the finite state machine in the environment where the energy prices change in time. Again, the goal is to have an optimization method incorporating the finite state machine model of the resource that would optimize overall energy consumption cost. Finally, the third problem defined by Honeywell company, involves energy-efficient scheduling of safety-critical tasks on a multiprocessor system on a chip under temporal isolation constraints.

Considering our solution approach, we analyze the behavior of the steel-hardening furnace and identify a bilinear system model that can faithfully simulate it. Then, we derive an optimal control of the furnace for any idle interval length. The optimal cost is then abstracted and captured by the so-called idle energy function. Using this abstraction, we propose a scheduling algorithm solving the studied single-machine fixed-sequence scheduling problem in polynomial time complexity. To solve the second problem, we extend the notion of idle energy function to include not only the idle period length but also the energy costs. Then, we propose a mathematical model that efficiently integrates this extended energy function. Finally, to solve the third problem, we analyze the behavior of several real physical platforms mounting modern multiprocessor systems on a chip. Then, we propose several power models and optimization methods and provide a comparative study addressing models fidelity, methods performance, and scalability.

The results across all studied problems show that with careful design choices, the optimization performance can be substantially improved, and the state-of-the-art methods used in the respective fields can be outperformed.

Keywords: scheduling, energy, modeling, optimization

Abstrakt

Energeticky efektivní rozvrhování si v posledních letech získaklo značnou pozornost vědecké komunity. Tento trend bude s největší pravděpodobností pokračovat i v budoucnu, protože energeticky efektivní rozvrhování pomáhá dosáhnout udržitelnosti výroby, zlepšit energetickou účinnost a snížit tak náklady na spotřebu energie.

Tato práce se zaměřuje na offline rozvrhování s důrazem na energetickou efektivitu v prostředích, kde se stav zdroje mění v čase a ovlivňuje spotřebu energie. Příklady takových zdrojů zahrnují například průmyslové pece používané pro kalení oceli, výrobní zařízení (CNC) nebo dokonce vestavěné systémy na čipu.

Velký důraz je v této práci kladen zejména na modelování stavu zdroje a návrh energeticky efektivních optimalizačních metod, které tento model zdroje integrují. Cílem je ukázat, jak může vhodné modelování zdrojů v kombinaci s vhodnou optimalizační metodou překonat doposud používané metody a postupy. Abychom tohoto cíle dosáhli, studujeme tři nezávislé problémy představující průřez teoretických i praktických výzkumných problémů zahrnujících energeticky efektivní offline rozvrhování s netriviálním zdrojem.

Motivace pro první problém vznikla během spolupráce s firmou Škoda Auto. Cílem je analyzovat a modelovat kalící pec, které je součástí linky na kalení oceli, a navrhnout optimalizační metodu pro minimalizaci spotřeby energie v časových intervalech, kdy je pec nečinná. Druhý problém se zaměřuje na obecný zdroj, který lze charakterizovat konečným automatem v prostředí, kde se ceny energie mění v čase. Opět je cílem navrhnout optimalizační metodu zahrnující model zdroje, která by optimalizovala celkové náklady na spotřebu energie. Třetí problém, definovaný ve spolupráci se společností Honeywell, zahrnuje energeticky efektivní rozvrhování bezpečnostně kritických úkolů na víceprocesorovém systému na čipu, kde rozvrhování jednotlivých úloh podléhá omezením časové izolace.

Pro řešení prvního problému analyzujeme chování pece na kalení oceli a identifikujeme bilineární model, který jej dokáže věrně simulovat. Na základě tohoto modelu jsme schopni poté odvodit optimální řízení pece pro jakoukoli délku intervalu její nečinnosti. Optimální spotřeba energie pro libovolně dlouhý interval je pak abstrahována a zachycena tzv. energetickou funkcí. Pomocí této abstrakce jsme schopni navrhnout rozvrhovací algoritmus řešící studovaný problém (rozvrhování na jednom zdroji s fixním pořadím úloh) v polynomiální časové složitosti. Abychom vyřešili druhý problém, rozšíříme definici energetické funkce tak, aby zahrnovala nejen délku intervalu nečinnosti, ale také náklady na energii. Poté můžeme vyvinout matematický model integrující tuto rozšířenou energetickou funkci. Abychom vyřešili třetí problém, analyzujeme chování několika reálných fyzických platforem reprezentující moderní multiprocesorové systémy. Poté navrhneme několik modelů příkonu, které integrujeme do několika optimalizačních metod. Výsledkem je srovnávací studie hodnotící věrnost modelů příkonu, výkon metod a jejich škálovatelnost.

Výsledky napříč všemi studovanými problémy ukazují, že pečlivým návrhem modelu a optimalizační metody lze dosáhnout výsledků podstatně překonávající nejmodernější metody a postupy používané v příslušných oblastech.

Klíčová slova: rozvrhování, energie, modelování, optimalizace

Goals and Objectives

This thesis investigates scheduling problems where the overall energy consumption depends on the state of the resource. The main objectives can be summarized as follows:

1. Study the selected scheduling problems where the state of the resource significantly impacts energy consumption.
2. Review the existing literature considering the resource-state modeling and energy optimization, identify the weak points, and suggest possible improvements.
3. For several selected use-case problems, propose suitable resource models and their integration with the optimization procedures in the context of energy consumption minimization.
4. Present how suitable resource modeling combined with appropriate optimization techniques can improve state-of-the-art solutions.

Contents

Preface 3

Literature Survey 5

Resource Modeling 5

Optimization 7

Contributions 8

Outline 9

1 Scheduling of Industrial Furnace **11**

1.1 Chapter Summary and Motivation 11

1.2 Introduction 12

1.2.1 Related Work 14

1.2.2 Contributions and Outline 15

1.3 Problem Statement 16

1.4 Scheduling Algorithm and Complexity Analysis 17

1.4.1 Definitions 18

1.4.2 Dominance of Block-Form Schedules 18

1.4.3 Finding an Energy-Optimal Block-Form Schedule 20

1.5 Electric Furnaces: Modeling, Optimal Control and Energy Function 24

1.5.1 A Bilinear Model Approximation of Furnaces 24

1.5.2 Solving the Ordinary Differential Equation with a Discontinuous Right-Hand Side 26

1.5.3 Minimum-Energy Control and the Related Idle Energy Function 27

1.6 Case Study: An Industrial Electric Furnace 32

1.6.1 Identification of the Furnace Model 32

1.6.2 Idle Energy Function of the Furnace 33

1.7 Comparison to the State-of-the-Art Approaches 33

1.7.1 Benchmark Instances 36

1.7.2 Transition Graph vs. Idle Energy Functions 37

1.7.3 Time Complexity Comparison 38

1.8 Chapter Conclusions 39

2 Scheduling with Resource States and Variable Energy Prices **41**

2.1 Chapter Summary and Motivation 41

2.2 Introduction 42

2.3 Problem Statement 43

2.4 Solution Approach 44

2.4.1 Instance Pre-processing: Computation of the Optimal Switching 45

2.4.2 Integer Linear Programming Model ILP-SPACES 48

2.4.3 Constraint Programming Models 49

2.4.4 Symmetry Breaking Constraints 52

2.5 Experiments 53

2.5.1 Instances 54

2.5.2 Preliminary CP Experiments: Results for PRELIM 54

2.5.3 Results for MEDIUM and LARGE Instances with Different Machine Transition Graphs 55
2.6 Chapter Conclusions 58

3 Scheduling for Multi-Processor Systems on a Chip 61
3.1 Chapter Summary and Motivation 61
3.2 Introduction 63
3.3 Related Work 65
3.3.1 Benchmarking 65
3.3.2 Optimization 66
3.3.3 Evaluation 67
3.3.4 Summary 67
3.4 Goal and System Model Formalization 68
3.4.1 Goal 68
3.4.2 System Model and Input Parameters 68
3.5 Hardware Platforms and Benchmarks 69
3.5.1 Physical Hardware for Evaluation 70
3.5.2 Benchmarking Kernels 70
3.6 Thermal Modeling 73
3.6.1 Thermal Experiments with Used Platforms 73
3.6.2 Transition from Thermal to Power Model 77
3.6.3 Power Models 78
3.7 Optimization Methods 85
3.7.1 ILP and Sum-Max Model 85
3.7.2 Optimization Based on the Linear Regression Model 86
3.7.3 Greedy Heuristic 90
3.7.4 Optimizer Minimizing/Maximizing the Idle Time 91
3.8 Experimental Evaluation and Results 92
3.8.1 Power Model Evaluation 92
3.8.2 Optimization Methods Comparison 93
3.8.3 Performance evaluation 95
3.8.4 Evaluation Summary 96
3.9 Chapter Conclusion 98

4 Thesis Conclusion 99
4.1 Fulfillment of the Goals 99
4.2 Future Work 101

Bibliography 103

List of Figures 118

List of Tables 119

List of Algorithms 121

A Curriculum Vitae 123

B List of Author's Publications **125**

Preface

Optimization has become an integral part of our everyday lives [67, 112]. Both the individuals and companies optimize, be it profit, welfare, or influence. Since the 1940s, the optimization trend has been on the rise[1]. Especially today, at times of tense geopolitical situation and turbulent energy prices, optimization shows its importance.

According to the Merriam-Webster dictionary [93], optimization is *a process of making something as good as it can be*. More specifically, within the scope of this thesis, we are interested in the specific mathematical procedures, models, and algorithms involved in the optimization process. However, to be able to apply the optimization techniques, we need the formal model of the system itself, which serves as an abstraction of reality, and enables us to perform the optimization without the need to conduct physical experiments.

As an example, assume that we want to optimize the utilization of the resources in a factory. It would be very inconvenient, time-consuming, and expensive to prepare, say, a thousand different scenarios, execute each of them in reality, and then select the one that performed the best. Instead, the system model (including models of the resources, production tasks, and other relevant factors) is prepared first. Then, an appropriate optimization technique is selected, and the whole system is optimized. Finally, the solution is tested in practice and possibly deployed.

In that context, the overall process from the problem specification up to the deployment of the optimized solution can be illustrated, as shown in Figure 1. The primary focus of this thesis is on the selection of the appropriate system model and optimization techniques and on the possible interplay between them. Specifically, we ask questions like:

- Does the model selection influence the optimization step?
- How much can we benefit by selecting the appropriate system model and optimization routine?

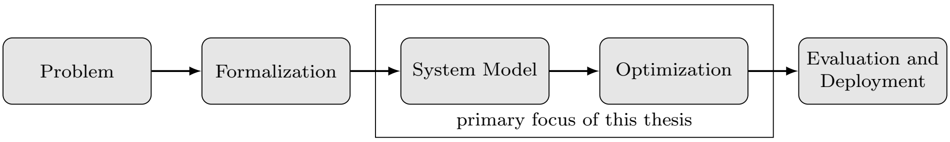


Figure 1: From problem specification towards the optimal solutions and their deployment.

Of course, this thesis does not have the ambition to solve all the optimization problems and provide the best models and optimization routines for each of them.

---

[1] We can simply observe the trend by examining the word ‘optimize’ in a wider context using the English language corpus. According to the Ngram Viewer by Google [94], the frequency of using the word ‘optimize’ increased by more than 15 percent between 2000 and 2019 and is still rising.

Instead, we concentrate on three use-case problems that serve as samples to illustrate the modeling, optimization, and their possible synergy. Each of these use-cases constitutes a single chapter of this thesis.

In the first chapter, we analyze a single-machine scheduling problem, for which we took inspiration from Škoda Auto. There, they have a segment of a production line where the steel parts are being hardened in electric vacuum furnaces. Each such furnace has a relatively high power consumption (up to 160 kW); furthermore, it takes a relatively long time to heat the furnace up to the operating temperature, and its overall dynamics is rather slow. The goal is to schedule the loading and unloading of the parts to the furnace, as well as the behavior of the furnace during the periods when nothing is being processed, such that the overall energy consumption is as low as possible.

In the second chapter, we change the problem a bit by considering any machine that can be modeled by a state transition graph, which is a model commonly used in the literature. Further, we consider time-dependent energy prices (e.g., low and high tariffs or 15-minutes pricing intervals). The goal is to find an optimization algorithm that will scale better than the state-of-the-art methods.

Finally, we study the problem of thermally-constrained safety-critical task allocation and scheduling on a heterogeneous multi-core embedded platform. Although this problem seems completely different from the previous two, it shows that the concepts are the same, be it production scheduling or embedded systems scheduling. Still, proper selection of the system's model and optimization technique has a crucial impact on the solution quality.

Although the problems are rather diverse, several unifying concepts put them to the same scope, namely:

- **Scheduling:** Scheduling is understood as a process of assigning tasks to resources in time. It has large importance across various domains. In production processes, plant and machinery resources are allocated to process the tasks such that some criterion (e.g., the production cost) is optimized. In the computing domain, resources such as processors or network links are assigned to perform tasks. Although scheduling is as old as humanity, it started to gain significance during the industrial revolution in the 18th century. The rapid growth followed in the 20th century mainly due to advances in computer technology, which allowed to solve the scheduling problems previously intractable by humans [64]. Nowadays, scheduling is an active research area with many practical application areas; further advances in modeling and optimization can therefore have a substantial impact in many domains.

- **Energy Optimization:** Typically, the scheduling is done with the aim of optimizing some objective function. In the past, this function was often connected to the timeliness of the system (finish as soon as possible, finish all tasks before their due dates, minimize the number of tardy jobs, etc.). However, with the increasing demand for sustainability, other factors gained their importance. Among those, energy optimization ranks among the most important. One of the reasons is that ever-increasing energy prices make energy one of the non-negligible production costs. Also, the recent geopolitical situation and unstable market have further contributed to the importance of energy management.

Not only is energy optimization important in the production processes, but it also has significant importance in embedded systems, be it to prolong the operating time of battery-powered devices or to avoid the permanent, irreversible failures caused by the system's overheating.

To conclude, although energy optimization became an important part of the decision process, it also brought additional complexity and new challenges to the optimization models and algorithms design. Therefore, it became an important research topic.

- **State of the Resource:** The complexity of the systems is steadily increasing. In manufacturing, we observe a shift from mass production toward mass customization and personalization. This is further complemented by robotization, digitization, and process automation, which are among the recent trends incorporated by the widely adopted Industry 4.0 paradigm. All of these factors contribute to a complexification of the manufacturing process, which leads to a need to develop new and more complex system models. In consequence, accurate modeling of each resource and its state is nowadays an indispensable part of any system model especially considering the system's heterogeneity, complex machine environments, including heat-intensive or reconfigurable machines, and the necessity to model multiple production stages to get the best possible results.

  This situation and trends are not exclusive to just manufacturing scheduling. Take, as an example, the scheduling of workloads in embedded systems. The ever-increasing high-computing demand has led to the development of rather complex heterogeneous multi-processor systems-on-chip, which are being deployed in many applications, including automotive and aerospace. The state of such a system can be associated with its temperature or other quantities, including the operating frequency or the bus/memory load. Having an accurate resource model integrating (some of) these quantities might be essential in order to avoid problems such as interference on the shared resources (e.g., memory, cache), thermal interference of the individual computing elements, etc.

Literature Survey

The detailed problem-specific literature reviews are presented in the respective chapters. Here, we summarize the commonly used resource modeling and optimization techniques.

Resource Modeling

In production, besides investing money to buy newer and more energy-efficient resources, the energy cost can be reduced by using one of two basic means [116, 51]: (i) by relying on the pricing mechanism (the energy consumption remains the same, but the workload is shifted towards time intervals when the energy price is cheaper), or (ii) directly through intelligent planning and scheduling. The latter option relies on the assumption that the resource can scale processing speed (changing the resource state during the task processing) or that the resource can be switched to some power-saving state (changing the resource state during idle time intervals). With this in regard, we analyze what resource models are being used in the literature.

Considering the resource state during idle time intervals, Gahm et al. distinguish between energy demand incurred by machine setups and machine power-saving states (turn-on/off or idling) [51]. Considering the setups, the resource model is often simple. Sometimes, just a constant cost is incurred for each setup [136]. Other times, the setup cost is job/machine-dependent [63] or even sequence-dependent [27, 81]. But even in such cases, the resource is characterized just by a matrix of pre-defined constant numbers representing the setup times/costs. Considering the power-saving states, the situation is very similar. The predominant approach relies on the finite state machine to describe the machine states and the transition between them. Typically, just several states are considered, e.g., ‘processing state’, ‘idle state’, and ‘turn-off state’ [97, 133, 121, 136, 29, 2]. There are some papers, especially related to heat-intensive processes, that adopt more complex machine models [61, 60]. Furnace loading and unloading are then modeled, possibly together with melting, holding, and other processes. These processes are integrated into standard mathematical models by introducing additional variables and constraints. Such machine models are rather problem-specific, and their re-usability in other domains becomes limited. Sometimes, the authors concentrate just on numerical simulation and energy performance analysis without implementing any optimization [62, 98]; such physics-based models are indeed complex; however, they are often unsuitable for further integration within common optimization methods, such as mathematical programming.

With regard to the resource state modeling during processing intervals, the energy consumption can depend on the type of the machine [47, 14, 79] or on the type of the processed task [47, 85, 86]. In both cases, the typical way to handle the resource model is to introduce a set of constants specifying the processing time and energy consumption of each task on each machine (and possibly also in each machine mode, which can be associated with machine processing speed).

In the embedded systems domain, we may find many parallels with the production scheduling domain. Again, energy consumption might be influenced by the type of the resource to which the task is allocated (heterogeneous systems provide CPU clusters of different power characteristics), resource processing mode (CPU frequency), or resource state during idle time intervals (some components has different power modes, such as ‘standby’, or ‘off’).

Considering the resource state during idle time intervals, there exists a research community centered around dynamic power management (DPM), i.e., a system-level power management technique that can selectively shut down idle electrical components to lower energy consumption [22, 77]. Contrary to the production scheduling domain, where the resource states and transitions between them are often encoded directly within the optimization method, DPM researchers often rely on the notion of the energy function [68, 56], which is an abstraction capturing an optimal energy consumption of the resource for any idle period length. Values of this function can be pre-computed based on the component characteristics described by the manufacturer. Often, however, the situation is simpler, as many authors assume two states only (on/off) [13, 7, 45]. Then, the resource can be again characterized by several parameters representing the switching time and cost.

Contrary to that, resource models are more complicated when considering the processing intervals and dynamic voltage and frequency scaling, which is a dominant power-saving approach in the literature [145, 120]. The majority of the models are based on the CMOS circuit power model, which describes power consumption as a function of supply voltage, clock frequency, and other parameters [120]. The so-called *short-circuit* part of the power is neglected, as it is deemed insignificant compared to the *dynamic* and *static* power. Static (leakage) power is associated with the existence of the leakage current; mostly, this part is also disregarded as well. However, as it depends on the temperature, techniques that aim to reduce the on-chip temperature also indirectly lead to a reduction of this leakage power [145]. The remaining part of the power consumption P_d corresponds to dynamic power and is often expressed as $P_d = \alpha V_{dd}^2 C f$, where α is the switching factor, V_{dd} is the supply voltage, C is the load capacitance, and f is the clock frequency [120].

Still, even such a model might be prone to errors. One problem is that the switching factor is dependent on the type of workload that is being executed [11, 20]. This is often disregarded in many works, where the experiments are mostly based on simulation only [32, 28, 6, 108]. Furthermore, it is often assumed that the execution time of each task scales linearly with frequency [6, 45, 108]. However, that is also not correct since the memory latency does not scale with processor frequency [40, 95]. Therefore, some of the authors prefer to learn the power model from data using, e.g., the AI techniques [122, 139, 41].

The resource model becomes even more complicated when the thermal behavior of the chip is considered. In that case, the authors rely on substitute resistance-capacitance (RC) networks that can model the thermal dissipation through the chip [119]. This leads to a system of differential equations relating the temperature in each thermal zone with the power consumption of the individual components. Given the initial condition, the system can be simulated; however, that is often rather computationally demanding. Therefore, for optimization purposes, the authors often simplify the model, e.g., by considering only the steady state [28].

In the context of related literature, this thesis includes problems with rather complex resource models. The heat-intensive industrial furnace is studied in Chapter 1. A general machine with states characterized by the finite state machine in a complex environment with changing energy prices is discussed in Chapter 2, and, finally, a heterogeneous multiprocessor system on a chip is studied in Chapter 3 in the context of the steady-state temperature minimization.

Optimization

Based on our experience, review of related works, and recent surveys [51, 52], we identify one dominant optimization approach used in the production scheduling community. The majority of the authors model the problem using a mixed integer (possibly non-linear) model and solve it with some of the off-the-shelf solvers. Then, if the scalability is not enough, the majority of the authors propose some heuristic algorithm (often metaheuristics, such as genetic algorithms, are used [52]). We attribute the popularity of this trend to its simplicity – it is simple to formulate the problems in mixed integer programming (even if it does not scale well), and it is simple to use metaheuristics (even if they do not necessarily need to perform well). Of course, other approaches exist, including decompositions [42], multi-stage heuristics [29], polynomial-time algorithms [34], etc., but the combination of mixed integer programming and metaheuristics is the most prominent.

The situation is more complex in the embedded community as the designed algorithms are often working online [69, 38, 5]. In that case, the algorithms are often rule-based, building upon EDF or similar rules [12]. As the online methods are out of the scope of this thesis, we further analyze offline ones, which might also be quite diverse [119, 120]. They include greedy methods [74], approximation methods [69], polynomial-time algorithms [7, 13, 56], decoupled methods [141], or even methods based on mathematical programming [32, 28, 45].

Regarding the optimization methods, this thesis has quite a wide coverage. In Chapter 1, a polynomial algorithm is proposed, integrating the information on the optimal control of the resource. In Chapter 2, we propose a mixed integer linear programming formulation that uses data obtained by polynomial pre-processing and therefore provides superior scalability to a generic MILP model. In Chapter 3, we compare many different approaches, including informed and uninformed heuristics, metaheuristics, and mathematical programming models.

Contributions

The key contributions of this thesis are:

- Three use-cases are presented, highlighting the appropriate system modeling and optimization importance.
- The results show that selecting the system model and optimization jointly brings significant benefits, independently of the specific problem to be solved.
- Individually, the state-of-the-art methods are compared with the proposed modeling and optimization approaches showing significant improvements achieved for each of the studied use-cases. Specifically:
  - In Chapter 1, we study optimal furnace control and its integration within the scheduling algorithm. An idle energy function is used to abstract the energy consumption of the furnace during idle periods. Based on this function, a polynomial algorithm is proposed to optimize the overall idle energy consumption. A set of simulations based on data obtained in real production show that the proposed solution provides significantly lower energy consumption compared to the existing modeling approach based on explicit modeling of the machine modes [97, 121, 29, 2].
  - In Chapter 2, we describe a novel pre-processing technique to solve problem $1, \mathrm{TOU} | \mathrm{states} | \mathrm{TEC}$ introduced by Shrouf et al. [121]. The pre-computed costs of the optimal resource state switchings allow us to

design an exact Integer Linear Programming model, which scales several times better compared to the previous state-of-the-art [4].

- In Chapter 3, we implement and compare several power models and optimization methods to solve thermal-aware task allocation under temporal isolation constraints on a heterogeneous multiprocessor system on a chip. Experimental results show that our methods outperform the greedy solution based on the principles commonly used in the relevant literature [142, 74].

Other specific contributions are further listed in each of the chapters.

Outline

This thesis is divided into three separate chapters, where each chapter describes a single use-case.

Chapter 1 studies a single-machine scheduling problem inspired by production in Škoda Auto company, where the scheduling resource is an electric vacuum furnace. The scheduling problem is formally defined (Section 1.3), and an appropriate optimization algorithm is proposed to solve it (Section 1.4). Further, the thermal model of the furnace is described and analyzed (Section 1.5). Then, model parameters are identified based on real data (Section 1.6). Finally, experiments show that the proposed resource model, in combination with the optimization algorithm, outperforms the commonly used approaches (Section 1.7).

Chapter 2 presents a use-case combining machine state modeling with the time-dependent energy prices. The scheduling problem is similar to the one discussed in Chapter 1, but the variable energy pricing makes the situation more complex. Throughout the chapter, we assume a widely studied general resource model describing the resource behavior by a transition graph. First, the problem and the resource model are formally introduced (Section 2.3). Then, a novel pre-processing technique is presented and integrated with the optimization models (Section 2.4). Finally, the proposed approach is validated on a set of benchmark instances (Section 2.5). The results show that significant improvements are achieved when the optimization model is integrated with the proposed pre-processing technique.

Chapter 3 focuses on thermal-aware task allocation on a heterogeneous multiprocessor system on a chip (MPSoC) under temporal isolation constraints. First, the problem is informally introduced (Section 3.2), and the related work is reviewed (Section 3.3). Second, system modeling is formalized in detail (Section 3.4). Then, as we target real modern MPSoC systems, several such systems used for solution validation are presented together with industrial benchmarks representing the workload (Section 3.5). Afterward, thermal modeling is discussed in detail (Section 3.6). Finally, optimization methods integrating described power models are described (Section 3.7). Everything is then experimentally validated (Section 3.8). The results underline the importance of appropriate resource model selection and its integration within the optimization method.

The individual chapters are based on the original scientific publications [17, 16, 20]. For the reader's convenience, each of the chapters starts with a summary introducing its content within the context of this thesis.

Chapter

Scheduling of Industrial Furnace

1

1.1 Chapter Summary and Motivation

This line of research originated from the collaboration with Škoda Auto. As part of his master's theses [43], Josef Dušek went to Škoda Auto and analyzed part of the steel-hardening production line, see Figure 1.1. The results of his analysis showed that there is a significant potential to reduce the energy production of the line, as vacuum hardening furnaces (the major energy consumer of the analyzed production line) were being unnecessarily heated to high temperatures even when processing nothing.

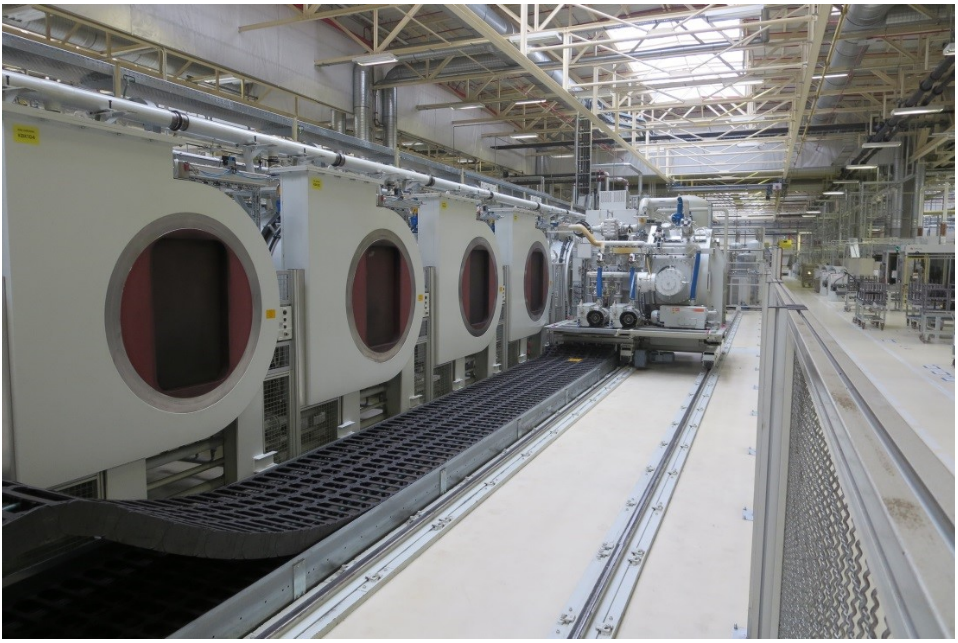


Figure 1.1: A steel hardening production line in Škoda Auto consisting of electrical vacuum furnaces [43].

Building upon this promising analysis, we extracted the data describing the behavior of the furnaces in time[1] and we started thinking about how to improve the energy consumption by means of scheduling. We published some preliminary results at the international conferences:

---

[1] Note that most of the original data were acquired only in the form of time plots (images) exported by diagnostic software in Škodad Auto; therefore, it actually took a great effort to digitize them.

- Ondřej Benedikt, Přemysl Šůcha, István Módos, Marek Vlk, and Zdeněk Hanzálek. “Energy-Aware Production Scheduling with Power-Saving Modes”. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Ed. by Willem-Jan van Hoeve. Cham: Springer International Publishing, 2018, pp. 72–81. ISBN: 978-3-319-93031-2. DOI: `10.1007/978-3-319-93031-2_6`,

- Ondřej Benedikt., Přemysl Šůcha., and Zdeněk Hanzálek. “On Idle Energy Consumption Minimization in Production: Industrial Example and Mathematical Model”. In: *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems - ICORES,*. INSTICC. SciTePress, 2020, pp. 35–46. ISBN: 978-989-758-396-4. DOI: `10.5220/0008877400350046`,

where the latter was awarded the Student Paper Award at the ICORES 2020 conference.

It showed that one of the challenging parts of this research is related to the furnace’s dynamics, as the heating and cooling take substantial time and cannot be neglected. Therefore, we decided to simplify the problem and analyze a single furnace in detail. This led to a major contribution

- Ondřej Benedikt, Baran Alikoç, Přemysl Šůcha, Sergej Čelikovský, and Zdeněk Hanzálek. “A polynomial-time scheduling approach to minimise idle energy consumption: An application to an industrial furnace”. In: *Computers & Operations Research* 128 (2021), p. 105167. ISSN: 0305-0548. DOI: `10.1016/j.cor.2020.105167`

accepted in Computers and Operations Research journal. There, we describe and identify a bilinear model that is used to simulate continuous-time furnace dynamics. Furthermore, we derive an energy-optimal control law to manage the furnace during the idle time intervals. Finally, we propose a scheduling method that incorporates the thermal model in a smart way (in the form of idle energy function) and provides an energy-optimal schedule for a set of tasks that are scheduled in a given fixed order in polynomial time complexity.

This whole study and its results highlight the importance of proper resource modeling (here, the thermal behavior modeled by a bilinear model) and its integration with the optimization method (here, the idle energy function abstraction and the problem reduction to the shortest path problem solvable in polynomial time complexity). The first chapter of this thesis presents this contribution (i.e., [17]).

1.2 Introduction

The machines in heat-intensive processes (such as furnaces) are highly energy-demanding, and therefore their energy consumption optimization usually provides a significant reduction in production costs. In this chapter, we focus on the idle energy consumption optimization, which has been widely studied in recent years, see, e.g., [59, 97, 121, 51, 29, 2]. The research presented in this chapter is inspired by a heat-intensive production process from Škoda Auto. There, steel hardening is performed in electric vacuum furnaces, which require high power input to reach and maintain the specific operating temperature. In this production line, all furnaces are heated to the operating temperature at the beginning of the week and turned off at its end. However, this strategy is very wasteful because a considerable amount of energy is consumed for heating even during periods when no material is being processed. The problem of energy-wasting during prolonged idle periods is not specific only to this particular plant. Similar observations have already been made in other companies as well [97].

A common approach in the area of the idle energy consumption optimization is to define a set of machine modes, typically "off", "on", and "stand-by" [97, 121, 29, 2]. The feasible transitions between the modes are then represented by a static *transition graph* defining the time and energy needed to switch from one mode to another and thus describing the machine dynamics to some extent. In this chapter, we argue that this type of model might be too restrictive for some types of machines (e.g., the furnaces). For such machines, we propose to employ the complete time-domain behavior of the machine, when available, in contrast to the use of the finite number of stand-by modes as in the existing literature. The relation between the length of the idle period and the possible minimal energy consumption is then represented by the *idle energy function*, which is used by the proposed scheduling algorithm. This way, the whole energy minimization problem is decomposed into two independent optimization problems: (i) determination of the idle energy function and (ii) optimal scheduling of the tasks.

For the scheduling part, we examine a single machine problem where tasks are characterized by release times, processing times, and deadlines while the objective is to minimize the idle energy consumption. Besides, we assume a fixed order of tasks. The reason for this assumption is that the single machine problem with release times and deadlines is already $\mathcal{NP}$-hard [53]. Therefore, it is reasonable to solve the entire production problem by a heuristic. In this case, a decision concerning the order of tasks and their assignment to machines is often determined by a local-search or meta-heuristic. These techniques can employ the scheduling approach proposed in this chapter for finding the optimal start times of the tasks given their order. We prove that whenever the idle energy function is concave, the scheduling problem can be solved in polynomial time by reduction to the shortest path problem. The main advantage of this transformation is that the size of the reduced problem is independent of the length of the scheduling horizon.

The determination of the idle energy function is specific to the considered machine. In this chapter, we take as an example electric furnaces widely used in industrial production lines such as steel hardening and glass tempering, operating at a specified temperature. Using the Pontryagin's minimum principle (PMP) to analyze a realistic bilinear model of the continuous-time furnace dynamics, we prove that the energy-optimal control law during any idle period is to switch from zero input power (cooling) to the maximum applicable input power (maximal heating) at some convenient switching time. This optimal control law is then shown to result in the concavity of the idle energy function, which enables to employ the proposed optimal scheduling algorithm. The theoretical approach and findings are validated through a case study investigating an industrial furnace in a real production line.

1.2.1 Related Work

Concerning the research of energy-efficient manufacturing systems, one of the first analyzes in this area was performed by Mouzon et al. [97], who observed that a significant amount of energy could be saved by managing the state of the machine. They proposed several dispatching rules for online production, considering *operating* and *idle* states of the machine. Specifically, rules were devised to turn the non-bottleneck machines off when they were idle for a certain amount of time. Experimental results showed that, compared to the worst-case policy (no switching), substantial energy savings could be achieved. This research laid the foundations for further works investigating the minimization of (idle) energy in production. Often, following the example of Mouzon et al., authors consider only a simple case with two states, the *processing* (operational) state and *off* state. That is also the case in the work of Che et al. [30], who proposed a mixed-integer linear programming (ILP) model and heuristics for bi-objective minimization of the energy and maximum tardiness. Another example can be found in the work of Zhou et al [143], who proposed a mathematical model and a differential evolution algorithm for a parallel batch processing machine scheduling problem considering minimization of the makespan and total energy consumption. Two states of the batch processing machine were assumed for the modeling, namely the *processing* and *idle* state. Angel et al. [7] analyzed a single machine problem with tasks characterized by release times and agreeable deadlines and showed that the problem of idle energy minimization can be solved in polynomial time when only on-off switching is considered. Machines characterized by three states (*processing*, *idle*, and *shutdown*) were studied by both Shrouf et al. [121] and Aghelinejad et al. [4], who addressed energy minimization under variable energy prices. A common aspect of all previously mentioned works is that the dynamics of the machine is simplified to several constants (representing the transition times/costs between pairs of modes) only. Contrary to that, we show that by using a more precise model of the machine dynamics, higher energy savings can be achieved. Our claim is supported by a case study examining a heat-intensive system employing a steel-hardening furnace.

Regarding scheduling for heat-intensive production systems and industrial furnaces, the literature is still very sparse. Some authors have studied re-heating furnaces [137, 125], which are used to heat steel slabs to a specified temperature before they enter the next production stage. Typically, the duration which the slabs spend inside the furnace (i.e., the processing time), and the sequence of the slabs are optimized. Haït et al. [61] studied the problem where the metal is melted in several induction furnaces. The melting time can be shortened by increasing the input power. In contrast, the processing time, as well as the temperature, are specified in our case to ensure the desired quality of the product. Liu et al. [87] addressed a glass production flow-shop problem, modeling multiple stages, and optimizing the makespan and total energy consumption. However, only the *processing* and *idle* states were considered to approximate the furnace dynamics in the scheduling model.

In addition to the manufacturing processes mentioned previously, the research on power-saving states has a broad base in the domain of embedded systems, where energy savings are crucial to prolonging the battery life [68, 13, 56]. The considered devices typically have only a small number of power-saving states [56], which are specified by the manufacturer. Sometimes authors assume only the *processing state* and the *off-state* [68, 13]. The studied problems commonly lead to online scheduling algorithms because of their real-time character or uncertainties in the arrival times of the tasks. In contrast to embedded systems, the dynamics of machines in production lines, e.g., for the heat-intensive systems investigated in our case study, is typically much slower. Thus, by assuming only *on* and *off* states for such machines, the idle periods between two consecutive tasks would need to be very long to make the transitions possible. Another difference is the possibility of solving the production problems offline with respect to known, or *a priori* approximated, parameters of the tasks and the identifiable dynamic behaviors. However, despite all differences, some concepts originating from the domain of embedded systems are general and can still be used even for production scheduling. Frequently, the idle energy consumption is captured by an *idle energy function*, $E : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, mapping the length of the idle period to energy consumption [56]. Such a function E is typically assumed to be non-decreasing piecewise-linear concave where each linear segment corresponds to a single power-saving state. Adopting this concept, we mainly propose a new polynomial-time scheduling algorithm, also suitable for production line machines whose dynamics can be captured by a concave idle energy function.

1.2.2 Contributions and Outline

The main contribution of this chapter is twofold. First, we propose a new polynomial scheduling algorithm using the concept of the idle energy function. Second, we show that the idle energy function can be used to better represent the dynamics of the machine compared to the approaches that are just approximating it with few states only. As the experimental results show, we can achieve much better energy savings. Further, we list the particular contributions of our article in the context of the present related works:

1. We define the problem of idle energy consumption minimization for a single machine scheduling with release times, deadlines, and the fixed order of tasks where the consumption of the machine is defined by the idle energy function (Section 1.3).

2. We suggest decomposing the studied problem to (i) the determination of the idle energy function with respect to the machine dynamics, and (ii) the optimal scheduling of tasks.

3. We show that the scheduling problem can be solved in $\mathcal{O}(n^3)$, where n is the number of tasks, assuming that the idle energy function is concave (Section 1.4). To the best of our knowledge, the closest work that can be adapted to our problem is the algorithm for a fixed sequence of tasks proposed by Aghelinejad et al. [3]. The complexity of their algorithm is $\mathcal{O}(|H|^2 n)$, where $|H|$ is the length of the scheduling horizon. Since for practical applications $|H| \gg n$, our approach exhibits a better complexity (Section 1.7.3).

4. Utilizing a bilinear system approximation of furnace dynamics, we propose an energy-optimal control law for fixed idle period lengths and show that the idle energy function under this control law is concave (Section 1.5).

5. Combining the scheduling approach and the idle energy function derived for a real industrial furnace at Škoda Auto (in Section 1.6), we verify the proposed approach on a set of instances and show (in Section 1.7.2) that the proposed solution provides significantly less energy consumption as compared with the existing modeling approach based on explicit modeling of the machine modes [97, 121, 29, 2].

The rest of the article is organized as follows. Section 1.3 provides the problem description and assumptions. In Section 1.4, the dominant structures in schedules are identified, and it is shown that the scheduling problem can be solved in polynomial time by finding the shortest path in a directed acyclic energy graph. Section 1.5 addresses the modeling of the furnace; a bilinear model is described, and the energy-optimal control law is derived. The case study in Section 1.6 describes a real furnace used in the production; bilinear model parameters are identified, and the idle energy function is derived. The case study is followed by Section 1.7, which shows the results of the numerical scheduling experiments using the identified model of the real furnace in contrast to the state-of-the-art modeling techniques assuming a finite number of machine modes. Finally, Section 1.8 concludes the article.

1.3 Problem Statement

We study a scheduling problem denoted $1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$, i.e., the minimization of the idle energy consumption on a single machine where the order of the tasks is fixed. Formally, let $T = \{1, 2, \dots, n\}$ denote the set of tasks sorted according to the given order. Each task $i \in T$ is characterized by three integers: release time $r_i \in \mathbb{Z}_{\geq 0}$, deadline $\tilde{d}_i \in \mathbb{Z}_{>0}$, and processing time $p_i \in \mathbb{Z}_{>0}$, such that $r_i + p_i \leq \tilde{d}_i \; \forall i \in T$.

A schedule is defined by vector of start times $\boldsymbol{s} = (s_1, s_2, \dots, s_n) \in \mathbb{R}^n_{\geq 0}$. A *feasible schedule* is such a schedule that satisfies the following constraints.

(C1) Each task i is processed within its execution time window $[r_i, \tilde{d}_i]$.

(C2) The processing order of the tasks is given and fixed.

(C3) At most, a single task is processed at one time.

(C4) The processing is done without preemption.

For the rest of this work, when we talk about a schedule, we always mean a feasible schedule.

We assume that the machine is turned *on* (e.g., heated to the operating temperature from *off* state in case of a furnace) just before the first task is processed, and it is turned *off* immediately after the last task is processed. When the machine is *off*, the power consumption is zero. Costs for turning the machine on and shutting it off are constant and cannot be optimized.

When a task is processed, the machine operates in the processing state given by the respective technological process (e.g., the furnace is heated to the operating temperature, which is the same for all tasks). Therefore, energy consumption cannot be optimized in this case, as well. However, during the idle periods, the machine can change its state to lower the energy consumption (i.e., the temperature of the furnace can be lowered to save energy). At the end of the idle period, the machine needs to be switched back to the processing state before the next task is processed.

The objective is to find start times $\boldsymbol{s}$, such that the idle energy consumption $E_{\text{total}}(\boldsymbol{s})$, i.e., the total energy consumption during idle periods, is minimized. An *idle period* is defined as the duration between the completion time of a task and start time of the following one. Since the execution order of the tasks is fixed, we can assume that the tasks are sorted in the given order, i.e., $s_i + p_i \leq s_{i+1} \, \forall i \in \{1, 2, \ldots, n-1\}$. Then, the objective can be written as

$$\min_{\boldsymbol{s}} E_{\text{total}}(\boldsymbol{s}) = \min_{\boldsymbol{s}} \sum_{i=1}^{n-1} E\left(s_{i+1} - (s_i + p_i)\right), \tag{1.1}$$

where $E : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ represents the *idle energy function*, which encodes the relationship between the idle period length and the consumed energy (taking into account various power-savings). The idle energy function is further discussed in Section 1.5.3, and a real example for an industrial furnace is shown in Figure 1.8 in Section 1.6.

Note that because of the fixed order, release times and deadlines can be propagated. Specifically, taking tasks from left to right, release times can be shifted such that

$$r_i := \max\{r_{i-1} + p_{i-1}, r_i\}, \ \forall i \in \{2, 3, \ldots, n\}, \tag{1.2}$$

and taking the tasks from right to left, deadlines can be adjusted such that

$$\tilde{d}_i := \min\{\tilde{d}_{i+1} - p_{i+1}, \tilde{d}_i\}, \ \forall i \in \{n-1, n-2, \ldots, 1\}. \tag{1.3}$$

If there exists a task such that its propagated execution window is shorter than its processing time, then the instance does not have a feasible solution for the given order. For the rest of this article, we assume that release times and deadlines are propagated and a feasible solution exists.

1.4 Scheduling Algorithm and Complexity Analysis

In this section, we show that $1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$ can be solved in polynomial time under the assumption that the energy function E is concave. Note that if the order was not fixed, the problem would be $\mathcal{NP}$-hard because its underlying problem $1|r_j, \tilde{d}_j|-$ is $\mathcal{NP}$-complete in a strong sense [53].

A special version of the problem studied here was addressed by Gerards and Kuper [56], who assumed a so-called *frame-based system*, i.e., a system where $r_i = (i-1) \cdot T$ and $\tilde{d}_i = i \cdot T$ for some constant number T. In frame-based systems, execution windows of the tasks do not overlap. Gerards and Kuper showed that idle energy minimization in frame-based systems can be done in polynomial time, assuming that the idle energy function is concave. We extend their result to

$1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$, i.e., to systems with arbitrary release times and deadlines, assuming that the execution order of the tasks is fixed.

Further, we describe the structure of the *energy graph*, and show that $1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$ can be solved by finding the shortest path in that graph. But first, we provide necessary definitions and show that only schedules in a special form (so-called *block-form schedules*) can be assumed for the optimization.

1.4.1 Definitions

A basic structure that appears in the feasible schedules is called a *block of tasks* or simply *block*, and is widely used [10, 13].

Definition 1.4.1 (Block of tasks)**.** A sequence of tasks $B = (b_1, \ldots, b_m)$, which are scheduled on the same machine, is called a block of tasks if the following properties hold:

$$s_{b_i} + p_{b_i} = s_{b_{i+1}}, \ \forall i \in \{1, 2, \ldots, m-1\}, \tag{1.4}$$

$$\forall i \in T \setminus B : (s_i + p_i < s_{b_1}) \vee (s_i > s_{b_m} + p_{b_m}). \tag{1.5}$$

Property (1.5) states that block B is maximal, i.e., it cannot be extended to the left or right. Every feasible schedule is composed of blocks of tasks, which are separated by idle intervals. Blocks are, therefore, fundamental building elements out of which the resulting schedule is created.

Even though all schedules are composed of blocks of tasks, some schedules are special in a certain sense. We call them *block-form schedules.*

Definition 1.4.2 (Block-form schedule)**.** A schedule consisting of k blocks B_1, B_2, $\ldots$, B_k is in the *block form* if each block of tasks B_j contains at least one task, which starts at its (propagated) release time or ends at its (propagated) deadline; such a task is called the *support* of block B_j.

Thanks to the properties of the block-form schedules, the idle energy optimization can be made simple, as shown in Section 1.4.2 and Section 1.4.3.

1.4.2 Dominance of Block-Form Schedules

In this section, we show that block-form schedules weakly dominate all other schedules. To prove this, we utilize the following lemma.

Lemma 1.4.3. *Given a concave idle energy function* $E : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$*, for* $0 \leq \epsilon \leq x \leq y$ *it holds that*

$$E(x - \epsilon) + E(y + \epsilon) \leq E(x) + E(y)\,. \tag{1.6}$$

Proof. Property (1.6) is directly implied by the concavity of E, see Gerards and Kuper [56]. $\square$

Lemma 1.4.3 implies that, in the case of having two idle periods x and y, energy $E(x) + E(y)$ decreases or remains the same even if the shorter idle period of length x is reduced on behalf of the longer idle period of length y. Then, we have the following theorem.

Theorem 1.4.4. *Given a concave idle energy function E, for every feasible schedule S_1 defined by start times $\boldsymbol{s}_1$, there exists a feasible schedule S_2 defined by start times $\boldsymbol{s}_2$, such that S_2 is in a block form and $E_{total}(\boldsymbol{s}_1) \geq E_{total}(\boldsymbol{s}_2)$.*

Proof. If S_1 is already in a block form, nothing has to be done. Otherwise, S_1 consists of k blocks

$$\{B_1, B_2, \ldots, B_k\} = \mathcal{B}_{\text{fixed}} \cup \mathcal{B}_{\text{free}},\ \mathcal{B}_{\text{fixed}} \cap \mathcal{B}_{\text{free}} = \emptyset,$$

where $\mathcal{B}_{\text{fixed}}$ is the set of blocks that contain at least one support, and $\mathcal{B}_{\text{free}}$ are the blocks without supports. The blocks in $\mathcal{B}_{\text{fixed}}$ will not be moved, while the blocks in $\mathcal{B}_{\text{free}}$ will be shifted to gain a support. By *shift*, we mean adding a non-zero constant to all start times of the tasks in the block.

Let us assume that there is an infinitely long idle period before the first block in S_1 and after the last one. Now, every block is separated from the other blocks by two idle periods (before and after the block).

Let us take an arbitrary block $B \in \mathcal{B}_{\text{free}}$. Since it does not contain a support, it can be shifted. The direction of the shift can be selected according to Lemma 1.4.3 such that the idle energy consumption does not increase (i.e., shift the block such that the shorter neighbouring idle period decreases its length). Note that the leftmost (rightmost) block is always shifted right (left) to prolong the time when the machine is off (idle energy consumption does not increase).

After the block is shifted as much as possible, there are two possible outcomes.

1. Some task $i \in B$ reaches its release time or deadline.

   In this case, block B gains a support and joins $\mathcal{B}_{\text{fixed}}$; the cardinality of $\mathcal{B}_{\text{free}}$ decreases by one.

2. Block B reaches its neighbouring block B_{neigh}.

   In this case, block B joins its neighbouring block. If $B_{\text{neigh}} \in \mathcal{B}_{\text{fixed}}$, then B gains a support and joins $\mathcal{B}_{\text{fixed}}$. Otherwise, $\mathcal{B}_{\text{free}} := (\mathcal{B}_{\text{free}} \setminus \{B, B_{\text{neigh}}\}) \cup \{B \oplus B_{\text{neigh}}\}$, i.e., B and B_{neigh} are joined (operator $\oplus$). Anyway, the cardinality of $\mathcal{B}_{\text{free}}$ decreases by one.

If cases 1. and 2. happen at the same time, both B and B_{neigh} gain a support, join $\mathcal{B}_{\text{fixed}}$, and the cardinality of $\mathcal{B}_{\text{free}}$ decreases by at least one.

It can be seen that after one shift, the cardinality of $\mathcal{B}_{\text{free}}$ decreases, and the idle energy consumption does not increase (by Lemma 1.4.3). By iteratively shifting the blocks without supports, every block will eventually join $\mathcal{B}_{\text{fixed}}$. Since there are at most n blocks in $\mathcal{B}_{\text{free}}$ at the beginning, and the cardinality of $\mathcal{B}_{\text{free}}$ decreases after each shift, $\mathcal{B}_{\text{free}}$ will be empty after at most n iterations. Also, there are at most n tasks in each block. Therefore, each shift can be done in $\mathcal{O}(n)$ steps (shifting one task after another). Hence, the transformation can be done in $\mathcal{O}(n^2)$ steps. Schedule S_2 is then given by the start times of the tasks in $\mathcal{B}_{\text{fixed}}$. □

Theorem 1.4.4 shows that it is sufficient to optimize only over schedules in the block form.

1.4.3 Finding an Energy-Optimal Block-Form Schedule

Here we show how the schedules can be represented as paths in an oriented directed acyclic *energy graph*. The graph-based approach was originally introduced for frame-based systems by and Kuper [56], but since the release times and deadlines in their frame-based systems do not overlap, the graph had a very simple structure. In our case, we need to non-trivially extend the idea, relying on Theorem 1.4.4.

By Definition 1.4.2, each block of a block-form schedule contains at least one support. The main idea leading to a graph-based approach is to represent the supports of the schedule by nodes of the energy graph. In the following, we will show that paths in the energy graph can be associated with the block-form schedules and that the shortest path corresponds to the optimal block-form schedule.

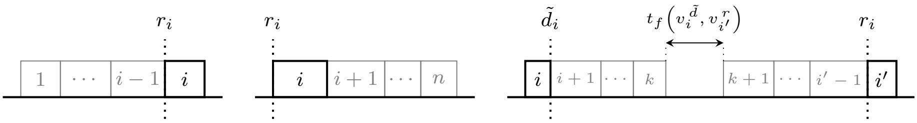
Our extended version of the energy graph can be represented as a triplet $G = (V_G, E_G, c)$, where V_G is set of its vertices, E_G is set of its oriented edges, and $c : E_G \to \mathbb{R}_{\geq 0}$ is the cost function. For each task $i \in T$, we define vertices v_i^r and $v_i^{\tilde{d}}$ representing situations when task i starts at its release time and ends at its deadline, respectively. Let $\mathrm{start}(v_i^x)$ be the actual start time of the task i represented by vertex v_i^x, i.e.,

$$\mathrm{start}(v_i^x) = \begin{cases} r_i, & \text{if } x \text{ is } r, \\ \tilde{d}_i - p_i, & \text{if } x \text{ is } \tilde{d}. \end{cases} \tag{1.7}$$

Furthermore, let us define two additional dummy vertices, the starting vertex v^s and the ending vertex v^e. We will define the edges in such a way that the paths between v^s and v^e represent block-form schedules. The set of edges E_G consists of three types of edges, $E_G = E_G^{(1)} \cup E_G^{(2)} \cup E_G^{(3)}$, where

$$E_G^{(1)} = \Big\{ (v^s, v_i^x) \;\Big|\; i \in T, x \in \{r, \tilde{d}\} \text{ such that the partial schedule given by}$$
$$s_i := \mathrm{start}(v_i^x),\; s_{i'} := s_i - \sum_{k=i'}^{i-1} p_k \;\forall i' \in \{1, 2, \ldots, i-1\} \text{ is feasible} \Big\}, \tag{1.8}$$

$$E_G^{(2)} = \Big\{ (v_i^x, v^e) \;\Big|\; i \in T, x \in \{r, \tilde{d}\} \text{ such that the partial schedule given by}$$
$$s_i := \mathrm{start}(v_i^x),\; s_{i'} := s_i + \sum_{k=i}^{i'-1} p_k \;\forall i' \in \{i+1, i+2, \ldots, n\} \text{ is feasible} \Big\}, \tag{1.9}$$



(a) (b) (c)

Figure 1.2: Examples of the partial schedules corresponding to the edges between (a) (v^s, v_i^r), (b) (v_i^r, v^e), and (c) $\left(v_i^{\tilde{d}}, v_{i'}^{r}\right)$.

$$\begin{aligned} E_G^{(3)} = \Big\{ & (v_i^x, v_{i'}^y) \;\big|\; i \in T, i' \in T, i < i',\; x, y \in \{r, \tilde{d}\} \text{ and} \\ & \exists k \in \{i, i+1, \dots, i'-1\} \text{ such that the partial schedule given by} \\ & s_i := \text{start}(v_i^x),\; s_{i'} := \text{start}(v_{i'}^y), \\ & s_a := s_i + \sum_{l=i}^{a-1} p_l \;\forall a \in \{i+1, i+2, \dots, k\}, \\ & s_b := s_{i'} - \sum_{l=b}^{i'-1} p_l \;\forall b \in \{k+1, \dots, i'-1\} \text{ is feasible} \Big\}. \end{aligned} \tag{1.10}$$

In $E_G^{(1)}$, edges connect the starting vertex v^s and vertex v_i^x, $x \in \{r, \tilde{d}\}$, $i \in T$, associated with task i. Each edge represents the situation when task i is the support and tasks $\{1, 2, \dots, i-1\}$ are aligned to the right, joining the block supported by task i, see Figure 1.2(a). Similarly, edges in $E_G^{(2)}$ link v_i^x, $x \in \{r, \tilde{d}\}$, $i \in T$, with the ending vertex v^e. Each edge represents the situations when task i is the support, and tasks $\{i+1, i+2, \dots, n\}$ are aligned to the left, joining the block supported by i, see Figure 1.2(b). Finally, set $E_G^{(3)}$ represents situations when there are two blocks of tasks supported by i and i', respectively. All the tasks $\{i+1, i+2, \dots, k\}$ are aligned to the left and join the block supported by i and tasks $\{k+1, k+2, \dots, i'-1\}$ are aligned to the right and join the block supported by task i', see Figure 1.2(c).

Now, we define the cost function c. We set the costs of edges in $E_G^{(1)}$ and $E_G^{(2)}$ to zero because the tasks represented by these edges are processed without any idle periods. The costs of edges in $E_G^{(3)}$ correspond to the idle energy consumption between two blocks of tasks. Even though there might be multiple possible ways to schedule the tasks between the two supports, the processing time of each task is assumed to be constant and so the length of the idle period is invariant for a fixed pair of supports. Let us denote the length of the idle period between blocks supported by v_i^x and $v_{i'}^y$, where $i' > i$, by $t_f(v_i^x, v_{i'}^y)$, defined by

$$t_f(v_i^x, v_{i'}^y) = \text{start}(v_{i'}^y) - (\text{start}(v_i^x) + p_i) - \sum_{k=i+1}^{i'-1} p_k. \tag{1.11}$$

Now, the cost function can be defined in the following way:

$$c(e) = \begin{cases} 0, \text{ if } e \in E_G^{(1)} \cup E_G^{(2)}, \\ E\left(t_f(v_i^x, v_{i'}^y)\right), \text{ if } e = (v_i^x, v_{i'}^y) \in E_G^{(3)}. \end{cases} \tag{1.12}$$

Explanatory example. To illustrate the energy graph, let us consider an arbitrary concave idle energy function E and four tasks characterized by parameters given in Table 1.1. The corresponding energy graph is shown in Figure 1.3. Each edge e is labeled by its cost $c(e)$, defined by (1.12).

Note that there is no edge between v^s and v_3^r because if task 3 started at its release time, it would not be possible to execute the previous tasks without introducing an idle period ($\tilde{d}_2 = 40 < 45 = r_3$). But in that case, the previous tasks would form a different block, having its own support. Therefore, edge (v^s, v_3^r) does not bring any additional useful information. The situation is similar for other 'missing' edges.

Table 1.1: Example task parameters.

i	1	2	3	4
r_i	0	15	45	80
$\tilde{d}_i$	20	40	70	100
p_i	10	15	5	10

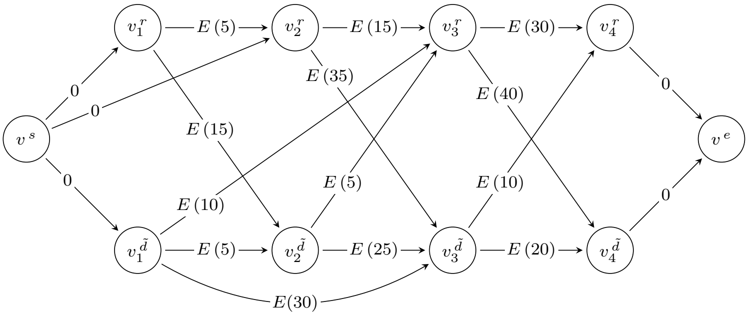


Figure 1.3: Energy graph constructed for the tasks specified by Table 1.1.

The connection between the paths in the energy graph and block-form schedules is explained by the following two lemmas.

Lemma 1.4.5. *For every block-form schedule S, there exists a path in the corresponding energy graph, such that length of the path equals the idle energy consumption of schedule S.*

Proof. This is assured by the structure of the energy graph. Given a block-form schedule with blocks $B_1, B_2, \ldots, B_k$ and their supports $a_1, a_2, \ldots, a_k$, the corresponding path in the energy graph is $v^s, v_{a_1}^{x(a_1)}, v_{a_2}^{x(a_2)}, \ldots, v_{a_k}^{x(a_k)}, v^e$, where

$$x(a_i) := \begin{cases} r & \text{if } a_i \text{ starts at its release time,} \\ \tilde{d} & \text{if } a_i \text{ ends at its deadline.} \end{cases} \tag{1.13}$$

Nodes on the path correspond to the supports of the individual blocks, and because the cost of each edge directly corresponds to the idle energy consumption, the length of the path is the same as the idle energy consumption of the schedule. ☐

Lemma 1.4.6. *For every path P between the start node v^s and end node v^e in the energy graph, there exists a feasible block-form schedule S, such that the idle energy consumption cost of S is the same as the length of path P.*

Proof. Again, this is trivially given by the structure of the energy graph, where nodes represent supports of the blocks. According to (1.8)–(1.10), an edge between two nodes representing the supports is added only if there exists a feasible schedule of the tasks between them. ☐

Finally, by Lemmas 1.4.5 and 1.4.6, we see that problem $1 \mid r_j, \tilde{d}_j, \text{fixed order} \mid \Sigma E$ can be solved by finding the shortest path in a directed acyclic graph. The graph contains $\mathcal{O}(n)$ vertices and at most $\mathcal{O}(n^2)$ edges. Whether edge e belongs to the graph or not can be verified according to (1.8)–(1.10) in linear time $\mathcal{O}(n)$. Therefore, the number of steps needed to build the graph is upper bounded by $\mathcal{O}(n^3)$. The shortest path itself can be found in linear time with respect to the size of the graph by the dynamic programming, see Sec. 24.2 in [37]. So the overall complexity is bounded by $\mathcal{O}(n^3)$.

Explanatory example (continued). The schedule corresponding to path $v^s, v_1^{\tilde{d}}, v_3^r, v_4^r, v^e$ is depicted in Figure 1.4. It consists of three blocks, $B_1 = (1, 2)$, $B_2 = (3)$, and $B_3 = (4)$. Supports of these blocks are tasks 1, 3 and 4, respectively. Idle energy consumption of the schedule equals the sum of energy consumed during the first idle period (from time 35 to time 45), plus energy consumed during the second idle period (from time 50 to time 80).

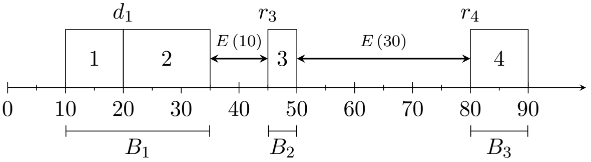


Figure 1.4: Feasible schedule corresponding to path $v^s, v_1^{\tilde{d}}, v_3^r, v_4^r, v^e$.

Remark 1. Note that edges in $E_G^{(3)}$ might not imply one particular schedule of the tasks between the supports. Therefore, for a given path, there might exist multiple feasible schedules with the same idle energy consumption. Similarly, as each block might contain multiple supports, there might be multiple different paths corresponding to one block-form schedule.

Remark 2. The graph-based approach described above can handle arbitrary concave idle energy function, which is a common shape of the idle energy function used in the literature [68, 56]. However, it is still an open question if the problem would be polynomial even if the idle energy function was not concave but arbitrary.

Remark 3. The energy graph could also be used to find the schedules minimizing the number of idle periods longer than 0. Such an application is useful when the stress of the machine caused by excessive switching needs to be minimized. The problem reduces again to the shortest path problem. The structure of the graph remains the same, but the edges in $E_G^{(3)}$ should be labeled by some positive constant, e.g., 1. Note that it is again possible to optimize only over the block-form schedules because the shifts described in the proof of Theorem 1.4.4 might join some blocks but never split them.

1.5 Electric Furnaces: Modeling, Optimal Control and Energy Function

Up till now, we have discussed how to solve scheduling problem $1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$, assuming that the energy function is given and concave. The majority of the existing papers addressing the idle energy optimization assume that the dynamics of the machine is described by a static transition graph, and its parameters are given. Obtaining those parameters or the idle energy function can be simple in some cases (e.g., for some hardware components in the embedded systems, the parameters or the idle energy function can be extracted from the data provided by the manufacturer), but becomes quite challenging in others. Since the idle energy optimization aims at a large variety of machines ranging from processors to huge furnaces, it is not possible to provide a single approach for obtaining the parameters of the transition graph or the idle energy function. Therefore, we concentrate on heat-intensive systems that are the most frequently addressed in connection with the idle energy optimization in production.

In this section, we discuss the electric furnace models and present a bilinear modeling approach, which is shown to provide a good approximation of industrial electric furnace dynamics. Further, the open-loop control for minimum energy consumption during idle periods, concerning the studied scheduling problem, is given based on the considered bilinear system approximation. Then, we show that the idle energy function as an input to the scheduling problem is concave under the proposed approximation and control, thus confirming the use of the above-proposed algorithm is correct.

1.5.1 A Bilinear Model Approximation of Furnaces

Obtaining and identifying a reasonable physical model of an industrial furnace is usually very difficult due to unspecified characteristics, imperfections or degradation of insulation materials, and time/temperature dependency of the physical parameters. Thus, instead of proposing a physical model and identifying its parameters, it is usual in practice to approximate the furnace dynamics with reasonable linear and nonlinear mathematical models; see, e.g., [129] for a linear model, [96] for a fuzzy system approximation, [130] for a direction-dependent model, and [135, 31] for bilinear system approximations.

Our decision to use the bilinear approximation of the furnace dynamics is motivated by the existing literature. For example, Derese and Nodulus [39] have reported that the bilinear model for heat-transfer processes is more suitable than the linear model. Chook and Tan [31] considered the identification of a first-order bilinear model for an electric tube furnace and showed experimentally that the bilinear model provides the most accurate description as compared with the linear and direction-dependent models. Another advantage of the bilinear model is its simplicity and well-understood behavior in the class of nonlinear systems. Thus, we also consider the approximation of the furnace dynamics similarly as in [31] with the bilinear model

$$\dot{x}(t) = -\alpha x(t) + \beta u(t) - \rho x(t)u(t), \quad x(t) \in \mathbb{R}, \quad u(t) \in [0, \bar{u}], \quad \alpha,\, \beta,\, \rho \in \mathbb{R}_{>0} \tag{1.14}$$

where u is the applied electric power (in kilowatts), i.e., the input to the system, and x is the deviation of the furnace temperature T_f (in kelvins) from the constant ambient temperature T_e, $x(t) := T_f(t) - T_e$, i.e., the variable to be controlled. The model (1.14) slightly differs from that in [31], because we additionally accommodate constraints on control and system parameters regarding the reality for furnaces. First, we do impose the upper bound $\bar{u}$ on the admissible control power, which is important in practice. Second, based on physical modeling considerations, it is assumed in (1.14) that the system parameters α, β, and ρ are positive constants. That is due to [31], where Section IV provides successful identification of α, β, ρ, resulting as positive numbers for their furnaces operational data. Note, that physical-principle-based modeling provided in [31] actually gives the following model

$$\dot{T}_f(t) = \frac{1}{C_f}\left(-\frac{T_f(t) - T_e}{R} + u(t) - K(T_f^4(t) - T_e^4)\right), \tag{1.15}$$

where C_f is the thermal capacitance, R is the thermal resistance, and K is a constant regarding the emissivity of the furnace. Obviously, $C_f,\ R,\ K > 0$; as already noted, T_e stands for the ambient temperature, which is assumed constant since its possible variations are negligible compared to extremely high furnace temperatures. Due to its complexity, instead of (1.15) [31] study simpler bilinear model (1.14) and provide some arguments for such a simplification. Indeed, there is a kind of trade-off: higher-order nonlinearity of (1.15) is replaced by bilinear dependence in (1.14), so rigorously (1.14) is not a simplification or approximation of (1.15). Yet, as shown in the sequel, (1.14) can be handled in an easier way, and some rigorous mathematical statements can be proved for it. Besides easier theoretical analysis, another argument justifying replacement of (1.15) by (1.14) given in [31] is that the nonlinearities that arise in heat-transfer processes may be represented by characteristics that are similar to those of a bilinear system. In such a way, we join the existing literature mainstream represented by Chook and Tan [31] and concentrate on the model (1.14) only. Note, that the constraints on $\alpha,\ \beta,\ \rho$ hold for the electrical vacuum furnace, which is studied in Section 1.6 as a case study. As shown in Section 1.6.1 later on, these parameters $\alpha,\ \beta,\ \rho$ can be quite precisely identified based on the real data, and the resulting estimates comply with the above assumptions.

1.5.2 Solving the Ordinary Differential Equation with a Discontinuous Right-Hand Side

Before formulating the main theorem of this section analyzing the optimal control of system (1.14), let us briefly recall the definition of the solution of the ordinary differential equation (ODE) with the possibly discontinuous right-hand side. This overview is presented in a rather casual way; rigorous and detailed theory can be found, e.g., in [49, Chapter 1]. Indeed, as it will be seen, the optimal control is a discontinuous function in time and thereby after substituting it into (1.14) one gets ODE with discontinuous (in time variable) right-hand side. Namely, consider ODE

$$\dot{x}(t) = f(x(t), t), \quad x \in \mathbb{R}^n. \tag{1.16}$$

The usual definition of the solution of (1.16) for its *continuous* right-hand side $f(x, t)$ is that the solution $x(t)$ is a continuously differentiable function of time converting the above ODE into equality valid for all times. As there are infinitely many such solutions, the specific unique solution is determined by the so-called *initial condition*

$$x(t_0) = x_0, \quad x_0 \in \mathbb{R}^n, \; t_0 \in \mathbb{R}, \tag{1.17}$$

where t_0, x_0 are given initial time and initial condition, respectively. The relations (1.16) and (1.17) are usually referred to as the *initial value problem*, or *Cauchy problem.* When the right-hand side of (1.16) is discontinuous, the solution of (1.16) cannot be continuously differentiable in time. When the discontinuity is with respect to time only, the usual way to handle this situation is to define the solution in *Caratheodory sense*; namely, the initial value problem (1.16) and (1.17) is replaced by the following integral equation

$$x(t) = x(t_0) + \int_{t_0}^{t} f(x(\tau), \tau) \mathrm{d}\tau, \tag{1.18}$$

where the solution $x(t)$ is required to be continuous only. Note, that the solution of the integral equation (1.18) automatically satisfies the initial condition (1.17) and, moreover, where $x(t)$ is in addition continuously differentiable, it implies the validity of (1.16). As already noted, Caratheodory approach helps to handle the discontinuity with respect to the time variable only. The discontinuity with respect to state variable x presents even more tough challenge and even more abstract solution is required, namely the so-called solution in the Fillipov's sense.

In the subsequent analysis, all the time discontinuities will be of the simplest kind, i.e., they will be piecewise continuous. In this case, Caratheodory solution can also be obtained in the following intuitively clear way. Namely, ODE is solved together with the initial condition on the largest time interval where $f(x, t)$ is continuous. When reaching discontinuity point $t_{dc} \in \mathbb{R}$, the resulting solution value $x(t_{dc})$ is taken as the initial condition for the next time interval where $f(x, t)$ is continuous; ODE is solved again and this procedure can be repeated.

Note that such an approach correctly represents reality. In the case of furnace heating, it means that discontinuous jump change of heating influences further development of the temperature, but the temperature has to stay continuous even at the point where heating intensity experiences jump, see Figure 1.7. Obviously, such an understanding of the solution of the ODE with time discontinuity at its right-hand side is the only acceptable one from the natural and practical point of view. Putting it in different words, under quite mild and reasonable mathematical technical assumptions imposed on the right-hand side $f(x,t)$, there is a unique solution that satisfies ODE in a classical sense everywhere except some isolated time moments, where this unique solution is at least continuous. In other words, many solutions are possible, but only one of them is everywhere at least continuous.

In the sequel, we will use exactly the latter approach to obtain the unique solution of the initial value problem when heating intensity (the input) is piecewise constant. Namely, we compute the solution to the initial value problem on time subinterval where heating intensity is constant. Then, at the time where heating intensity jumps to a different constant value, we use the terminal value of temperature on the first time subinterval as the initial condition for the ODE solution on the next time subinterval.

1.5.3 Minimum-Energy Control and the Related Idle Energy Function

This subsection aims to study the optimal control of furnaces during an idle period, based on the approximate bilinear model (1.14).

Recall that our aim is to find an energy-efficient behavior of the furnace in an idle period. Thus, we look for an optimal control law, which minimizes the power consumption for any fixed idle period length. Then, our problem for furnaces turns into finding a control minimizing the performance index

$$J(u) = \int_0^{t_f} |u(t)|dt \tag{1.19}$$

which is called as minimum-control-effort problem [73]. Obviously, t_f can be considered as the idle period length, i.e., $(s_{i+1} - (s_i + p_i))$ in (1.1). Then $J(u)$ is the energy (in kilowatt-hours) consumed during the corresponding idle period, i.e., $E(s_{i+1} - (s_i + p_i))$ in (1.1). Note, that it is sufficient to consider an open-loop control to heat the furnace to the (close neighborhood of) operating temperature at the end of the idle period (assuming constant ambient temperature), whereas a closed-loop control is necessary to maintain the operating temperature. Such a control strategy is actually common in process control applications, e.g., see Figure 1.5 with the temperature data of the real industrial furnace controlled to operate at different temperatures in our case study. As we seek a control minimizing energy consumption during the idle periods, we give the following theorem for the open-loop optimal control problem for the industrial furnaces which can be modeled as the bilinear system in (1.14).

Theorem 1.5.1. *Consider the following optimal control problem: minimize the performance index (1.19) subject to constraints*

$$x(0) = x(t_f) = x_0 \in \mathbb{R},\ x_0 > 0, \tag{1.20}$$

where $x(t)$ is the solution of the system (1.14) *and $t_f > 0$ is a given fixed terminal time. Further, assume that*

$$(\beta - \rho x_0)\bar{u} - \alpha x_0 > 0, \tag{1.21}$$

where $\bar{u}$ *is the upper bound on* $u(t)$*. Then there exists the unique optimal control* $u^*(t)$ *solving the above-defined optimal control problem and this optimal control takes the following form*

$$u^*(t) = \begin{cases} 0, & \forall t \in [0, t_{sw}) \\ \bar{u}, & \forall t \in [t_{sw}, t_f], \end{cases} \tag{1.22}$$

where $t_{sw} \in (0, t_f)$ *is the switching time. Finally,* t_{sw} *is the solution of the following equation*

$$x_0 = \exp\left((-\alpha - \rho\bar{u})(t_f - t_{sw})\right)\left(x_0 \exp(-\alpha t_{sw}) - \frac{\beta\bar{u}}{\alpha + \rho\bar{u}}\right) + \frac{\beta\bar{u}}{\alpha + \rho\bar{u}}, \tag{1.23}$$

this solution exists and is unique for any given $t_f > 0$*. Furthermore, defined in such a way function* $t_{sw}(t_f)$ *satisfies*

$$\frac{\mathrm{d}t_{sw}}{\mathrm{d}t_f} = 1 - \frac{\alpha x_0}{(\beta \exp(\alpha t_{sw}) - \rho x_0)\bar{u}}. \tag{1.24}$$

Proof. Pontryagin's minimum principle (PMP) is used [73]. To do so, realise that $|u(t)|$ in (1.19) can be replaced simply by $u(t)$ because $u(t) > 0\,\forall t$ in (1.14). Further, the appropriate Hamiltonian function for the performance index (1.19) and the system (1.14) is given by

$$H(x(t), u(t), \psi(t)) = u(t) - \alpha\psi(t)x(t) + \psi(t)[\beta - \rho x(t)]u(t) \tag{1.25}$$

where $\psi(t)$ represents the usual adjoint variable. By PMP, the necessary conditions for $u^*(t)$ to be an optimal control are

$$\dot{x}^*(t) = \frac{\partial H(x^*, u^*, \psi^*)}{\partial \psi} = -\alpha x^*(t) + \beta u^*(t) - \rho x^*(t)u^*(t), \tag{1.26a}$$

$$\dot{\psi}^*(t) = -\frac{\partial H(x^*, u^*, \psi^*)}{\partial x} = \psi^*(t)(\rho u^*(t) + \alpha), \quad \psi(0) = \psi_0 \in \mathbb{R} \setminus \{0\}, \tag{1.26b}$$

$$H(x^*(t), u^*(t), \psi^*(t)) = \min_{u \in [0, \bar{u}]} H(x^*(t), u(t), \psi^*(t))\ \forall t \in [0, t_f] \quad \Rightarrow \tag{1.26c}$$

$$u^*(t) + \psi^*(t)[\beta - \rho x^*(t)]u^*(t) = \min_{u \in [0, \bar{u}]} \left(u(t) + \psi^*(t)[\beta - \rho x^*(t)]u(t)\right)\ \forall t \in [0, t_f]. \tag{1.26d}$$

Indeed, the boundary conditions (1.20) of the investigated control problem are fixed, so that $\psi(t)$ can be any nontrivial solution of the adjoint equation (1.26b).

Before analyzing the above necessary condition for the optimality, let us give the following property useful later on. Namely, (1.26a) and (1.26b) can be solved analytically giving that

$$x^*(t) = \exp\left(-\alpha t - \rho \int_0^t u^*(\eta)d\eta\right)\left(x_0 + \beta \int_0^t \exp\left(\alpha\eta + \rho \int_0^\eta u^*(s)ds\right) u^*(\eta)d\eta\right), \tag{1.27}$$

$$\psi^*(t) = \psi_0 \exp\left(\alpha t + \rho \int_0^t u^*(\eta)d\eta\right). \tag{1.28}$$

To analyze (1.26a)–(1.26d) subject to the control constraint $u(t) \in [0, \bar{u}]$, consider the function

$$\phi(\psi^*(t), x^*(t)) = \psi^*(t)(\beta - \rho x^*(t)) + 1 \tag{1.29}$$

to investigate the minimum of the Hamiltonian with respect to u. Further, realise that the necessary condition (1.26c)–(1.26d) implies that $u(t) = \bar{u}$ if $\phi(\psi^*(t), x^*(t)) < 0$; $u(t) = 0$ if $\phi(\psi^*(t), x^*(t)) > 0$; whereas for $\phi(\psi^*(t), x^*(t)) = 0$ it is always satisfied. As a consequence, the optimal control, if it exists, satisfies

$$u^*(t) \begin{cases} = \bar{u}, & \text{for} \quad \phi(\psi^*(t), x^*(t)) < 0 \\ = 0, & \text{for} \quad \phi(\psi^*(t), x^*(t)) > 0 \\ \in [0, \bar{u}], & \text{for} \quad \phi(\psi^*(t), x^*(t)) = 0. \end{cases} \tag{1.30}$$

Furthermore, by (1.27) and (1.28) it holds that

$$\begin{aligned} \phi(t) &= 1 - \psi_0 x_0 \rho + \psi_0 \beta \exp\left(\alpha t + \rho \int_0^t u(\eta) d\eta\right) \\ &\quad -\psi_0 \beta \rho \int_0^t \exp\left(\alpha t + \rho \int_0^\eta u(s) ds\right) u(\eta) d\eta, \\ \frac{d\phi(t)}{dt} &= \psi_0 \beta \left(\alpha + \rho u(t)\right) \exp\left(\alpha t + \rho \int_0^t u(\eta) d\eta\right) \\ &\quad -\psi_0 \beta \rho \, u(t) \exp\left(\alpha t + \rho \int_0^t u(\eta) d\eta\right), \end{aligned} \tag{1.31}$$

which implies

$$\frac{d\phi(t)}{dt} = \psi_0 \alpha \beta \exp\left(\alpha t + \rho \int_0^t u(\eta) d\eta\right). \tag{1.32}$$

Now, using (1.31) and (1.32) one concludes that

$$\phi(0) = \psi_0(\beta - \rho x_0) + 1, \tag{1.33}$$

$$\operatorname{sign}\left(\frac{d\phi}{dt}\right) = \operatorname{sign}(\psi_0), \ \psi_0 \neq 0. \tag{1.34}$$

Note that by (1.34) $\phi(t)$ is obviously a strictly monotonous function. In such a way, $\phi(t)$ either vanishes at a single isolated point only, or it never vanishes. As $\psi_0 \neq 0$, only the following four options are possible for $u^*(t)$ to be optimal.

1. If $\psi_0 > (\rho x_0 - \beta)^{-1} > 0$, then $\phi(0) > 0$ and $\frac{d\phi(t)}{dt} > 0, \forall t \geq 0$, which means $\phi(t) > 0, \forall t \geq 0$. By (1.30), then $u^*(t) \equiv 0$. However, it is clear from (1.27) that (1.14) with $u(t) \equiv u^*(t) \equiv 0$ does not satisfy (1.20).

2. If $(\rho x_0 - \beta)^{-1} > \psi_0 > 0$, then $\phi(0) < 0$ and $\frac{d\phi(t)}{dt} > 0, \forall t \geq 0$. By (1.30), then $u^*(t) = \bar{u},\ t < t_{sw}$ and $u^*(t) = 0,\ t > t_{sw}$. However, this option is not possible because $(\rho x_0 - \beta) > 0$ contradicts the assumption (1.21) as α, $\bar{u}$ and x_0 are positive.

3. If $\psi_0 < (\rho x_0 - \beta)^{-1} < 0$, then $\phi(0) < 0$ and $\frac{d\phi(t)}{dt} < 0, \forall t \geq 0$, which means $\phi(t) < 0, \forall t \geq 0$. By (1.30), then $u^*(t) \equiv \bar{u}$. However, by assumption (1.21) and by (1.27) it holds that $x(t_f) > x_0$. Thus, (1.20) is violated.

4. If $(\rho x_0 - \beta)^{-1} < \psi_0 < 0$, then $\phi(0) > 0$ and $\frac{d\phi(t)}{dt} < 0$, $\forall t \geq 0$. By (1.30), then
$$u^*(t) = 0,\ t < t_{sw},\ u^*(t) = \bar{u},\ t > t_{sw}; \quad t_{sw} = \alpha^{-1} \log\left((\rho x_0 \psi_0 - 1)/(\beta\psi_0)\right). \tag{1.35}$$
Moreover, it can be seen through some straightforward analysis that when ψ_0 ranges through $((\rho x_0 - \beta)^{-1}, 0)$, the expression $((\rho x_0 \psi_0 - 1)/(\beta\psi_0))$ ranges through $(1, \infty)$, i.e., ψ_0 can always be chosen in such a way that any $t_{sw} \in (0, \infty)$ is possible.

Summarizing, the control satisfying PMP and (1.20) under assumption (1.21) should have the form (1.35) for some suitable switching time t_{sw}. To conclude the proof, it remains to show that there is a unique $t_{sw} \in [0, t_f)$ such that (1.14) with $u(t) \equiv u^*(t)$ given by (1.35) satisfies the boundary conditions (1.20). Such a property follows straightforwardly by (1.27) and (1.21), moreover, also by (1.27), the switching time t_{sw} is the solution of
$$x_0 = \exp\left((-\alpha - \rho\bar{u})(t_f - t_{sw})\right)\left(\exp(-\alpha t_{sw})x_0 - \frac{\beta\bar{u}}{\alpha + \rho\bar{u}}\right) + \frac{\beta\bar{u}}{\alpha + \rho\bar{u}}. \tag{1.36}$$

Indeed, on the right-hand side of (1.36) there is a value of temperature trajectory $x(t)$ at time t_f obtained by solving (1.14) on subinterval $[0, t_{sw})$ with initial condition $x(0) = x_0$ applying the input (applied power) $u \equiv 0$ and then solving (1.14) with initial condition $x(t_{sw}) = \exp(-\alpha t_{sw})x_0$ and the input $u \equiv \bar{u}$ on subinterval $[t_{sw}, t_f]$.

Note, that t_{sw} solving (1.36) exists and is unique for any given $t_f > 0$. Indeed, the right-hand side of (1.36) is a smooth function of t_{sw} and it is equal to $\exp(-\alpha t_f)\, x_0 < x_0$ if $t_{sw} = t_f$ and to
$$\exp\left((-\alpha - \rho\bar{u})t_f\right)\left(x_0 - \frac{\beta\bar{u}}{\alpha + \rho\bar{u}}\right) + \frac{\beta\bar{u}}{\alpha + \rho\bar{u}} > x_0,$$
if $t_{sw} = 0$. The last inequality straightforwardly holds thanks to the assumption (1.21) and $\exp\left((-\alpha - \rho\bar{u})t_f\right) \in (0, 1)$. As a consequence, there exists at least one t_{sw} solving (1.36) thanks to the well-known basic property of continuous functions. To show that such t_{sw} is unique, note that the right-hand side of (1.36) is strictly decreasing function of t_{sw} since its derivative with respect to t_{sw} is
$$\bar{u} \cdot \exp((-\alpha - \rho\bar{u})(t_f - t_{sw})) \cdot (\rho x_0 \exp(-\alpha t_{sw}) - \beta),$$
which is negative since by the assumption (1.21) $\beta > \rho x_0$ and obviously $\rho x_0 > \rho x_0 \exp(-\alpha t_{sw})$ as $\alpha > 0, t_{sw} \geq 0$. In such a way, the value t_{sw} solving (1.36) exists and is unique. Finally, to prove (1.24) apply the well-known formula to compute the derivative of the implicitly defined function and perform some straightforward, though laborious computations. The proof is complete. ☐

Remark 4. The assumption (1.21) is equivalent to $\alpha x_0/(\beta - \rho x_0) \in (0, \bar{u})$. The value $\alpha x_0/(\beta - \rho x_0)$ is the constant trim control keeping the state x_0 as the equilibrium, i.e., $x(t) \equiv x_0$ and therefore the assumption (1.21) should be valid in any reasonable practical setting. Indeed, if the assumption (1.21) is to be replaced by $(\beta - \rho x_0)\bar{u} - \alpha x_0 = 0$, then the optimal control is $u^*(t) = \bar{u}$, $\forall t \in [0, t_f]$, *i.e., as if* $t_{sw} = 0$ *in* (1.35). As such, $\bar{u} = \alpha x_0/(\beta - \rho x_0)$ is the trim control value that ensures $x(t) \equiv x_0$; practically, such a situation is not acceptable because any small perturbation pushing the state to a value slightly lower than x_0 cannot be compensated for.

Remark 5. We consider the optimal control law with the state constraint (1.20) because a single operating temperature x_0 for the scheduling problem is considered. Definitely, the furnace temperature is x_0 at the beginning of each idle period and should also be x_0 at the end of the idle period to execute the consecutive task. In fact, Theorem 1.5.1 can be easily extended to a more general case with boundary conditions of the form $x(0) = x_0$, $x(t_f) = x_f$, $x_0 > 0$, $x_f > 0$ and, possibly, $x_0 \neq x_f$.

Let us finally show that the energy function of the idle period length, for a furnace described by the bilinear model (1.14) and optimally controlled as proposed in Theorem 1.5.1, is concave.

Theorem 1.5.2. *The idle energy function* $E : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ *of system (1.14) under control (1.22) assuming (1.21) is described by equation* $E(t_f) = \bar{u} \cdot (t_f - t_{sw}(t_f))$ *for any* $t_f \in \mathbb{R}_{\geq 0}$*, where* $t_{sw}(t_f)$ *is the function existing by* (1.23)*. Moreover,* $E(t_f)$ *is concave.*

Proof. Recall that $\bar{u}$ is constant maximal value of the applied electric power in (1.14). Also recall from the proof of Theorem 1.5.1 that t_{sw} in control (1.22) applied to system (1.14) is uniquely determined with the implicit solution of (1.23) for given t_f and fixed parameters α, β, ρ, x_0, and $\bar{u}$. Thus, the energy consumption during an idle period, i.e., idle energy function, can be described as

$$E(t_f) = \bar{u} \cdot (t_f - t_{sw}(t_f)). \tag{1.37}$$

Then, for concavity of $E(t_f)$, it remains to show that

$$\frac{\partial^2 E(t_f)}{\partial t_f^2} = -\bar{u}\,\frac{\mathrm{d}^2 t_{sw}}{\mathrm{d}t_f^2} \tag{1.38}$$

is negative $\forall t_{sw}$. Substituting further differentiation of (1.24) to (1.38) gives

$$\frac{\partial^2 E(t_f)}{\partial t_f^2} = -\frac{\alpha^2 \beta x_0 \exp(\alpha t_{sw})}{(\rho x_0 - \beta \exp(\alpha t_{sw}))^2}\,\frac{\mathrm{d}t_{sw}}{\mathrm{d}t_f}. \tag{1.39}$$

To prove (1.39), first note that by assumption (1.21), it holds $\beta > \rho x_0$ and therefore, the denominator of the fraction in (1.39) is positive. The numerator $\alpha^2 \beta x_0 \exp(\alpha t_{sw})$ is positive as well, since $\beta > 0$ by definition (1.14) and $x_0 > 0$ by (1.20). Therefore, to prove that (1.39) is negative, it remains to show that $(\mathrm{d}t_{sw})/(\mathrm{d}t_f) > 0\,\forall t_{sw}$. By (1.21) we have $\alpha x_0 < (\beta - \rho x_0)\bar{u}$, which also implies $\alpha x_0 < (\beta \exp(\alpha t_{sw}) - \rho x_0)\bar{u}$ since $\alpha > 0$ by definition (1.14) and $t_{sw} \geq 0$. It follows that $\frac{\alpha x_0}{(\beta \exp(\alpha t_{sw}) - \rho x_0)\bar{u}} < 1$, which in turn proves that $(\mathrm{d}t_{sw})/(\mathrm{d}t_f) > 0$. As a consequence, $E(t_f)$ is concave, and the proof is complete. □

By Theorem 1.5.2, we conclude that problem $1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$ can be solved in polynomial time for furnaces that can be modeled as (1.14), and controlled by (1.22). In the following section, the proposed approach is shown on a real industrial electric furnace from Škoda Auto.

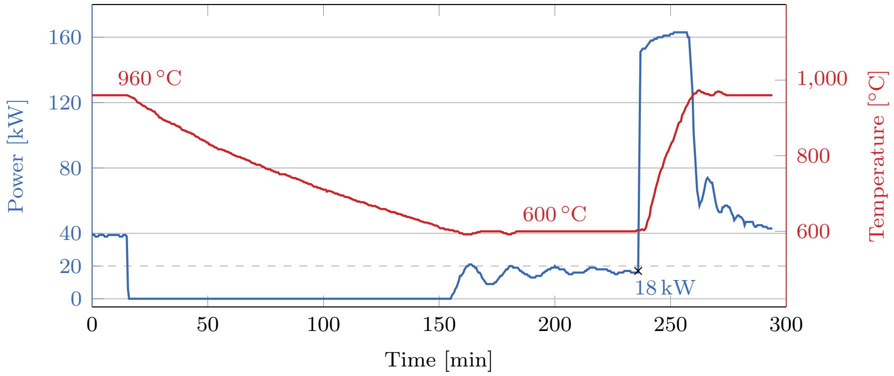


Figure 1.5: Relationship between the temperature and power when cooling to 600 °C and heating back to operating temperature

1.6 Case Study: An Industrial Electric Furnace

Škoda Auto has a production line employing a ModulTherm® system by ALD, containing electric vacuum furnaces used for the steel hardening. The outer steel shells of the furnaces are cooled by a central cooling system of circulating water at ∼35 °C to avoid overheating of the system. Thus, we can assume that the ambient temperature (T_e) is constant. The operating temperature of the furnaces is set to 960 °C for the hardening process, which takes about 2.5 hours on average.

The heating of the furnaces has a substantial energy demand across the whole production line. In a normal regime, all furnaces are turned on and heated to the operating temperature. The operating temperature is preserved even if nothing is being processed. To investigate the potential for energy savings, an experiment has been performed, during which the furnace was cooled to 600 °C, and its steady-state power consumption was measured. Afterwards, the furnace was heated back to the operating temperature again. Measured data are shown in Figure 1.5 [43]. It can be seen that the steady-state power consumption for 600 °C and 960 °C is about 18 kW and 40 kW, respectively.

Clearly, if the idle period is long enough, significant energy savings can be achieved by lowering the temperature of the furnace, i.e., turning off the furnace for a longer time and then reheating it back at the right time. This can be achieved by the optimal control law described in the previous section. The rest of the section documents the identification of the furnace in Škoda Auto and shows the resulting idle energy function.

1.6.1 Identification of the Furnace Model

We employ the bilinear model given by (1.14) to the furnace mentioned above and estimate the parameters α, β, ρ in the model. For this purpose, we use the temperature data collected by Dušek [43] shown by dashed lines in Figure 1.6, with a sampling time of 30 s. The system parameters are estimated as

$$\alpha = 0.003821964, \quad \beta = 0.175187494, \quad \rho = 0.000094367 \tag{1.40}$$

by the least-squares method using the measured temperature samples and their derivatives obtained via a polynomial regression. The simulated response of the system (1.14) with (1.40) is illustrated by red lines in Figure 1.6, when the experimental input power is applied. It is seen that the utilized bilinear model provides a reasonable fit to the measured temperature values of the furnace. Note, that all the measurements were carried out during production and it was not possible to test arbitrary input signals (i.e., power). Nevertheless, the mean absolute percentage error over all experiments for the identified model is found as 4.49 %, which is sufficiently accurate for the system identification.

1.6.2 Idle Energy Function of the Furnace

To reveal the idle energy function of the furnace, let us first demonstrate the furnace temperature response under the proposed energy-optimal control law given by Theorem 1.5.1. In Figure 1.7, the time response of the furnace model (1.14) with the parameters (1.40) is illustrated via simulations for two different terminal times ($t_f^{(1)}$ and $t_f^{(2)}$), i.e., idle periods, when the optimal control (1.22) is applied. Indeed, the applied input power is switched from zero to the maximum applicable power $\bar{u}$ (160 kW) at the appropriate switching times $t_{sw}(t_f^{(1)})$ and $t_{sw}(t_f^{(2)})$ calculated by (1.23), to ensure reaching the operating temperature (960 °C) at the end of each idle period. The corresponding minimal energy consumption $E(t_f^{(1)})$, and $E(t_f^{(2)})$ (calculated by (1.37)), is also illustrated in the lower part of Figure 1.7.

Performing the above explained calculations for an appropriate sampling of the idle period length t_f, one can obtain the idle energy function E, as shown in Figure 1.8. Function E is bounded by a constant shown by the dashed line, which is the energy for heating the machine from the ambient temperature (35 °C) to the operating temperature. Clearly, it is seen that E is concave, as declared by Theorem 1.5.2.

Remark 6. Note that for the real furnace application the proposed control may not be precisely optimal, and the operating temperature may not be reached exactly at $t = t_f$, inherently due to the uncertain dynamics and the approximate modeling. Nevertheless, the proposed approximation is acceptable for achieving almost optimal control in practice. The reach of the operating temperature can be guaranteed with a simple *if case* control as is actually done in switching to feedback control around the operating point in practical process control approaches.

1.7 Comparison to the State-of-the-Art Approaches

As it was explained in the introduction, conventional scheduling approaches to idle energy optimization assume only a small number of machine modes to approximate the dynamics of the machine [97, 121, 29, 2]. To represent the machine modes,

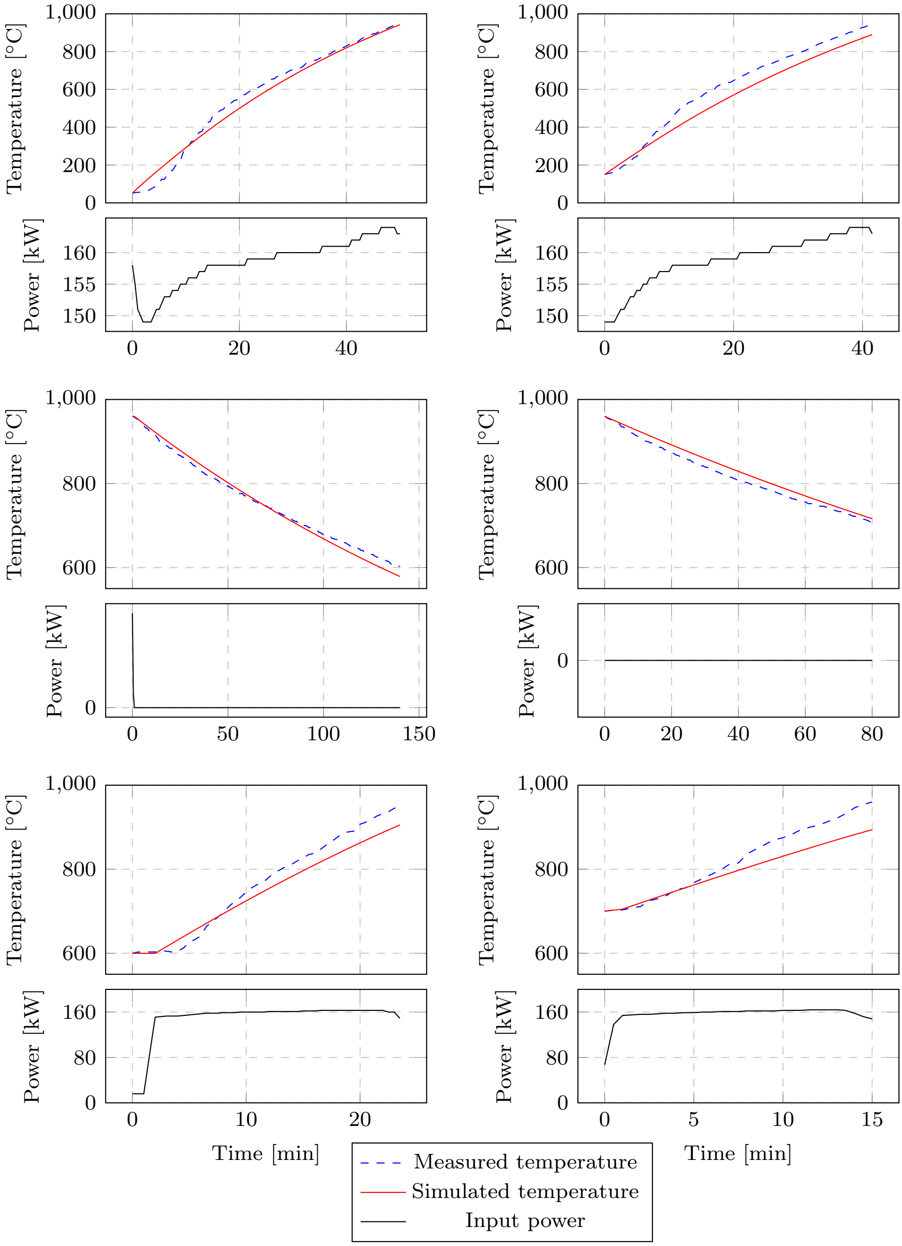


Figure 1.6: Comparison of the measured data and simulation using a bilinear model.

the authors typically use the static transition graph, where the vertices represent the modes, and the edges represent the available transitions between them. The edges are labeled by the time, which is needed for the transition, and the power,

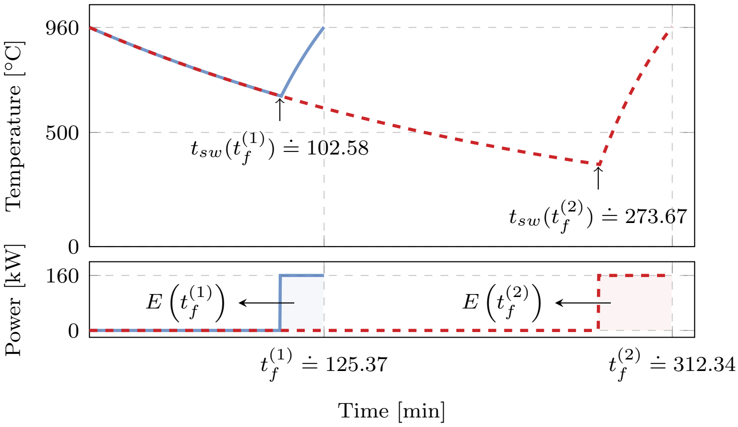


Figure 1.7: Example of the optimal control for two different terminal times $t_f^{(1)}$ and $t_f^{(2)}$.

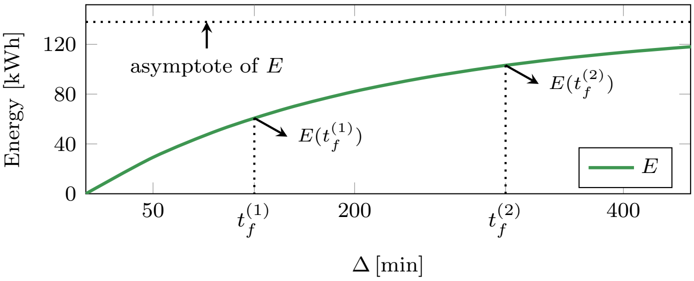


Figure 1.8: Idle energy function E and two different idle period lengths $t_f^{(1)}$ and $t_f^{(2)}$ with the corresponding idle energy consumption $E\left(t_f^{(1)}\right)$ and $E\left(t_f^{(2)}\right)$.

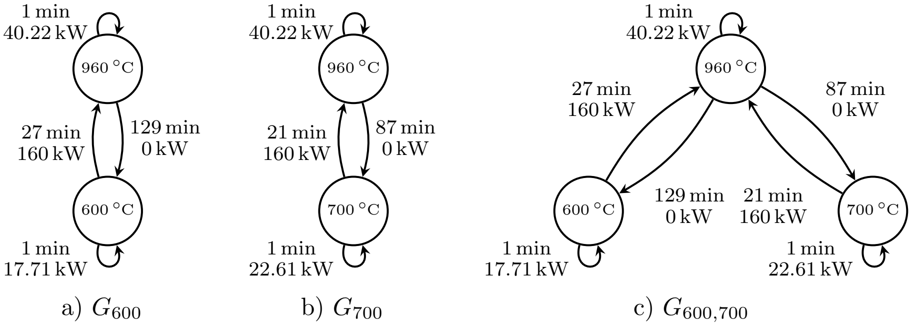


Figure 1.9: Examples of the transition graphs for the furnace model (1.14) with parameters (1.40).

which is consumed during the transition. Examples of the transition graphs for the furnace model (1.14) with parameters (1.40) are shown in Figure 1.9. These graphs represent simple scenarios, with a single processing mode (960 °C) and one (G_{600}, G_{700}), or two ($G_{600,700}$), standby modes. The standby modes correspond to allowed temperatures, to which the furnace can be cooled during the idle periods (here 600 °C, and 700 °C).

The primal aim of this section is to show, why representation via an idle energy function is better than a transition graph. This is illustrated by an experiment described in Section 1.7.2. Secondly, we compare complexity of the algorithm for problem $1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$ described in Section 1.4 with the state-of-the-art approaches. This analysis is described in Section 1.7.3.

1.7.1 Benchmark Instances

Considering the behavior of the machine, we use the idle energy function E depicted in Figure 1.8 for the minimization of the objective (1.1). Our approach is compared to the dynamic programming adopted from [3], which represents the behavior of the machine by a finite transition graph. For the comparison, we use the transition graphs G_{600}, G_{700}, and $G_{600,700}$ depicted in Figure 1.9.

Now we describe, how we generate the tasks parameters for the benchmarks instances. A set of 6750 instances was generated using Algorithm 1. Specifically, 10 instances were generated for each combination of $n \in \{30, 40, 50\}$, $\gamma \in \{0.2, 0.4, \ldots, 3.0\}$, and $\delta \in \{0.2, 0.4, \ldots, 3.0\}$. A wide range of parameters γ and δ was used to generate data of different characteristics. Constants $p_{\min}$ and $p_{\max}$, denoting the minimal and the maximal processing time, were set to 1 and 300, respectively. Note that Algorithm 1 is designed such that only feasible instances are generated. By $U\{a, b\}$, we denote integer uniform distribution on set $\{a, a+1, \ldots, b\}$; here $Exp(x)$ denotes exponential distribution with scale parameter x.

One of the factors influencing the final energy savings is the utilization of the machine, which is calculated as the ratio between the sum of processing times and

```
input  : Number of tasks $n$, bounds on processing time $p_{\min}$, $p_{\max}$, params. $\gamma$, $\delta$
output : Vectors $\boldsymbol{r}$, $\tilde{\boldsymbol{d}}$, $\boldsymbol{p}$
// generate processing times
foreach $i \leftarrow 1$ to $n$ do $p_i \sim U\{p_{\min}, p_{\max}\}$;
// generate release times and deadlines
$r_1 := 0$ ;
$\tilde{d}_1 \sim \lceil r_1 + p_1 + Exp(\delta \cdot \texttt{Average}(\boldsymbol{p})) \rceil$ ;
foreach $i \leftarrow 2$ to $n$ do
    $r_i \sim \lceil r_{i-1} + p_{i-1} + Exp(\gamma \cdot \texttt{Average}(\boldsymbol{p})) \rceil$;
    $\tilde{d}_i \sim \lceil r_i + p_i + Exp(\delta \cdot \texttt{Average}(\boldsymbol{p})) \rceil$ ;
// propagate deadlines by (1.3) (release times are already propagated)
foreach $i \leftarrow (n-1)$ to 1 do $\tilde{d}_i := \min\{\tilde{d}_{i+1} - p_{i+1}, \tilde{d}_i\}$;
```

Algorithm 1: Generation of task parameters

Table 1.2: Number of generated instances with respect to utilization (columns) and number of tasks (rows).

		Utilization							
		(0.1, 0.2]	(0.2, 0.3]	(0.3, 0.4]	(0.4, 0.5]	(0.5, 0.6]	(0.6, 0.7]	(0.7, 0.8]	(0.8, 0.9]
	30	12	532	621	391	286	193	125	90
n	40	2	508	672	383	273	191	113	108
	50	5	520	672	376	252	195	121	109
Total		19	1560	1965	1150	811	579	359	307

length of the scheduling horizon, i.e., $\sum_{i=1}^{n} p_i \,/\, (\tilde{d}_n - r_1)$. Based on the machine utilization, the generated instances were divided, as indicated by Table 1.2.

1.7.2 Transition Graph vs. Idle Energy Functions

For the experiment, we optimized all generated instances with respect to the idle energy functions E (our approach), and transition graphs G_{600}, G_{700}, and $G_{600,700}$ (representing the state-of-the-art approaches assuming only a small number of modes). The instances with transition graphs G_{600}, G_{700}, and $G_{600,700}$ were optimized using the dynamic programming adopted from [3].

To compare the results, we define the average power per idle time $\overline{P}$ as

$$\overline{P} = \frac{E^{\star}_{\text{total}}}{(\tilde{d}_n - r_1) - \sum\limits_{i=1}^{n} p_i}, \tag{1.41}$$

where $E^{\star}_{\text{total}}$ is the optimal total idle energy consumption (with respect to given idle energy function or transition graph). It is assumed that the machine is underutilized, i.e., $(\tilde{d}_n - r_1) - \sum_{i=1}^{n} p_i > 0$. For the considered models, it holds that $0 \leq \overline{P} \leq \overline{P}_{\max}$, where $\overline{P}_{\max}$ is the theoretical worst case, representing the situation when the furnace is heated to the operating temperature all the time.

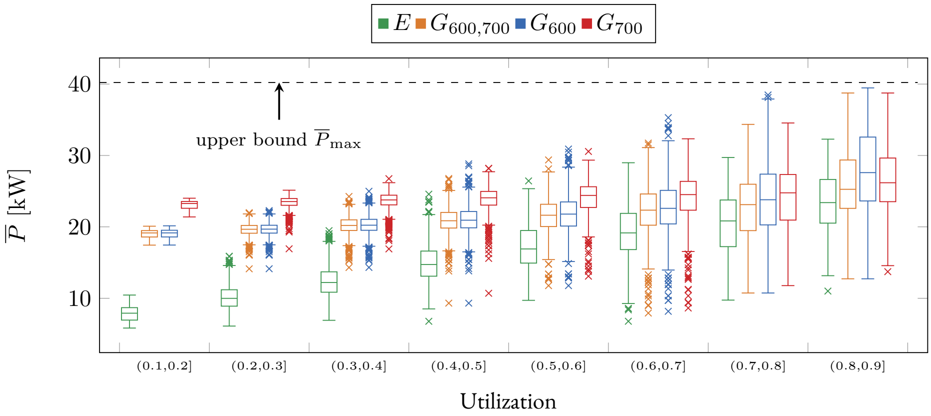


Figure 1.10: Average power per idle time $\overline{P}$ depending on the modeling of the machine dynamics and utilization of the machine.

Results for different utilizations of the machines are shown in the form of boxplots in Figure 1.10. Clearly, our approach using E dominates all the transition graphs, as the power saving modes modeled by G_{600}, G_{700}, and $G_{600,700}$ are only a subset of all possible modes implicitly encoded in E. The difference increases when utilization is lowered as the idle periods become longer. For example, the average $\overline{P}$ for E is less than half compared to $G_{600,700}$ for utilization $(0.1, 0.2]$.

It can be seen that $\overline{P}$ optimized with respect to G_{600} nearly converges to steady-state power compensating for the energy loss at 600 °C, which is approximately 18 kW. Similar observation also holds for G_{700}, and $G_{600,700}$. Using G_{700} is slightly better than G_{600} only when the utilization is high because shorter idle periods do not allow the standby mode corresponding to 600 °C to be reached.

1.7.3 Time Complexity Comparison

The authors of conventional scheduling approaches to idle energy optimization use the ILP formalism for the modeling [97, 121, 29, 2]. The scheduling horizon is discretized into a set of intervals H (e.g., one minute long), and for each interval $k \in H$ and each possible mode of the machine m, binary variables encode whether the machine operates in mode m during interval k or not [2, 121, 4]. The main weakness in these approaches is that the size of the model depends on the number of intervals in H as well as on the number of machine states. Therefore, the model can be used successfully only for small instances of the problem. When long scheduling horizon is considered (e.g., 7200 minutes in a work-week), building and optimization of such model become intractable.

To the best of our knowledge, the nearest polynomial-time approach that can be adopted to solve the problem addressed in this chapter is described in [3]. Assuming that the scheduling horizon is discretized and the order of the tasks if fixed, the problem can be transformed to the shortest path problem. Aghelinejad et al. construct graph G having $|H|$ layers, each of which is containing about $\sum_{i \in T} p_i$

nodes. Node $n(i,k)$ in layer k encodes that i intervals were spent for the processing from the beginning till time k. The graph contains $\mathcal{O}(|H| \sum_{i \in T} p_i)$ nodes, and $\mathcal{O}(|H|^2 \sum_{i \in T} p_i)$ edges. The shortest path representing the schedule with lowest energy consumption can be found by dynamic programming in $\mathcal{O}(|H|^2 \sum_{i \in T} p_i)$. In the original paper [3], the authors did not assume release times and deadlines. However, their approach can be easily extended by removing the edges, which would cause the processing of the task i outside of its execution window defined by $[r_i, \tilde{d}_i]$. Further, in the case of the problem studied in this chapter, it is not necessary to model every unit of tasks' processing times. Thus, term $\sum_{i \in T} p_i$ can be substituted by n (processing units corresponding to a single job can be joined together). Therefore, the complexity of solving our problem by the approach described in [3] is $\mathcal{O}(|H|^2 n)$ assuming that the scheduling horizon is discretized into $|H|$ intervals.

In comparison, the energy graph proposed in this chapter contains $\mathcal{O}(n)$ nodes and $\mathcal{O}(n^2)$ edges and can be constructed in $\mathcal{O}(n^3)$ steps. The overall complexity of our approach is therefore $\mathcal{O}(n^3)$. Taking into account that for a real production $|H|$ is typically larger than n, the complexity of our approach is significantly better.

Summarizing, we believe that there are two main drawbacks in the adaptation of the state-of-the-art approaches (including both the ILP models as well as the graph proposed in [3]). First, the complexity of the state-of-the-art approaches sharply grows with the length of the scheduling horizon H, while our approach is independent on it. Second, a finite number of machine modes cannot fully describe the behavior of more complex systems. For example, see function E in Figure 1.8 representing the energy consumption w.r.t. the length of the idle period for our case study. The shape of this function cannot be reasonably approximated by a simple transition graph with several modes only.

1.8 Chapter Conclusions

This chapter has two aims. The first one is to show that for some machines, e.g., furnaces and other heat-intensive systems, when approximating their dynamics by a simple transition graph, the scheduling algorithm cannot achieve the maximum energy savings. For such systems, we propose a different concept incorporating the complete dynamics and the optimal control of the machine into the idle energy function, which represents the energy consumption of the machine much better. The analysis in Section 1.7.2 on an electric furnace from Škoda Auto company shows the significant difference between these two concepts. Second, we show that problem $1 \,|\, r_j, \tilde{d}_j, \text{fixed order} \,|\, \Sigma E$ can be solved in polynomial time, assuming that the idle energy function is concave. The time complexity of our algorithm is better than the complexity of related state-of-the-art algorithms, as it is explained in Section 1.7.3.

Our analysis is focused on heat-intensive processes, as the most typical applications in the domain of idle energy optimization and scheduling. Indeed, our analysis cannot be applied to an arbitrary machine, and we cannot analyze every possible one. Nevertheless, many energy demanding systems have very similar properties, often resulting in a concave idle energy function. Moreover, the concept of energy function allows integrating the system dynamics and its energy-optimal control, studied in the control engineering domain, into the scheduling domain. As we believe, this synergy is essential for achieving maximal energetic efficiency. A related example can be found in papers [23, 24] studying energy optimization of robotic cells, where very complex dynamics of a robotic manipulator is also encoded into an energy function. Those papers do not study idle energy consumption but address the relation between the speed limit of a robot movement and its energy consumption. Unlike the case with the furnaces, this function is convex; nevertheless, the idea of the decomposition is the same. Therefore, as we believe, there are other applications where the complex dynamics of a machine can be expressed using a nonlinear function and exploited in a scheduling algorithm to achieve the best savings. Therefore, finding other scenarios where an energy function can be used is the real challenge for future research.

Chapter 2

Scheduling with Resource States and Variable Energy Prices

2.1 Chapter Summary and Motivation

In Chapter 1, we successfully applied the concept of idle energy function in the scope of hardening furnace scheduling. This smart abstraction incorporating the optimal control applied to the furnace bilinear dynamics model was crucial for the design of the efficient scheduling algorithm. Albeit this application was successful, industrial furnace modeling and optimization are not widely studied in the research community.

In this chapter, we extend the ideas of the idle energy function to a different problem, this time combining two highly researched paradigms – the variable energy prices and a machine with several energy-saving states modeled by a finite state machine/automaton (FSM). As in the previous chapter, the goal is to schedule the workload together with machine states such that the overall energy consumption is minimized.

Previously, the situation was simple in the sense that the energy price was assumed to be constant, and so the optimal energy consumption of the furnace depended only on the idle period length Δ giving us the idle energy function $E : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ capturing the optimal energy consumption $E(\Delta)$ during any idle period of length Δ. Now, however, the situation is more complicated. The energy cost varies in time, and so the optimal energy consumption cost depends not only on the idle period length Δ but also on its start. Still, we can abstract the optimal behavior of the furnace in a similar manner, and derive an idle energy (cost) function, this time two-dimensional.

We show that not only the idle energy function extension is possible, but also it brings significant benefits in terms of computation speed when properly integrated within the optimization method. This time, we compare the state-of-the-art mathematical model based on integer linear programming (ILP) with the ILP model integrating the pre-computed optimal costs (idle energy cost function). This chapter is based on paper

- Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Constraints* 25.3–4 (Dec. 2020), pp. 300–318. ISSN: 1383-7133. DOI: `10.1007/s10601-020-09317-y`

published in Constraints journal. Also, the contribution was presented at the CPAIOR 2020 conference, where it was awarded the Best Student Paper award.

2.2 Introduction

Energy-efficient scheduling has been attracting a considerable amount of attention lately, as reported by both Gahm et al. [51] and Gao et al. [52]. The trend is most likely to continue in the future since the energy-efficient scheduling helps to achieve sustainability of the production by both decreasing the production cost and minimizing its environmental impact. Gahm et al. [51] identified promising approaches to energy-aware scheduling, including, among others, (i) the optimization of the energy demand by considering the power-saving states of the machines, and (ii) the participation in demand response programs, which are used by the electric utilities to reward the energy consumers for shifting their energy consumption to *off-peak* intervals [92].

In this chapter, we study a single machine scheduling problem to minimize the total energy cost (TEC) of the production. We consider both the power-saving states of the machine and the time-of-use (TOU) pricing. The TOU pricing is one of the demand response programs in which the electricity price may differ every hour. The scheduling problems with TOU pricing have been extensively addressed in the literature [48, 57, 60].

Considering the power-saving states of the machine, Mouzon et al. [97] identified that a significant energy cost reduction could be attained. However, the switchings between the machine states need to be planned carefully because of their non-negligible energy costs and transition times.

The integration of the power-saving states and the TOU pricing was initially proposed by Shrouf et al. [121], who designed an integer linear programming (ILP) model for the single machine problem with the fixed order of the jobs. However, it was proven by Aghelinejad et al. [3] that the problem with the fixed order of the jobs can be solved in polynomial time. Aghelinejad et al. [4] improved and generalized the existing ILP model by removing the restriction on the fixed order of the jobs, in which case the problem is $\mathcal{NP}$-hard [3]. However, in both works of Aghelinejad et al. [4] and Shrouf et al. [121], only small instances of the problem have been solved optimally.

In this chapter, we study a single machine problem introduced by Shrouf et al. [121] and further studied by Aghelinejad et al. [4], and describe a novel pre-processing technique to solve it efficiently. Our pre-processing technique pre-computes the optimal switching behavior in time w.r.t. the energy costs. The pre-computed costs of the optimal switchings allow us to design exact ILP and constraint programming (CP) models. In contrast, the ILP model proposed by Aghelinejad et al. [4] explicitly formulates the transition behavior of the machine, which needs to be optimized jointly with the scheduling of the jobs. As shown by the experiments, our approach outperforms the existing ILP model [4], which is, to the best of our knowledge, the state-of-the-art among the exact methods for this problem. Our ILP model can solve all benchmark instances with up to 190 jobs and 1277 pricing intervals within the time limit. On the other hand, the state-of-the-art ILP model from the literature scales only up to instances with 60 jobs and 316 intervals. This shows that our approach can tackle production-size problems. For example, creating a schedule of one workweek (5 days) with start times granularity of 5 minutes requires 1440 intervals.

2.3 Problem Statement

Let $\mathcal{I} = \{I_1, I_2, \dots, I_h\}$ be a set of *intervals*, which partition the scheduling horizon. The *energy costs* for the intervals are given by the vector $\boldsymbol{c} = (c_1, c_2, \dots, c_h)$, where $c_i \in \mathbb{Z}_{\geq 0}$ is the energy (electricity) cost associated with interval I_i. It is assumed, that every interval is one time unit long, i.e., $I_1 = [0, 1)$, $I_2 = [1, 2)$, …, $I_h = [h-1, h)$. Note that the physical representation of the time unit length can be different depending on the required granularity of the scheduling horizon.

Let $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$ be a set of *jobs*, which must be scheduled on a single machine that is available throughout the whole scheduling horizon; we assume that $n \geq 1$. Each job J_j is characterized by its *processing time* $p_j \in \mathbb{Z}_{>0}$, given in the number of intervals. Scheduling of the jobs is non-preemptive, and the machine can process at most one job at a time. All the jobs are available at the beginning of the scheduling horizon.

During each interval, the machine is operating in one of its *states* $s \in \mathcal{S}$ or transits from one state to another. Let us denote the *transition time function* by $T : \mathcal{S} \times \mathcal{S} \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$, and the *transition power function* by $P : \mathcal{S} \times \mathcal{S} \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$. The direct transition from state s to state $s' \neq s$ lasts $T(s, s')$ intervals and has power consumption $P(s, s')$, which is the constant rate of the consumed energy at every time unit. The value ∞ means that the direct transition does not exist. For ease of notation, we set $T(s, s) = 1$ for each $s \in \mathcal{S}$, by which we represent that the machine is remaining in the state s for the duration of one interval. Correspondingly, we assume that $P(s, s)$ denotes the power consumption of the machine while staying in state s for the duration of one interval.

Note that the transition time/power functions are general enough to represent many kinds of machines used throughout the literature [4, 19, 97, 121]. Often, the machine states and transitions are represented by a *transition graph* instead of the transition time/power function. Then, the graph nodes correspond to the machine states, while the edges represent the allowed direct transitions between the states. The edges are labeled by the corresponding values of the transition time/power functions. An example of a transition graph is shown in Fig. 2.1.

During the first and the last interval, the machine is assumed to be in off state `off` $\in \mathcal{S}$. Besides, the machine has a single processing state, `proc` $\in \mathcal{S}$, which must be active during the processing of the jobs. Due to the transition from/to the initial/last `off` state, the machine cannot be in `proc` state during the early/late intervals. Hence, we denote the *earliest* and the *latest* interval during which the machine can be in `proc` state by I_{earl} and I_{late}, respectively.

A *solution* is a pair $(\boldsymbol{\sigma}, \boldsymbol{\Omega})$, where $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots, \sigma_n) \in \mathbb{Z}^n_{\geq 0}$ is the vector denoting the start time of the jobs, and $\boldsymbol{\Omega} = (\Omega_1, \Omega_2, \dots, \Omega_h) \in (\mathcal{S} \times \mathcal{S})^h$ represents the active state or transition in each interval. The solution is feasible if the following four conditions are satisfied.

1. the machine processes at most one job at a time;

2. the jobs are processed when the machine is in `proc` state, i.e.,
   $\forall J_j \in \mathcal{J}\ \forall i \in \{\sigma_j + 1, \dots, \sigma_j + p_j\} : \Omega_i = (\texttt{proc}, \texttt{proc})$,

   where $\{a, \dots, b\}$ is $\{a, a+1, \dots, b\}$;

3. the machine is in `off` state during the first and the last interval, i.e., $\Omega_1 = (\texttt{off}, \texttt{off})$, and $\Omega_h = (\texttt{off}, \texttt{off})$;
4. all transitions are valid with respect to the transition time function, i.e., $\forall i \in \{1, 2, \ldots, h-1\}$ such that $\Omega_i = (s, s')$, $\Omega_{i+1} = (s'', s''')$, it holds that
   - (a) Ω_i and Ω_{i+1} encode only feasible states/transitions: $0 < T(s, s') < \infty$, $0 < T(s'', s''') < \infty$,
   - (b) only feasible zero-time transitions are allowed between Ω_i and Ω_{i+1}: either $s' = s''$ or there exists a sequence of states $(s', s_1, \ldots, s_k, s'')$ such that $T(s', s_1) = T(s_1, s_2) = \cdots = T(s_{k-1}, s_k) = T(s_k, s'') = 0$,
   - (c) the non-zero-time transitions respect the transition time function: if $s'' \neq s'''$ and $\Omega_i \neq \Omega_{i+1}$ then there exists Ω_ℓ with ℓ being the smallest index larger than $(i+1)$ such that $\Omega_\ell \neq \Omega_{i+1}$, and it holds that $\ell - i - 1 = T(s'', s''')$.

The total energy cost (TEC) of solution $(\boldsymbol{\sigma}, \boldsymbol{\Omega})$ is

$$\sum_{I_i \in \mathcal{I}} c_i \cdot P(\Omega_i), \tag{2.1}$$

where $P(\Omega_i)$ represents $P(s, s')$ for $\Omega_i = (s, s')$. The goal of the scheduling problem is to find a feasible solution minimizing the total energy cost (2.1).

The above-defined problem was introduced by Shrouf et al. [121] and is denoted in standard Graham's notation as $1, \mathrm{TOU} | \, \mathrm{states} \, | \, \mathrm{TEC}$. The problem was shown to be $\mathcal{NP}$-hard [3].

Example: *Here, we present a small example to illustrate the proposed notation. Let us consider a scheduling horizon consisting of 16 intervals,* $\mathcal{I} = \{I_1, \ldots, I_{16}\}$, *and the associated energy costs* $\boldsymbol{c} = (2, 1, 2, 1, 6, 16, 14, 3, 2, 5, 3, 15, 3, 2, 1, 2)$. *Let us have three jobs,* $\mathcal{J} = \{J_1, J_2, J_3\}$ *with processing times* $p_1 = 2$, $p_2 = 1$, *and* $p_3 = 2$. *Considering the machine states, we assume* $\mathcal{S} = \{\texttt{proc}, \texttt{off}, \texttt{idle}\}$. *The values of the transition time function and the transition power function are given in Fig. 2.1. For the given transition time function, we have* $I_{\mathrm{earl}} = I_4$ *and* $I_{\mathrm{late}} = I_{14}$. *Note that the same machine states and transitions were originally proposed by Shrouf et al. [121].*

The optimal solution to the given instance is depicted in Fig. 2.2, where

$$\begin{aligned}
\boldsymbol{\sigma} &= (9, 3, 12), \ \textit{and} \\
\boldsymbol{\Omega} &= ((\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{proc}), (\texttt{off}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{off}), (\texttt{off}, \texttt{off}), \\
&\quad (\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{proc}), (\texttt{off}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{idle}, \texttt{idle}), \\
&\quad (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{off}), (\texttt{off}, \texttt{off})).
\end{aligned}$$

The TEC of this optimal solution is equal to 133.

2.4 Solution Approach

In this section, we first describe how to pre-compute the optimal switching behavior of the machine and the corresponding costs. Afterward, we design efficient ILP

$s \backslash s'$	`proc`	`idle`	`off`
`proc`	4	0	1
`idle`	0	2	∞
`off`	5	∞	0

$P(s, s')$

$s \backslash s'$	`proc`	`idle`	`off`
`proc`	1	0	1
`idle`	0	1	∞
`off`	2	∞	1

$T(s, s')$

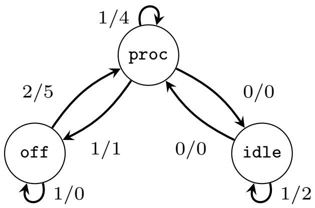
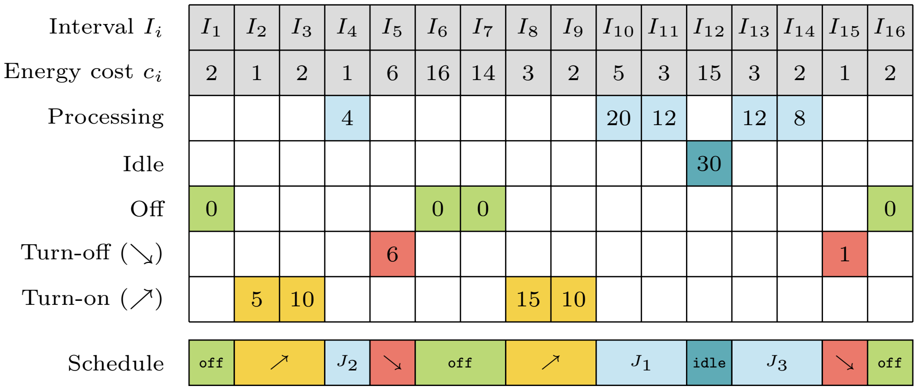


Figure 2.1: Parameters of the transition power function $P(s, s')$ and the transition time function $T(s, s')$, and the corresponding transition graph, where every edge from s to s' is labeled by $T(s, s')/P(s, s')$.



Interval I_i	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
Energy cost c_i	2	1	2	1	6	16	14	3	2	5	3	15	3	2	1	2
Processing				4						20	12		12	8		
Idle												30				
Off	0					0	0									0
Turn-off (↘)					6										1	
Turn-on (↗)		5	10					15	10							
Schedule	`off`	↗		J_2	↘	`off`		↗		J_1		`idle`	J_3		↘	`off`

Figure 2.2: The optimal schedule for the example instance. Each cell, corresponding to interval I_i and a state/transition, contains the value $c_i \cdot P(\Omega_i)$. The sum over all these values gives the TEC equal to 133.

(called ILP-SPACES) and CP models that integrate the pre-computed optimal switching costs.

2.4.1 Instance Pre-processing: Computation of the Optimal Switching

Given two states s, s' in which the machine is during two intervals $I_i, I_{i'}$ such that $i < i'$, the pre-processing computes the optimal transitions from (s, I_i) to $(s', I_{i'})$ over all possible states w.r.t. the energy cost. Formally, the pre-processing solves the following optimization problem

$$\min_{\Omega_{i+1}, \Omega_{i+2}, \ldots, \Omega_{i'-1}} \sum_{i''=i+1}^{i'-1} c_{i''} \cdot P(\Omega_{i''}). \tag{2.2}$$

such that $((s, s), \Omega_{i+1}, \Omega_{i+2}, \ldots, \Omega_{i'-1}, (s', s'))$ are valid transitions w.r.t. to the transition time function. We call this an *optimal switching problem.* As an illustration, the cost of the optimal switching in Fig. 2.2 from $(\texttt{proc}, I_4)$ to $(\texttt{proc}, I_{10})$ equals to 31. Interestingly, the optimal switching problem can be solved in polynomial time by finding the shortest path in an *interval-state* graph, which is explained in the rest of this section.

The interval-state graph is defined by a triplet (V, E, w), where V is the set of *vertices*, E is the set of *edges* and $w : E \to \mathbb{Z}_{\geq 0}$ are the *weights* of the edges. The set of the vertices and edges of this graph are defined as follows:

$$V = \{v_{1,\texttt{off}}\} \cup \{v_{i,s} : I_i \in \mathcal{I} \setminus \{I_1\}, s \in \mathcal{S}\} \cup \{v_{h+1,\texttt{off}}\}, \tag{2.3}$$

$$\begin{aligned} E = &\{(v_{1,\texttt{off}}, v_{2,\texttt{off}})\} \\ &\cup \{(v_{i,s}, v_{i+T(s,s'),s'}) : s, s' \in \mathcal{S}, I_i \in \mathcal{I} \setminus \{I_1\}, \\ &\qquad\qquad T(s,s') \neq \infty, (i-1) + T(s,s') \leq h-1\} \\ &\cup \{(v_{h,\texttt{off}}, v_{h+1,\texttt{off}})\}\,. \end{aligned} \tag{2.4}$$

Informally, each vertex $v_{i,s} \in V$ represents that at the beginning of interval I_i the machine is in state s. Each edge $(v_{i,s}, v_{i',s'}) \in E$ corresponds to the direct transition from state s to state s' that lasts $T(s,s') = (i' - i)$ intervals. The condition $(i-1) + T(s,s') \leq h-1$ ensures, that only transitions completing at most at the beginning of interval I_h are present in the interval-state graph.

The edges are weighted by the total energy cost of the corresponding transition w.r.t. the costs of energy in intervals, i.e., *weight* of edge $(v_{i,s}, v_{i',s'}) \in E$ is defined as

$$w(v_{i,s}, v_{i',s'}) = \sum_{i''=i}^{i'-1} c_{i''} \cdot P(s,s')\,. \tag{2.5}$$

Note that by the definition, the interval-state graph encodes all the feasible transitions between the machine states in time.

Returning to the optimal switching problem (2.2), the optimal transitions from (s, I_i) to $(s', I_{i'})$ w.r.t. the energy cost can be obtained by finding the shortest path from $v_{i+1,s}$ to $v_{i',s'}$ in the interval-state graph. We denote the cost of the optimal switching by function $l : V \times V \to \mathbb{Z}_{\geq 0}$.

Example (continued): *Continuing with the Example, Fig. 2.3 shows the whole interval-state graph for the given instance. The green dashed path shows the optimal transition of the machine assuming that the machine is in* `proc` *state during intervals* I_4 *and* I_{10}*; at first the machine is turned off (during* I_5*), then it remains off (during intervals* I_6 *and* I_7*), and is turned on afterward (intervals* I_8*,* I_9*). In this case,* $l(v_{5,\text{proc}}, v_{10,\text{proc}}) = 31$.

The values of l can be computed using the Floyd-Warshall algorithm in $\mathcal{O}(h^3 \cdot |\mathcal{S}|^3)$ time. However, for the scheduling decisions, only some of the switchings are interesting. Since all the jobs need to be scheduled in the `proc` state, the optimal switchings have to be resolved only in the ‘space’, i.e., the sequence of intervals: (i) between the two consecutive intervals with `proc`; (ii) between the first `off` and the first `proc`; and (iii) the last `proc` and the last `off`. The cost of the switchings between $s, s' \in \{\texttt{off}, \texttt{proc}\}^2$ are recorded by function $c^\star : \mathcal{I}^2 \to \mathbb{Z}_{\geq 0}$ defined as

$$c^\star(i, i') = \begin{cases} l(v_{i+1,\texttt{proc}}, v_{i',\texttt{proc}}) & i > 1, i' < h \qquad \text{case (i)} \\ l(v_{2,\texttt{off}}, v_{i',\texttt{proc}}) & i = 1, i' < h \qquad \text{case (ii)} \\ l(v_{i+1,\texttt{proc}}, v_{h,\texttt{off}}) & i > 1, i' = h \qquad \text{case (iii)} \end{cases} \tag{2.6}$$

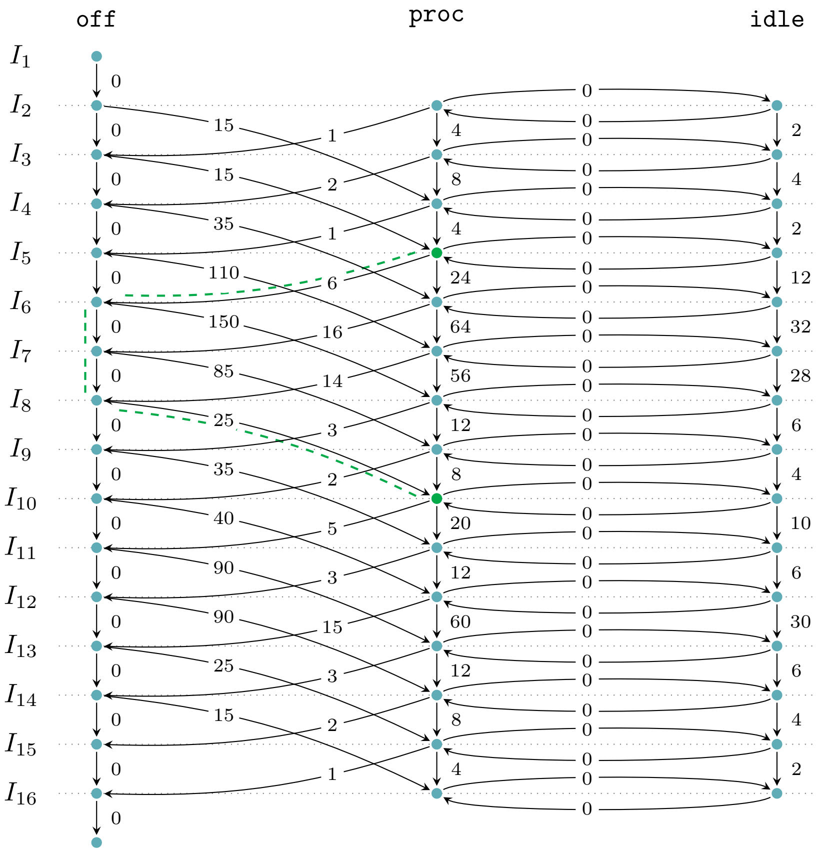


Figure 2.3: Interval-state graph for the Example instance from Section 2.3 with highlighted optimal switching behavior from $(\texttt{proc}, I_4)$ to $(\texttt{proc}, I_{10})$.

for each $i < i'$. The vector of states corresponding to $c^\star(i, i')$, i.e., the *optimal switching behavior* of the machine between i and i', is denoted by $\boldsymbol{\Omega}^\star(i, i')$. As an example, see the Fig. 2.2, where intervals $\{I_5, I_6, \ldots, I_9\}$ represent the space between two consecutive jobs J_2, J_1 with cost $c^\star(4, 10) = l(v_{5,\texttt{proc}}, v_{10,\texttt{proc}}) = 31$. The optimal switching behavior is

$$\boldsymbol{\Omega}^\star(4, 10) = ((\texttt{proc}, \texttt{off}), (\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{proc}), (\texttt{off}, \texttt{proc})), \quad (2.7)$$

which is depicted by the green dashed path in Fig. 2.3.

Values of $c^\star$ can be computed efficiently using an algorithm that we call the *Shortest Path Algorithm for Cost Efficient Switchings* (SPACES). In every iteration $I_i \in \mathcal{I} \setminus \{I_h\}$, SPACES computes all values $c^\star(i, i+1), c^\star(i, i+2), \ldots, c^\star(i, h)$ by finding the shortest paths from $v_{i+1,\texttt{proc}}$ (or $v_{2,\texttt{off}}$ if $i = 1$) to all other vertices in the interval-state graph. The shortest paths are obtained with Dijkstra algorithm that runs in $\mathcal{O}(|E| + |V| \cdot \log |V|)$ if implemented using the priority queues. Since the Dijkstra algorithm is started h times, the complexity of SPACES is

$$\mathcal{O}(h \cdot (|E| + |V| \cdot \log |V|)) = \mathcal{O}(h^2 \cdot |\mathcal{S}| \cdot (|\mathcal{S}| + \log h \cdot |\mathcal{S}|))\,. \tag{2.8}$$

To increase the performance further, iterations i can be computed in parallel since they are independent of each other. Moreover, the shortest paths between states could be cached, and might be re-used between the iterations of the algorithm. However, for the size of the benchmark instances considered in the experiments, the runtime of the SPACES without caching is negligible in comparison to the total solving time. Hence, we did not implement SPACES with the shortest paths' caching.

2.4.2 Integer Linear Programming Model ILP-SPACES

In the ILP model proposed by Aghelinejad et al. [4], the state transition functions are explicitly encoded. In contrast, our ILP-SPACES model works only with the optimal switching costs pre-computed by the SPACES algorithm, thus encoding the transitions implicitly without sacrificing the optimality. The only task of the ILP solver is then to schedule the jobs, and select appropriate spaces in between, such that the TEC is minimized. Thus, the structure of our model is greatly simplified, with positive impact on its performance.

Formally, the variables used in the ILP-SPACES model are

- job start time $s_{j,i} \in \{0,1\}$: equals 1 if job J_j starts at the beginning of interval I_i, otherwise 0;

- space activation $x_{i,i'} \in \{0,1\}$: equals 1 if the machine undergoes the optimal switching defined by $\mathbf{\Omega}^\star(i,i')$, otherwise 0.

The complete model follows.

$$\min \sum_{\substack{I_i, I_{i'} \in \mathcal{I} \\ i < i'}} x_{i,i'} \cdot c^\star(i,i') + \sum_{\substack{J_j \in \mathcal{J} \\ I_i \in \mathcal{I}}} s_{j,i} \cdot c^{(\text{job})}_{j,i}, \tag{2.9}$$

$$\sum_{I_i \in \mathcal{I}} s_{j,i} = 1,\ \forall J_j \in \mathcal{J}, \tag{2.10}$$

$$s_{j,i} = 0,\ \forall J_j \in \mathcal{J}, \forall i \in \{1, \ldots, \text{earl} - 1\} \cup \{\text{late} - p_j + 2, \ldots, h\}, \tag{2.11}$$

$$\sum_{J_j \in \mathcal{J}} \sum_{i'=\max\{2, i-p_j+1\}}^{i} s_{j,i'} + \sum_{i'=1}^{i-1} \sum_{i''=i+1}^{h} x_{i',i''} = 1,\ \forall I_i \in \{I_2, I_3, \ldots, I_{h-1}\}. \tag{2.12}$$

The objective (2.9) minimizes the total energy cost, consisting of the optimal switching cost of the active spaces, and the cost of the jobs processing, where

$$c^{(\text{job})}_{j,i} = \sum_{i'=i}^{i+p_j-1} c_{i'} \cdot P(\texttt{proc}, \texttt{proc}) \tag{2.13}$$

for job $J_j \in \mathcal{J}$ and $i \in \{\text{earl}, \ldots, \text{late} - p_j + 1\}$.

Constraint (2.10) forces every job to be scheduled exactly once, and constraint (2.11) forbids the job to be scheduled before I_{earl} and after I_{late}. Finally, the last constraints (2.12) force the machine to be processing a job or to be undergoing some transition during every interval and forbid overlaps between them.

2.4.2.1 Search Space Reduction

Various methods can be employed to reduce the search space without sacrificing the optimality. One of such methods is pruning of the spaces variables that lead to infeasible solutions if activated.

The pruning works as follows. For each $I_i, I_{i'}$ such that $i < i'$, the available time for processing the jobs is computed for both left (before I_i) and right (after $I_{i'}$) part of the scheduling horizon, i.e., $i - \text{earl} + 1$ and $\text{late} - i' + 1$, respectively. Then, activating the switching behavior $\mathbf{\Omega}^{\star}(i, i')$ leads to an infeasible solution if one of the following *pruning conditions* holds

PC.1: The largest job can be fitted in neither part, i.e.,

$$\max_{J_j \in \mathcal{J}} p_j > i - \text{earl} + 1 \quad \wedge \quad \max_{J_j \in \mathcal{J}} p_j > \text{late} - i' + 1\,. \tag{2.14}$$

PC.2: The total available time for processing is less than the sum of all the processing times, i.e.,

$$(i - \text{earl} + 1) + (\text{late} - i' + 1) < \sum_{J_j \in \mathcal{J}} p_j\,. \tag{2.15}$$

If any of these conditions holds, the corresponding space variable $x_{i,i'}$ is not created in ILP-SPACES.

2.4.3 Constraint Programming Models

Thanks to the expressiveness of the CP, there are multiple possibilities on how to model our scheduling problem. Since the performance of each model is not easily predictable beforehand, we decided to try different combinations of the jobs' and spaces' modeling.

In the end, we selected the best model by performing a preliminary experiment, see Section 2.5.2. The best model is called Element-Free-SumLengths and is denoted as CP-SPACES.

In the following, we describe the CP-SPACES model formally. High-level description of all the tested CP models is in Section 2.4.3.3. Also, all the source codes are publicly available at `https://github.com/CTU-IIG/EnergyStatesAndCostsScheduling`. In the text, we use the IBM CP formalism [75] for describing the models.

2.4.3.1 CP-SPACES Model

The idea of the CP-SPACES model is similar to the ILP-SPACES, with the exception that the spaces are not fixed – they are allowed to 'float' within the scheduling horizon. In consequence, the spaces do not have fixed costs because the cost depends on the position of the space in the horizon and its length. In our CP-SPACES model, costs are formulated with an *Element* expression, which is integrated in the objective.

Variables. Two types of interval variables are used in the CP-SPACES model.

To represent each job $J_j \in \mathcal{J}$ and the intervals in which the job is allocated, we use interval variable z_j of fixed length p_j. This interval variable represents the time interval in which the corresponding job is processed, i.e, the job starts at time $\mathrm{StartOf}(z_j)$ and completes at time $\mathrm{EndOf}(z_j)$.

The spaces in the schedule are modeled by using the optional interval variables, where $x_{\ell,k}$ represents the 'floating' spaces of fixed length ℓ. For each possible *length* $\ell \in \{1, 2, \dots, h - 2 - \sum_{J_j \in \mathcal{J}} p_j\}$, we create $K(\ell) = \left\lfloor \frac{h-2-\sum_{J_j \in \mathcal{J}} p_j}{\ell} \right\rfloor$ variables that are indexed by $k \in \{1, 2 \dots, K(\ell)\}$. Note that the number $K(\ell)$ gives the upper bound on the number of the spaces of length ℓ that may appear in a feasible schedule, while $\ell_{\max} = h - 2 - \sum_{J_j \in \mathcal{J}} p_j$ gives an upper bound on the space length.

Constraints. Since the machine is assumed to be in `off` state during I_1 and I_h, the earliest and the latest interval during which a switching might occur is I_2 and I_{h-1}, respectively. Hence, starts (ends) of the spaces are restricted by

$$\left.\begin{aligned} \mathrm{StartOf}(x_{\ell,k}) &\geq 1 \\ \mathrm{EndOf}(x_{\ell,k}) &\leq h - 1 \end{aligned}\right\} \forall \ell \in \{1, \dots, \ell_{\max}\}, k \in \{1, \dots, K(\ell)\}. \tag{2.16}$$

As mentioned previously, the spaces have fixed lengths, i.e.,

$$\mathrm{LengthOf}(x_{\ell,k}) = \ell, \ \forall \ell \in \{1, \dots, \ell_{\max}\}, k \in \{1, \dots, K(\ell)\}. \tag{2.17}$$

To ensure that the jobs and the spaces are not overlapping, we use the *NoOverlap* constraint,

$$\mathrm{NoOverlap}(\{x_{\ell,k} : \ell \in \{1, \dots, \ell_{\max}\}, k \in \{1, \dots, K(\ell)\}\} \cup \{z_j : J_j \in \mathcal{J}\}). \tag{2.18}$$

The lengths of the spaces are constrained by

$$\sum_{\ell=1}^{\ell_{\max}} \sum_{k=1}^{K(\ell)} \mathrm{LengthOf}(x_{\ell,k}) = \ell_{\max}, \tag{2.19}$$

to ensure that the whole scheduling horizon is filled.

Objective. The objective is to minimize the TEC, here expressed as

$$\begin{aligned} &\sum_{\ell=1}^{\ell_{\max}} \sum_{k=1}^{K(\ell)} \mathrm{Element}(\boldsymbol{c}_\ell^{(\mathrm{space})}, \mathrm{StartOf}(x_{\ell,k})) + \\ &\quad + \sum_{J_j \in \mathcal{J}} \mathrm{Element}(\boldsymbol{c}_j^{(\mathrm{job})}, \mathrm{StartOf}(z_j) + 1), \end{aligned} \tag{2.20}$$

where the first part corresponds to the cost for optimal switchings between the job processings, and the second part corresponds to the cost for job processing. To compute the cost of the present spaces, vector

$$\boldsymbol{c}_\ell^{(\mathrm{space})} = (c^\star(1, 1 + \ell + 1), c^\star(2, 2 + \ell + 1), \dots, c^\star(h - \ell - 1, h)) \tag{2.21}$$

is used to represent the optimal switching costs for the given length ℓ addressed by the start of space $x_{\ell,k}$ (indexed from 1). Similarly, to compute the cost of the jobs, vector

$$\boldsymbol{c}_j^{(\mathrm{job})} = (c_{j,1}^{(\mathrm{job})}, c_{j,2}^{(\mathrm{job})}, \ldots, c_{j,h}^{(\mathrm{job})}) \tag{2.22}$$

is used.

2.4.3.2 Interval-state Graph as a Global Constraint

Note that a structure like the interval-state graph could be used even for the filtering of the variables domains, and could be embedded to a global constraint in CP. Imagine having variables $x_1, \ldots, x_h$ with domains $D_1, \ldots, D_h$, representing the states of the machine in each interval. Then whenever a value is filtered from a domain, we could remove the edges in the interval-state graph that are leading to the state corresponding to the removed value. Afterward, by forward and backward search in the graph, we could make the other domains consistent. This is similar to the filtering techniques used for the grammar constraints [109, 71] or knapsack constraints [128]. We believe that efficient global propagation based on the interval-state graph for this unrolled transition diagram can be designed. However, its formal derivation is beyond the scope of this chapter.

2.4.3.3 Evaluated CP models

In this section, we describe the variations of the CP-SPACES model, which were implemented and evaluated in the preliminary experiment. The description is divided into three parts, namely the *modeling of the jobs*, the *modeling of the spaces*, and the *linking constraints.*

Modeling of the jobs: All the models contain an interval variable z_j with a fixed length of p_j for each job $J_j \in \mathcal{J}$. This interval variable represents the time interval in which the corresponding job is processed, i.e, the job starts at time $\mathrm{StartOf}(z_j)$ and completes at time $\mathrm{EndOf}(z_j)$. The models for the jobs differ primarily in how the cost of scheduling the jobs is formulated in the objective.

- OPTIONAL: For each job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$, create an *optional* interval variable $z_{j,i}$ having fixed length p_j and fixed start meaning that the job J_j starts to be processed at the beginning of interval I_i. By enforcing $\mathrm{Alternative}(z_j, \{z_{j,i} : \forall I_i \in \mathcal{I}\})$ on every job $J_j \in \mathcal{J}$, we constraint that only one such variable will be present in the schedule. Then, every pair of job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$ adds term $\mathrm{PresenceOf}(z_{j,i}) \cdot c_{j,i}^{(\mathrm{job})}$ into the objective.
- LOGICAL: Every pair of job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$ adds term $(\mathrm{StartOf}(z_j) = i - 1) \cdot c_{j,i}^{(\mathrm{job})}$ into the objective, i.e., if J_j starts at the beginning of I_i, the contribution of J_j into objective is $c_{j,i}^{(\mathrm{job})}$.
- ELEMENT: Each $J_j \in \mathcal{J}$ adds term $c_{j,\mathrm{StartOf}(z_j)+1}^{(\mathrm{job})}$ into the objective. The indexing by a variable can be done using Element expression.
- OVERLAP: Every pair of job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$ adds term $\mathrm{Overlap}(z_j, I_i) \cdot c_i \cdot P(\texttt{proc}, \texttt{proc})$ into the objective.

- STEPFN: For every unique processing time $p \in \{p_j : J_j \in \mathcal{J}\}$, create a *step function* F_p representing the cost of starting a job with processing time p at the beginning of an interval. Each job $J_j \in \mathcal{J}$ adds term $\mathrm{StartEval}(\mathrm{StartOf}(z_j), F_{p_j})$ into the objective.

Modeling of the spaces: Similarly as with the jobs, the modeling techniques for the spaces differ in how they contribute into the objective.

- FIXED: For each pair of intervals $I_i, I_{i'} \in \mathcal{I}$ such that $i < i'$, create an *optional* interval variable $x_{i,i'}$ having fixed start to i and having fixed end to $i' - 1$. Each such pair of intervals adds term $\mathrm{PresenceOf}(x_{i,i'}) \cdot c^\star(i, i')$ into the objective.
- FREE: For each possible space length $\ell \in \{1, \ldots, h\}$, we create $K(\ell) = \left\lfloor \frac{h}{\ell} \right\rfloor$ optional interval variables $x_{\ell,k}$ that are indexed by $k \in \{1, 2 \ldots, K(\ell)\}$ and have fixed length ℓ. Each pair of length $\ell \in \{1, \ldots, h\}$ and $k \in \{1, 2 \ldots, K(\ell)\}$ adds term $c^\star(\mathrm{StartOf}(x_{\ell,k}), \mathrm{EndOf}(x_{\ell,k}) + 1)$ into the objective.

  The difference between FIXED and FREE is that in FREE the start times of the space variables are not fixed.
- NOVARS: In this case, the spaces are not modeled by variables at all. Instead, we create a sequence variable π over all jobs variables z_j. Each position $\ell \in \{1, \ldots, n-1\}$ in π adds term $c^\star(\mathrm{EndOf}(\pi_\ell), \mathrm{StartOf}(\pi_{\ell+1}) + 1)$ into the objective (for brevity of the description, the cost of the switching from the first `off` and to the last `off` is omitted).

Linking constraints: The formulations of the jobs and spaces must be linked together with a linking constraint ensuring that every time instant of the scheduling horizon is occupied by either a job or a space interval variable. Moreover, we use NoOverlap on the jobs and spaces variables so that they do not overlap each other. Note that since NOVARS does not use variables for modeling the spaces, the linking constraint is not necessary in this case.

- SUM: Here we simply constraint that the sum of the lengths of all present jobs and spaces variables equals to the length of the scheduling horizon.
- PULSE: For each job and space variable, we create a *pulse* function having *height* of 1. Then, all the pulse functions are summed together into a *cumul* expression, which is forced to be 1 in every time instant of the scheduling horizon.
- STARTOFNEXT: We create a sequence variable π over all jobs and spaces variables. Then we constraint that each variable on position π_ℓ ends at the start of variable $\pi_{\ell+1}$.

2.4.4 Symmetry Breaking Constraints

Symmetries in connection with the CP and ILP are widely studied [36, 54, 91]. Some of the symmetries in combinatorial problems arise when multiple feasible solutions with the same objective value, and which differ only in the values of the variables, correspond to the same "canonical" feasible solution. For example, the jobs in our scheduling problem are identical, thus replacing one job in a feasible solution with another one having the same processing time will not influence the objective value.

The symmetries might negatively affect the performance of the models due to enlarged search space. To break the symmetries without losing the optimality, *symmetry breaking constraints* are employed. In the CP models, we use the following symmetry breaking constraints.

1. Fixed ordering on the jobs having the same processing time: Let $J_j, J_{j'} \in \mathcal{J}$, such that $j < j'$, be two jobs having the same processing time. The symmetry breaking constraint is EndBeforeStart$(z_j, z_{j'})$.

   To reduce the size of the models, the constraint is not created for every pair of jobs; instead, the jobs having the same processing time are sorted by their indices and the constraint is created only for every pair of two consecutive jobs along this sorted sequence.

2. Fixed ordering on the spaces having the same length: Similarly as with breaking the symmetries on identical jobs, we can break symmetries on identical spaces for FREE space modeling. That is, let $\ell \in \{1, \ldots, h\}$ be a space length and let $k, k' \in \{1, 2 \ldots, K(\ell)\}$ such that $k < k'$. Then we add the following two constraints

$$\text{PresenceOf}(x_{\ell,k'}) \leq \text{PresenceOf}(x_{\ell,k}) \,, \tag{2.23}$$

$$\text{EndBeforeStart}(x_{\ell,k}, x_{\ell,k'}) \,. \tag{2.24}$$

We also tried to fix the ordering of the jobs with the same processing time in the ILP-SPACES model using either constraint (2.25a) or (2.25b)

$$\sum_{I_i \in \mathcal{I}} s_{j,i} \cdot i + p_j \leq \sum_{I_i \in \mathcal{I}} s_{j',i} \cdot i, \quad \forall J_j, J_{j'} \in \mathcal{J}, p_j = p_{j'}, j < j' \,, \tag{2.25a}$$

$$s_{j',i} \leq \sum_{I_{i'} \in \mathcal{I} : i' < i} s_{j,i'}, \quad \forall J_j, J_{j'} \in \mathcal{J}, p_j = p_{j'}, j < j', \forall I_i \in \mathcal{I} \,. \tag{2.25b}$$

However, the performance of the resulting model was inferior to the original model without the constraints. Thus, the symmetry breaking constraints are omitted for the ILP-SPACES.

2.5 Experiments

This section evaluates how ILP-SPACES and the CP models perform in comparison to the ILP-REF model proposed by Aghelinejad et al. [4]. The comparison is made on a set of randomly generated instances; see Section 2.5.1 for the description of the generated dataset. Due to the space constraints, we only compare the best CP model which is selected according to the results of the preliminary experiment; see Section 2.5.2. The final results are presented in Section 2.5.3.

All the experiments were executed on 2x Intel(R) Xeon(R) Silver 4110 CPU 2.10 GHz with 188 GB of RAM (16 cores in total). For solving the ILP and CP models, we used Gurobi 8 and IBM CP Optimizer 12.9, respectively. Except for the time-limit and the search phases in CP-SPACES, which branched on the jobs first, all the solver parameters were set to the default values.

The source codes and the experimental data (instances and solutions) are publicly available at `https://github.com/CTU-IIG/EnergyStatesAndCostsScheduling` and `https://github.com/CTU-IIG/EnergyStatesAndCostsSchedulingData`, respectively.

2.5.1 Instances

The instances in the dataset can be divided according to

1. a number of jobs:
   (a) MEDIUM: medium instances with $n \in \{30, 60, 90\}$;
   (b) LARGE: large instances with $n \in \{150, 170, 190\}$;
2. a machine transition graph:
   (a) NOSBY: a simple graph with no standby state [4, 121], see Fig. 2.1 for its description;
   (b) TWOSBY: a graph with two standby states shown in Fig. 2.4.

For fixed n and a machine transition graph, 12 random instances are generated in the following way (48 instances in the whole dataset). The processing times of the jobs are randomly sampled from discrete uniform distribution $\mathcal{U}\{1, 5\}$. The number of intervals in each instance is obtained as a multiple of the total processing time plus the required number of intervals to turn the machine on and off, where this multiple H^{mul} is taken from set $\{1.3, 1.6, 1.9, 2.2\}$. The energy cost in each interval is randomly sampled from $\mathcal{U}\{1, 10\}$. For instances differing only in the number of intervals, the energy costs are sampled gradually, i.e., the energy costs of all the intervals in an instance with a shorter horizon are the same as for the corresponding intervals in an instance with a longer horizon.

The dataset for the preliminary CP experiments is generated using a similar scheme as described above with the following differences: $n \in \{30, 60\}$, $H^{\text{mul}} = 1.2$ and NOSBY transition graph is used. For each fixed n, we generate 6 random instances. Thus, the dataset for the preliminary experiments has 12 instances in total. We denote this dataset as PRELIM.

Note that the distributions for sampling the processing times and the energy costs of the intervals for all the datasets are the same as proposed by Aghelinejad et al. [4] and Shrouf et al. [121].

2.5.2 Preliminary CP Experiments: Results for PRELIM

The purpose of the preliminary experiments is to compare different CP modeling strategies and to select the best-performing model that will be compared against ILP-SPACES and ILP-REF.

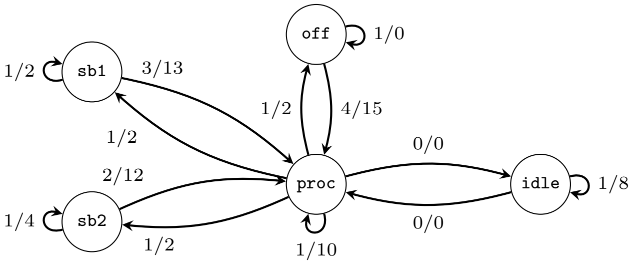


Figure 2.4: Example of a transition graph with multiple standby states; every edge from s to s' is labeled by $T(s,s')/P(s,s')$.

The results aggregated over the instances are presented in Table 2.1, where *gap* stands for the average optimality gap, *time* is the average runtime in seconds and *#best* represents the number of times the model achieved the best upper bound among all the tested models including the ILP-REF. The optimality gap on each instance is defined as

$$\frac{ub - lb^{\text{best}}}{ub} \cdot 100\ [\%]\,, \tag{2.26}$$

where lb^{best} is the best lower bound obtained over all models (including the ILP-REF) on that instance. The time limit is set to 600 s per instance.

For each instance of PRELIM dataset, none of the CP models is able to prove the optimality of the found solution in the given time-limit. In Table 2.1, it can be observed how the quality of the best found solution varies across the tested models. Based on the lowest achieved gap, we choose the model Element-Free-SumLengths for the following experiments, which we denote as CP-SPACES.

Since we are looking for the optimal solutions, we have also experimented with the settings of `FailureDirectedSearchEmphasis` parameter for the CP-SPACES model. We run one additional experiment with CP-SPACES model on PRELIM dataset, but with `FailureDirectedSearchEmphasis` set to 16, i.e., 16 threads of the CP solver were dedicated to failure-directed search. However, we were not able to obtain better lower bounds or upper bounds within the given time-limit, compared with the default setting of the parameter. Hence, the default setting of `FailureDirectedSearchEmphasis` is used for the rest of the experiments.

2.5.3 Results for MEDIUM and LARGE Instances with Different Machine Transition Graphs

Both the presented Tables 2.2, 2.3 have the same structure: each row represents one instance characterized by the number of jobs n and the number of intervals h. The objective value ub of the best found feasible solution, lower bound lb and the running time t are reported for each tested model. If the objective value or the lower bound is in bold font, the corresponding value is known to be optimal. Therefore, if both objective and the lower bound are in bold, the solver was able to prove the solution optimality within the time-limit. If the solver reached its given

Table 2.1: Comparison of different CP modeling techniques on PRELIM dataset.

Model	gap [%]	time [s]	#best [-]
Element-Free-SumLengths	0.15	600.0	6
Optional-Free-Pulse	0.18	600.0	6
StepFunction-Free-SumLengths	0.22	600.0	6
Optional-Free-StartOfNext	0.25	600.0	4
Element-Free-StartOfNext	0.26	600.0	5
StepFunction-Free-Pulse	0.28	600.0	5
Element-Free-Pulse	0.33	600.0	6
Optional-No	0.33	600.0	5
Optional-Free-SumLengths	0.35	600.0	4
Logical-Free-SumLengths	0.37	600.0	6
Logical-Fixed-Pulse	0.42	600.0	4
Logical-Free-StartOfNext	0.43	600.0	5
StepFunction-Free-StartOfNext	0.43	600.0	4
Logical-Free-Pulse	0.47	600.0	4
Element-Fixed-Pulse	0.52	600.0	4
Overlap-Free-SumLengths	0.54	600.0	3
Element-Fixed-SumLengths	0.59	600.0	4
Overlap-Free-Pulse	0.61	600.0	4
Logical-Fixed-SumLengths	0.65	600.0	2
StepFunction-Fixed-Pulse	0.70	600.0	4
StepFunction-Fixed-SumLengths	0.73	600.0	3
Overlap-Free-StartOfNext	0.82	600.0	2
Optional-Fixed-SumLengths	0.82	600.0	3
Optional-Fixed-Pulse	0.83	600.0	3
StepFunction-Fixed-StartOfNext	0.98	600.0	4
Optional-Fixed-StartOfNext	0.98	600.0	4
StepFunction-No	1.02	600.0	2
Overlap-Fixed-Pulse	1.06	600.0	2
Overlap-Fixed-SumLengths	1.12	600.0	2
Overlap-Fixed-StartOfNext	1.17	600.0	3
Element-Fixed-StartOfNext	1.19	600.0	3
Logical-Fixed-StartOfNext	1.22	600.0	3
Overlap-No	1.66	600.0	2
Element-No	1.69	600.0	1
Logical-No	2.24	600.0	0
ILP-SPACES	0.00	4.6	12

time-limit on an instance without proving the optimality of a solution, the value in the corresponding cell in t column is TLR.

The last rows in each table show the average running time of each model and the average optimality gap, defined by (2.26). The average time is computed over all instances; if the solver timed-out on some instance, the specified time-limit is taken as the running time on that instance.

Additionally, we report the pre-processing time P-P for the large instances. For the medium-size instances, the comparison between the pre-processing time and the solving time of ILP-SPACES is shown in Fig. 2.5. The pre-processing takes in average only 2.8 % of the the total solving time (pre-processing plus the solving of the ILP-SPACES model) for NOSBY and 1.8 % for TWOSBY. Overall medium-size instances, the average and maximum pre-processing times were 0.69 s and 2.93 s, respectively.

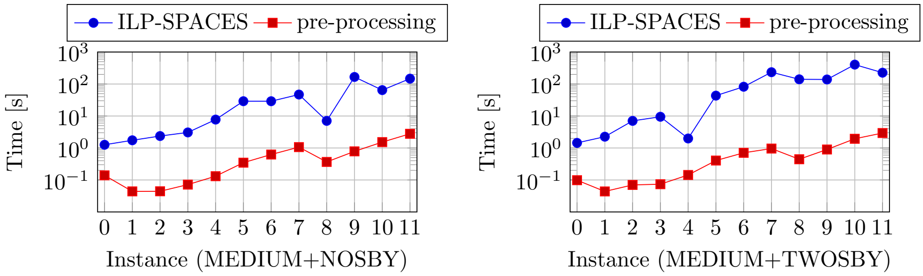


Figure 2.5: Comparison between the running times of the pre-processing and ILP-SPACES model for MEDIUM dataset.

2.5.3.1 Results for Medium Instances

The results of this experiment are shown in Table 2.2. In this table we can see that ILP-SPACES solves all the instances within the time-limit (600 s). On the other hand, the model ILP-REF proposed by Aghelinejad et al. [4] finds the optimal solution and proves the optimality only for 11 instances out of 24 within the time-limit. Moreover, some of the non-optimal solutions found by ILP-REF are far from the optimum, for example, the objective of the solution found for $n = 90, h = 621$ on TWOSBY is more than twice the objective of the optimal one found by ILP-SPACES.

Unfortunately, CP-SPACES is not able to prove the optimality of any instance within the time-limit. However, the average optimality gaps (1.73 % for NOSBY and 0.84 % for TWOSBY) reveals that it can find near-optimal solutions. The performance of both CP-SPACES and ILP-SPACES is slightly influenced by a more complex transition graph, whereas the performance of ILP-REF deteriorates significantly (average optimality gap 1.99 % for NOSBY increased to 16.02 % for TWOSBY).

2.5.3.2 Results for Large Instances

The results of this experiment are shown in Table 2.3. The results for CP-SPACES are not included, since we were unable to obtain solutions to all the instances from the IBM CP Optimizer. We observed that the solver used all the available RAM and started swapping, which negatively affected the runtime. Thus, we excluded CP-SPACES from the comparison on the LARGE dataset.

Looking at the results of ILP-SPACES, we can see that it solved all 24 instances within the time-limit (3600 s). On the other hand, ILP-REF was able to find the optimal solutions for only two smallest instances. Comparing the average optimality gaps, ILP-REF achieved 7.58 % on NOSBY transition graph and 34.39 % on TWOSBY, whereas ILP-SPACES achieved 0 % optimality gap on both transition graphs.

Table 2.2: Comparison of upper bound *ub*, lower bound *lb* and runtime *t* between the models on MEDIUM dataset. Time-limit is 600 s and TLR stands for time-limit reached.

Instance		ILP-REF [4]			CP-SPACES			ILP-SPACES		
n	h	ub [-]	lb [-]	t [s]	ub [-]	lb [-]	t [s]	ub [-]	lb [-]	t [s]
30	104	1426	1426	3.7	1435	496	TLR	1426	1426	1.3
30	127	1394	1394	4.9	1396	488	TLR	1394	1394	1.7
30	150	1394	1394	5.7	1396	484	TLR	1394	1394	2.4
30	173	1394	1394	7.0	1408	484	TLR	1394	1394	3.1
60	258	4290	4290	88.5	4339	1724	TLR	4290	4290	7.7
60	316	3994	3994	344.7	4048	1584	TLR	3994	3994	29.0
60	374	3836	3826	TLR	3925	1424	TLR	3836	3836	29.0
60	432	3956	3800	TLR	3986	1380	TLR	3833	3833	46.9
90	363	6044	5839	TLR	5963	2328	TLR	5920	5920	7.0
90	445	5778	5567	TLR	5769	2232	TLR	5686	5686	166.0
90	528	5916	4695	TLR	5670	2168	TLR	5431	5431	64.5
90	610	5901	4514	TLR	5590	1832	TLR	5373	5373	147.1
Average time [s]:				337.9			>600			42.1
Average optimality gap [%]:				1.99			1.73			0.00

MEDIUM+TWOSBY

Instance		ILP-REF [4]			CP-SPACES			ILP-SPACES		
n	h	ub [-]	lb [-]	t [s]	ub [-]	lb [-]	t [s]	ub [-]	lb [-]	t [s]
30	106	3815	3815	29.4	3815	1240	TLR	3815	3815	1.4
30	129	3804	3804	30.7	3815	1220	TLR	3804	3804	2.3
30	152	3804	3804	42.0	3815	1210	TLR	3804	3804	7.0
30	175	3804	3804	61.4	3815	1210	TLR	3804	3804	9.5
60	254	10 863	10 863	588.1	10 863	4190	TLR	10 863	10 863	2.0
60	311	10 289	10 087	TLR	10 401	3860	TLR	10 248	10 248	43.3
60	368	9917	9696	TLR	10 104	3470	TLR	9917	9917	82.1
60	426	20 346	9133	TLR	9954	3340	TLR	9874	9874	233.9
90	370	17 179	14 818	TLR	15 401	5900	TLR	15 379	15 379	140.2
90	454	22 808	12 951	TLR	14 973	5680	TLR	14 923	14 923	138.6
90	538	25 992	11 868	TLR	14 729	5500	TLR	14 548	14 548	403.8
90	621	29 558	11 406	TLR	14 900	4620	TLR	14 392	14 392	225.8
Average time [s]:				412.6			>600			107.5
Average optimality gap [%]:				16.02			0.84			0.00

2.6 Chapter Conclusions

Continuing on the recent research of the single-machine scheduling problem with the variable energy costs and power-saving machine states, we propose a pre-processing algorithm SPACES, which pre-computes the optimal switching behavior of the machine for all possible spaces in the schedule. The pre-processing runs in polynomial time and works well even for large instances of the problem, e.g., it takes 23 s to pre-process our largest benchmark instance with 190 jobs and 1277 intervals. The pre-computed switching costs are successfully integrated into novel ILP and CP models, which are compared to the state-of-the-art exact ILP model on a set of benchmark instances. Results show that our approach outperforms the existing methods considering all aspects – the runtime, the provided lower bounds, and the upper bounds. Using our models, we obtain the optimal solutions even

Table 2.3: Comparison of upper bound ub, lower bound lb and runtime t between the models on LARGE dataset. Time-limit is 3600 s and TLR stands for time-limit reached.

LARGE+NOSBY

Instance		ILP-REF [4]			ILP-SPACES			P-P
n	h	ub [-]	lb [-]	t [s]	ub [-]	lb [-]	t [s]	t [s]
150	527	8582	8567	TLR	8582	8582	187	1.0
150	647	8726	8240	TLR	8409	8409	277	2.9
150	767	8557	7787	TLR	8132	8132	624	5.5
150	888	8976	6780	TLR	8078	8078	511	9.1
170	650	10 596	9628	TLR	10 068	10 068	290	2.3
170	799	10 794	8832	TLR	9820	9820	1087	4.6
170	948	10 940	8343	TLR	9637	9637	806	9.3
170	1097	11 189	8124	TLR	9620	9620	1345	13.4
190	757	12 555	11 206	TLR	12 008	12 008	246	3.9
190	930	12 882	10 521	TLR	11 758	11 758	942	6.9
190	1104	12 791	9949	TLR	11 611	11 611	3147	13.3
190	1277	12 757	0	TLR	11 465	11 465	1348	22.7
Average time [s]:				>3600			901	7.9
Average optimality gap [%]:				7.58			0.00	

LARGE+TWOSBY

Instance		ILP-REF [4]			ILP-SPACES			P-P
n	h	ub [-]	lb [-]	t [s]	ub [-]	lb [-]	t [s]	t [s]
150	529	21 910	21 562	TLR	21 910	21 910	130	1.1
150	649	29 425	20 685	TLR	21 821	21 821	702	3.1
150	769	37 764	18 140	TLR	21 353	21 353	949	5.2
150	890	43 929	16 799	TLR	21 266	21 266	701	8.5
170	651	28 425	24 983	TLR	25 807	25 807	809	2.6
170	799	39 095	21 981	TLR	25 518	25 518	1244	5.0
170	948	46 083	20 709	TLR	25 279	25 279	2922	8.5
170	1096	53 177	20 091	TLR	25 279	25 279	2162	14.1
190	756	38 471	27 984	TLR	30 563	30 563	797	4.2
190	929	46 319	26 166	TLR	30 224	30 224	1069	7.5
190	1102	53 751	24 630	TLR	30 224	30 224	2069	13.6
190	1275	61 547	0	TLR	30 071	30 071	2572	23.7
Average time [s]:				>3600			1344	8.1
Average gap [%]:				34.39			0.00	

for the large instances with up to 190 jobs and 1277 intervals, which have been previously tackled only heuristically [4].

Chapter

3

Scheduling for Multi-Processor Systems on a Chip

3.1 Chapter Summary and Motivation

The content of the third chapter of this thesis resulted from our collaboration with Honeywell. As part of the European Union Clean Sky 2 Joint Undertaking under the H2020 Framework Programme, we have researched software techniques aimed at reducing the on-chip temperature while executing some given workload.

The global problem defined by Honeywell includes two types of workload – the safety-critical (SC) tasks and the best-effort (BE) tasks. All tasks are executed within windows that implement the temporal isolation needed for system certifiability. Each window contains a part reserved for the safety-critical tasks and a part reserved for the best-effort tasks. The safety-critical tasks are indispensable for the proper operation of the system. They are executed at a pre-defined clock frequency. Contrary to them, the best-effort tasks represent a workload that might improve the quality of the service but can be skipped if a thermal emergency occurs. Best-effort tasks do not have an assigned fixed frequency – it can be changed at runtime as necessary. To goal is to produce an offline thermal-efficient schedule (and possibly some online policies managing the best-effort tasks in case of emergencies) that will be periodically repeated. A simple illustration of one hyper-period (called a major frame) containing three windows is provided in Figure 3.1.

The validation of the proposed methods should be done on a real physical platform. Honeywell proposed a platform mounting i.MX 8QuadMax processor by NXP, which is a modern heterogeneous multi-core processor based on ARM big.LITTLE architecture with the potential to be used in future aerospace systems.

During the course of the project, we put a lot of effort into our testbed, which was described in

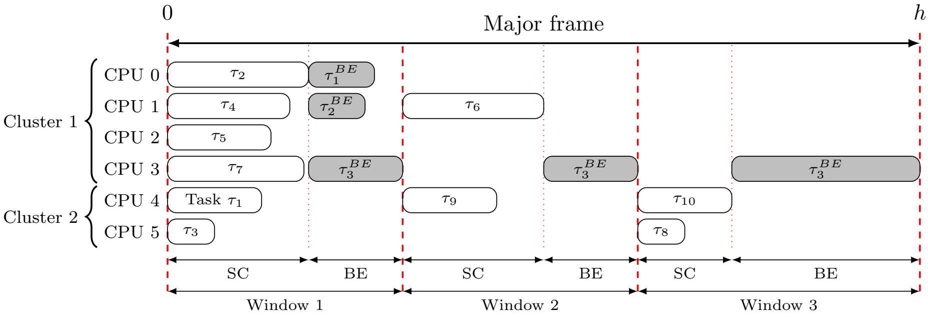


Figure 3.1: Schema of time-partitioned scheduling of safety-critical and best-effort tasks.

- Michal Sojka, Ondřej Benedikt, Zdeněk Hanzálek, and Pavel Zaykov. "Testbed for thermal and performance analysis in MPSoC systems". In: *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. 2020, pp. 683–692. DOI: `10.15439/2020F174`,

and presented at the IWCPS workshop (part of the IEEE FedCSIS conference). In order to have accurate and repeatable measurements, an open-source tool called Thermobench was developed (https://github.com/CTU-IIG/thermobench). Furthermore, to mimic the avionics operating systems implementing time-partitioned scheduling, another open-source tool called DEmOS was created (https://github.com/CTU-IIG/demos-sched). Finally, to perform thermal imaging, open-source software for finding thermal hot spots was programmed (https://github.com/CTU-IIG/thermocam-pcb), and some preliminary results were published at the EMSOFT conference:

- Michal Sojka, Ondřej Benedikt, and Zdeněk Hanzálek. "Work-in-Progress: Determining MPSoC Layout from Thermal Camera Images". In: *2021 International Conference on Embedded Software (EMSOFT)*. 2021, pp. 39–40.

Since the SC and BE tasks are isolated, the expected hyper-period length is in the range of seconds, and the system used for validation has slower thermal dynamics, we decided to take a decoupled approach and tackle safety-critical and best-effort task scheduling separately.

For the BE tasks, we use DVFS to lower the platform temperature. The preliminary study, including task-to-frequency mapping and some simple online backup strategies, was done in conjunction with student Radek Bumbálek, resulting in his master's thesis

- Radek Bumbálek. "Proactive and reactive approaches for non-critical tasks scheduling under thermal constraints in the avionics domain". MA thesis. 2022. URL: `https://dspace.cvut.cz/handle/10467/99111`,

which has further publication potential.

Because the frequency of SC tasks is given, we use task allocation as a primary power-saving mechanism. Originally, we intended to apply the idle energy function in this context as well since some of the system's components support power-saving states as part of so-called dynamic power management (DPM). However, direct control of the component states proved to be quite difficult to achieve in practice. For the thermal-aware SC task allocation, we develop an empirical power model and integrate it within an integer linear programming model. The preliminary study

- Ondřej Benedikt, Michal Sojka, Pavel Zaykov, David Hornof, Matěj Kafka, Přemysl Šůcha, and Zdeněk Hanzálek. "Thermal-Aware Scheduling for MPSoC in the Avionics Domain: Tooling and Initial Results". In: *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2021, pp. 159–168. DOI: `10.1109/RTCSA52859.2021.00026`

was presented at the RTCSA 2021 conference and was awarded the Best Paper Award. The rest of this chapter presents an extension to this submission, which is (at the time of writing) under review in a highly impacted scientific journal.

In the context of this thesis, modeling a multi-processor system on a chip (MPSoC) presents a great challenge – the chip contains multiple cores, which consume power and heat up. As the system is heterogeneous, the power consumption of different cores might be different. Furthermore, the individual cores influence each other after heating up. Moreover, MPSoCs are quite complex nowadays, and so other undesired events, such as memory interference, influencing the power consumption might take effect. Considering the optimization, the problem at hand is also quite challenging; not only do we look for optimal task-to-core mapping, but we also need to find task-to-window assignment such that the windows fit into the pre-defined hyper-period. This temporal isolation further complexifies the already complex assignment problem. Finally, as we look for a thermal-efficient schedule, the thermal model must be integrated into the optimization somehow, but classical thermal models might be too complex to be efficiently optimized.

3.2 Introduction

Increasing demand for computing power has led to the deployment of Multi-Processor System-on-Chips (MPSoC) in many industrial domains. These include even the ones in safety-critical domains, such as aerospace and automotive [115, 80]. Inevitably, the usage of such complex systems in safety-critical applications brings new challenges related to thermal management. Overheating has a negative impact on the reliability and safety of the systems [76], e.g., due to thermal degradation of hardware components. Moreover, the rise in the operational temperature of on-chip components may lead to irreversible permanent failures [106]. Furthermore, reduction of the on-chip temperature leads to a reduction of the leakage power, which is nowadays a significant part of the modern MPSoC power consumption [145, 110]. Therefore, the thermal-aware design became a crucial part of the development across all system levels.

Many approaches exist to reduce MPSoC temperature, including (i) thermally aware hardware design, (ii) active cooling, and (iii) software-based optimization. Examples of the latter-most subsume dynamic power management (DPM), dynamic voltage and frequency scaling (DVFS), or thermal-aware task mapping and scheduling. Many works address individual problems such as power or thermal modeling [66, 134], design of efficient scheduling algorithms [142, 33, 143], the study of thermal behavior of hardware platforms [84, 113]. However, these are often studied separately, while our ambition is to look at a subset of those problems relevant for safety-critical applications and study them together.

In the deployment of safety-critical applications, it is required to avoid unexpected interference between tasks. One potential source of interference is DVFS, which is therefore deemed unsuitable for these applications [46]. Another source of interference can be the operating system scheduler. Therefore, avionics applications adopt time-partitioned scheduling [35], where the tasks are scheduled into temporal isolation windows. Although some authors have already addressed thermal-efficient task mapping on heterogeneous MPSoCs [142, 1], the integration with temporal isolation windows, which makes the problem even more complex, has not yet been studied.

In this chapter, we study thermal modeling in connection with the design of optimization algorithms and how the imprecision of the former influences the efficiency of the latter. Throughout this chapter, we target the problem of thermally efficient task allocation under temporal isolation constraints on heterogeneous MPSoC. We assume that the allocation is computed offline (in the design phase) and that DVFS is not used due to the safety certification requirements [46]. The temporal isolation constraints are used to separate the critical workloads in time as implemented in many avionics operating systems based on the ARINC-653 standard [9]. In our experiments, we use an open-source ARINC-653-like Linux scheduler called DEmOS[1], providing independence of proprietary avionics RTOSes.

We focus on analyzing three hardware platforms (I.MX8QM MEK, I.MX8QM Ixora, NVIDIA TX2) representing the modern MPSoCs. Further, we discuss thermal and power modeling and propose multiple optimization methods incorporating the power models. Finally, we evaluate the models and methods on physical platforms and summarize their advantages and disadvantages. As required by our industrial partner, we emphasize the data-driven evaluation based on measured characteristics of real physical platforms.

Contributions.

- We demonstrate that an empirical sum-max power model (`SM`) integrated within an Integer Linear Programming formalism outperforms the other methods across all tested scenarios on all platforms.
- For comparison, we implement a new power model based on linear regression (`LR`) and its simplified variant providing an upper bound of the estimated power consumption (`LR-UB`).
- We discuss the trade-offs between the power model accuracy and optimization method performance based on the integration of all our power models with several optimization methods for task allocation on heterogeneous MPSoC.
- Overall, we evaluate three power models, four optimization methods integrating our power models, one local informed heuristic based on related works, and two uniformed heuristics. All of this is evaluated on three hardware platforms. All measured data as well as the source code of the optimization methods are publicly available at `https://github.com/benedond/safety-critical-scheduling`.

This chapter extends the preliminary study [20] as follows:

- The original empirical power model proposed in [20] is properly evaluated and compared with another model based on linear regression.
- The optimization model proposed in [20] is compared with several other informed and uninformed models and heuristics. Their performance is tested, and strengths and weaknesses are discussed.
- Two additional hardware platforms (NXP I.MX8 Ixora, NVIDIA TX2) are used in the experimental measurements to validate the proposed solution approach.

---

[1] `https://github.com/CTU-IIG/demos-sched`

- New benchmarks based on the industrial standard EEMBC Autobench 2.0 are used in addition to small and synthetic benchmarks used in [20]. Many new measurements are collected, and the results are reported.

Outline. The rest of this chapter is organized as follows. Section 3.3 summarizes the related works. Section 3.4 describes the system model and formalizes the scheduling problem definition. The physical hardware used for the experiments and the benchmarking kernels are summarized in Section 3.5. Thermal modeling with regard to the allocation of the safety-critical tasks and implementation of specific models are addressed in Section 3.6. Integration of the thermal models with the optimization procedures is discussed in Section 3.7. Experimental evaluation follows in Section 3.8. Finally, Section 3.9 concludes the chapter.

3.3 Related Work

Thermal-aware and energy-efficient scheduling for real-time systems has been studied for many years [55]. The approaches found in the literature differ in many aspects. We identified three steps, illustrated in Figure 3.2, that need to be performed to implement thermally efficient scheduling: the *benchmarking*, the *optimization* and the *evaluation*. The decisions made at each step relate to the other steps and have an influence on the overall properties of achieved results. In the next subsections, we describe each step and related decisions in more detail and review the relevant literature. Finally, we summarize the choices made in this chapter and relate them to the current state-of-the-art.

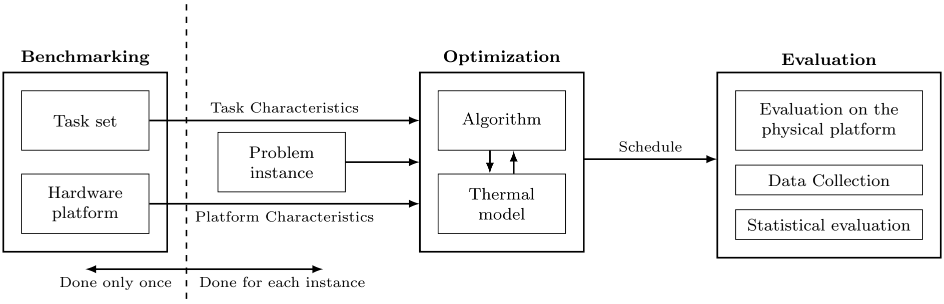


Figure 3.2: Three steps (bechmarking, optimization and evaluation) towards the thermally efficient scheduling.

3.3.1 Benchmarking

To perform the thermal optimization, the algorithm needs to be provided with information about the platform itself and the tasks to be executed. In the rest of the chapter, we call this information the *platform characteristics* and *task characteristics*. In the literature, the characteristics are usually obtained by two means: from documentation or by benchmarking.

Platform characteristics describe the thermal/power behavior without relation to the executed workload. The platform characteristics might include, e.g., the area of the chip, power consumption under various frequency settings, and thermal conductances and capacitances of the thermal nodes to be modeled. Often, the authors take these parameters for granted, relying on the information from the technical documentation of the platform [107, 100], or using the pre-defined configurations provided with the simulation software like HotSpot [89, 72]. This, however, might not reflect reality accurately as thermal parameters are often influenced by the printed circuit board (PCB) layout chosen by a particular board manufacturer and/or by variations in the manufacturing process. Therefore, identifying target platform characteristics by benchmarking [124, 8] provides more accurate parameters.

Task characteristics allow us to distinguish between the individual workloads and describe their thermal effects, which can differ due to their different nature, such as using different parts of the chip, being more or less memory intensive, etc. Task characteristics are typically obtained by benchmarking, but authors use characteristics of varying complexities. Some works assume all tasks to be identical (therefore, no task-specific characteristics are needed) [32], while other works assume a single numerical coefficient [140, 89], multiple coefficients [20, 114, 11], or even very complex characteristics obtained, e.g., from many CPU performance counters [118] or by training a neural network [139].

Since benchmarking tends to be time-consuming, proper care must be taken when selecting the characteristics. The benchmark engineer needs to ensure that they describe the platform and task behavior well while avoiding unnecessary complexity. In this chapter, we show that even simple characteristics can be sufficient for significant temperature reduction.

3.3.2 Optimization

Platform and task characteristics, together with the instance parameters, serve as an input to the optimization procedure, which integrates the scheduling algorithm and the thermal model. The algorithms can operate online, at system run time, or offline. Online algorithms are typically used in areas like cloud or high-performance computing [126]. Offline algorithms are often required for safety-critical applications such as in avionics [35], which is the target domain for the work in this chapter. In offline scheduling, the algorithm decides the task's allocations and their start times while ensuring that all other timing, thermal, and other resource constraints are met. For this, the algorithm must interact with the thermal model.

3.3.2.1 Thermal Model

The thermal model serves to predict the evolution of on-chip temperature in time based on the state of the system and the workload that is being executed. From the viewpoint of the systems dynamics we differentiate between the *transient-state* and the *steady-state* models [28]. The former is more general and can be used to predict the transient states of the system (i.e., the heating and cooling trajectories) [8, 50, 142, 82]. Contrary to that, the latter model provides only the steady-state temperature but tends to be much simpler to implement [144, 114, 89, 1]. According to Chantem et al. [28], the steady-state model is sufficient if the temporal parameters of the workload are short enough.

From the spacial point of view, we distinguish between the *single-output* and *multi-output* thermal models. Single-output model provides a prediction for a single thermal node only, modeling, e.g., the average on-chip temperature [20, 70]. The multi-output model can be used when more precise spatial granularity is needed (modeling, e.g., per-cluster or per-core temperatures) [8, 1, 83].

In this chapter, we evaluate the thermal properties of our platforms experimentally and relate them to the thermal and power models.

3.3.2.2 Optimization Algorithms

Many different approaches to task scheduling and allocation have been studied. Due to the inherent complexity of thermal-aware scheduling, authors often rely on local or greedy heuristics even when constructing the schedule offline [1, 106, 83, 144, 74, 142]. Other approaches include meta-heuristics such as evolutionary algorithms [89, 26, 87], or even exhaustive (optimal) approaches mainly based on mixed-integer linear programming [89, 1, 28, 70]. In this chapter, we use all three approaches and compare them.

The scheduling algorithm and the thermal model can interact in many ways; (i) In the simplest form, the thermal model is used to validate whether the schedule provided by the scheduling algorithm can be executed under the given thermal constraints or not [8]; (ii) In more complex cases, the loop is closed, and the thermal model provides the information about the violation of the thermal constraints back to the scheduling algorithm, which, in turn, tries to rebuild the schedule [1, 107]; (iii) Finally, the thermal model and the thermal constraints can be integrated directly within the scheduling algorithm, thus providing the most integrated solution [28, 20]. In this chapter, we implement and evaluate all three approaches.

3.3.3 Evaluation

When the optimization is completed, one needs to evaluate the performance of the resulting schedule. In that regard, two common approaches can be found in the literature: (i) evaluation by simulation and (ii) evaluation on a physical platform. Simulation-based approaches are found more often [1, 106, 83, 143, 72, 28, 74, 70]. The reasons justifying this approach include simpler execution of the experiments and better reproducibility of the results. On the other hand, simulation is always based on models, which might fail to capture all details of the hardware platform properly. Thus some authors evaluate thermal effects of the schedules experimentally on real hardware [88, 90, 78, 105, 8]. In this chapter, we follow this experimental approach.

3.3.4 Summary

None of the above-mentioned works tackles the same problem as this chapter. Our work is unique in combining avionics ARINC-653-inspired time-partitioned scheduling of safety-critical workloads with thermal issues on real hardware platforms.

Table 3.1: Design choices and their coverage (✓) in this chapter.

Design choice	Options 1	Option 2	Reference
Thermal model dynamics	✓ **Steady-state**	✗ Transient	Section 3.6.1.1
Thermal model output	✓ **Single**	✗ Multiple	Section 3.6.1.2
Optimization w.r.t model	✓ **Heuristics**	✓ **Exhaustive**	Section 3.7
Search strategy w.r.t. temperature parameters	✓ **Uninformed**	✓ **Informed**	Section 3.7
Evaluation	✗ Simulation	✓ **Experimental**	Section 3.8

With respect to the design choices presented in this section, we target the steady-state thermal model due to the character of the workload (short task execution time). In addition, we adopt a single-output system model based on experimental evaluation and benchmarking performed on three different hardware platforms. Considering the optimization methods, we compare both heuristics and exhaustive approaches. The overview of design choices and their coverage in this chapter is summarized in Table 3.1.

3.4 Goal and System Model Formalization

In this section, we define the goal of the thermal optimization and formalize the system model and input parameters defining the scheduling problem of the thermal-aware safety-critical tasks allocation on MPSoC under temporal isolation constraints.

3.4.1 Goal

We want to find an assignment of tasks to CPUs of a heterogeneous multi-core platform together with an allocation of tasks to the temporal isolation windows such that the steady-state temperature of the platform is minimized.

There are at least three factors that make this problem complex. (i) All the tasks must be scheduled within the pre-defined scheduling hyper-period, which repeats indefinitely; therefore, any task allocation with the makespan exceeding the hyper-period is not feasible. (ii) The number of isolation windows and their lengths are not known a priori. (iii) The steady-state temperature depends on the thermal interference of the tasks running in parallel on different CPUs.

3.4.2 System Model and Input Parameters

We define our model and parameters as follows.

Model of Processing Elements. We assume a heterogeneous architecture, i.e., MPSoC having m computing *clusters* (possibly of different hardware architecture) denoted by $\{C_1, C_2, \ldots, C_m\} = \mathcal{C}$. Cluster C_k has $c_k \in \mathbb{Z}_{>0}$ cores, which are assumed to be identical. All cores in the cluster share the same clock frequency, which we assume to be fixed due to typical safety requirements [46]. Different clusters can have different frequencies. We assume that the platform has so-called *platform characteristics* (features) denoted as $\mathcal{F}^{(p)}$, which are obtained by benchmarking and characterize the platform's thermal/power behavior.

Task Model. We assume a set of independent, non-preemptive, periodic tasks $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$. By $e_{i,k} \in \mathbb{Z}_{>0}$ we denote the *worst-case execution time* of task $\tau_i \in \mathcal{T}$ on cluster $C_k \in \mathcal{C}$. All tasks are ready at time 0 and have a common period $h \in \mathbb{Z}_{>0}$, which is called a *major frame length.* We assume that the deadline of each task is equal to period h. Each task represents a single-threaded safety-critical process that needs to be executed on one of the platform cores. We denote the task characteristics associated with task τ_i as $\mathcal{F}_i^{(t)}$. Note that both task characteristics and platform characteristics are platform-specific, i.e., they need to be obtained separately for each tested platform.

Temporal Isolation. The temporal isolation of the safety-critical tasks is ensured by so-called *scheduling windows*, which are non-overlapping intervals partitioning the hyper-period (see Fig. 3.3) inspired by ARINC-653 standard; more details are discussed in [20]. We denote the set of such windows as $\mathcal{W} = \{W_1, W_2, \ldots, W_q\}$. Length of window $W_j \in \mathcal{W}$ is denoted by $l_j \in \mathbb{Z}_{\geq 0}$. Each task needs to be assigned to a single window, within which it will be executed at the core of one of the clusters. At most one task per core can be executed within each window. Note that the number of windows q is not known a priori but can be upper-bounded by n since the tasks are non-preemptive and so only one task will be present in each window in the worst case.

Used notation is illustrated in Figure 3.3, where the schedule of seven tasks is presented.

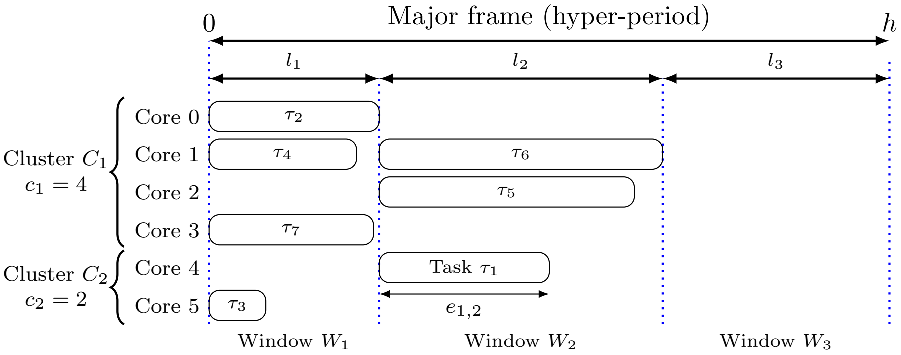
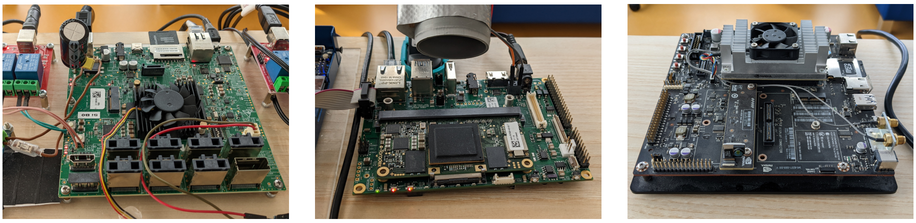


Figure 3.3: Illustration of the used notation.

3.5 Hardware Platforms and Benchmarks

As we discussed previously, we opt for an experimental evaluation instead of simulation. To make the comparison of optimization methods and power models more representative, we conduct the benchmarking and evaluation phases on three platforms, which are briefly described in Section 3.5.1. Further, we describe the benchmarking kernels selected for the experiments in Section 3.5.2. Finally, Section 3.6 summarizes the thermal model together with the selected platform and task characteristics.



(a) (b) (c)

Figure 3.4: Embedded platforms used for the evaluation: (a) I.MX8QuadMax Multisensory Enablement Kit by NXP (I.MX8 MEK), (b) Toradex Apalis I.MX8 board (I.MX8 Ixora), (c) Nvidia Jetson TX2 Developer Kit (TX2).

3.5.1 Physical Hardware for Evaluation

We selected modern high-performing MPSoCs for the evaluation, namely I.MX 8QuadMax by NXP [102] and Nvidia Tegra X2 T186 [101]. Both of these are based on ARM big.LITTLE heterogeneous architecture hosting two CPU clusters including so-called *high-performing* and *energy-efficient* cores.

The I.MX 8QuadMax features four ARM Cortex-A53 cores and two ARM Cortex-A72 cores. Each of the cores has 32 kB data cache, and each cluster has 1 MB L2 cache. We set the clock frequency of each cluster to the highest values, which is 1200 MHz for the A53 cluster, and 1600 MHz for the A72 cluster, respectively.

Similarly, Nvidia Tegra X2 T186 hosts four energy cores and two high-performing cores, which are of ARM Cortex-A57 architecture and Nvidia Denver architecture, respectively. Each A57 core has 32 kB data cache, and each Denver core has 64 kB data cache. The size of the L2 cache of each cluster is 2 MB. We set the clock frequency of both clusters to 2035 MHz.

In our testbed, we have two boards with I.MX8, namely I.MX8QuadMax Multisensory Enablement Kit (MEK) [103], and Ixora carrier board with Toradex Apalis I.MX8 module [127]. In the further text, these platforms are denoted as I.MX8 MEK and I.MX8 Ixora, respectively. Besides their different form factor and PCB layout, the first one has an Aluminum heat sink mounted on the chip while the latter has none, but we cool it by airflow from an external fan. In this way, the latter chip can be observed by a Workswell infrared camera [132]. We have extended both I.MX8 boards with external power meters. Besides I.MX8, we have Nvidia MPSoC, which is mounted on NVIDIA Jetson TX2 Developer Kit carrier board [99]. We henceforth denote this platform simply as TX2. A part of our testbed is shown in Figure 3.4. The configuration and used sensors are described in more detail in [124].

3.5.2 Benchmarking Kernels

To mimic the safety-critical workloads used in avionics and other similar domains, we use a set of relatively simple applications (kernels) written in C. The set contains selected kernels based on EEMBC AutoBench 2.0 [44] together with custom memory stressing tool *membench* and software rendering tool based on OpenGL *tinyrenderer*

[124]. We use tinyrenderer in two configurations – rendering boggie objects (*-boggie*) and diablo objects (*-diablo*).

AutoBench is a general-purpose benchmark set containing generic workload tests, as well as automotive and signal-processing algorithms. We use twelve of its kernels including: *a2time* (angle to time conversion), *aifirf* (finite impulse response filter), *bitmnp* (bit manipulation), *canrdr* (CAN remote data request), *idctrn* (inverse discrete cosine transform), *iirflt* (infinite impulse response filter), *matrix* (matrix arithmetic), *pntrch* (pointer chasing), *puwmod* (pulse width modulation), *rspeed* (road speed calculation), *tblook* (table lookup and interpolation), and *ttsprk* (tooth to spark). Each benchmark is used in two variants, i.e. *-4K* and *-4M*, representing two different input data sizes (4 kB and 4 MB). Further information about the benchmarks can be found in [111].

Membench is a tool that stresses the memory hierarchy. It can be configured in many ways. We use it in three different configurations with respect to the working set size (WSS), i.e. *-1K*, *-1M* and *-4M*, representing WSS of 1 kB, 1 MB, and 4 MB, respectively. Further, we test both sequential (*-S*) and random (*-R*) memory accesses in both read-only (*-RO*) and read-and-write (*-RW*) variants. Therefore, we have twelve membench kernels in our benchmark set.

Each of the kernels (12×2 autobench, 12 membench, 2 tinyrenderer) is wrapped inside of an infinite loop. A single iteration represents one execution of the kernel. We report the iterations per second (IPS) of each kernel (executed on a single core, without any interference) for each tested hardware platform in Table 3.2.

Figure 3.5 shows the relative speedup s on a high-performing (big) cluster compared to the energy-efficient (little) cluster, i.e., the ratio between runtimes e on these two clusters normalized by their frequencies f. We calculate the relative speedup s as:

$$s = \frac{\frac{e_{\text{little}}}{f_{\text{little}}}}{\frac{e_{\text{big}}}{f_{\text{big}}}} = \frac{\text{IPS}_{\text{big}} f_{\text{big}}}{\text{IPS}_{\text{little}} f_{\text{little}}}. \tag{3.1}$$

We observe that big cluster of I.MX8 (TX2) platform is, on average, about 2.8× and (1.3×) more performant than the little one.

Table 3.2: Iterations per second (IPS) of used kernels.

	I.MX8 MEK		I.MX8 Ixora		TX2 Developer Kit	
	little (A53) 1200.00 MHz	big (A72) 1600.00 MHz	little (A53) 1200.00 MHz	big (A72) 1600.00 MHz	little (A57) 2035.00 MHz	big (Denver) 2035.00 MHz
a2time-4K	32 612.42	56 012.34	32 392.47	55 925.02	64 788.87	90 732.49
a2time-4M	32.10	54.60	31.59	54.51	58.05	117.38
aifirf-4K	3179.22	8806.25	3152.15	8800.54	10 512.20	15 104.39
aifirf-4M	3.11	8.75	3.09	8.74	10.39	15.24
bitmnp-4K	11 763.41	24 485.25	11 655.71	24 536.22	27 287.52	47 289.58
bitmnp-4M	9.84	21.65	9.76	21.54	20.20	34.08
canrdr-4K	33 409.00	62 938.10	33 078.74	63 262.99	163 382.51	189 621.55
canrdr-4M	35.57	68.12	35.20	68.02	182.91	226.16
idctrn-4K	6966.36	20 301.27	6914.02	20 269.73	24 190.64	28 626.42
idctrn-4M	6.86	20.49	6.81	20.47	24.24	28.86
iirflt-4K	8253.34	23 629.72	8178.40	23 748.03	23 785.59	35 237.47
iirflt-4M	8.23	18.83	8.17	18.81	22.54	35.31
matrix-4K	4029.86	10 552.83	4002.32	11 235.61	12 562.78	9796.33
matrix-4M	14.76	41.73	14.60	41.65	43.02	34.85
membench-1K-RO-R	11.89	11.87	11.80	11.87	14.93	20.04
membench-1K-RO-S	11.88	11.87	11.80	11.87	14.93	20.04
membench-1K-RW-R	4.46	7.92	4.42	7.91	14.94	20.00
membench-1K-RW-S	4.46	7.92	4.42	7.91	14.93	19.99
membench-1M-RO-R	2.11	1.63	1.70	1.50	2.09	2.44
membench-1M-RO-S	4.23	3.32	3.99	3.17	6.18	5.02
membench-1M-RW-R	1.42	1.27	1.15	1.29	1.89	2.56
membench-1M-RW-S	3.78	2.94	3.49	2.80	5.27	4.68
membench-4M-RO-R	0.15	0.20	0.14	0.18	0.23	0.18
membench-4M-RO-S	0.71	2.42	0.63	2.01	1.49	4.25
membench-4M-RW-R	0.15	0.19	0.13	0.17	0.23	0.18
membench-4M-RW-S	0.63	1.50	0.48	1.36	1.46	3.69
pntrch-4K	279.39	773.88	277.13	815.54	843.55	1053.43
pntrch-4M	0.27	0.79	0.27	0.79	1.01	1.44
puwmod-4K	18 709.96	35 919.98	18 547.46	35 863.73	45 702.33	79 401.37
puwmod-4M	19.03	37.41	18.86	37.38	47.51	87.53
rspeed-4K	53 498.40	136 629.65	52 959.46	136 940.36	171 090.58	140 909.26
rspeed-4M	56.07	168.75	54.88	168.47	198.18	188.11
tblook-4K	19 232.01	59 599.44	19 025.71	60 082.12	73 168.14	82 280.66
tblook-4M	19.59	63.66	19.33	62.50	73.65	86.65
tinyrenderer-boggie	3.25	6.33	3.17	6.23	5.67	6.26
tinyrenderer-diablo	2.85	5.72	2.81	5.68	5.01	5.68
ttsprk-4K	67 723.01	138 148.80	67 350.74	137 949.16	172 158.44	242 422.26
ttsprk-4M	183.36	390.49	180.60	388.72	483.17	645.50

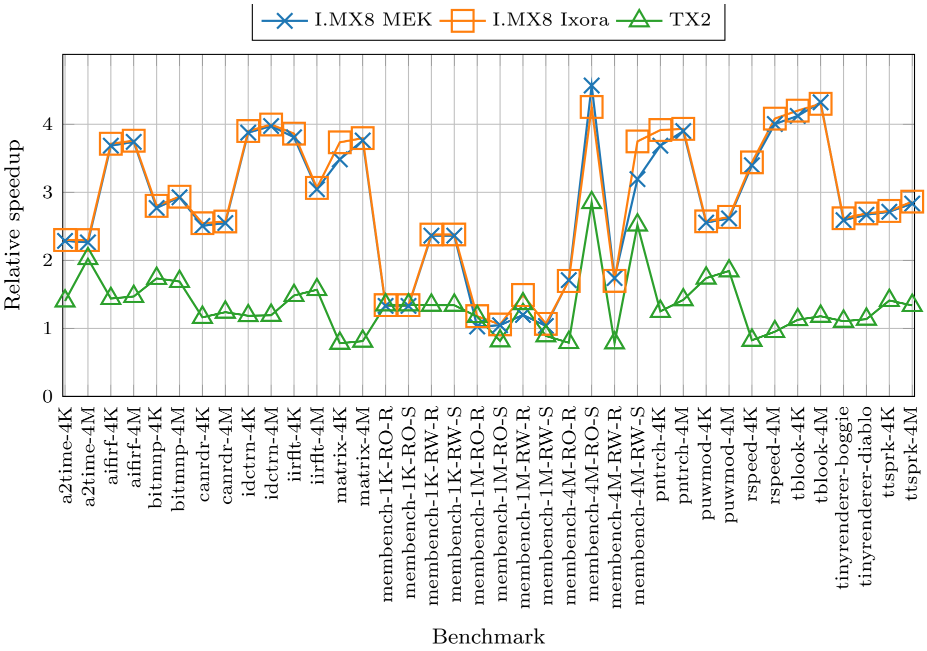


Figure 3.5: Relative speedup on a CPU from the high-performing cluster.

3.6 Thermal Modeling

A lot of attention must be paid when designing a thermal model. In our view, the main aspects of the thermal model to be considered and balanced are its *simplicity* and *accuracy*. A simpler model is easier to integrate with the optimization procedures. Also, it takes less effort to identify its parameters. However, a too simple model fails to predict the system's behavior accurately. Therefore, the trade-off between simplicity and accuracy needs to be taken into account.

The rest of this section is divided into three parts. In Section 3.6.1, we experimentally justify the use of our thermal model. Then, we discuss the transition from thermal to power modeling in Section 3.6.2. Finally, we describe the specific power models in Section 3.6.3.

3.6.1 Thermal Experiments with Used Platforms

To justify our thermal model selection (steady-state, single-output), we perform a set of experiments. First, we analyze the platform's thermal dynamics in Section 3.6.1.1, and then we assess the relationship between the temperatures of little and big CPU clusters in Section 3.6.1.2.

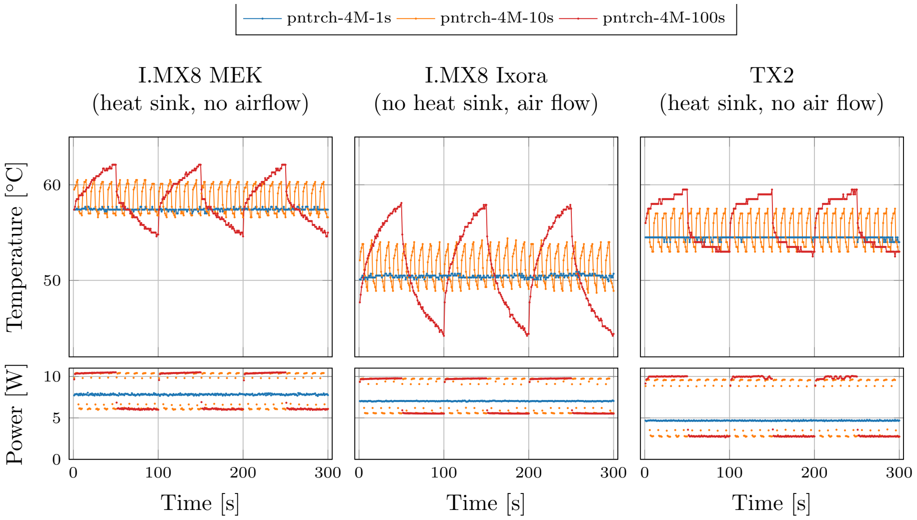


Figure 3.6: Influence of the on-chip temperature near the high-performing cluster on the major frame length (hyper-period) for three instance alternating between computing and idling.

3.6.1.1 Thermal Dynamics and the Major Frame Length

Based on the typical lengths of the major fame used in avionics applications (less than one second), we decided to have a steady-state thermal model. This decision is supported by the following experiment: a schedule containing two windows of the same length is created. These two windows constitute the major frame. In the first window, all the cores are loaded, executing some workload (here *pntrch-4M*), whereas, in the second window, all cores are idling. We alternately execute these two windows and monitor the temperature and power consumption of the platform. We create three instances, which differ in the major frame length – the first (denoted with suffix *-1s*) has the major frame length equal to 1 s (each window is 500 ms long), the second (*-10s*) has the major frame length 10 s, and the third (*-100s*) has the major frame length 100 s. The resulting temperatures measured in the proximity of the big cluster are shown in Figure 3.6.

Indeed, when the major frame length of the instance are long enough, such as in the *-100s* case, we clearly observe the heating and cooling curves corresponding to the individual scheduling windows. However, for our use-case (*-1s*), we see that the temperature is almost constant.

Note that the power consumption of both platforms based on I.MX8 is nearly the same; however, their thermal trajectories differ significantly due to different physical parameters (heat sink versus no heat sink, with/without airflow).

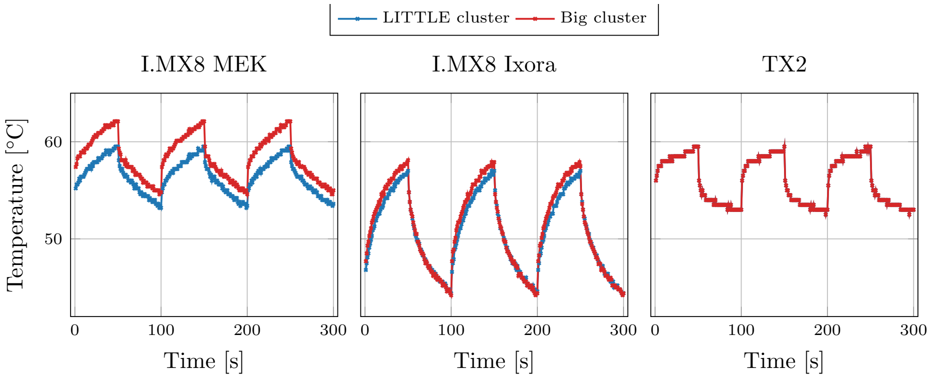


Figure 3.7: Temperatures obtained for *pntrch-4M-100s* from on-chip sensors near little and big cluster thermal zones.

3.6.1.2 Investigating Spatial On-Chip Temperatures

One of the decisions to take into consideration is whether to use a single- or multi-output thermal model. We decided on a single-output model. Here, we reason why.

When scheduling a workload, different parts of the chip start to produce heat. Ideally, we would like to monitor the temperature of each core. However, per-core temperature monitoring might not be possible for many platforms, including ours. Our three platforms provide us with just several temperature sensors associated with the major thermal zones (little cluster, big cluster, PMIC, GPU, etc.). We visualize the temperatures measured for the *pntrch-4M-100s* benchmark (the one used in the previous section) near little and big clusters in Figure 3.7.

We observe that the temperature difference on I.MX8 Ixora is smaller compared to I.MX8 MEK, because of the absence of a heat sink on the Ixora board and active cooling that is employed. Considering the TX2 platform, we observe that both thermal zones report the same value. This might be caused by a massive heatsink, combined with the imprecision of the sensors and their possible spatial proximity.

To further investigate the thermal behavior near the CPU clusters, we look at I.MX8 Ixora using the Thermal camera. We execute *pntrch-4M* on all cores of each cluster and compare the resulting images. Figure 3.8 shows the spatial on-chip temperature $T(x, y)$, where the x and y coordinates are in pixels (each pixel corresponds to 0.29 mm). Also, we show the heat sources on a chip $h(x, y)$, where $h(x, y) = \max\{0, -\kappa\nabla^2 T(x, y)\}$ is a positive part of negative Laplacian of $T(x, y)$ scaled by factor $\kappa > 0$, which follows from heat diffusion equation as explained in [138, 124].

Figure 3.8 shows that the big cluster is heating the platform much more (the peak of $h(x, y)$ is about 2.5× higher) compared to the little one. Also, the left part of the figure shows how the on-chip heat spreader distributes the heat from the heat source to the borders of the chip. When only the little cluster is executing the workload, the difference between the individual cluster zones' temperatures is nearly

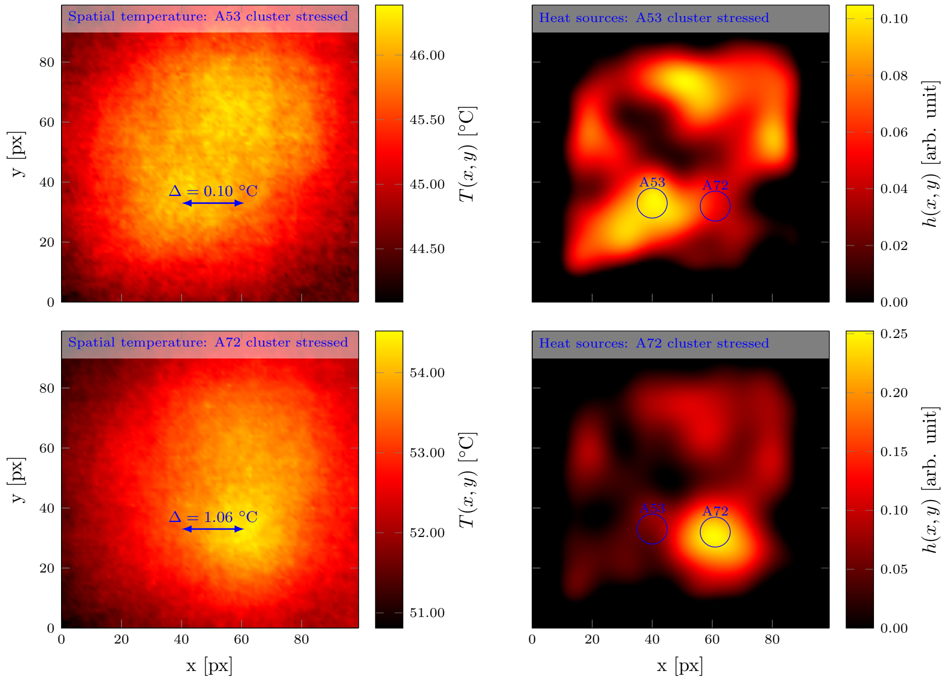


Figure 3.8: Spatial on-chip temperature $T(x, y)$ on the left and hot spots $h(x, y)$ on the right of I.MX8 Ixora with little (A53) cluster stressed at the top, and big (A72) cluster stressed at the bottom.

negligible. When the big cluster is performing the computations, the difference is more apparent, but still only about 1 °C for this particular workload.

To summarize the observations: although we see some differences between the temperatures measured in the vicinity of the individual clusters, both of their thermal trajectories are similar, as seen in Figure 3.7. Due to the heat spreader and relative proximity of both clusters, the change of the temperature near one of the clusters influences the temperature near the other one as shown in Figure 3.8. Taking that into account, we decided to model only the temperature near the big cluster, which is thermally dominant.

3.6.2 Transition from Thermal to Power Model

A single-output steady-state thermal model of MPSoC adopted in this chapter can be tightly related to a simpler model based on average power consumption. In some sense, they can be used interchangeably, but from the practical viewpoint, the difference is very important. Deviations in the ambient temperature cause deviations in the steady-state temperature. Therefore, measuring the power input is usually more stable and easily reproducible. Further, the time needed to reach stabilized temperature can be rather long, whereas the power measurements reflect the immediate state.

Widely used methodology for creating thermal models of MPSoC relies on resistance-capacitance (RC) thermal networks [66, 104, 107]. The system is modeled as a set of thermal nodes, that are interconnected via thermal conductances and associated with thermal capacitances. The relation between the temperature of every thermal node, its power consumption, and the ambient temperature can then be expressed by a set of differential equations [104]:

$$\boldsymbol{A}\boldsymbol{T}' + \boldsymbol{B}\boldsymbol{T} = \boldsymbol{P} + \boldsymbol{G}T_{\text{amb}}, \tag{3.2}$$

where η is the number of thermal nodes, $\boldsymbol{A} \in \mathbb{R}^{\eta\times\eta}$ is a matrix of capacitances, $\boldsymbol{B} \in \mathbb{R}^{\eta\times\eta}$ is a matrix of thermal conductances, $\boldsymbol{T} \in \mathbb{R}^{\eta\times 1}$ is a vector of temperatures at each node, $\boldsymbol{P} \in \mathbb{R}^{\eta\times 1}$ is a vector of power consumption of the nodes, and $\boldsymbol{G} \in \mathbb{R}^{\eta\times 1}$ is a vector containing the thermal conductance between each node and the ambient.

When the system reaches a steady state, $\boldsymbol{A}\boldsymbol{T}'$ becomes zero as the temperature remains constant in time. Then, considering a single thermal node only, $\boldsymbol{B}$, $\boldsymbol{G}$ and $\boldsymbol{P}$ become scalars (denoted by B, G, and P) and the whole system reduces to linear relation with respect to P:

$$T = \frac{1}{B}P + \frac{G}{B}T_{\text{amb}}, \tag{3.3}$$

where T is the steady-state temperature at the thermal node (here, at the thermal zone near the big cluster), and P is the power consumption.

We, indeed, observe this linear relation in reality, as shown in Figure 3.9. There, we plot the average power and steady-state temperature of various benchmarks (both memory and CPU-bound) executed on our platforms. Clearly, both measured quantities are strongly correlated.

In the rest of the chapter, we work with the power model instead of the thermal model, assuming that the final transformation from the average power to the steady-state temperature can be done according to (3.3).

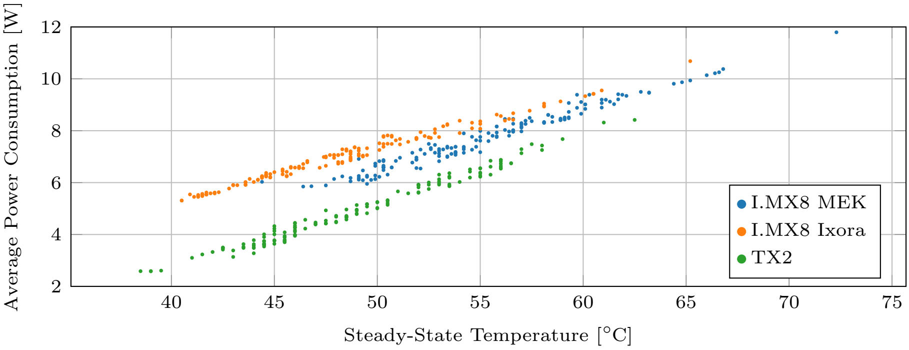


Figure 3.9: Average power and steady-state temperature of various benchmarks executed on tested platforms.

Note that model (3.2) does not take into account the temperature-dependent leakage power, contrary to, e.g., Guo et al. [58]. While this might look like a significant drawback, our results in Section 3.8 show that even such a model is sufficient for temperature reduction when integrated within the optimization framework.

3.6.3 Power Models

Following the general discussion on thermal modeling, we continue with descriptions of specific models. As noted in Section 3.6.2, since the average power and the steady-state temperature are linearly related, we implement just the models estimating the average power consumption.

3.6.3.1 Empirical Sum-Max Model

First, we summarize the *sum-max* model (`SM`) proposed in [20]. The model is purely empirical; given a scheduling window with the allocated tasks, it predicts the average power consumed during the execution of such a window.

Specifically, given window W_j of length l_j and tasks allocation $(\mathcal{T}_1^j, \ldots, \mathcal{T}_m^j)$, where set $\mathcal{T}_k^j$ represents tasks allocated to cluster C_k in window W_j, `SM` model predicts the average power consumption $P(W_j)$ as:

$$P(W_j) = \sum_{C_k \in \mathcal{C}} \sum_{\tau_i \in \mathcal{T}_k^j} \left(a_{i,k} \cdot \frac{e_{i,k}}{l_j} \right) + \max_{\substack{C_k \in \mathcal{C} \\ \tau_i \in \mathcal{T}_k^j}} o_{i,k} + P_{\text{idle}}, \tag{3.4}$$

where P_{idle} is the idle power consumption of the platform, and $o_{i,k}$ and $a_{i,k}$ are task-specific coefficients obtained via benchmarking. The average power of a schedule consisting of multiple windows is calculated as a weighted average of their individual contributions (the weights correspond to the window lengths).

The model is built upon the assumption that power consumption of z instances of task τ_i executed independently and in parallel on $z \in \{1, 2, \ldots, c_k\}$ cores of cluster C_k can be expressed as $(z \cdot a_{i,k} + o_{i,k} + P_{\text{idle}})$. Coefficients $a_{i,k}$ and $o_{i,k}$ can be related to dynamic and static power consumption incurred by execution of task τ_i on C_k. At the end of this chapter, we present a numerical example illustrating the calculation of `SM` power model for one specific window.

Platform Characteristics. In the context of the modeling and optimization framework discussed in this chapter, platform characteristics $\mathcal{F}^{(p)}$ constitute a single parameter only, which is the idle power consumption, $\mathcal{F}^{(p)} = (P_{\text{idle}})$. The idle power consumption of each tested platform is listed in Table 3.3.

Table 3.3: Idle power consumption P_{idle} of tested platforms.

Platform	P_{idle} [W]
I.MX8 MEK	5.5
I.MX8 Ixora	5.5
TX2	2.6

Task Characteristics. The sum-max model needs two coefficients for each task and cluster, i.e., task characteristics $\mathcal{F}_i^{(t)}$ are represented by a four-tuple for each $\tau_i \in \mathcal{T}$, $\mathcal{F}_i^{(t)} = (a_{i,1}, o_{i,1}, a_{i,2}, o_{i,2})$. Following the methodology introduced in [20], we identify the coefficients for all benchmarks on each tested platform. Their values are visualized in Figure 3.10, and further listed in Table 3.4 and Table 3.5. Note that the sum of $o_{i,k}$ and $a_{i,k}$ (i.e., the height of the bar in Figure 3.10) represents the increase the power consumption of the platform w.r.t. P_{idle} when executing the benchmark on a single core of cluster C_k.

Table 3.4: Task characteristics coefficients identified for used kernels on little cluster.

	I.MX8 MEK		I.MX8 Ixora		TX2 Developer Kit	
	$o_{i,1}$ [W]	$a_{i,1}$ [W]	$o_{i,1}$ [W]	$a_{i,1}$ [W]	$o_{i,1}$ [W]	$a_{i,1}$ [W]
a2time-4K	0.25	0.25	0.24	0.24	0.29	0.58
a2time-4M	1.38	0.35	0.95	0.32	0.51	0.70
aifirf-4K	0.20	0.26	0.23	0.24	0.34	0.84
aifirf-4M	0.95	0.37	0.66	0.32	0.51	0.89
bitmnp-4K	0.21	0.22	0.25	0.20	0.28	0.46
bitmnp-4M	1.20	0.29	0.83	0.27	0.50	0.49
canrdr-4K	0.16	0.37	0.23	0.33	0.34	0.75
canrdr-4M	1.36	0.41	0.93	0.38	0.59	1.00
idctrn-4K	0.23	0.28	0.23	0.26	0.36	0.86
idctrn-4M	1.15	0.35	0.77	0.31	0.50	0.92
iirflt-4K	0.21	0.25	0.22	0.23	0.33	0.77
iirflt-4M	1.20	0.32	0.80	0.29	0.59	0.80
matrix-4K	0.22	0.35	0.20	0.33	0.33	0.92
matrix-4M	0.65	0.51	0.46	0.43	0.43	0.94
membench-1K-RO-R	0.22	0.17	0.23	0.15	0.25	0.34
membench-1K-RO-S	0.24	0.16	0.22	0.15	0.23	0.34
membench-1K-RW-R	0.25	0.20	0.21	0.18	0.29	0.62
membench-1K-RW-S	0.23	0.19	0.21	0.18	0.31	0.63
membench-1M-RO-R	0.37	0.57	0.57	0.35	0.32	0.41
membench-1M-RO-S	0.55	0.77	0.57	0.53	0.40	0.56
membench-1M-RW-R	0.48	0.63	0.57	0.40	0.29	0.54
membench-1M-RW-S	0.40	0.90	0.60	0.53	0.28	0.88
membench-4M-RO-R	1.81	0.24	1.23	0.19	0.48	0.37
membench-4M-RO-S	1.81	0.45	1.27	0.36	0.69	0.48

membench-4M-RW-R	1.78	0.31	1.24	0.23	0.49	0.49
membench-4M-RW-S	2.02	0.48	1.37	0.34	1.21	0.65
pntrch-4K	0.23	0.26	0.21	0.24	0.40	0.76
pntrch-4M	0.44	0.41	0.31	0.34	0.40	0.77
puwmod-4K	0.25	0.32	0.22	0.29	0.32	0.69
puwmod-4M	1.36	0.36	0.91	0.33	0.59	0.81
rspeed-4K	0.19	0.38	0.20	0.33	0.41	0.84
rspeed-4M	1.52	0.49	1.04	0.43	0.91	1.23
tblook-4K	0.21	0.25	0.20	0.24	0.40	0.80
tblook-4M	1.33	0.26	0.82	0.27	0.43	0.90
tinyrenderer-boggie	1.37	0.29	0.90	0.26	0.49	0.59
tinyrenderer-diablo	1.33	0.28	0.86	0.26	0.46	0.58
ttsprk-4K	0.19	0.40	0.19	0.34	0.39	0.67
ttsprk-4M	−0.16	0.71	0.16	0.48	0.38	0.71

Table 3.5: Task characteristics coefficients identified for used kernels on big cluster.

	I.MX8 MEK		I.MX8 Ixora		TX2 Developer Kit	
	$o_{i,2}$ [W]	$a_{i,2}$ [W]	$o_{i,2}$ [W]	$a_{i,2}$ [W]	$o_{i,2}$ [W]	$a_{i,2}$ [W]
a2time-4K	0.16	1.05	0.19	0.98	0.36	1.51
a2time-4M	1.63	1.14	1.10	1.08	0.50	2.05
aifirf-4K	0.14	1.47	0.17	1.39	0.38	1.81
aifirf-4M	1.08	1.72	0.70	1.61	0.48	1.89
bitmnp-4K	0.22	0.74	0.21	0.72	0.35	1.49
bitmnp-4M	1.31	0.88	0.93	0.79	0.53	1.45
canrdr-4K	0.29	1.08	0.14	1.11	0.37	1.84
canrdr-4M	1.59	1.32	1.05	1.26	0.55	2.23
idctrn-4K	0.19	1.51	0.10	1.50	0.36	2.11
idctrn-4M	1.27	1.68	0.71	1.65	0.49	2.21
iirflt-4K	0.30	1.31	0.21	1.28	0.42	1.38
iirflt-4M	1.43	1.47	0.85	1.44	0.54	1.59
matrix-4K	0.90	1.30	0.41	1.59	0.38	1.79
matrix-4M	0.99	1.92	0.59	1.79	0.38	1.83
membench-1K-RO-R	0.25	0.41	0.21	0.41	0.37	0.60
membench-1K-RO-S	0.25	0.42	0.19	0.41	0.38	0.58
membench-1K-RW-R	0.22	0.83	0.18	0.78	0.42	1.34
membench-1K-RW-S	0.21	0.81	0.17	0.77	0.39	1.36
membench-1M-RO-R	1.48	0.67	1.05	0.52	0.97	0.28
membench-1M-RO-S	1.22	1.24	0.96	0.90	−0.27	1.87
membench-1M-RW-R	1.58	0.81	1.15	0.55	1.01	0.60
membench-1M-RW-S	1.86	1.01	1.43	0.66	−0.75	2.86
membench-4M-RO-R	1.80	0.51	1.33	0.37	0.39	0.54
membench-4M-RO-S	3.18	0.26	2.19	0.30	0.90	1.66
membench-4M-RW-R	1.90	0.55	1.37	0.43	0.43	0.67
membench-4M-RW-S	3.17	0.35	2.35	0.29	1.63	2.02
pntrch-4K	0.39	1.25	0.10	1.33	0.32	1.58
pntrch-4M	0.74	1.59	0.40	1.47	0.44	1.55
puwmod-4K	0.24	1.19	0.17	1.14	0.37	1.80
puwmod-4M	1.53	1.39	0.95	1.35	0.52	1.99
rspeed-4K	0.15	1.65	0.14	1.47	0.39	1.24
rspeed-4M	1.76	2.21	1.14	2.08	0.75	1.70
tblook-4K	0.29	1.50	0.15	1.45	0.38	1.48
tblook-4M	1.35	1.68	0.78	1.62	0.47	1.61
tinyrenderer-boggie	1.47	1.09	1.02	0.96	0.52	0.98
tinyrenderer-diablo	1.34	1.14	0.93	1.00	0.52	0.91
ttsprk-4K	0.20	1.29	0.24	1.15	0.41	1.46
ttsprk-4M	−0.34	2.06	0.05	1.58	0.35	1.50

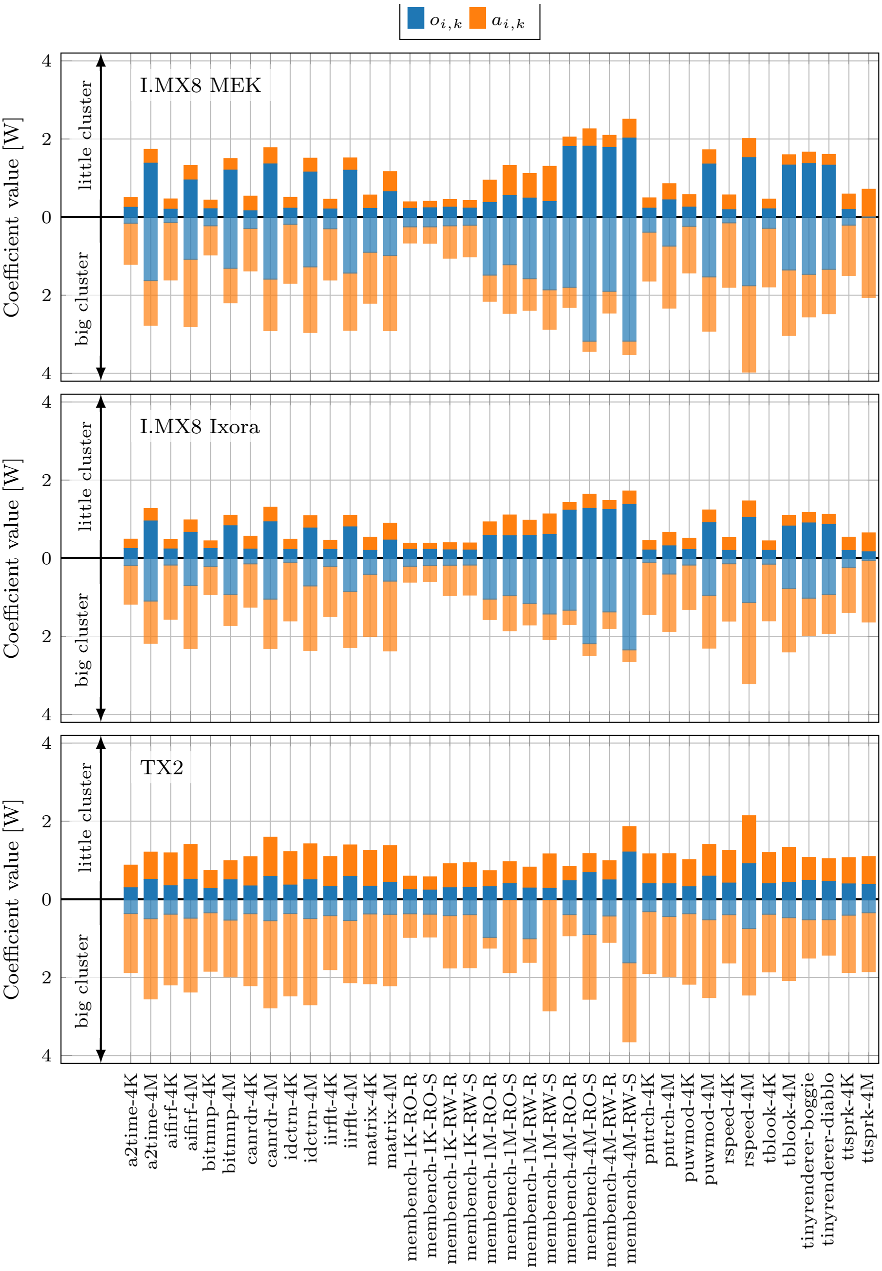


Figure 3.10: Values of task characteristics coefficients $o_{i,k}$ and $a_{i,k}$ of tested benchmarks on little (top semi-axis) and big (bottom semi-axis) clusters.

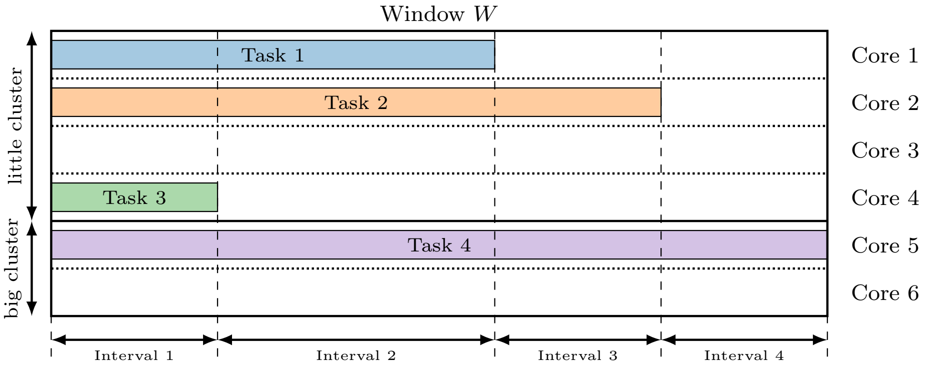


Figure 3.11: Illustration of processing-idling intervals needed for the `LR` model.

3.6.3.2 Linear Regression Model

The sum-max model was designed to estimate the average power consumption of a whole isolation window. Its simple form allows for relatively straightforward integration with the optimization methods, e.g., with Integer Linear Programming, as shown in [20]. However, the model may fail to provide accurate prediction when the tasks of very distinct lengths are present in the window. Specifically, the max term counts with the largest $o_{i,k}$ only, which might represent one of the short tasks that possibly ends early in the window. In that case, the predicted power might overestimate the actual one.

To overcome this shortcoming, we designed another power estimation model based on linear regression, which was successfully used in the context of power consumption estimation [117]. Instead of estimating the average power of the whole window, we split the window into several intervals, within which each core either executes a single benchmark for the whole time or remains idle for the whole time. We call them *processing-idling* intervals. The situation is illustrated in Figure 3.11. Then, the model estimates the power consumption of each such interval. Similarly to the `SM` model, the overall average power consumption is then estimated by a weighted average of the intervals' individual contributions.

Thanks to the decomposition into the processing-idling intervals, data acquisition and fitting of the model become easier since the timing and overlaps of the individual tasks do not need to be considered (each core is either processing or idling for the whole interval). A further advantage is that such a model can be used even if the temporal isolation constraints (windows) are not considered since essentially any multi-core schedule can be divided into such intervals (simply by projecting the start times and end times of tasks to the time axis; the intervals are then defined by every two consecutive projected time points). On the contrary, the integration of the `LR` model with the optimization might be harder since the lengths of the processing-idling intervals are not known a priori as they depend on the allocation of the tasks to windows.

To describe the linear regression model (`LR`), let us assume that we have some interval I with allocated tasks. By $i(k, r)$ we denote the index of task, which is allocated in interval I to cluster $C_k \in \mathcal{C}$ and its core $r \in \{1, \ldots, c_k\}$. If the core is idle, we assume $i(k, r) = 0$ (where τ_0 represents an idle task). Now, let us assume that behavior of task τ_i on each cluster C_k is characterized by a vector of real numbers $\hat{\boldsymbol{x}}_{i,k}$ (e.g., $\hat{\boldsymbol{x}}_{i,k} = (o_{i,k}, a_{i,k})$). We assume that idle task τ_0 is characterized by zero vector, $\hat{\boldsymbol{x}}_{0,k} = \mathbf{0}\ \forall C_k \in \mathcal{C}$. Then, the average power consumption of interval I can be estimated by `LR` model as:

$$P(I) = \sum_{C_k \in \mathcal{C}} \sum_{r=1}^{c_k} \left(\hat{\boldsymbol{x}}_{i(k,r),k} \circ \boldsymbol{\beta}_{k,r} \right) + P_{\text{idle}}, \tag{3.5}$$

where $\circ$ is the scalar product operator and P_{idle} is the constant intercept term. Note that when no task is executed (all are idle), the prediction is exactly P_{idle}, which is the platform's idle power consumption. In this general form, the regression coefficients $\boldsymbol{\beta}_{k,r}$ are possibly different for each core. However, we assume that all cores of each cluster are identical, so arbitrary permutation of tasks allocated to a single cluster should lead to the same power consumption. By this, we can simplify the model (3.5) to:

$$\begin{aligned} P(I) &= \sum_{C_k \in \mathcal{C}} \sum_{r=1}^{c_k} \left(\hat{\boldsymbol{x}}_{i(k,r),k} \circ \boldsymbol{\beta}_{k} \right) + P_{\text{idle}} \\ &= \sum_{C_k \in \mathcal{C}} \boldsymbol{\beta}_k \circ \left(\sum_{r=1}^{c_k} \hat{\boldsymbol{x}}_{i(k,r),k} \right) + P_{\text{idle}}, \end{aligned} \tag{3.6}$$

where all cores of the single cluster have the same regression coefficients. For simpler understanding of the calculation of `LR` power model, we present a numerical example at the end of this chapter.

Platform and Tasks Characteristics. Similarly to `SM` model, `LR` model needs just the idle power P_{idle} as platform characteristic. On the other hand, the individual tasks characteristics now correspond to the elements of input vector $\hat{\boldsymbol{x}}_{i,k}$. For simplicity and better comparison, we can use already identified coefficients $o_{i,k}$ and $a_{i,k}$, which are also the only tasks characteristics used by `SM` model. In general, we could also include more characteristics obtained, e.g., by monitoring the performance counters during the tasks execution [118].

Identification of Regression Coefficients. To identify the regression coefficients, we created 1000 unique instances (each representing one interval I), which were randomly populated with the benchmarking kernels described in Section 3.5.2. Specifically, each interval was 1 s long and contained from zero (all cores idling) up to 6 (all cores processing) kernels randomly picked from the set. All these instances were executed on all tested platforms (each for 180 s), and the average power consumption was measured. Detailed evaluation of `LR` model and its comparison with `SM` model is presented in Section 3.8.1. Here, we just report the identified coefficients (i.e., elements of vectors $\boldsymbol{\beta}_k$ for each cluster $C_k \in \mathcal{C}$) in Table 3.6.

Example. To illustrate the the calculation of both power models numerically, let us assume three tasks, as illustrated in Fig. 3.11. The first task executes *a2time-4K* kernel for 450 ms at little cluster ($k = 1$). The second task, also assigned to a the little cluster, executes *canrdr-4M* kernel for 550 ms. Finally, the third

Table 3.6: Regression coefficients identified for all tested platforms and coefficient of determination R^2.

	little cluster ($\boldsymbol{\beta_1}$)		big cluster ($\boldsymbol{\beta_2}$)		
platform	$\beta_{1,1}$	$\beta_{2,1}$	$\beta_{1,2}$	$\beta_{2,2}$	R^2
I.MX8 MEK	1.205	0.270	0.969	0.456	0.822
I.MX8 Ixora	1.227	0.232	0.981	0.420	0.814
TX2	0.857	0.648	0.946	0.801	0.974
corresponding independent var. →	$a_{i,1}$	$o_{i,1}$	$a_{i,2}$	$o_{i,2}$	

task, which is assigned to the big cluster ($k = 2$), executes *membench-1M-RO-S* kernel for 700 ms. The length of the window is hence 700 ms. Considering the I.MX8 MEK, the relevant task characteristics coefficients (see Table 3.4 and Table 3.5) are: $a_{1,1} = 0.25$, $o_{1,1} = 0.25$, $a_{2,1} = 0.41$, $o_{2,1} = 1.36$, $a_{3,2} = 1.24$, and $o_{3,2} = 1.22$. The `SM` power model just takes the maximum offset coefficient (1.36) and adds the activity coefficients contributions of the individual tasks, i.e., the estimated power consumption is $P_{\text{idle}} + 1.36 + \left(0.25 \cdot \frac{450}{700} + 0.41 \cdot \frac{550}{700} + 1.24 \cdot \frac{700}{700}\right) \doteq 8.58\,\text{W}$. To evaluate the `LR` model, the whole window is split to three processing-idling intervals. The first interval is 450 ms long and all three tasks are executed during it. The second interval is 100 ms long and covers the second and third task. The last interval is 150 ms long, and covers only the third task. The power is estimated individually for each interval, and averaged out in the end. The linear regression coefficients are listed in Table 3.6. For the first interval, the prediction is $[\beta_{1,1}(a_{1,1} + a_{2,1}) + \beta_{1,2} \cdot a_{3,2}] + [\beta_{2,1}(o_{1,1} + o_{2,1}) + \beta_{2,2} \cdot o_{3,2}]$, which is $[1.205 \cdot (0.25 + 0.41) + 0.969 \cdot 1.24] + [0.270 \cdot (0.25 + 1.36) + 0.456 \cdot 1.22] \doteq 2.99$. Since this interval lasts only 450 ms, this number is then averaged to $2.99 \cdot \frac{450}{700} \doteq 1.92\,\text{W}$. The calculations for the other two intervals are analogous; we report just their averaged contributions, which are 0.37 W and 0.38 W, respectively. In total, the power predicted by `LR` power model is $P_{\text{idle}} + 1.92 + 0.37 + 0.38 = 8.17\,\text{W}$. The visual comparison is shown in Fig. 3.12.

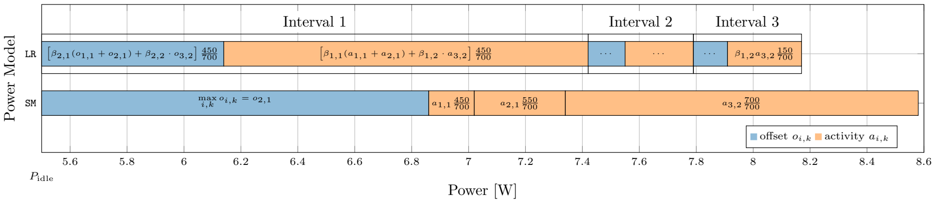


Figure 3.12: Visualization of power models evaluation.

3.7 Optimization Methods

In Section 3.6, we revised the thermal modeling, and summarized two specific power models, namely `SM` and `LR` models. Further, we identified all the necessary platform and task characteristics. In this section, we continue with the integration. We take the power models and design optimization methods that solve the problem of safety-critical task allocation while minimizing the estimated power consumption.

First, we summarize the optimizer based on Integer Linear Programming (ILP) and `SM` model proposed in [20] in Section 3.7.1. Second, we discuss the optimization based on `LR` power model in Section 3.7.2. Finally, we introduce an informed greedy heuristic and uninformed idle-time optimizers used for further comparison in Section 3.7.3 and Section 3.7.4, respectively.

3.7.1 ILP and Sum-Max Model

The original implementation proposed in [20] relies on a simple encoding of the scheduling problem to the ILP formalism. Binary variable $x_{i,j,k}$ is equal to 1 if and only if task $\tau_i \in \mathcal{T}$ is allocated to window $W_j \in \mathcal{W}$ and cluster $C_k \in \mathcal{C}$. All the resource constraints can be then simply written as:

$$\sum_{W_j \in \mathcal{W}} \sum_{C_k \in \mathcal{C}} x_{i,j,k} = 1 \quad \forall \tau_i \in \mathcal{T}, \tag{3.7}$$

$$\sum_{\tau_i \in \mathcal{T}} x_{i,j,k} \leq c_k \quad \forall W_j \in \mathcal{W}, C_k \in \mathcal{C}, \tag{3.8}$$

meaning that each task is assigned to some cluster and core and that capacity of each cluster is respected. To model the length of the individual windows, continuous variable $\hat{l}_j$ is introduced for each window $W_j \in \mathcal{W}$. Length is then linked to the assignment variables by

$$\hat{l}_j \geq x_{i,j,k} \cdot e_{i,k} \quad \forall \tau_i \in \mathcal{T}, W_j \in \mathcal{W}, C_k \in \mathcal{C}, \tag{3.9}$$

and constrained by the major frame length h:

$$\sum_{W_j \in \mathcal{W}} \hat{l}_j \leq h. \tag{3.10}$$

Finally, the `SM` model (3.4) is linearized to fit the ILP formalism and rewritten to the objective function (3.11). The idle power P_{idle} is not included in the objective since it is considered to be constant. The power consumption predictions are averaged over all windows with the weights corresponding to the windows lengths. The non-linear *max* term is (in each window $W_j \in \mathcal{W}$) replaced by continuous variable y_j, which serves as its upper bound. When the solver reaches the optimum, this upper bound becomes tight. The link between Y_j and $x_{i,j,k}$ is formed by *big-M*, where M is sufficiently large constant. The objective (3.11) represents the estimated average power consumption. The whole ILP model encoding the scheduling problem and integrating `SM` model then becomes:

$$\texttt{ILP-SM:} \quad \min \frac{1}{h} \sum_{W_j \in \mathcal{W}} \left(\sum_{\tau_i \in \mathcal{T}} \sum_{C_k \in \mathcal{C}} (x_{i,j,k} \cdot a_{i,k} \cdot e_{i,k}) + y_j \right) \tag{3.11}$$

subject to:

$$y_j \geq o_{i,k} \cdot \hat{l}_j - M \cdot (1 - x_{i,j,k}) \quad \forall \tau_i \in \mathcal{T}, W_j \in \mathcal{W}, C_k \in \mathcal{C}, \tag{3.12}$$

(3.7), (3.8), (3.9), (3.10).

3.7.2 Optimization Based on the Linear Regression Model

Contrary to `SM` model, which predicts the power for each isolation window, `LR` model predicts the power for processing-idling intervals only. Direct integration within the ILP formalism proved to be quite laborious[2]; therefore, we followed different paths. First, we neglect the processing-idling intervals and assume that each task executes for the whole duration of the window to which it is allocated. We call this simplified model as `LR-UB`. Then we can simply formulate the optimization as a quadratic programming optimization problem as shown in Section 3.7.2.1. Second, we use a different optimization framework, namely the black-box optimization based on a genetic algorithm, which can simply integrate the `LR` model as a part of the fitness evaluation. We describe this approach in Section 3.7.2.2.

3.7.2.1 Integration of LR to QP

We assume that all tasks allocated to one isolation window are executed for the whole length of the window. Therefore, the execution times of the individual tasks are potentially assumed to be longer than they are in reality. In consequence, processing-idling intervals are completely neglected (each whole window becomes a single processing-idling interval). In Fig. 3.13, we illustrate the assumed extensions of the task execution times in gray (the original processing-idling intervals are shown in Fig. 3.11). In a sense, the idea is similar to the *max* term in the `SM` model. We hope that by minimizing the upper bound instead of the original objective, we get a schedule that performs reasonably well in practice while keeping the formulation relatively simple.

Then, we follow the same steps as for the integration of the `SM` model with the ILP. We use the same binary variables $x_{i,j,k}$ deciding whether task $\tau_i \in \mathcal{T}$ is allocated to window $W_j \in \mathcal{W}$ and cluster $C_k \in \mathcal{C}$. Constraints modeling the task allocation and the resource capacity and limiting the major frame length are the same as before, only the objective changes, as the power is now predicted by the `LR` model for each window. The whole model is now as follows:

---

[2] We implemented the model, but it became rather complex, mainly due to the necessary linearization, and its performance was poor even for small instances (we failed to obtain reasonable upper and lower bounds for instance with 20 tasks even after one day of solving by state-of-the-art solver Gurobi).

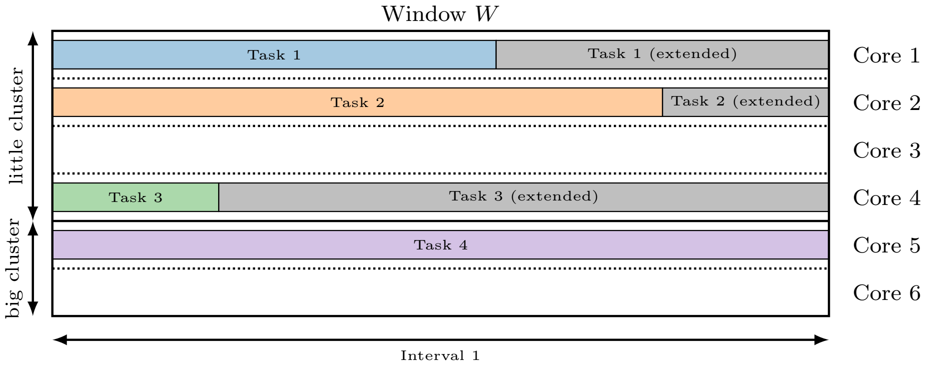


Figure 3.13: Illustration of a simplified window with a single processing-idling interval only, as used by `LR-UB`.

$$\texttt{QP-LR-UB:} \quad \frac{1}{h} \cdot \sum_{W_j \in \mathcal{W}} \hat{l}_j \cdot \sum_{\tau_i \in \mathcal{T}} \sum_{C_k \in \mathcal{C}} x_{i,j,k} \cdot \underbrace{(a_{i,k} \cdot \beta_{1,k} + o_{i,k} \cdot \beta_{2,k})}_{\star} \tag{3.13}$$

subject to:
(3.7), (3.8), (3.9), (3.10).

Clearly, the objective becomes quadratic (due to multiplication of $x_{i,j,k}$ and $\hat{l}_j$).[3] Note that for all the texted benchmarking kernels (see Table 3.4 and Table 3.5) and identified linear regression coefficients (see Table 3.6), the expression denoted by $\star$ in (3.13) is positive. Furthermore, $\star$ is zero for the idle task. Therefore, the objective (3.13) becomes an upper bound on the original `LR` value (by the original `LR` we mean the `LR` model, which does not neglecting the processing-idling intervals).

3.7.2.2 Black-Box Optimizer

Instead of using Mathematical Programming and building a complicated model, we could also use the black-box optimization framework, which is conceptually different. The objective function is not given in a closed form, but only its outputs can be observed provided the inputs. For us, given the full tasks allocation (schedule), we can compute the average power consumption based on the `LR` model (or possibly any other model). The black-box optimization algorithm searches through the space of all allocations and tries to find the best one w.r.t the given fitness function (here the `LR` model).

There are many algorithms that can be used for black-box optimization, including, for example, Particle Swarm Optimization, Differential Evolution, or Genetic Algorithms. Some of these algorithms are already implemented in existing libraries, which are often optimized for speed, easily usable, and open-source. We use *Genetic algorithm* (`GA`) from *Evolutionary* package implemented in Julia [131]. We use

---

[3] We use Gurobi solver for both Linear and Quadratic optimization.

standard two-point crossover and BGA mutation; mutation and crossover rates are set to 0.2 and 0.8, respectively. The selection is done according to a uniform ranking scheme (discarding the lowest 10 % of the population), and the population size is set to $50 \cdot |\mathcal{T}|$.

We represent the position of each task $\tau_i \in \mathcal{T}$ in the schedule by continuous variable $x_i \in [0, 1)$. In order to optimize the allocation problem using the continuous variables, we introduce the following transformation: Each variable x_i is evenly split to m intervals, i.e., for two clusters, we obtain interval $[0, 0.5)$ representing the allocation to the first cluster and interval $[0.5, 1)$ representing the allocation to the second one. Each such sub-interval is then again evenly split to q intervals representing the allocation to the individual windows.

Still, it might happen that allocation represented by variables x_i would be infeasible – either due to the major frame length (when allocated windows are too long) or due to resource capacity constraints (when too many tasks are allocated to the same window and resource). There are several ways to handle this issue. One option is to use such a black-box solver that supports constrained optimization (`GA` can do that). Another option is to introduce post-processing that would try to reconstruct some feasible solution from the infeasible assignment. Even though it would appear that solely the former option solves the problem, too many infeasible solutions slow down the convergence of the optimization algorithm. Therefore, we use both presented options – the former for the major frame length constraint and the latter for the resource constraints.

The post-processing (reconstruction) procedure is described by Algorithm 2. Informally, the preferred allocation of the tasks is pre-computed based on the transformation described above. Then, starting from the first window, the allocation of the tasks is iteratively fixed. If the task cannot be added to the current window (i.e., the resource capacity would be exceeded), its preferred allocation is moved to the next window (in a cyclic manner). The iteration over all windows is repeated twice. If there are still some unassigned tasks or the major frame length is exceeded, the solution is discarded; otherwise, feasible allocation of the tasks to windows and clusters is returned.

The black-box optimizer iterates over many possible instantiations of x_i. Every time some instantiation is tested, the schedule (defined by allocations created by Algorithm 2) is reconstructed and evaluated by the `LR` power model. After a termination condition is met (e.g., time limit or iteration limit is exceeded), the best-so-far solution is returned. We execute the algorithm with a pre-defined time limit; if it terminates sooner, it is restarted from a random instantiation of x_i until exhausting the time limit.

```
input  : instantiation of variables $x_i \ \forall \tau_i \in \mathcal{T}$
output : Functions mapping tasks to windows and tasks to clusters (or failure)
$TaskAssignmentPreference(i) \leftarrow \frac{x \% \frac{1}{m}}{\frac{1}{q \cdot m}} \quad \forall \tau_i \in \mathcal{T}$
$TaskAssignmentCluster(i) \leftarrow \lfloor \frac{x}{m} \rfloor + 1 \quad \forall \tau_i \in \mathcal{T}$
$TaskAssignmentWindow(i) \leftarrow \lfloor TaskAssignmentPreference(i) \rfloor + 1 \quad \forall \tau_i \in \mathcal{T}$
$WindowCapacity(j, k) \leftarrow c_k \quad \forall W_j \in \mathcal{W}, C_k \in \mathcal{C}$
$TasksInWindow(j) \leftarrow \{\} \quad \forall W_j \in \mathcal{W}$
$WindowLength(j) \leftarrow 0 \quad \forall W_j \in \mathcal{W}$
$TaskAssigned(i) \leftarrow False \quad \forall \tau_i \in \mathcal{T}$
for $Iteration \in \{0, 2, \ldots, 2 \cdot q - 1\}$ do
    $CurrentWindow \leftarrow (Iteration \,\%\, q) + 1$
    $TasksToCurrentWindow \leftarrow \{i \mid \tau_i \in \mathcal{T} \wedge TaskAssignmentWindow(i) = CurrentWindow \wedge \neg TaskAssigned(i)\}$
    sort $TasksToCurrentWindow$ by $TaskAssignmentPreference$ in non-decreasing order
    for $i \in TasksToCurrentWindow$ do
        if $WindowCapacity(j, TaskAssignmentCluster(i)) > 0$ then
            $WindowCapacity(j, TaskAssignmentCluster(i)) \mathrel{-}= 1$
            $TasksInWindow(j) \leftarrow TasksInWindow(j) \cup \{i\}$
            $TaskAssigned(i) \leftarrow True$
            $WindowLength(j) \leftarrow \max\{WindowLength(j), e_{i,TaskAssignmentCluster(i)}\}$
        else
            if $CurrentWindow = q - 1$ then
                $TaskAssignmentWindow(i) \leftarrow CurrentWindow + 1$
            else
                $TaskAssignmentWindow(i) \leftarrow (CurrentWindow + 1) \,\%\, q$
            $TaskAssignmentPreference(i) \leftarrow 0$
if $\sum_{W_j \in \mathcal{W}} WindowLength(j) > h \vee \neg \bigwedge_{\tau_i \in \mathcal{T}} TaskAssigned(i)$ then
    return Schedule reconstruction failed.
else
    return $(TaskAssignmentWindow, TaskAssignmentCluster)$
```

Algorithm 2: Get a feasible allocation from the instantiation of variables x_i, or report a failure.

3.7.3 Greedy Heuristic

As a reference method, we describe a greedy heuristic. Contrary to all the previous methods based on ILP, QP, or black-box optimization, the greedy heuristic does not try to search through the whole optimization space (here, set of all possible allocations). Instead, the search space is intentionally restricted in order to decrease the computation time and improve the scalability.

The heuristic that we present is based on the works of Zhou et al. [142] and Kuo et al. [74], but its main idea is rather general and applicable in the wider context. The tasks are sorted by their energy consumption and processed one by one in a non-increasing order (the most energy-consuming task goes first). In each iteration, the currently processed task is assigned to the cheapest computing cluster (w.r.t. energy consumption). The assignment is done only if some feasible schedule exists even for all the remaining (still unprocessed) tasks, i.e., the assignment cannot be fixed if it would cause infeasibility.

For the tasks ordering, we use analogous methodology that is used in [142] (in Algorithm 1) – we can identify the parameter μ_i used in [142] with task characteristic $a_{i,k}$ since both of these parameters represent tasks dynamic power consumption to some extent. Then, the task τ_i is assigned to cluster $C_k \in \mathcal{C}$ that minimizes $a_{i,k} \cdot e_{i,k}$ (i.e., expected task energy consumption). Before each assignment, feasibility needs to be checked. When considering the temporal isolation windows, it becomes a bit tricky because these windows make the scheduling on the individual clusters and their cores dependent on each other (without the windows, the situation is much simpler since only the utilization bound needs to be checked).

To check the feasibility, we use a modified ILP model as presented in Section 3.7.1:

$$
\begin{aligned}
\texttt{ILP-FEAS:}\quad & \min 0 && (3.14)\\
& \text{subject to:} \\
& \sum_{W_j \in \mathcal{W}} x_{i,j,r(\tau_i)} = 1 \quad \forall \tau_i \in \mathcal{T}_{\text{fixed}} && (3.15)\\
& (3.7),\ (3.8),\ (3.9),\ (3.10),
\end{aligned}
$$

where $\mathcal{T}_{\text{fixed}}$ represents the set of tasks with already fixed assignment and $r : \mathcal{T} \rightarrow \{1, \ldots, m\}$ maps the tasks with fixed assignment to the index of their assigned cluster. The whole greedy heuristic is summarized in Algorithm 3.

Note that solving `ILP-FEAS` model is easier compared to `ILP-SM` as the solver can terminate after finding any feasible solution, whereas in the latter case, it needs to explore the whole search space somehow (iterating over multiple feasible solutions).

```
input  : set of tasks $\mathcal{T}$, set of clusters $\mathcal{C}$, major frame length $h$
output : assignment of the tasks to resources
Function CheckFeasibility($\mathcal{T}_{fixed} \subseteq \mathcal{T}, r : \mathcal{T}_{fixed} \rightarrow \{1, \dots, m\}$) is
    if a feasible solution to ILP-FEAS with fixed tasks assignment of $\mathcal{T}_{fixed}$ given by
      $r$ exists then
        return true
    else return false;
sort all tasks $\tau_i \in \mathcal{T}$ by $\max_{C_k \in \mathcal{C}}\{a_{i,k} \cdot e_{i,k}\}$ in non-increasing order
$\mathcal{T}_{\text{fixed}} = \{\}$
foreach task $\tau_i \in \mathcal{T}$ do
    $\mathcal{C}_{\text{sorted}} \leftarrow$ sort $\mathcal{C}$ by $\max_{C_k \in \mathcal{C}}\{a_{i,k} \cdot e_{i,k}\}$ in non-decreasing order
    foreach cluster $C_k \in \mathcal{C}_{sorted}$ do
        assign $\tau_i$ to $C_k$, $r(\tau_i) = k$
        if CheckFeasibility($\mathcal{T}_{fixed} \cup \{\tau_i\}, r$) then
            $\mathcal{T}_{\text{fixed}} \leftarrow \mathcal{T}_{\text{fixed}} \cup \{\tau_i\}$
            break
if $\mathcal{T}_{fixed} = \mathcal{T}$ then
    return assignment of tasks to clusters and windows given by the solution of
      ILP-FEAS with fixed cluster assignments defined by r
else
    error: feasible assignment of tasks to resources does not exist
```

Algorithm 3: Greedy heuristic.

3.7.4 Optimizer Minimizing/Maximizing the Idle Time

Finally, we present two more optimizers, this time uninformed, i.e., not using any task or platform characteristics. These methods simply optimize the idle (i.e., non-processing) time within the major frame. We present them as ILP models.

First, the total processing time $t_{\text{processing}}$, can be expressed in terms of variables $x_{i,j,k}$ introduced in `ILP-SM` as follows:

$$t_{\text{processing}} = \sum_{\tau_i \in \mathcal{T}} \sum_{W_j \in \mathcal{W}} \sum_{C_k \in \mathcal{C}} x_{i,j,k} \cdot e_{i,k}. \tag{3.16}$$

Then the total idle time t_{idle} within the major frame is

$$t_{\text{idle}} = \left(h \cdot \sum_{C_k \in \mathcal{C}} c_k \right) - t_{\text{processing}}. \tag{3.17}$$

Now, the first model, `ILP-IDLE-MAX`, maximizes the idle time in the hope that long idle periods allow for the platform to cool down. Also, a schedule with maximal idle time is beneficial from the perspective of the practitioner. Sometimes the instance changes and some more tasks need to be added for the execution; in such a case, schedules with long idle periods offer the space to do so. The model can be formalized as:

$$\texttt{ILP-IDLE-MAX:} \quad \max t_{\text{idle}} \quad \text{subject to: } (3.7), (3.8), (3.9), (3.10), (3.16), (3.17). \tag{3.18}$$

Contrary to that, the second model, `ILP-IDLE-MIN` minimizes the idle time. The idea is that longer execution time is typically associated with the little cluster (see Figure 3.5), which might be, however, more power-efficient. The model is described as

$$\texttt{ILP-IDLE-MIN:} \quad \min t_{\text{idle}} \quad \text{subject to: } (3.7), (3.8), (3.9), (3.10), (3.16), (3.17). \tag{3.19}$$

3.8 Experimental Evaluation and Results

To evaluate the described power models and the optimization methods, we conduct a series of experiments on the three physical platforms. First, we assess the quality of the power models in Section 3.8.1. Next, we compare the optimization methods with respect to the capability of reducing the peak temperature (Section 3.8.2) and based on their computational complexity and scalability (Section 3.8.3).

3.8.1 Power Model Evaluation

For each tested platform, we generate one thousand instances of *CPU-bound workload* (all kernels except membench) and one thousand instances of *mixed workload* (all kernels including membench), that is two thousand distinct instances in total. The workload consists of a single periodically repeated window with a length of 1 s. In that window, each CPU executes either nothing (probability 0.5) or a random kernel. The kernels are executed for duration uniformly selected from interval 1 ms to 1000 ms.

Each instance was executed for 60 s; this gives more than four days (100 hours) of measured data in total. The power consumption was sampled every 10 ms, and the average value was reported. Further, we calculated the predicted power by `SM`, `LR`, and `LR-UB` power models.

The results for mixed workload instances are shown in Fig. 3.14. The instances are sorted by the measured power consumption on I.MX8 MEK. The Table 3.7 shows the mean absolute error of all power models on both types of workload. We observe that the lowest prediction error is achieved by the linear regression (`LR`) model. In relative terms (w.r.t. the idle power consumption), its error is 4.3 %, 4.5 % and 13.3 % for I.MX8 MEX, I.MX8 Ixora and TX2, respectively. `SM` model performed slightly worse, with average relative error of 11.2 %, 5.6 %, and 16.0 % for I.MX8 MEX, I.MX8 Ixora and TX2. Finally, `LR-UB` model failed to deliver satisfactory predictions; its relative error is 24.3 %, 19.3 %, and 74.4 % for I.MX8 MEX, I.MX8 Ixora and TX2.

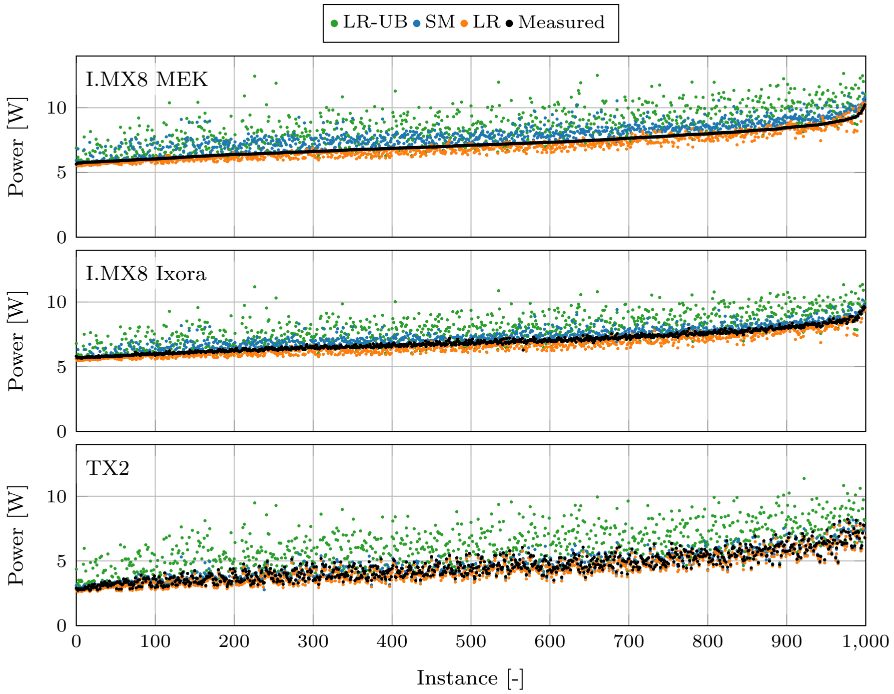


Figure 3.14: Measured and predicted power consumption of 1000 testing instances (mixed workload windows); instances are sorted by I.MX8 MEK measured power consumption.

The trends are also clearly visible in Fig. 3.14; `SM` model is more pessimistic than the `LR` model, which is expected due to the max term in Eq. (3.4). However, it steadily provides an upper bound on the measured power consumption. Even though the `LR-UB` mostly provides an upper bound as well, it is not as tight.

3.8.2 Optimization Methods Comparison

Here, we discuss how well the power models integrate with the optimization. We compare the optimization methods on two types of workloads as in the previous section: CPU-bound and mixed. For each workload type, we construct six different instances. We follow the same instance generation scheme as in [20]. We generate

Table 3.7: Mean absolute error (in Watts) of the tested power models.

	SM		LR		LR-UB	
Platform	mixed	CPU	mixed	CPU	mixed	CPU
I.MX8 MEK	0.67	0.56	0.26	0.21	1.30	1.38
I.MX8 Ixora	0.35	0.27	0.28	0.21	1.00	1.12
TX2	0.17	0.66	0.24	0.45	1.54	2.33

Table 3.8: List of compared optimization methods and corresponding power models.

Acronym	Power model	Optimization method
`ILP-SM`	`SM`	ILP
`QP-LR-UB`	`LR-UB`	QP
`BB-LR`	`LR`	BB (generic alg.)
`BB-SM`	`SM`	BB (generic alg.)
`HEUR`	expected energy	greedy
`ILP-IDLE-MIN`	—	ILP
`ILP-IDLE-MAX`	—	ILP

20 tasks; each of them executes a randomly selected kernel. Each task is assigned a randomly generated execution time on the big cluster in the range 40 ms to 160 ms. Execution time for the little cluster is scaled appropriately to perform the same work. The scaling coefficient is calculated from Table 3.2. The major frame length h is calculated as $h = \frac{n \cdot \bar{e}}{\kappa}$, where $\bar{e}$ is the average execution time across all clusters, n is the number of tasks (here 20), and κ is empirical constant changing the tightness of the schedules (here set to 3.5).

For each instance, all optimization methods, as listed in Table 3.8, were executed to generate a schedule for each platform. For better comparison, we execute the black-box optimizer with both `SM` and `LR` power models. The solving time limit was set to 300 s per instance. The schedules found for the first instance are illustrated in Fig. 3.15.

During the experiment, each schedule was executed on the respective platform for 30 min; this gives 42 hours of measured data per platform. We measured the average power consumption and steady-state temperature. The power offset ($P_{\text{measured}} - P_{\text{idle}}$) is reported in Table 3.9 (the rows are sorted by the average power consumption on I.MX8 MEK). The `ILP-SM` method achieved the best results in almost all cases. The difference from the lowest result is negligible in the few cases where it was not the best. Slightly worse, but still good results were obtained by the `BB-SM` method. One practical difference between these two methods is that `ILP-SM` requires an ILP solver (here commercial Gurobi solver) for its operation, while `BB-SM` can be implemented with freely available tools.

An interesting observation is that the best results are obtained with the `SM` power model. Recall that the most accurate power model was `LR`, not `SM`. We account that to the fact that even though the `SM` is systematically overestimating the power consumption, it is consistent in a sense that windows with higher predicted power consumption indeed consume more than windows with lower predicted power consumption.

Figure 3.16 shows the temperatures near the individual clusters averaged over all six instances. We observe that the difference between the worst- and the best-performing methods are 5.5 °C, 4.9 °C, and 3.6 °C, corresponding to 22 %, 19.6 %, and 14.4 % differences relative to the ambient temperature (25 °C) for I.MX8 MEK, I.MX8 Ixora, and TX2.

When comparing the greedy local heuristic `HEUR` with the `ILP-SM` method, the exhaustive `ILP-SM` can save, in average, up to 1.6 °C, 1.3 °C, and 0.6 °C for

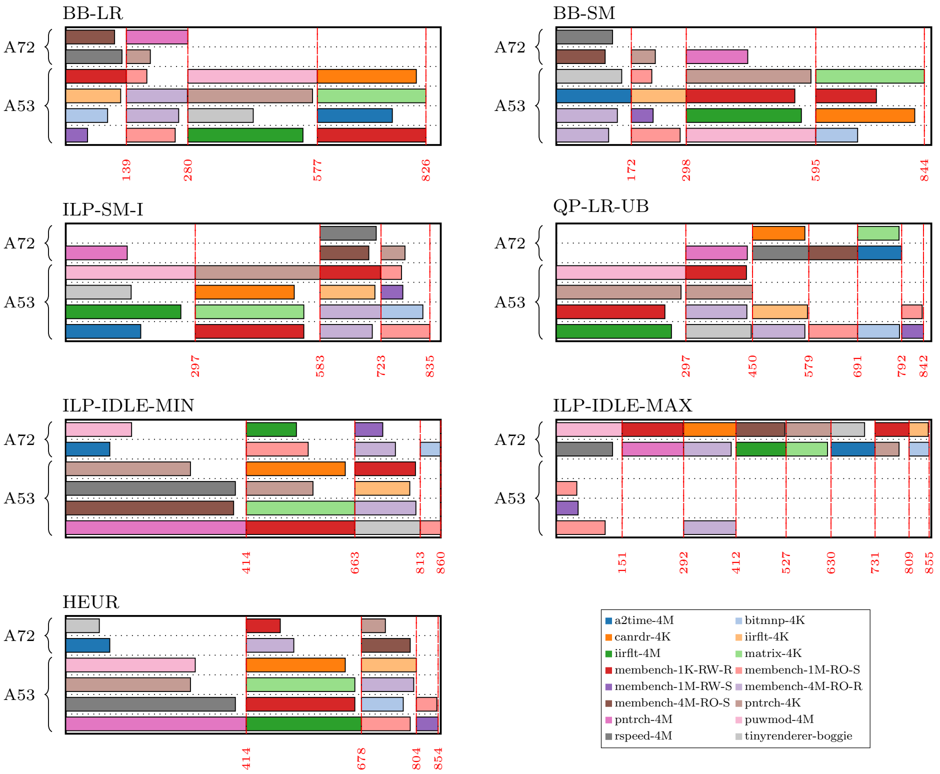


Figure 3.15: Example schedules for instance no. 1 on I.MX8 MEK (mixed-workload).

I.MX8 MEK, I.MX8 Ixora, and TX2, respectively.

3.8.3 Performance evaluation

Finally, we evaluate the scalability of tested optimization methods. We study how the computation time increases with the increasing instance size corresponding to the number of tasks n.

Ten instances are randomly generated for each $n \in \{5, 10, \ldots, 60\}$ (120 instances in total). Each optimization method is then executed for every instance; as some of the methods might be rather time-demanding for larger instance sizes, we limit the maximum computation time per instance to 300 s. We use the same generator as in Section 3.8.2, but we perform the experiment only with the characteristics based on I.MX8 MEK. The outcome would be quite similar for the other platforms.

The average computation times for different values of n are shown in Fig. 3.17. As the black-box optimizer (`BB`) is programmed to randomly restart each time it converges to some solution, it always consumes all the provided time. Besides, the models globally optimizing the schedule w.r.t. the provided objective, i.e., `QP-LR-UB` and `ILP-SM`, are the first to run out of time. Out of these two, the more complex

Table 3.9: Power offset ($P_{\text{measured}} - P_{\text{idle}}$ [W]) observed for six instances with mixed workloads and six instances with CPU-bound workloads on tested platforms.

	Mixed workloads instances						CPU-bound instances						
Method	1	2	3	4	5	6	1	2	3	4	5	6	average
							I.MX8 MEK						
ILP-SM	2.77	2.26	2.99	2.79	2.38	2.17	2.47	2.69	2.72	2.80	2.90	2.78	2.64
BB-SM	2.72	2.48	3.12	3.04	2.61	2.39	2.81	2.97	2.97	3.02	3.06	3.10	2.86
HEUR	3.16	2.71	3.17	3.20	3.12	2.44	2.90	3.01	3.00	3.26	3.16	3.36	3.04
BB-LR	3.09	2.82	3.36	3.22	3.01	2.63	2.94	3.03	2.98	3.18	2.93	3.31	3.04
ILP-IDLE-MIN	3.44	2.70	3.36	3.24	3.19	2.67	2.98	3.09	2.95	3.37	3.22	3.23	3.12
QP-LR-UB	3.61	2.84	3.50	3.48	3.24	2.44	3.12	3.51	3.11	3.35	3.30	3.21	3.23
ILP-IDLE-MAX	3.74	3.25	3.90	4.06	3.41	3.05	3.47	3.88	3.69	3.96	3.92	4.12	3.70
							I.MX8 Ixora						
ILP-SM	2.19	1.97	2.52	2.36	2.00	1.94	2.15	2.25	2.28	2.31	2.38	2.35	2.23
BB-SM	2.33	2.18	2.66	2.54	2.09	2.13	2.32	2.41	2.41	2.44	2.53	2.63	2.39
HEUR	2.54	2.21	2.58	2.53	2.46	2.23	2.45	2.41	2.42	2.59	2.56	2.77	2.48
BB-LR	2.54	2.28	2.65	2.63	2.44	2.12	2.40	2.48	2.41	2.51	2.51	2.75	2.48
ILP-IDLE-MIN	2.81	2.04	2.66	2.67	2.59	2.23	2.38	2.51	2.42	2.72	2.56	2.61	2.52
QP-LR-UB	2.69	2.24	2.87	2.86	2.37	2.05	2.51	2.90	2.69	2.69	2.71	2.94	2.63
ILP-IDLE-MAX	3.30	2.82	3.28	3.36	3.02	2.91	3.04	3.33	3.18	3.45	3.22	3.44	3.20
							TX2						
ILP-SM	3.53	3.22	3.23	3.78	3.44	3.03	3.55	3.90	3.45	3.96	3.57	4.16	3.57
BB-SM	3.71	3.41	3.36	3.88	3.74	3.29	3.76	3.98	3.64	4.18	3.76	4.32	3.75
HEUR	3.59	3.27	3.25	4.00	3.70	3.11	3.74	3.91	3.54	4.15	3.73	4.19	3.68
BB-LR	3.70	3.30	3.43	3.92	3.71	3.26	3.78	3.98	3.71	4.10	3.88	4.26	3.75
ILP-IDLE-MIN	4.34	3.51	4.21	4.28	4.02	3.88	3.80	4.06	4.44	4.31	3.93	4.53	4.11
QP-LR-UB	3.65	3.35	3.28	3.82	3.61	3.13	3.63	3.98	3.55	3.97	3.57	4.29	3.65
ILP-IDLE-MAX	4.63	4.02	4.37	4.82	4.49	3.88	4.71	4.87	4.66	5.19	4.99	5.31	4.66

model based on the quadratic programming (`QP-LR-UB`) is about 6× slower than `ILP-SM` on instances with 15 and 20 tasks. Comparing the global methods to the local one (`HEUR`) on instances with 20 tasks, we see that the global methods `ILP-SM` and `QP-LR-UB` need about 18× and 95× more time, respectively. Performance of the heuristic method (`HEUR`) is comparable with `ILP-IDLE-MIN` on instances with 30 and more tasks; for smaller instances, `HEUR` is a bit slower, especially due to the overhead caused by performing the feasibility check (solving `ILP-FEAS`) multiple times. Even though the `ILP-IDLE-MAX` scales the best, it fails to produce thermally efficient schedules, as shown in Section 3.8.2.

3.8.4 Evaluation Summary

To summarize the results, the linear regression-based power model (`LR`) exhibited lower errors than the empirical sum-max model (`SM`), but it proved to be harder to integrate with the optimization methods. Its simplified variant `LR-UB` failed to provide a tight upper bound and therefore performed rather poorly.

Considering the optimization methods, global `ILP-SM` based on the integer linear programming and simpler `SM` power model provided overall best results. The black-box approach based on metaheuristics proved to be competitive as well; especially, it might be preferred for large-size instances, for which the integer linear programming fails to deliver high-quality solutions in a reasonable time. Also, the `BB` approach is based on an open-source implementation of a genetic algorithm, which might be an advantage when compared to the other tested methods based on the commercial Gurobi solver.

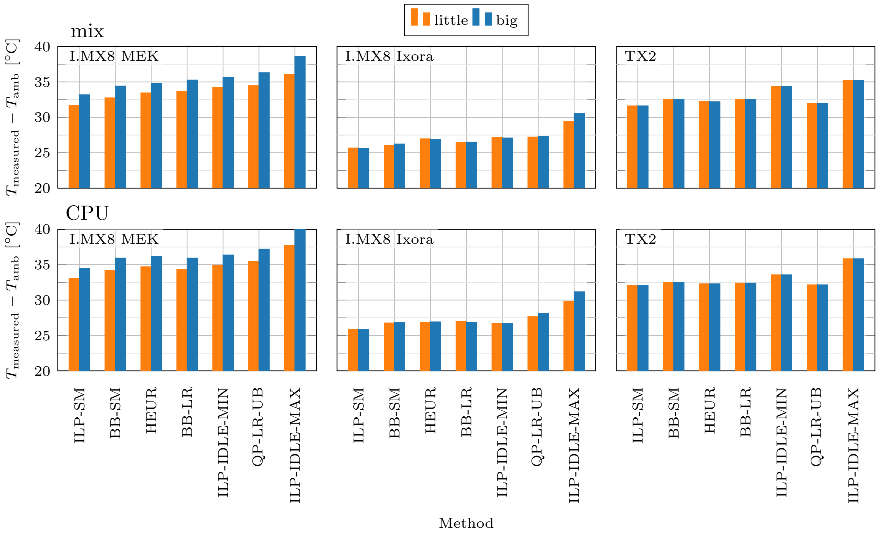


Figure 3.16: Average difference between the measured steady-state temperature T_{measured} and the average ambient temperature T_{amb} for tested optimization methods. Recall that Ixora's lower temperatures are caused by applied air flow.

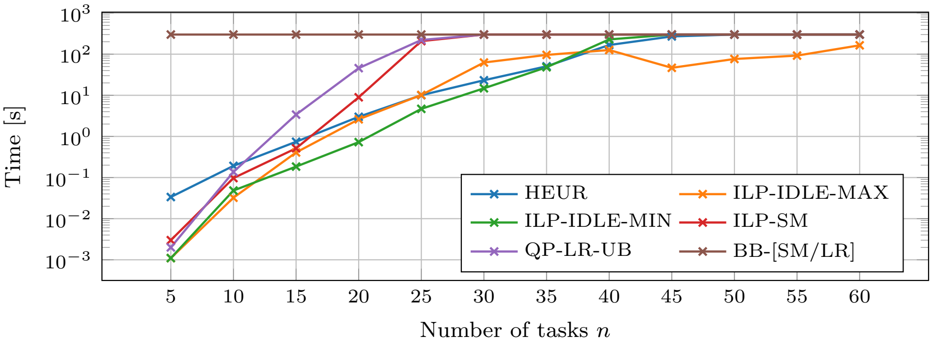


Figure 3.17: Average computation time of different methods w.r.t. the instance size n.

3.9 Chapter Conclusion

To conclude, this work studied the problem of offline task allocation on a heterogeneous multi-core platform for safety-critical avionics applications with the aim of minimizing the platform's steady-state temperature. The main limitations were unavailability of DVFS due to safety requirements and necessity to schedule the tasks into temporal isolation windows. Three power models were compared, including the empirical sum-max model, the linear regression model, and its simplified variant providing the upper bound. Furthermore, their integration within the optimization procedures was discussed. Several optimization approaches, including those based on mathematical programming, and both informed and uninformed heuristics, were described and evaluated on three hardware platforms. The data and source code is publicly available[4].

Extensive experimental evaluation showed that the best-performing method `ILP-SM` reduces the platform temperature by up to 16 %, 14 % and 10 % for I.MX8 Ixora, I.MX8 MEK and TX2, when compared to the worst method `ILP-IDLE-MAX`. Moreover, `ILP-SM` saves up to 4.7 %, 4.6 %, and 1.8 % when comparing to the heuristic method `HEUR`, which is a model-free approach popular due to its simplicity. Furthermore, the second best method `BB-SM` provides better scalability and just 3 % higher temperatures on average while being implemented solely with open-source software.

Surprisingly, all the best methods rely on the `SM` power model, which has the mean absolute error by 66 % higher (in average) than the `LR` model. This shows the importance of the power model and optimization method co-design.

---

[4] `https://github.com/benedond/safety-critical-scheduling`

Chapter

Thesis Conclusion

4

This thesis investigated offline scheduling problems involving resource state modeling and energy consumption (cost). As the research area itself is rather broad, we have concentrated on several use-cases highlighting the importance of a 'conjugate approach' involving the problem-tailored design jointly considering the resource model and optimization method. The results showed that with careful design choices, the optimization performance was substantially improved and, as such, the state-of-the-art methods used in the respective fields were outperformed. In the following lines, we summarize how the individual research objectives were fulfilled.

4.1 Fulfillment of the Goals

1. *Study the scheduling problems where the state of the resource significantly impacts the energy consumption.*

   This thesis studied three specific problems. First, we considered a production scheduling problem involving an electric vacuum steel-hardening furnace in Chapter 1. The problem inspiration and data came from our industrial partner, Škoda Auto. Undeniably, heat-intensive furnaces rank among the most-demanding energy consumers. Therefore, any energy optimization of their operation could bring significant benefits.

   Second, we moved to a general research problem in Chapter 2 involving the resource states modeled by a finite state machine and variable energy prices. Both of these provide rather general design concepts. Consider that: (i) the majority of the research works involving general resources with power-saving states adopt the finite state machine concept, and (ii) the time-of-use energy pricing provides a general mechanism that can accurately reflect the cost of electricity on the grid. Therefore, the proposed advances in the solution methods scalability provide important theoretical results with potential practical implications.

   Third, we studied thermal-aware safety-critical task allocation on a heterogeneous multiprocessor system on a chip under temporal isolation constraints in Chapter 3. That problem, defined by our industrial partner Honeywell, is of great importance in the aerospace domain as the modern systems need the high-computing power provided by modern MPSoC on the one hand but need to compensate for the produced heat without the use of massive heat sinks or mechanical fans on the other hand.

2. *Review the existing literature considering the resource-state modeling and energy optimization, identify the weak points, and suggest possible improvements.*

   We studied the related literature. Even though energy optimization has been extensively addressed in the last few decades, we conclude that there is still a lot of space for improvement.

One of the possible ways is to design a problem-tailored model and optimization procedure, as we did in Chapter 1. That way, the model can include problem-specific features, and thus, by being more informed than general models, improvements can be achieved.

Many scheduling problems involving energy are, in fact, $\mathcal{NP}$-hard. Therefore, we can often see that the researchers formulate the model in a mathematical programming language, but claiming that it lacks scalability, they move to design various metaheuristics without even trying to improve the model itself. Indeed, there will always be the instance size intractable by the mathematical model. However, thanks to advances in the modeling, as well as in the solvers, this size can be rather large. Thus, by proper pre-processing and integration, we may be able to solve instances of the problem that are practically relevant, as we show in Chapter 2.

Finally, as in embedded systems, modeling and optimization are two research topics that are often studied separately. The resource model might be very complex. As such, it might be rather hard to integrate it within an optimization method that could scale well. However, the model is just a tool serving to describe the system. In the end, it will always be prone to error because there will always be some things neglected. Therefore, we might sacrifice some of the model complexity in order to gain better integration ability and better optimization scalability. This is also the path that we follow and investigate in Chapter 3. We show that even though the resource model itself is not perfect, it may still bring satisfactory and practically applicable results.

3. *For several selected use-case problems, propose suitable resource models and their integration with the optimization procedures in the context of energy consumption minimization.*

   As mentioned above, we have studied three use-case problems in detail. In Chapter 1, we studied furnace behavior and proposed a strategy for its efficient management during idle periods. Further, we integrated the model within the optimization method using the idle energy function abstraction. That way, the addressed optimization problem was solved in polynomial time.

   In Chapter 2, we extended the idea of idle energy function to work even with variable energy prices. We designed a pre-processing technique that could efficiently compute the function values. Further, we studied how to integrate the function efficiently within the mathematical models.

   Finally, in Chapter 3, we provided a comparative study of several power models and optimization methods. We put a great emphasis on practicality; all the approaches were tested on three hardware platforms regarding both thermal efficiency and accuracy. We also included an additional experiment testing the scalability of the method.

4. *Present how suitable resource modeling combined with appropriate optimization techniques can improve the state-of-the-art solutions.*

   In all chapters, we compared the proposed approaches to the relevant techniques proposed in the literature. The results proved that suitable resource modeling and integration indeed bring improvement over the state-of-the-art.

4.2 Future Work

As the research path is never ending, there is still much to be done. The following list presents several ideas that could suggest possible research directions.

1. In the first two chapters, the concept of idle energy function was successfully applied in the process of solving two seemingly unrelated problems. Further, the literature shows that it can also be successfully used in the embedded systems domain for the management of components states during idle periods. This indicates a unifying concept that could be used for idle time management of any resource. Moreover, specific models and methods could be devised, integrating this concept. Therefore, we believe that a unifying framework for idle time resource management could be developed.

2. In Chapter 2, we integrated the idle energy function into the mathematical model, which brought significant improvements in scalability. Besides this, we have conducted a preliminary (unpublished) experiment, where we integrated the idle energy function directly into a dedicated branch-and-bound procedure. Preliminary results showed further 100× speed-up on the same instance set, which is rather promising. Therefore, we believe that the development of specialized algorithms integrating the models' abstractions is also one of the perspective ways.

3. We believe that there exists a gap between industrial problems and theoretical research. The practical problems are often overly constrained and therefore deemed unsuitable for scientific publications. On the other hand, the theoretical works are often highly abstract and frequently rely on models and simulations that might be transferable to practice only with great difficulties and unclear results. Thus we think that the works that will marginalize the gap between theory and practice will be highly appraised.

Bibliography

[1] Meisam Abdollahi et al. “THAMON: Thermal-aware High-performance Application Mapping onto Opto-electrical Network-on-Chip”. In: *Journal of Systems Architecture* 121 (Dec. 2021), p. 102315. ISSN: 1383-7621. DOI: `10.1016/j.sysarc.2021.102315`.

[2] J.B. Abikarram, K. McConky, and R. Proano. “Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing”. In: *Journal of Cleaner Production* 208 (2019), pp. 232–242. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2018.10.048`.

[3] M. Aghelinejad, Y. Ouazene, and A. Yalaoui. “Complexity analysis of energy-efficient single machine scheduling problems”. In: *Operations Research Perspectives* 6 (2019), p. 100105. ISSN: 2214-7160. DOI: `10.1016/j.orp.2019.100105`.

[4] M. Aghelinejad, Y. Ouazene, and A. Yalaoui. “Production scheduling optimisation with machine state and time-dependent energy costs”. In: *International Journal of Production Research* 56.16 (2018), pp. 5558–5575. DOI: `10.1080/00207543.2017.1414969`.

[5] Rehan Ahmed, Parameswaran Ramanathan, and Kewal K. Saluja. “On thermal utilization of periodic task sets in uni-processor systems”. In: *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications.* 2013, pp. 267–276. DOI: `10.1109/RTCSA.2013.6732227`.

[6] Haider Ali et al. “Contention & Energy-Aware Real-Time Task Mapping on NoC Based Heterogeneous MPSoCs”. In: *IEEE Access* 6 (2018), pp. 75110–75123. DOI: `10.1109/ACCESS.2018.2882941`.

[7] E. Angel, E. Bampis, and V. Chau. “Low Complexity Scheduling Algorithm Minimizing the Energy for Tasks with Agreeable Deadlines”. In: *LATIN 2012: Theoretical Informatics.* Ed. by D. Fernández-Baca. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 13–24. ISBN: 978-3-642-29344-3. DOI: `10.1007/978-3-642-29344-3_2`.

[8] Gabriele Ara, Tommaso Cucinotta, and Agostino Mascitti. “Simulating Execution Time and Power Consumption of Real-Time Tasks on Embedded Platforms”. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing.* SAC ’22. New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 491–500. ISBN: 978-1-4503-8713-2. DOI: `10.1145/3477314.3507030`.

[9] *Avionics Application Software Standard Interface, Part 1, Required Services, ARINC Specification 653 Part 1 Supplement 4 (653P1-4).* 2015.

[10] Kenneth Baker and Dan Trietsch. *Principles of Sequencing and Scheduling.* Wiley Publishing, Sept. 2018. ISBN: 9781119262565. DOI: `10.1002/9781119262602`.

[11] Alessio Balsini, Luigi Pannocchi, and Tommaso Cucinotta. "Modeling and Simulation of Power Consumption and Execution Times for Real-Time Tasks on Embedded Heterogeneous Architectures". In: *SIGBED Rev.* 16.3 (Nov. 2019), pp. 51–56. DOI: `10.1145/3373400.3373408`.

[12] Mario Bambagini et al. "Energy-Aware Scheduling for Real-Time Systems: A Survey". In: 15.1 (Jan. 2016). ISSN: 1539-9087. DOI: `10.1145/2808231`.

[13] Philippe Baptiste, Marek Chrobak, and Christoph Dürr. "Polynomial-Time Algorithms for Minimum Energy Scheduling". In: 8.3 (July 2012). ISSN: 1549-6325. DOI: `10.1145/2229163.2229170`.

[14] Andres Bego, Lin Li, and Zeyi Sun. "Identification of reservation capacity in critical peak pricing electricity demand response program for sustainable manufacturing systems". In: *International Journal of Energy Research* 38.6 (2014), pp. 728–736. DOI: `10.1002/er.3077`.

[15] Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Abstracts of Fast-Track Journal Paper. xxi-xxiii. Springer, Cham, 2020. ISBN: 978-3-030-58941-7. DOI: `10.1007/978-3-030-58942-4`.

[16] Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Constraints* 25.3–4 (Dec. 2020), pp. 300–318. ISSN: 1383-7133. DOI: `10.1007/s10601-020-09317-y`.

[17] Ondřej Benedikt et al. "A polynomial-time scheduling approach to minimise idle energy consumption: An application to an industrial furnace". In: *Computers & Operations Research* 128 (2021), p. 105167. ISSN: 0305-0548. DOI: `10.1016/j.cor.2020.105167`.

[18] Ondřej Benedikt et al. "Energy-Aware Production Scheduling with Power-Saving Modes". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Ed. by Willem-Jan van Hoeve. Cham: Springer International Publishing, 2018, pp. 72–81. ISBN: 978-3-319-93031-2. DOI: `10.1007/978-3-319-93031-2_6`.

[19] Ondřej Benedikt et al. "Energy-Aware Production Scheduling with Power-Saving Modes". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Ed. by Willem-Jan van Hoeve. Cham: Springer International Publishing, 2018, pp. 72–81. DOI: `10.1007/978-3-319-93031-2_6`.

[20] Ondřej Benedikt et al. "Thermal-Aware Scheduling for MPSoC in the Avionics Domain: Tooling and Initial Results". In: *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).* 2021, pp. 159–168. DOI: `10.1109/RTCSA52859.2021.00026`.

[21] Ondřej Benedikt., Přemysl Šůcha., and Zdeněk Hanzálek. "On Idle Energy Consumption Minimization in Production: Industrial Example and Mathematical Model". In: *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems - ICORES,* INSTICC. SciTePress, 2020, pp. 35–46. ISBN: 978-989-758-396-4. DOI: `10.5220/0008877400350046`.

[22] L. Benini, A. Bogliolo, and G. De Micheli. "A survey of design techniques for system-level dynamic power management". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8.3 (2000), pp. 299–316. DOI: `10.1109/92.845896`.

[23] Libor Bukata, Přemysl Šůcha, and Zdeněk Hanzálek. "Optimizing energy consumption of robotic cells by a Branch & Bound algorithm". In: *Computers & Operations Research* 102 (2019), pp. 52–66. ISSN: 0305-0548. DOI: `10.1016/j.cor.2018.09.012`.

[24] Libor Bukata et al. "Energy Optimization of Robotic Cells". In: *IEEE Transactions on Industrial Informatics* 13.1 (2017), pp. 92–102. DOI: `10.1109/TII.2016.2626472`.

[25] Radek Bumbálek. "Proactive and reactive approaches for non-critical tasks scheduling under thermal constraints in the avionics domain". MA thesis. 2022. URL: `https://dspace.cvut.cz/handle/10467/99111`.

[26] Kun Cao et al. "A Survey of Optimization Techniques for Thermal-Aware 3D Processors". In: *Journal of Systems Architecture* 97 (Aug. 2019), pp. 397–415. ISSN: 1383-7621. DOI: `10.1016/j.sysarc.2019.01.003`.

[27] Elisabet Capón-García et al. "Multiobjective optimization of multiproduct batch plants scheduling under environmental and economic concerns". In: *AIChE Journal* 57.10 (2011), pp. 2766–2782. DOI: `10.1002/aic.12477`.

[28] T. Chantem, X. S. Hu, and R. P. Dick. "Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19.10 (Oct. 2011), pp. 1884–1897. DOI: `10.1109/TVLSI.2010.2058873`.

[29] A. Che, S. Zhang, and X. Wu. "Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs". In: *Journal of Cleaner Production* 156 (2017), pp. 688–697. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2017.04.018`.

[30] A. Che et al. "Energy-efficient bi-objective single-machine scheduling with power-down mechanism". In: *Computers & Operations Research* 85 (2017), pp. 172–183. ISSN: 0305-0548. DOI: `10.1016/j.cor.2017.04.004`.

[31] K. Chee Chook and A.H. Tan. "Identification of an Electric Resistance Furnace". In: *IEEE Transactions on Instrumentation and Measurement* 56 (6 2007), pp. 2262–2270. DOI: `10.1109/TIM.2007.907960`.

[32] Gang Chen, Kai Huang, and Alois Knoll. "Energy Optimization for Real-Time Multiprocessor System-on-Chip with Optimal DVFS and DPM Combination". In: *ACM Trans. Embed. Comput. Syst.* 13.3s (Mar. 2014). ISSN: 1539-9087. DOI: `10.1145/2567935`.

[33] Ting-Hsuan Chien and Rong-Guey Chang. “A thermal-aware scheduling for multicore architectures”. In: *Journal of Systems Architecture* 62 (2016), pp. 54–62. ISSN: 1383-7621. DOI: `https://doi.org/10.1016/j.sysarc.2015.12.003`.

[34] Marek Chrobak et al. “Scheduling with gaps: new models and algorithms”. In: *Journal of Scheduling* 24.4 (Aug. 2021), pp. 381–403. ISSN: 1099-1425. DOI: `10.1007/s10951-021-00691-w`.

[35] Darren D. Cofer and Murali Rangarajan. “Formal Modeling and Analysis of Advanced Scheduling Features in an Avionics RTOS”. In: *Proceedings of the Second International Conference on Embedded Software*. EMSOFT ’02. Berlin, Heidelberg: Springer-Verlag, Oct. 2002, pp. 138–152. ISBN: 978-3-540-44307-0.

[36] David Cohen et al. “Symmetry Definitions for Constraint Satisfaction Problems”. In: *Constraints* 11 (July 2006), pp. 115–137. DOI: `10.1007/s10601-006-8059-8`.

[37] T.H. Cormen et al. *Introduction to Algorithms*. 2nd. Cambridge, MA, USA: MIT Press, 2001. ISBN: 9780262032933.

[38] Robert I. Davis and Alan Burns. “A Survey of Hard Real-Time Scheduling for Multiprocessor Systems”. In: *ACM Comput. Surv.* 43.4 (Oct. 2011). ISSN: 0360-0300. DOI: `10.1145/1978802.1978814`.

[39] I. Derese and E.J. Nodulus. “Nonlinear control of bilinear systems”. In: *IEE Proceedings D: Control Theory and Applications* 127.4 (1980), pp. 169–175. DOI: `10.1049/ip-d.1980.0026`.

[40] Vinay Devadas and Hakan Aydin. “On the Interplay of Voltage/Frequency Scaling and Device Power Management for Frame-Based Real-Time Embedded Applications”. In: *IEEE Transactions on Computers* 61.1 (2012), pp. 31–44. DOI: `10.1109/TC.2010.248`.

[41] Somdip Dey et al. “Asynchronous Hybrid Deep Learning (AHDL): A Deep Learning Based Resource Mapping in DVFS Enabled Mobile MPSoCs”. In: *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*. 2021, pp. 303–308. DOI: `10.1109/WF-IoT51360.2021.9594956`.

[42] Jian-Ya Ding et al. “Parallel Machine Scheduling Under Time-of-Use Electricity Prices: New Models and Optimization Approaches”. In: *IEEE Transactions on Automation Science and Engineering* 13.2 (2016), pp. 1138–1154. DOI: `10.1109/TASE.2015.2495328`.

[43] J. Dušek. “Návrh úpravy řízení výrobní linky s ohledem na snížení její spotřeby”. MA thesis. the Czech republic: Czech Technical University in Prague, 2016. URL: `https://dspace.cvut.cz/handle/10467/65284`.

[44] EEMBC. *Autobench 2.0*. 2022. URL: `https://www.eembc.org/autobench/` (visited on 07/07/2022).

[45] Amirhossein Esmaili, Mahdi Nazemi, and Massoud Pedram. “Modeling Processor Idle Times in MPSoC Platforms to Enable Integrated DPM, DVFS, and Task Scheduling Subject to a Hard Deadline”. In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference.* ASPDAC ’19. Tokyo, Japan: Association for Computing Machinery, 2019, pp. 532–537. ISBN: 9781450360074. DOI: `10.1145/3287624.3287630`.

[46] Maher Fakih et al. “SAFEPOWER Project: Architecture for Safe and Power-Efficient Mixed-Criticality Systems”. In: *Microprocessors and Microsystems* 52 (July 2017), pp. 89–105. ISSN: 0141-9331. DOI: `10.1016/j.micpro.2017.05.016`.

[47] Kan Fang et al. “A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction”. In: *Journal of Manufacturing Systems* 30.4 (2011). Selected Papers of 39th North American Manufacturing Research Conference, pp. 234–240. ISSN: 0278-6125. DOI: `10.1016/j.jmsy.2011.08.004`.

[48] Kan Fang et al. “Scheduling on a single machine under time-of-use electricity tariffs”. In: *Annals of Operations Research* 238.1 (Mar. 2016), pp. 199–227. DOI: `10.1007/s10479-015-2003-5`.

[49] A.F. Filippov. *Differential Equations with Discontinuous Righthand Sides: Control Systems.* en. Mathematics and its Applications. Springer Netherlands, 1988. ISBN: 978-90-277-2699-5. DOI: `10.1007/978-94-015-7793-9`.

[50] Nathan Fisher et al. “Thermal-Aware Global Real-Time Scheduling and Analysis on Multicore Systems”. In: *Journal of Systems Architecture.* Special Issue on Multiprocessor Real-time Scheduling 57.5 (May 2011), pp. 547–560. ISSN: 1383-7621. DOI: `10.1016/j.sysarc.2010.09.010`.

[51] Christian Gahm et al. “Energy-efficient scheduling in manufacturing companies: A review and research framework”. In: *European Journal of Operational Research* 248.3 (2016), pp. 744–757. ISSN: 0377-2217. DOI: `10.1016/j.ejor.2015.07.017`.

[52] Kaizhou Gao et al. “A review of energy-efficient scheduling in intelligent production systems”. In: *Complex & Intelligent Systems* (2019). DOI: `10.1007/s40747-019-00122-6`.

[53] M. Garey and D. Johnson. “Two-Processor Scheduling with Start-Times and Deadlines”. In: *SIAM Journal on Computing* 6.3 (1977), pp. 416–426.

[54] Ian P. Gent, Karen E. Petrie, and Jean-François Puget. “Chapter 10 - Symmetry in Constraint Programming”. In: *Handbook of Constraint Programming.* Ed. by Francesca Rossi, Peter van Beek, and Toby Walsh. Vol. 2. Foundations of Artificial Intelligence. Elsevier, 2006, pp. 329–376. DOI: `10.1016/S1574-6526(06)80014-3`.

[55] M.E.T. Gerards, J.L. Hurink, and P.K.F. Hölzenspies. “A survey of offline algorithms for energy minimization under deadline constraints”. In: *Journal of Scheduling* 19.1 (2016), pp. 3–19. ISSN: 1099-1425.

[56] M.E.T. Gerards and J. Kuper. “Optimal DPM and DVFS for Frame-based Real-time Systems”. In: *ACM Trans. Archit. Code Optim.* 9.4 (2013), 41:1–41:23. ISSN: 1544-3566. DOI: `10.1145/2400682.2400700`.

[57] Xu Gong et al. “Energy- and labor-aware flexible job shop scheduling under dynamic electricity pricing: A many-objective optimization investigation”. In: *Journal of Cleaner Production* 209 (2019), pp. 1078–1094. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2018.10.289`.

[58] Xingxing Guo et al. “Leakage-Aware Thermal Management for Multi-Core Systems Using Piecewise Linear Model Based Predictive Control”. In: Jan. 2019, pp. 64–69. DOI: `10.1145/3287624.3287665`. URL: `https://wanghaiuestc.github.io/papers/ASPDAC19_pwl.pdf`.

[59] T. Gutowski et al. “Environmentally benign manufacturing: Observations from Japan, Europe and the United States”. In: *Journal of Cleaner Production* 13 (2005), pp. 1–17. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2003.10.004`.

[60] Hubert Hadera et al. “Optimization of steel production scheduling with complex time-sensitive electricity cost”. In: *Computers & Chemical Engineering* 76 (2015), pp. 117–136. ISSN: 0098-1354. DOI: `10.1016/j.compchemeng.2015.02.004`.

[61] A. Haït and C. Artigues. “A hybrid CP/MILP method for scheduling with energy costs”. In: *European Journal of Industrial Engineering* 5 (2011), pp. 471–489. DOI: `10.1504/EJIE.2011.042742`.

[62] N. Hajaliakbari and S. Hassanpour. “Analysis of thermal energy performance in continuous annealing furnace”. In: *Applied Energy* 206 (2017), pp. 829–842. DOI: `10.1016/j.apenergy.2017.08.246`.

[63] Yan He et al. “An energy-responsive optimization method for machine tool selection and operation sequence in flexible machining job shops”. In: *Journal of Cleaner Production* 87 (2015), pp. 245–254. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2014.10.006`.

[64] Jeffrey W. Herrmann. “A History of Production Scheduling”. In: *Handbook of Production Scheduling*. Ed. by Jeffrey W. Herrmann. Boston, MA: Springer US, 2006, pp. 1–22. ISBN: 978-0-387-33117-1. DOI: `10.1007/0-387-33117-4_1`.

[65] David Hornof. “Offline scheduling of the safety-critical tasks within the isolation time-windows”. MA thesis. 2021. URL: `https://dspace.cvut.cz/handle/10467/95363`.

[66] Wei Huang et al. “HotSpot: A compact thermal modeling methodology for early-stage VLSI design”. In: *IEEE Transactions on Very Large Scale Integration Systems - VLSI* 14 (May 2006), pp. 501–513. DOI: `10.1109/TVLSI.2006.876103`.

[67] Michael D. Intriligator. *Mathematical Optimization and Economic Theory*. USA: Society for Industrial and Applied Mathematics, 2002. ISBN: 0898715113.

[68] S. Irani, S. Shukla, and R. Gupta. "Online Strategies for Dynamic Power Management in Systems with Multiple Power-saving States". In: *ACM Trans. Embed. Comput. Syst.* 2.3 (2003), pp. 325–346. ISSN: 1539-9087. DOI: `10.1145/860176.860180`.

[69] Sandy Irani, Sandeep Shukla, and Rajesh Gupta. "Algorithms for Power Savings". In: *ACM Trans. Algorithms* 3.4 (Nov. 2007), 41–es. ISSN: 1549-6325. DOI: `10.1145/1290672.1290678`.

[70] Xiaowen Jiang et al. "Energy-Efficient Scheduling of Periodic Applications on Safety-Critical Time-Triggered Multiprocessor Systems". In: *Electronics* 7.6 (June 2018), p. 98. ISSN: 2079-9292. DOI: `10.3390/electronics7060098`.

[71] Serdar Kadioglu and Meinolf Sellmann. "Grammar Constraints". In: *Constraints* 15.1 (Jan. 2010), pp. 117–144. ISSN: 1383-7133. DOI: `10.1007/s10601-009-9073-4`.

[72] A. Kanduri et al. "adBoost: Thermal Aware Performance Boosting Through Dark Silicon Patterning". In: *IEEE Transactions on Computers* 67.8 (Aug. 2018), pp. 1062–1077. ISSN: 0018-9340. DOI: `10.1109/TC.2018.2805683`.

[73] D.E. Kirk. *Optimal Control Theory: An Introduction*. Dover Books on Electrical Engineering Series. Dover Publications, 2004. ISBN: 9780486434841.

[74] Chin-Fu Kuo and Yung-Feng Lu. "Task Assignment with Energy Efficiency Considerations for Non-DVS Heterogeneous Multiprocessor Systems". In: *SIGAPP Appl. Comput. Rev.* 14.4 (Jan. 2015), pp. 8–18. ISSN: 1559-6915. DOI: `10.1145/2724928.2724929`.

[75] Philippe Laborie et al. "IBM ILOG CP optimizer for scheduling". In: *Constraints* 23.2 (Apr. 2018), pp. 210–250. ISSN: 1572-9354. DOI: `10.1007/s10601-018-9281-x`.

[76] V. Lakshminarayanan and N. Sriraam. "The effect of temperature on the reliability of electronic components". In: *2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. 2014, pp. 1–6. DOI: `10.1109/CONECCT.2014.6740182`.

[77] Wai-Kong Lee, Sze-Wei Lee, and Wee-Ong Siew. "Hybrid model for dynamic power management". In: *IEEE Transactions on Consumer Electronics* 55.2 (2009), pp. 656–664. DOI: `10.1109/TCE.2009.5174436`.

[78] Youngmoon Lee, Kang G. Shin, and Hoon Sung Chwa. "Thermal-Aware Scheduling for Integrated CPUs–GPU Platforms". In: *ACM Trans. Embed. Comput. Syst.* 18.5s (Oct. 2019). ISSN: 1539-9087. DOI: `10.1145/3358235`.

[79] Deming Lei and Xiuping Guo. "An effective neighborhood search for scheduling in dual-resource constrained interval job shop with environmental objective". In: *International Journal of Production Economics* 159 (2015), pp. 296–303. ISSN: 0925-5273. DOI: `10.1016/j.ijpe.2014.07.026`.

[80] George Lentaris et al. "High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation". In: *Journal of Aerospace Information Systems* 15.4 (2018), pp. 178–192. DOI: `10.2514/1.I010555`. eprint: `https://doi.org/10.2514/1.I010555`.

[81] Ming Li and Deming Lei. "An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times". In: *Engineering Applications of Artificial Intelligence* 103 (2021), p. 104307. ISSN: 0952-1976. DOI: `10.1016/j.engappai.2021.104307`.

[82] Tiantian Li, Ge Yu, and Jie Song. "Minimizing Energy by Thermal-Aware Task Assignment and Speed Scaling in Heterogeneous MPSoC Systems". In: *Journal of Systems Architecture* 89 (Sept. 2018), pp. 118–130. ISSN: 1383-7621. DOI: `10.1016/j.sysarc.2018.08.003`.

[83] Tiantian Li et al. "Minimizing Temperature and Energy of Real-Time Applications with Precedence Constraints on Heterogeneous MPSoC Systems". In: *Journal of Systems Architecture* 98 (Sept. 2019), pp. 79–91. ISSN: 1383-7621. DOI: `10.1016/j.sysarc.2019.07.001`.

[84] Xin Li et al. "Accurate On-Chip Temperature Sensing for Multicore Processors Using Embedded Thermal Sensors". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.11 (2020), pp. 2328–2341. DOI: `10.1109/TVLSI.2020.3012833`.

[85] C.-H. Liu and D.-H. Huang. "Reduction of power consumption and carbon footprints by applying multi-objective optimisation via genetic algorithms". In: *International Journal of Production Research* 52.2 (2014), pp. 337–352. DOI: `10.1080/00207543.2013.825740`.

[86] ChenGuang Liu et al. "Production planning of multi-stage multi-option seru production systems with sustainable measures". In: *Journal of Cleaner Production* 105 (2015). Decision-support models and tools for helping to make real progress to more sustainable societies, pp. 285–299. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2014.03.033`.

[87] Ming Liu et al. "Energy-oriented bi-objective optimization for the tempered glass scheduling". In: *Omega* 90 (2020), p. 101995. ISSN: 0305-0483. DOI: `10.1016/j.omega.2018.11.004`.

[88] Jan Lucas and Ben Juurlink. "MEMPower: Data-Aware GPU Memory Power Model". In: *Architecture of Computing Systems – ARCS 2019*. Ed. by Martin Schoeberl et al. Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 195–207. ISBN: 978-3-030-18656-2.

[89] Kanchan Manna et al. "Thermal-Aware Application Mapping Strategy for Network-on-Chip Based System Design". In: *IEEE Transactions on Computers* 67.4 (Apr. 2018), pp. 528–542. ISSN: 1557-9956. DOI: `10.1109/TC.2017.2770130`.

[90] Filippo Mantovani and Enrico Calore. "Performance and Power Analysis of HPC Workloads on Heterogeneous Multi-Node Clusters". In: *Journal of Low Power Electronics and Applications* 8.2 (June 2018), p. 13. DOI: `10.3390/jlpea8020013`.

[91] François Margot. "Symmetry in Integer Linear Programming". In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art.* Ed. by Michael Jünger et al. Berlin, Heidelberg: Springer Berlin Heidelberg, Nov. 2010, pp. 647–686. ISBN: 978-3-540-68279-0. DOI: `10.1007/978-3-540-68279-0_17`.

[92] Lennart Merkert et al. "Scheduling and energy – Industrial challenges and opportunities". In: *Computers & Chemical Engineering* 72 (2015), pp. 183–198. ISSN: 0098-1354. DOI: `10.1016/j.compchemeng.2014.05.024`.

[93] Merriam-Webster. *Optimize.* In: *Merriam-Webster.com dictionary.* URL: `https://www.merriam-webster.com/dictionary/optimization` (visited on 05/02/2022).

[94] Jean-Baptiste Michel et al. "Quantitative Analysis of Culture Using Millions of Digitized Books". In: *Science* 331.6014 (2011), pp. 176–182. DOI: `10.1126/science.1199644`.

[95] Rustam Miftakhutdinov, Eiman Ebrahimi, and Yale N. Patt. "Predicting Performance Impact of DVFS for Realistic Memory Systems". In: *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture.* 2012, pp. 155–165. DOI: `10.1109/MICRO.2012.23`.

[96] U.-C. Moon and K.Y. Lee. "Hybrid algorithm with fuzzy system and conventional PI control for the temperature control of TV glass furnace". In: *IEEE Transactions on Control Systems Technology* 11.4 (2003), pp. 548–554. DOI: `10.1109/TCST.2003.813385`.

[97] Gilles Mouzon, Mehmet B. Yildirim, and Janet Twomey. "Operational methods for minimization of energy consumption of manufacturing equipment". In: *International Journal of Production Research* 45.18–19 (2007), pp. 4247–4271. DOI: `10.1080/00207540701450013`.

[98] "Numerical investigation on energy performance of hot stamping furnace". In: *Applied Thermal Engineering* 147 (2019), pp. 694–706. DOI: `10.1016/J.APPLTHERMALENG.2018.10.083`.

[99] NVIDIA. *Harness AI at the Edge with the Jetson TX2 Developer Kit.* 2022. URL: `https://developer.nvidia.com/embedded/jetson-tx2-developer-kit` (visited on 07/07/2022).

[100] NVIDIA. *Jetson TX2 Series Thermal Design Guide (TDG-09420-001_v1.0).* 2019. URL: `http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-thermal-design-guide`.

[101] NVIDIA. *Tegra X2 Series SoC.* Technical Reference Manual Parker_TRM_DP07821001p. June 2017. URL: `https://developer.nvidia.com/embedded/downloads#?search=Tegra%20X2`.

[102] NXP. *I.MX 8QuadMax Applications Processor.* IMX8QMRM. Reference Manual. Sept. 2021. URL: `https://www.nxp.com/webapp/Download?colCode=IMX8QMRM`.

[103] NXP. *i.MX 8QuadMax/QuadPlus Multisensory Enablement Kit*. 2021. URL: `https://www.nxp.com/design/development-boards/i-mx-evaluation-and-development-boards/i-mx-8quadmax-multisensory-enablement-kit-mek:MCIMX8QM-CPU` (visited on 11/18/2021).

[104] Santiago Pagani et al. "TSP: Thermal Safe Power: Efficient Power Budgeting for Many-Core Systems in Dark Silicon". In: *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*. CODES '14. New Delhi, India: Association for Computing Machinery, 2014. ISBN: 9781450330510. DOI: `10.1145/2656075.2656103`.

[105] James Pallister, Simon J. Hollis, and Jeremy Bennett. "Identifying Compiler Options to Minimize Energy Consumption for Embedded Platforms". In: *The Computer Journal* 58.1 (Jan. 2015), pp. 95–109. ISSN: 0010-4620. DOI: `10.1093/comjnl/bxt129`.

[106] Suraj Paul, Navonil Chatterjee, and Prasun Ghosal. "Dynamic Task Mapping and Scheduling with Temperature-Awareness on Network-on-Chip based Multicore Systems". In: *Journal of Systems Architecture* 98 (2019), pp. 271–288. ISSN: 1383-7621. DOI: `https://doi.org/10.1016/j.sysarc.2019.08.002`.

[107] Javier Perez Rodriguez and Patrick Meumeu Yomsi. "Thermal-Aware Schedulability Analysis for Fixed-Priority Non-preemptive Real-Time Systems". In: *2019 IEEE Real-Time Systems Symposium (RTSS)*. 2019, pp. 154–166. DOI: `10.1109/RTSS46320.2019.00024`.

[108] Javier Pérez Rodríguez and Patrick Meumeu Yomsi. "An Efficient Proactive Thermal-Aware Scheduler for DVFS-Enabled Single-Core Processors". In: *29th International Conference on Real-Time Networks and Systems*. RTNS'2021. NANTES, France: Association for Computing Machinery, 2021, pp. 144–154. ISBN: 9781450390019. DOI: `10.1145/3453417.3453430`.

[109] Gilles Pesant. "A Regular Language Membership Constraint for Finite Sequences of Variables". In: *Principles and Practice of Constraint Programming – CP 2004*. Ed. by Mark Wallace. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 482–495. ISBN: 978-3-540-30201-8. DOI: `10.1007/978-3-540-30201-8_36`.

[110] Karuppanan Pi et al. "A Fully Differential Operational Amplifier with Slew Rate Enhancer and Adaptive Bias for Ultra Low Power". In: *Journal of Low Power Electronics* 13 (Mar. 2017), pp. 67–75. DOI: `10.1166/jolpe.2017.1467`.

[111] Jason A. Poovey et al. "A Benchmark Characterization of the EEMBC Benchmark Suite". In: *IEEE Micro* 29.5 (2009), pp. 18–29. DOI: `10.1109/MM.2009.74`.

[112] S.S. Rao. *Engineering Optimization: Theory and Practice*. Wiley, 2019. ISBN: 9781119454717. URL: `https://books.google.cz/books?id=oG21DwAAQBAJ`.

[113] Sheriff Sadiqbatcha et al. "Real-Time Full-Chip Thermal Tracking: A Post-Silicon, Machine Learning Perspective". In: *IEEE Transactions on Computers* 71.6 (2022), pp. 1411–1424. DOI: `10.1109/TC.2021.3086112`.

[114] Bagher Salami et al. “Physical-Aware Predictive Dynamic Thermal Management of Multi-Core Processors”. In: *Journal of Parallel and Distributed Computing.* Special Issue on Energy Efficient Multi-Core and Many-Core Systems, Part I 95 (Sept. 2016), pp. 42–56. ISSN: 0743-7315. DOI: `10.1016/j.jpdc.2016.03.008`.

[115] Christian El Salloum et al. “The ACROSS MPSoC – A New Generation of Multi-core Processors Designed for Safety-Critical Embedded Systems”. In: *2012 15th Euromicro Conference on Digital System Design.* 2012, pp. 105–113. DOI: `10.1109/DSD.2012.126`.

[116] Sven Schulz, Udo Buscher, and Liji Shen. “Multi-objective hybrid flow shop scheduling with variable discrete production speed levels and time-of-use energy prices”. In: *Journal of Business Economics* 90 (Nov. 2020). DOI: `10.1007/s11573-020-00971-5`.

[117] Arsalan Shahid et al. “A Comparative Study of Techniques for Energy Predictive Modeling Using Performance Monitoring Counters on Modern Multicore CPUs”. In: *IEEE Access* 8 (2020), pp. 143306–143332. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2020.3013812`.

[118] Arsalan Shahid et al. “Improving the accuracy of energy predictive models for multicore CPUs by combining utilization and performance events model variables”. en. In: *Journal of Parallel and Distributed Computing* 151 (May 2021), pp. 38–51. ISSN: 0743-7315. DOI: `10.1016/j.jpdc.2021.01.007`.

[119] Hafiz Fahad Sheikh et al. “An overview and classification of thermal-aware scheduling techniques for multi-core processing systems”. In: *Sustainable Computing: Informatics and Systems* 2.3 (2012), pp. 151–169. ISSN: 2210-5379. DOI: `10.1016/j.suscom.2011.06.005`.

[120] Saad Zia Sheikh and Muhammad Adeel Pasha. “Energy-Efficient Multicore Scheduling for Hard Real-Time Systems: A Survey”. In: *ACM Trans. Embed. Comput. Syst.* 17.6 (Dec. 2018). ISSN: 1539-9087. DOI: `10.1145/3291387`.

[121] Fadi Shrouf et al. “Optimizing the production scheduling of a single machine to minimize total energy consumption costs”. In: *Journal of Cleaner Production* 67 (2014), pp. 197–207. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2013.12.024`.

[122] Amit Kumar Singh et al. “Learning-Based Run-Time Power and Energy Management of Multi/Many-Core Systems: Current and Future Trends”. In: *Journal of Low Power Electronics* 13.3 (2017), pp. 310–325. ISSN: 1546-1998. DOI: `doi:10.1166/jolpe.2017.1492`.

[123] Michal Sojka, Ondřej Benedikt, and Zdeněk Hanzálek. “Work-in-Progress: Determining MPSoC Layout from Thermal Camera Images”. In: *2021 International Conference on Embedded Software (EMSOFT).* 2021, pp. 39–40.

[124] Michal Sojka et al. “Testbed for thermal and performance analysis in MPSoC systems”. In: *2020 15th Conference on Computer Science and Information Systems (FedCSIS).* 2020, pp. 683–692. DOI: `10.15439/2020F174`.

[125] L. Tang, H. Ren, and Y. Yang. “Reheat furnace scheduling with energy consideration”. In: *International Journal of Production Research* 53 (2014), pp. 1–19.

[126] ChaudhryMuhammad Tayyab et al. “Thermal-Aware Scheduling in Green Data Centers”. In: *ACM Computing Surveys (CSUR)* (Feb. 2015). DOI: `10.1145/2678278`.

[127] Toradex. *NXP i.MX 8 Computer on Module*. 2021. URL: `https://www.toradex.com/computer-on-modules/apalis-arm-family/nxp-imx-8` (visited on 11/18/2021).

[128] Michael Trick. “A Dynamic Programming Approach for Consistency and Propagation for Knapsack Constraints”. In: *Ann Oper Res* 118 (Dec. 2001). DOI: `10.1023/A:1021801522545`.

[129] J. Wang and C.J. Spanos. “Real-time furnace modeling and diagnostics”. In: *IEEE Transactions on Semiconductor Manufacturing* 15.4 (2002), pp. 393–403. ISSN: 08946507. DOI: `10.1109/TSM.2002.804874`.

[130] Q.-G. Wang, C.-C. Hang, and W. Zou. “Automatic tuning of nonlinear PID controllers for unsymmetrical processes”. In: *Computers & Chemical Engineering* 22.4-5 (1998), pp. 687–694. ISSN: 00981354. DOI: `10.1016/S0098-1354(97)00220-2`.

[131] Wildart. *Evolutionary.jl*. online. Julia package hosted on Github. Aug. 2022. URL: `https://wildart.github.io/Evolutionary.jl/stable/` (visited on 07/21/2022).

[132] Workswell. *Workswell InfraRed Camera*. 2022. URL: `https://workswell-thermal-camera.com/workswell-infrared-camera-wic/#specifications` (visited on 07/02/2022).

[133] Mehmet Bayram Yildirim and Gilles Mouzon. “Single-Machine Sustainable Production Planning to Minimize Total Energy Consumption and Total Completion Time Using a Multiple Objective Genetic Algorithm”. In: *IEEE Transactions on Engineering Management* 59.4 (2012), pp. 585–597. DOI: `10.1109/TEM.2011.2171055`.

[134] Chanmin Yoon et al. “Accurate power modeling of modern mobile application processors”. In: *Journal of Systems Architecture* 81 (2017), pp. 17–31. ISSN: 1383-7621. DOI: `https://doi.org/10.1016/j.sysarc.2017.10.001`.

[135] D.-L. Yu. “Diagnosing simulated faults for an industrial furnace based on bilinear model”. In: *IEEE Transactions on Control Systems Technology* 8.3 (2000), pp. 435–442. ISSN: 1063-6536. DOI: `10.1109/87.845874`.

[136] Simone Zanoni, Laura Bettoni, and Christoph H. Glock. “Energy implications in a two-stage production system with controllable production rates”. In: *International Journal of Production Economics* 149 (2014). The Economics of Industrial Production, pp. 164–171. ISSN: 0925-5273. DOI: `10.1016/j.ijpe.2013.06.025`.

[137] B. Zhang et al. “The modeling and control of a reheating furnace”. In: *Proceedings of the 2002 American Control Conference*. Vol. 5. 2002, 3823–3828 vol.5. DOI: `10.1109/ACC.2002.1024524`.

[138] Jinwei Zhang et al. "Accurate Power Density Map Estimation for Commercial Multi-Core Microprocessors". In: *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2020, pp. 1085–1090. DOI: `10.23919/DATE48585.2020.9116545`.

[139] Kaicheng Zhang et al. "Machine Learning-Based Temperature Prediction for Runtime Thermal Management Across System Components". In: *IEEE Transactions on Parallel and Distributed Systems* 29.2 (2018), pp. 405–419. DOI: `10.1109/TPDS.2017.2732951`.

[140] Junlong Zhou et al. "Security-Critical Energy-Aware Task Scheduling for Heterogeneous Real-Time MPSoCs in IoT". In: *IEEE Transactions on Services Computing* 13.4 (July 2020). Conference Name: IEEE Transactions on Services Computing, pp. 745–758. ISSN: 1939-1374. DOI: `10.1109/TSC.2019.2963301`.

[141] Junlong Zhou et al. "Thermal-Aware Task Scheduling for Energy Minimization in Heterogeneous Real-Time MPSoC Systems". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.8 (2016), pp. 1269–1282. DOI: `10.1109/TCAD.2015.2501286`.

[142] L. Zhou et al. "Energy consumption model and energy efficiency of machine tools: a comprehensive literature review". In: *Journal of Cleaner Production* 112 (2016), pp. 3721–3734. ISSN: 0959-6526.

[143] Shengchao Zhou et al. "A multi-objective differential evolution algorithm for parallel batch processing machine scheduling considering electricity consumption cost". In: *Computers & Operations Research* 96 (2018), pp. 55–68. ISSN: 0305-0548. DOI: `10.1016/j.cor.2018.04.009`.

[144] Di Zhu et al. "TAPP: Temperature-aware Application Mapping for NoC-based Many-Core Processors". In: *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Mar. 2015, pp. 1241–1244.

[145] Sergey Zhuravlev et al. "Survey of Energy-Cognizant Scheduling Techniques". In: *IEEE Transactions on Parallel and Distributed Systems* 24.7 (2013), pp. 1447–1464. DOI: `10.1109/TPDS.2012.20`.

List of Figures

1 From problem specification towards the optimal solutions and their deployment. . . . 3

1.1 A steel hardening production line in Škoda Auto consisting of electrical vacuum furnaces [43]. . . . 11
1.2 Examples of the partial schedules corresponding to the edges between (a) (v^s, v_i^r), (b) (v_i^r, v^e), and (c) $\left(v_i^{\tilde{d}}, v_{i'}^r\right)$. . . . 21
1.3 Energy graph constructed for the tasks specified by Table 1.1. . . . 22
1.4 Feasible schedule corresponding to path $v^s, v_1^{\tilde{d}}, v_3^r, v_4^r, v^e$. . . . 23
1.5 Relationship between the temperature and power when cooling to 600 °C and heating back to operating temperature . . . 32
1.6 Comparison of the measured data and simulation using a bilinear model. . . . 34
1.7 Example of the optimal control for two different terminal times $t_f^{(1)}$ and $t_f^{(2)}$. . . . 35
1.8 Idle energy function E and two different idle period lengths $t_f^{(1)}$ and $t_f^{(2)}$ with the corresponding idle energy consumption $E\left(t_f^{(1)}\right)$ and $E\left(t_f^{(2)}\right)$. . . . 35
1.9 Examples of the transition graphs for the furnace model (1.14) with parameters (1.40). . . . 36
1.10 Average power per idle time $\overline{P}$ depending on the modeling of the machine dynamics and utilization of the machine. . . . 38

2.1 Parameters of the transition power function $P(s, s')$ and the transition time function $T(s, s')$, and the corresponding transition graph, where every edge from s to s' is labeled by $T(s, s')/P(s, s')$. . . . 45
2.2 The optimal schedule for the example instance. Each cell, corresponding to interval I_i and a state/transition, contains the value $c_i \cdot P(\Omega_i)$. The sum over all these values gives the TEC equal to 133. 45
2.3 Interval-state graph for the Example instance from Section 2.3 with highlighted optimal switching behavior from $(\texttt{proc}, I_4)$ to $(\texttt{proc}, I_{10})$. 47
2.4 Example of a transition graph with multiple standby states; every edge from s to s' is labeled by $T(s, s')/P(s, s')$. . . . 55
2.5 Comparison between the running times of the pre-processing and ILP-SPACES model for MEDIUM dataset. . . . 57

3.1 Schema of time-partitioned scheduling of safety-critical and best-effort tasks. . . . 61
3.2 Three steps (bechmarking, optimization and evaluation) towards the thermally efficient scheduling. . . . 65
3.3 Illustration of the used notation. . . . 69

3.4 Embedded platforms used for the evaluation: (a) I.MX8QuadMax Multisensory Enablement Kit by NXP (I.MX8 MEK), (b) Toradex Apalis I.MX8 board (I.MX8 Ixora), (c) Nvidia Jetson TX2 Developer Kit (TX2). 70

3.5 Relative speedup on a CPU from the high-performing cluster. 73

3.6 Influence of the on-chip temperature near the high-performing cluster on the major frame length (hyper-period) for three instance alternating between computing and idling. 74

3.7 Temperatures obtained for *pntrch-4M-100s* from on-chip sensors near little and big cluster thermal zones. 75

3.8 Spatial on-chip temperature $T(x, y)$ on the left and hot spots $h(x, y)$ on the right of I.MX8 Ixora with little (A53) cluster stressed at the top, and big (A72) cluster stressed at the bottom. 76

3.9 Average power and steady-state temperature of various benchmarks executed on tested platforms. 78

3.10 Values of task characteristics coefficients $o_{i,k}$ and $a_{i,k}$ of tested benchmarks on little (top semi-axis) and big (bottom semi-axis) clusters. . 81

3.11 Illustration of processing-idling intervals needed for the `LR` model. . . 82

3.12 Visualization of power models evaluation. 84

3.13 Illustration of a simplified window with a single processing-idling interval only, as used by `LR-UB`. 87

3.14 Measured and predicted power consumption of 1000 testing instances (mixed workload windows); instances are sorted by I.MX8 MEK measured power consumption. 93

3.15 Example schedules for instance no. 1 on I.MX8 MEK (mixed-workload). 95

3.16 Average difference between the measured steady-state temperature T_{measured} and the average ambient temperature T_{amb} for tested optimization methods. Recall that Ixora's lower temperatures are caused by applied air flow. 97

3.17 Average computation time of different methods w.r.t. the instance size n. 97

List of Tables

1.1 Example task parameters. 22
1.2 Number of generated instances with respect to utilization (columns) and number of tasks (rows). 37

2.1 Comparison of different CP modeling techniques on PRELIM dataset. 56
2.2 Comparison of upper bound ub, lower bound lb and runtime t between the models on MEDIUM dataset. Time-limit is 600 s and TLR stands for time-limit reached. 58
2.3 Comparison of upper bound ub, lower bound lb and runtime t between the models on LARGE dataset. Time-limit is 3600 s and TLR stands for time-limit reached. 59

3.1 Design choices and their coverage (✓) in this chapter. 68
3.2 Iterations per second (IPS) of used kernels. 72
3.3 Idle power consumption P_{idle} of tested platforms. 79
3.4 Task characteristics coefficients identified for used kernels on little cluster. 79
3.5 Task characteristics coefficients identified for used kernels on big cluster. 80
3.6 Regression coefficients identified for all tested platforms and coefficient of determination R^2. 84
3.7 Mean absolute error (in Watts) of the tested power models. 93
3.8 List of compared optimization methods and corresponding power models. 94
3.9 Power offset ($P_{\text{measured}} - P_{\text{idle}}$ [W]) observed for six instances with mixed workloads and six instances with CPU-bound workloads on tested platforms. 96

List of Algorithms

1 Generation of task parameters 37

2 Get a feasible allocation from the instantiation of variables x_i, or report a failure. 89
3 Greedy heuristic. 91

Chapter

A

Curriculum Vitae

Ondřej Benedikt was born in Prague, Czech Republic, in 1993. He received his bachelor's and master's degrees in 2016 and 2018, respectively, in the Open Informatics study program at Czech Technical University in Prague. During his studies, Ondřej specialized in Computer Science and Artificial Intelligence. He graduated with honors and ranked first among the other graduates. After obtaining the master's degree, Ondřej started a Ph.D. program at the Department of Control Engineering on scheduling with energy consumption considerations.

During the first two years of his Ph.D. studies, Ondřej worked on productions optimization problems. Two papers were published in impacted journals. The first one (in Computers & Operations Research) studied the scheduling of tasks on an industrial furnace, assuming its thermal behavior. The thermal state of the furnace was considered, and its model was integrated with the optimization procedure. The research was motivated by problems tackled in real production in Škoda Auto. In the second paper (published in Constraints), a more theoretical problem assuming both machine states and variable energy prices was considered. Besides publishing the papers, Ondřej participated in several conferences. He received Student Paper Awards at ICORES-2020 and CPAIOR-2020 conferences.

During the other two years of his Ph.D. studies, Ondřej participated in the Thermac project, which is a Horizon-2020 project tackling thermal management challenges for avionics systems in small aircraft. The behavior of the physical multi-core heterogeneous platform (i.MX8 by NXP) was analyzed, and allocation and scheduling techniques were proposed with the aim to reduce the peak thermal temperature of the platform attained during the workload execution. Again, several conference papers were published (FedCSIS, EMSOFT, RTCSA), and the RTCSA-2021 paper was awarded the Best Paper Award.

Throughout the whole Ph.D. study, Ondřej participated in teaching activities. He led the labs for the Combinatorial Optimization course at FEE CTU and helped with the digitalization of the lab's materials. Moreover, he successfully supervised two master's students, David Hornof and Radek Bumbálek.

Ondřej Benedikt
Prague, Fall 2022

Chapter B

List of Author’s Publications

Publications in Journals with Impact Factor

Ondřej Benedikt, Baran Alikoç, Přemysl Šůcha, Sergej Čelikovský, and Zdeněk Hanzálek. “A polynomial-time scheduling approach to minimise idle energy consumption: An application to an industrial furnace”. In: *Computers & Operations Research* 128 (2021), p. 105167. ISSN: 0305-0548. DOI: `10.1016/j.cor.2020.105167`
IF Q1, AIS Q1, Coauthorship 33%, Citations 3

Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. “Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States”. In: *Constraints* 25.3–4 (Dec. 2020), pp. 300–318. ISSN: 1383-7133. DOI: `10.1007/s10601-020-09317-y`
IF Q3, AIS Q3

International Conferences and Workshops

Ondřej Benedikt, Přemysl Šůcha, István Módos, Marek Vlk, and Zdeněk Hanzálek. "Energy-Aware Production Scheduling with Power-Saving Modes". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Ed. by Willem-Jan van Hoeve. Cham: Springer International Publishing, 2018, pp. 72–81. ISBN: 978-3-319-93031-2. DOI: `10.1007/978-3-319-93031-2_6`
CPAIOR, CORE B

Ondřej Benedikt., Přemysl Šůcha., and Zdeněk Hanzálek. "On Idle Energy Consumption Minimization in Production: Industrial Example and Mathematical Model". In: *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems - ICORES,.* INSTICC. SciTePress, 2020, pp. 35–46. ISBN: 978-989-758-396-4. DOI: `10.5220/0008877400350046`
ICORES, CORE C, Citation 1
Student Paper Award

Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Abstracts of Fast-Track Journal Paper. xxi-xxiii. Springer, Cham, 2020. ISBN: 978-3-030-58941-7. DOI: `10.1007/978-3-030-58942-4`
CPAIOR, CORE B
Best Student Paper Award

Michal Sojka, Ondřej Benedikt, Zdeněk Hanzálek, and Pavel Zaykov. "Testbed for thermal and performance analysis in MPSoC systems". In: *2020 15th Conference on Computer Science and Information Systems (FedCSIS).* 2020, pp. 683–692. DOI: `10.15439/2020F174`
FedCSIS, CORE B, Citation 1

Michal Sojka, Ondřej Benedikt, and Zdeněk Hanzálek. "Work-in-Progress: Determining MPSoC Layout from Thermal Camera Images". In: *2021 International Conference on Embedded Software (EMSOFT).* 2021, pp. 39–40
EMSOFT, CORE A

Ondřej Benedikt, Michal Sojka, Pavel Zaykov, David Hornof, Matěj Kafka, Přemysl Šůcha, and Zdeněk Hanzálek. "Thermal-Aware Scheduling for MPSoC in the Avionics Domain: Tooling and Initial Results". In: *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).* 2021, pp. 159–168. DOI: `10.1109/RTCSA52859.2021.00026`
RTCSA, CORE B, Citation 1
Best Paper Award

Supervised Master's Thesis

David Hornof. "Offline scheduling of the safety-critical tasks within the isolation time-windows". MA thesis. 2021. URL: `https://dspace.cvut.cz/handle/10467/95363`

Radek Bumbálek. "Proactive and reactive approaches for non-critical tasks scheduling under thermal constraints in the avionics domain". MA thesis. 2022. URL: `https://dspace.cvut.cz/handle/10467/99111`

Awards

ICORES 2020 Conference	Best Student Paper Award
CPAIOR 2020 Conference	Student Paper Award
RTCSA 2021 Conference	Best Paper Award

CTU

CZECH TECHNICAL
UNIVERSITY
IN PRAGUE