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

Doctoral Thesis

Scheduling under energy consumption limits

István Módos

January 2021

Scheduling under energy consumption limits

by

István Módos

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

Doctoral Thesis

Supervisors: doc. Ing. Přemysl Šůcha, Ph.D.
Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics
Submission date: January 2021

István Módos: Scheduling under energy consumption limits, Ph.D. Thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, January 2021, Prague.

Acknowledgments

First, I would like to thank my supervisor Přemysl Šůcha, who guided me through my studies, starting with my master thesis. Mr. Šůcha was always motivating, keen to provide any advice, and willing to patiently discuss every research problem. I learned a lot from him, and for that, I am grateful. I would like to also express my thanks to Zdeněk Hanzálek, the head of our research group. He strived to provide the best working conditions possible, and I respect him greatly, both scientifically and personally.

Without friends and colleagues, one would definitely not enjoy his work as with them. I had the luck to collaborate, laugh, share the passion for the teaching and optimization with the following amazing people: Antonín Novák, Ondřej Benedikt, Jan Dvořák, Anna Minaeva, Libor Bukata, Marek Vlk, Michal Bouška, Aasem Ahmad, and Roman Václavík. Moreover, fist bump to all the friends from our climbing party. The time spent together on solving interesting boulder problems meant really a lot to me.

I want to express my deep gratitude to my parents, Jarmila Módos and István Módos. Without their hard work and love of their children, I would not be where I am today.

Finally, I acknowledge all the grants and projects that funded my research. Namely, the work was supported by (i) the ARTEMIS initiative funded by the European Commission under the project DEMANES 295372 and (ii) the Technology Agency of the Czech Republic under the Centre for Applied Cybernetics TE01020197.

CHPMV

István Módos
Prague, January 2021

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.

István Módos
Prague, January 2021

Abstract

With the increasing costs for energy consumption of large manufacturing companies, the newest trend of energy-efficient and sustainable scheduling appeared within the scientific literature. To design an automated scheduling system for such companies, it not sufficient anymore to consider only traditional objectives such as makespan. The production schedules must also take the energy-related aspects into account. Otherwise, such schedules are not usable in practice, or they would require many hand-made changes by human schedulers.

One of such aspects addressed by this work is the energy consumption limit, representing an upper limit of the energy consumption within every metering interval of a day. The energy consumption limits are specified in a contract between a manufacturing company and an electric utility, and its violation is monetarily penalized.

The motivational example for this research is a company producing tempered glass, which violated the energy consumption limits a few times per month. Usually, the violation was caused due to uncertainty in the preparation time of the jobs. In such a situation, energy-demanding jobs were executed within the same metering interval; thus, the energy consumption increased, and the energy limit was violated.

This thesis investigates the scheduling problem with the energy limits, which is considered in both deterministic and stochastic environments. In the deterministic environment, it is expected that the start times of the jobs are never delayed; this problem has its application in fully automated production or in companies, which strictly penalize its workers for such delays. We studied the computation complexity of different problem variants: the main result is that the problem with fixed ordering of the jobs, constant number of machines, and assumption that the jobs cannot cross multiple metering intervals can be solved in polynomial time.

The study of the problem within a stochastic environment focuses on the robustness of the production schedules, i.e., how to obtain schedules that do not violate the energy consumption limits even in case of unexpected disturbances. Due to the complex nature of such a problem, only a single machine problem is investigated. However, we discovered that if the order of the jobs is fixed, such a problem can be solved in a pseudo-polynomial time; the developed algorithm is then utilized in exact and heuristic algorithms.

For both deterministic and stochastic environments, we proposed exact (optimization models, Branch-and-Bound, Logic-based Benders decomposition) and heuristic (adaptive local search) algorithms so that large instances with hundreds of jobs can be solved. The proposed methods were compared with with the adapted ones from the literature, which we improved and outperformed in the number of better solutions found.

Keywords: production scheduling, electric energy, energy consumption limits, robust scheduling

Abstrakt

S roustoucími náklady za spotřebu elektřiny pro velké výrobní podniky se v posledních letech objevil trend výzkumu v rámci energeticky efektivního a udržitelného rozvrhování. Při návrhu automatizovaných rozvrhovacích systémů pro takové podniky již nestačí brát v úvahu pouze tradiční kritéria jakými jsou např. délka rozvrhu. Výrobní rozvrhy musí zohledňovat i energetické aspekty, jinak jsou v praxi nepoužitelné, popř. je nutné je ručně upravit lidskými plánovači.

Jedním z takových aspektů, kterým se tato práce zabývá, je tzv. čtvrthodinové maximum. Je to horní limit na spotřebu elektrické energie v každou čtvrthodinu během dne. Tento limit si výrobní podniky sjednávají s distributorem elektřiny a jeho překročení je finančně penalizováno.

Motivační příklad pro tento výzkum byl podnik vyrábějící tvrzené sklo, kde k překročení čtvrthodinového maxima docházelo několikrát do měsíce. Často k této situaci došlo kvůli nepředvídatelnému prodloužení přípravného času operací, což vedlo k tomu, že se k sobě přiblížily energeticky náročné operace během stejné čtvrthodiny, během které spotřeba energie překročila sjednaný limit.

V této práci se proto zabýváme tímto rozvrhovacím problémem, který dále uvažujeme jak v deterministickém, tak v stochastickém prostředí. V deterministickém prostředí očekáváme, že začátky vykonávání operací nepodléhají zpoždění, což má své uplatnění buď v plně automatizované výrobě anebo v podnicích, kde pracovníci jsou za prodlevy penalizováni. Studovali jsme výpočetní složitost různých variant tohoto problému; hlavním výsledkem je, že varianta s fixním pořadím operací, neměnným počtem strojů a předpokladem, že operace nemohou protínat více intervalů, se dá řešit v polynomiálním čase.

Studium rozvrhovacího problému v stochastickém prostředí se zaměřuje studium postupů, jak zaručit, že výsledný výrobní rozvrh je robustní, tedy že i v případě, že nastanou neočekávané události, tak čtvrthodinová maxima nebudou porušena. Vzhledem k náročnosti podstaty problému bylo v této části uvažováno pouze rozvrhování na jednom stroji. Nicméně, objevili jsme, že pokud se zafixuje pořadí operací, tak se takový rozvrhovací problém dá vyřešit v pseudo-polynomálním čase; navržený algoritmus využíváme v exaktních a heuristických algoritmech.

Pro rozvrhování v deterministickém a stochastickém prostředí jsme navrhli několik algoritmů, jak exaktních (optimalizační modely, metoda větví a mezí, Logická Bendrova dekompozice), tak heuristických (adaptivní lokální prohledávání), abychom mohli řešit i velké instance se stovkami operací. Srovnali jsme naše přístupy s adaptovanými přístupy z literatury, které jsme dále vylepšili a překonali v počtu nejlepších nalezených řešení.

Klíčová slova: rozvrhování ve výrobě, elektrická energie, čvrthodinová maxima, robustnostní rozvrhování

Goals and Objectives

The thesis deals with production scheduling under energy consumption limits. The main goals of the work are:

1. Study the related energy-aware scheduling literature.
2. Formalize the scheduling problem with the energy consumption limits in both deterministic and stochastic production environments.
3. Investigate the proposed scheduling problem. Determine its computational complexity and design algorithms (both exact and heuristic) for solving the problem.
4. Propose a parametrized generator of the benchmark instances. Experimentally evaluate the performance of the developed algorithms.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

Chapter 1

Introduction

For successful integration of an advanced planning and scheduling system into energy-demanding manufacturing, it is no longer sufficient to consider only traditional aspects such as schedule makespan and deadlines. To provide efficient, practical, and realizable schedules, the machines' energy consumption has to be taken into account [41]. Although integration of the energy-awareness into the manufacturing scheduling is getting more and more attention [8, 20], there is still a gap between industrial needs and academic research [56].

This thesis focuses on one such energy-related aspect called *peak energy demand*. Usually, the manufacturing companies have a contract with an electric utility specifying a maximum energy limit that the companies may consume within periodically repeating intervals. Violation of this maximum limit leads to significant penalty fees; thus, the scheduling system must consider such constraints to provide feasible schedules. Unfortunately, even if the scheduling system provides schedules that satisfy the maximum energy consumption limits, the disturbances in the production (such as machine breakdowns or unpredictable preparation time) may lead to their violations. Therefore, the robustness of the schedules against such disturbances is necessary for their practical applicability.

The examples, which motivated this research, are two manufacturing companies: the first one produces tempered glass, and the second one is an automotive company manufacturing gearboxes. The production process in both companies is similar; that is, a material is heated to a high temperature in one of the many furnaces to obtain the desired physical properties. The heating is highly energy demanding; the furnaces consume dozens of kilowatts to keep their inner temperature at the operating level. Therefore, if multiple furnaces start heating the material simultaneously, the contracted limit is violated, and the companies pay the penalty fee. Moreover, especially in the glass production, the preparation time's uncertainty causes delays of material heating. Such randomness occasionally leads to violation of the maximum energy consumption limits, even if the energy limits were satisfied in the original production plan.

This chapter first reviews the components of the electricity bill, which helps to categorize the aims and contributions of the thesis within the energy-aware scheduling domain. Then, the considered problem of satisfying the maximum energy consumption limits in both deterministic and stochastic environments is explained in more detail. The rest of the chapter is devoted to the related work, thesis contributions, and its outline.

1.1 A Review of Electricity Bill Components

This section describes how industrial customers are billed for their electric energy consumption. The aim is to provide a broader context of the current situation to classify the research of this thesis better. The schema of the electricity bill components is illustrated in Fig. 1.1. In summary, this thesis focuses on avoiding

the peak energy demand charge for violating the maximum energy consumption limits within fixed metering intervals.

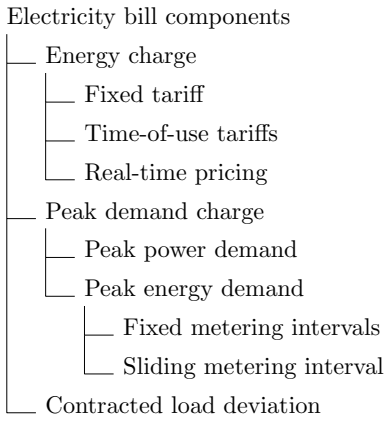

Figure 1.1: The electricity bill components.

The electricity bill has two primary components: *energy charge* and *peak demand charge*. The energy charge is related to the total consumed energy, i.e., the customers pay for every consumed kWh. The simplest energy contracts assume fixed tariffs, i.e., the energy price is the same regardless of time. From the electric utility point of view, it would be the most convenient if all the customers had a flat energy consumption profile since it is easy to determine the amount of power to be produced for constant consumption. However, this is not the case in reality, and the energy consumption varies during the day. Therefore, to reward the customers for flattening their energy consumption profiles, the electric utilities provide *demand response programs* [41]. Common demand response programs are *price-based* ones such as Time-of-Use (TOU) tariffs or Real-Time Pricing (RTP). In both programs, the cost of consumed energy may vary every hour; therefore, the customers are rewarded for shifting their electricity load to off-peak periods with lower energy costs. Within TOU response programs, there are few different pre-specified tariffs (such as on-peak, mid-peak, critical-peak, and off-peak tariffs), whereas in RTP the hourly energy cost follows the energy market and is subject to its non-deterministic nature.

The second component of the electricity bill, the peak demand charge, is related to the peak electricity consumption. The peak demand is charged either from the *peak power demand* or *peak energy demand*. For the peak energy demand, the computation is usually based on the energy consumption over a fixed or sliding metering intervals with 15 to 60 minutes duration [66], and the demand is charged for every kWh. For example, in the Czech Republic, the industrial customers are obliged to have a contracted *maximum energy consumption limit* with the electric utility. If the customer's demand is greater than the contracted limit in any metering interval, the customer pays a penalty fee, which is given by law [67].

For large-scale industrial and energy-intensive customers such as steel plants [51, 25], the electricity bill contains another component called *load deviation penalty*, reflecting the deviation from the contracted energy consumption in every hour of a day. Both over-consumption and under-consumption of the energy are penalized,

although small deviations from the contracted consumption are usually tolerated.

1.2 Scheduling with Maximum Energy Consumption Limits

This section explains the problem of scheduling with the maximum energy consumption limits in more detail. First, the scheduling problem in a deterministic environment is introduced. After that, we discuss the necessity to consider the robustness of the schedules w.r.t. the disturbances in the production.

When measuring the peak energy demand, the time interval of each day of the scheduling horizon is divided into *metering intervals* of a fixed length. The length of the metering intervals may vary from electric utility to electric utility. Moreover, in some countries, the length can be given by law, e.g., the length of the metering intervals is specified to be 15 minutes in the Czech Republic. The contract between the electric utility and the manufacturing companies requires that the total energy demand is below a contracted peak energy demand, which is called a *maximum energy consumption limit*. A visualization is shown in Fig. 1.2 illustrating the power consumption over a part of the scheduling horizon. The scheduling horizon is divided into 15-minutes metering intervals in which the total energy consumption is measured as the integral of the power consumption specified by function $P(t)$. For example, Fig. 1.2 shows the total energy consumption in the metering interval starting at 10:45 and ending at 11:00. The contract then enforces that the total energy consumption in every metering interval is below a contracted energy consumption limit $E^{\max}$.

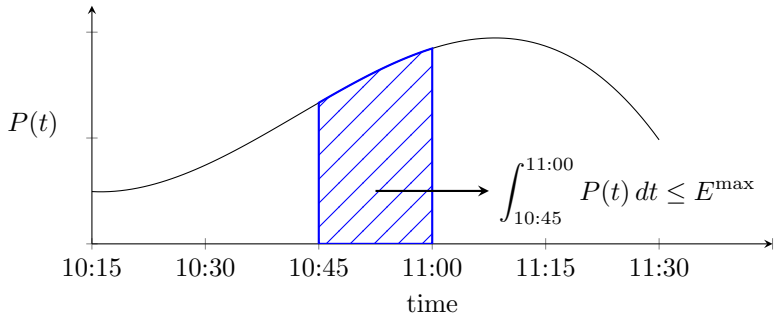

Figure 1.2: An illustration of the maximum energy consumption limit constraining the total energy consumption in the metering interval 10:45-11:00. $P(t)$ is a function representing the power consumption over time.

Not every schedule of the production jobs is feasible w.r.t. the contracted maximum energy consumption limit; consider an example illustrated in Fig. 1.3. The figure contains two parts: the upper one is a schedule of four production jobs on a single machine and the bottom one represents the total energy consumption in every metering interval I_i. The machine consumes power while processing a job, which may vary from job to job. The energy consumption of a machine while processing a job in a specific metering interval is represented by the height of the corresponding box in the bottom part of the figure. The figure reveals that

although this schedule is optimal w.r.t. the makespan, the total energy consumption in the first metering interval I_1 is larger than the maximum energy consumption limit $E^{\max}$, denoted by the red dashed line. Using more advanced algorithms, the scheduling system can provide feasible schedules w.r.t. these limits. One of such schedules is shown in Fig. 1.4, which has the same makespan as the previous one, but due to more suitable re-ordering of the jobs, the maximum energy consumption limit is satisfied in every metering interval.

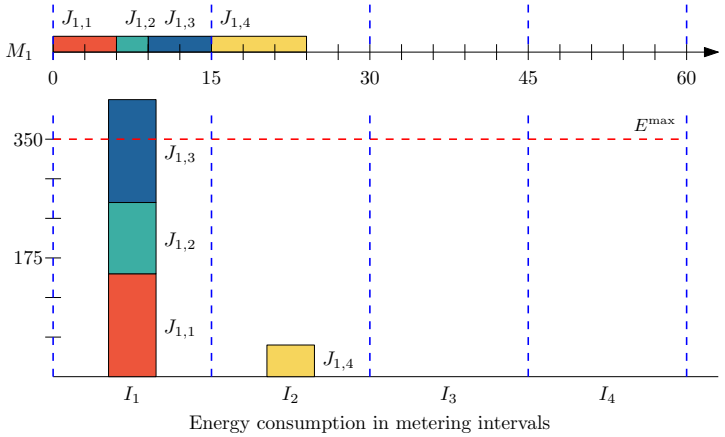

Figure 1.3: A schedule with an optimal makespan, which violates the maximum energy consumption limit in metering interval I_1.

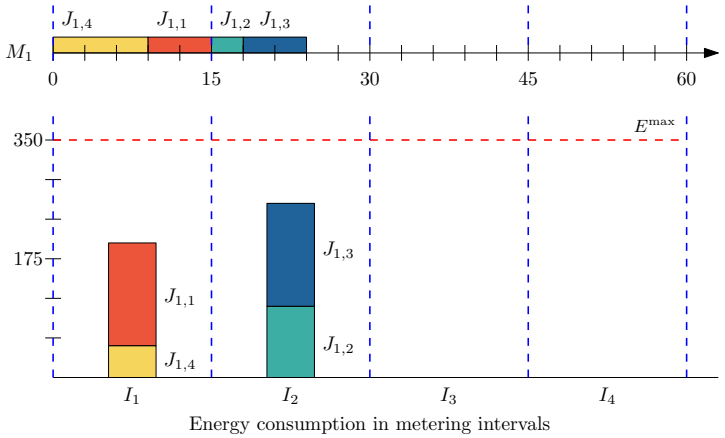

Figure 1.4: A schedule with an optimal makespan, which satifies the maximum energy consumption limit in every metering interval of the scheduling horizon.

However, in reality, unexpected events (e.g., inexperienced seasonal workers or delays in the preceding manufacturing stages) may delay the jobs' start times. We call the delayed start time a *realized start time*, whereas the initial non-delayed start time is a *baseline start time*. The issue is that if the energy-demanding jobs are scheduled close to each other in the baseline schedule, delaying a job may increase the energy consumption in some metering interval above the energy limit. For

example, in the schedule shown in Fig. 1.5 the start of job $J_{1,1}$ is delayed by 6 minutes. In such a situation, the energy-demanding jobs $J_{1,1}, J_{1,3}$ are processed within the same metering interval; thus, the maximum energy consumption limit is violated in I_2, and the manufacturing company must pay a penalty fee even though the baseline schedule (see Fig. 1.4) is feasible.

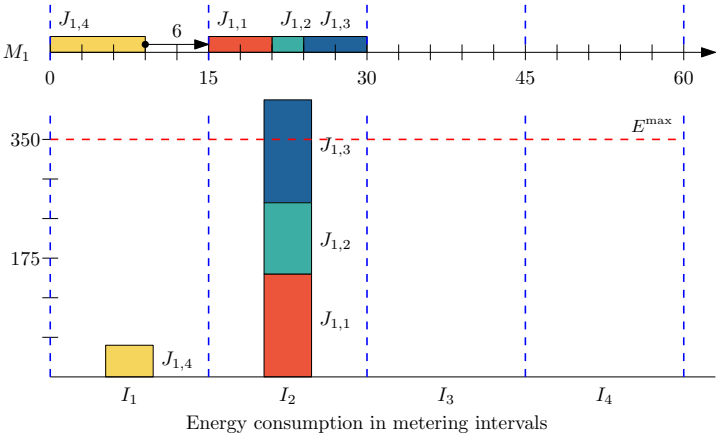

Figure 1.5: Delaying job $J_{1,1}$ by 6 minutes leads to the violation of the maximum energy consumption limit in metering interval I_2, even if the baseline schedule (shown in Fig. 1.4) satisfies the energy limits.

One possible approach to tackle these uncertainties is to use *a peak energy demand controller*, which monitors the total energy consumption. When the total energy consumption approaches the energy limit, the machines are turned off until the start of the next metering interval. However, such a solution may cause long downtimes in manufacturing if the jobs' order is not chosen reasonably beforehand. A more viable approach is to employ *pro-active scheduling*, i.e., the baseline schedule is designed to avoid the hazardous situations if the delays of the jobs are within a pre-specified tolerance. For example, the baseline schedule shown in Fig. 1.6 incorporates both better ordering of the jobs and forced idle times to ensure robustness against disturbances.

1.3 Related Work

1.3.1 Energy-aware Scheduling

The integration of automated scheduling and energy-awareness in the manufacturing is an ongoing and active field of research [20]. One of the critical properties in which the existing works differ is how they incorporate the energy-awareness into a production schedule. The first research direction considers the energy consumption to be a part of the objective function [39, 62, 25, 19, 11, 12, 69, 70, 71, 22, 1, 2, 13, 59, 52]. For example, the authors of [1] optimize both the energy and peak demand charges simultaneously in a parallel machine environment. Other works, such as [25], consider a more complex objective that combines electricity bill components with

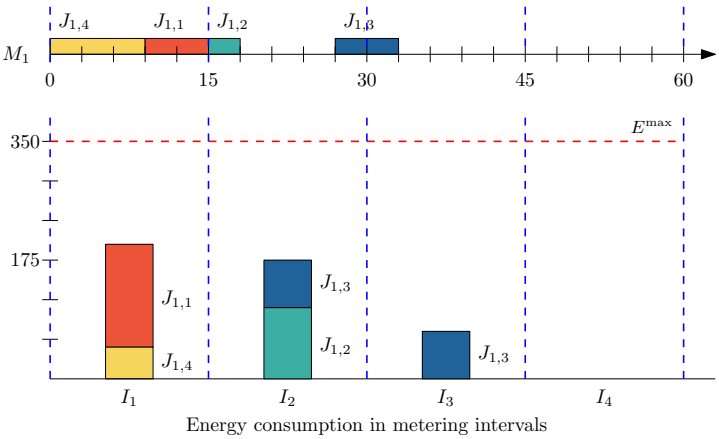

Figure 1.6: A baseline schedule that is robust against the delays of the jobs.

traditional scheduling objectives, e.g., the weighted start time of the tasks. The advantage of this multi-objective approach is that if each objective part can be expressed as a monetary cost, the found schedules' quality is easily interpretable by the end-user of such a scheduling system. However, since the underlying scheduling problems without considering the energy-awareness are already hard to solve, it is even more complicated to design efficient methods that exploit the combinatorial structure of such multi-objective problems.

On the other hand, the second research direction focuses on optimizing the traditional scheduling objectives, such as makespan or total tardiness, while formulating the energy-related constraints as *hard*, e.g., the contracted peak demand cannot be violated. Such an approach is suitable for industrial customers with fixed electricity tariffs. For example, [19] tackles the flow shop scheduling problem, where the goal is to design schedules that minimize the makespan and do not violate the contracted peak power demand. By considering a less complicated objective function, the authors exploited the structure of the problem and discovered valid inequalities that strengthened the mixed-integer linear programming models. The main disadvantage of these approaches is that they cannot tackle the multi-objective problems directly. However, by incorporating them in a meta-optimization algorithm that searches the optimal energy-related parameters, such as peak power limit, these approaches can indirectly handle even the multi-objective problems.

Few works are dealing with the scheduling and the demand charge. Since different electric utilities may have their conditions regarding the demand charge billing, the studied scheduling problems can differ significantly. However, two main streams can be identified: *peak power demand* [10, 18, 50, 61, 40, 68, 15] and *peak energy demand* [1]. In the case of peak power demand, the utilities charge the customers by their peak power consumption in every time unit. On the other hand, in the case of peak energy demand, the customers are billed according to the measured energy consumption in either fixed or sliding *metering intervals* of a given length (e.g., 15 minutes). Sometimes, the electric utilities limit the maximum peak energy demand by a contract.

In [1], the authors optimize both the energy charge and peak energy demand

measured in a sliding metering interval. The authors proposed a mixed-integer linear programming model and studied how the model parameters influence the resulting schedules. The proposed model's disadvantage is that it allows only one job to be processed by a machine in a metering interval. In the case of longer metering intervals and shorter jobs, this model may lead to significant idling of the machines.

Minimum and maximum energy consumption limits are considered in the electrical load commitment problem [51, 26, 27, 25, 21, 53] (also referred to as *load tracking* or *load deviation*), where the industrial customers must follow the pre-agreed energy consumption in every hour. Although over-consumption and under-consumption of the energy are penalized, a penalty-free region between the limits is assumed. The works mentioned above consider these limits to be soft, and they minimize the absolute difference between the actual and the contracted energy consumption measured in every metering interval. In contrast to the works mentioned earlier, this thesis assumes that these consumption limits are hard constraints.

There are some similarities between the energy consumption limit and *limited-capacity buffers* [3], as both can constraint the throughput. However, the key difference is that the energy consumption limit modeling requires computing the energy consumption over all the machines within each time interval. On the other hand, a buffer affects only the adjacent machines. Assume the following example, where the buffers cannot model the energy consumption limit. Consider a serial production line where all the machines have the same processing rate, and there are no machine failures. In such a case, the buffers are not used at all, i.e., a job completed on a machine is immediately processed on the succeeding one. Suppose that the energy consumption of the machines is high during the processing state. Thus, the sum of the energy consumption over all the machines within some time period can be larger than the limit, even if the buffers are empty all the time Hence, buffers alone do not provide information on whether the energy consumption limit is violated or not.

The problem considering the energy consumption limit is a special case of *resource-constrained project scheduling with partially renewable resources* [4] denoted as RCPSP/π. The partially renewable resources allow the capacities of the resources to be renewed in the specified intervals; thus, they are a generalization of renewable and non-renewable resources. A new partially renewable resource is created for every metering interval to transform the scheduling problem with the maximum energy consumption limit to RCPSP/π. Each resource's available capacity equals the maximum energy limit, and the set of the associated periods for this new resource is the set of all time instants within the corresponding metering interval. The heuristic algorithm for RCPSP/π presented in [4] achieves good results. However, since it is a time-based algorithm, its computational complexity depends on the scheduling horizon's length, and thus it is only suitable for instances with short scheduling horizons.

1.3.2 Robust Scheduling

Robust scheduling is a well-studied problem in the domain of resource-constrained project scheduling [29, 37]. The robustness is obtained either by a robust resource

allocation or inserting time buffers between activities. In the resource-constrained project scheduling domain, the closest problem to ours is presented in [57]. This work aims to find a partial-ordering of the activities so that if the activities are arbitrarily delayed (w.r.t. the ordering), the total demand of the resources in every time instant is below the respective capacities. The difference from the problem considered in this thesis is that considered problem limits the integral of the jobs' demands.

A particular interest for us is the modeling using *uncertainty scenarios* [16, 14], which are used when the probability distribution of the uncertain events is either not known or is uniform. An uncertainty scenario is one realization of the uncertain events, e.g., a time occurrence of a machine breakdown. In general, the scheduling with uncertainty scenarios aims to mitigate the worst-case execution over all uncertainty scenarios.

To the best of our knowledge, only a few works deal with robust scheduling and energy consumption limits. One such work is [49], where the goal is to reduce the peak energy consumption of flow shop schedules under uncertain processing times of the jobs. The method proposed by the authors inserts idle times into the schedule to reduce the expected peak energy demand. The time points for idle times are computed by evaluating all the possible schedules originating from the set of possible scenarios and. Therefore, the running time of the algorithm can increase significantly with the size of this set.

1.4 Contributions

This thesis studies the scheduling problem with the maximum energy consumption limits in both deterministic and stochastic environments. The scheduling problems are investigated from theoretical and practical standpoints. The theoretical study is vital for designing the algorithms, i.e., it clarifies whether some sub-problems can be solved efficiently. For example, if a global optimization method, such as *a genetic algorithm*, is used to tackle a scheduling problem, an individual's encoding must be decided. One possible encoding is to represent the individual solution using its start times directly; however, such encoding might not be efficient as it does not guarantee active schedules. Instead, a more natural encoding is to consider a fixed ordering of the jobs on which the mutation and crossover are applied. To obtain the actual schedule from the fixed ordering, a *scheduling operator* is used; that is, the scheduling operator solves the sub-problem of the original scheduling problem where the ordering of the jobs is assumed to be fixed. If such a sub-problem is proven to be efficiently solvable, then the genetic algorithm with an exact scheduling operator may quickly find high-quality solutions. On the other hand, by proving that the sub-problem is hard justifies using a heuristic scheduling operator.

The key contributions are

- deterministic environment:

    1. A complexity study of different variants of the scheduling problem with the energy consumption limits. The studied variants consider unit processing times, fixed ordering of the jobs on the machines, constant

number of the metering intervals, and a no-crossing assumption, i.e., the jobs may not cross the interval boundaries.

2. A polynomial algorithm finding the shortest schedule satisfying the maximum energy consumption limits for a fixed permutation of the jobs on a fixed number of machines with the assumption that the jobs cannot cross multiple metering intervals.
3. Efficient Constraint Programming (CP) model for the multi-machine scheduling under maximum energy consumption limits. We also improve the existing Integer Linear Programming (ILP) model from the literature for the same problem by identifying a property that enables to reduce the number of constraints in this model.
4. An adaptive local search algorithm with a heuristic scheduling operator for the large instances.

- stochastic environment:

1. A pseudo-polynomial algorithm for a single machine scheduling problem with a fixed permutation of the jobs, which finds an optimal schedule w.r.t. any regular objective function and which does not violate the energy consumption limits even if the jobs are delayed. A regular objective is an objective that is non-decreasing in the completion times of the jobs. Many common objectives, such as makespan, total tardiness, and total completion times are regular; therefore, the algorithm has a wide range of applications.
2. The algorithm mentioned above for the fixed permutation of the jobs is exploited in exact methods (Branch-and-Bound and Logic-based Benders Decomposition) for finding the optimal and robust permutation of the jobs. Moreover, the same algorithm is also utilized as a scheduling operator in an adaptive local search heuristic for tackling larger instances.

1.5 Outline

The following Chapter 2 formally states the scheduling problem considered in this thesis. The problem statement is general enough to encapsulate different variants that are studied in the later chapters. The next Chapter 3 lays the preliminaries for the thesis. Specifically, the ILP and CP modeling formalism are introduced, which are then used throughout the work. Moreover, Chapter 3 describes the general adaptive local search framework that is used for designing heuristic algorithms.

The main parts of the thesis are Chapters 4 and 5. The former Chapter 4 investigates different variants of the scheduling problem with the energy consumption limits in the deterministic environment, i.e., the jobs are not subject to disturbances in the manufacturing. The results of this chapter were described in [44].

The design of the algorithms that provide robust schedules against start times disturbances is studied in the latter Chapter 5. The content of this chapter was published in [42] (albeit with different objective function under study).

The last Chapter 6 concludes the work, discusses the fullfilment of the goals and possible future work.

Chapter 2

General Problem Statement

This section provides the formal statement of the scheduling problem with the energy consumption limit. The problem statement is general as much as possible so that its special variants can be investigated in the later chapters.

Let $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ be a set of *machines*. Each machine $M_k \in \mathcal{M}$ is *dedicated* to a set of *jobs* $\mathcal{J}_k^M = \{J_{k,1}, J_{k,2}, \ldots, J_{k,n_k}\}$, on which the jobs must be processed without preemption. The set of all jobs is defined as $\mathcal{J} = \bigcup_{M_k \in \mathcal{M}} \mathcal{J}_k^M$, and its cardinality will be denoted as n. It is assumed that $m \leq n$.

Each job $J_{k,j} \in \mathcal{J}$ is defined by its

- *processing time* $p_{k,j} \in \mathbb{Z}_{>0}$: time required to process the job by the dedicated machine.
- *power consumption* $P_{k,j} \in \mathbb{Z}_{\geq 0}$: the power consumption of the dedicated machine when processing the job, i.e., it is the constant rate at which the energy is consumed by the machine at every time instant. The total consumed energy by the dedicated machine when processing job $J_{k,j}$ is $p_{k,j} \cdot P_{k,j}$.

  In general, a machine's power consumption profile can be either non-linear or may be split into intervals with approximately constant consumption [60, 65]. In this thesis, it is assumed that the machines consume constant power while processing a job; the constant power represents either the average power (if the power consumption profile is non-linear) or the actual power in the processing state (in case of the constant-consumption intervals).

The jobs have to be scheduled within *scheduling horizon* $H \in \mathbb{Z}_{>0}$, which is divided into a set of *metering intervals* $\mathcal{I} = \{I_1, I_2, \ldots, I_{\frac{H}{D}}\}$ with equal *length* $D \in \mathbb{Z}_{>0}$ (it is assumed that H is a multiple of D). Each metering interval $I_i \in \mathcal{I}$ starts at time $\tau_i^{\text{start}} = (i-1) \cdot D$ and ends at time $\tau_i^{\text{end}} = i \cdot D$. Moreover, *a maximum energy consumption limit* $E_i^{\max}$ is defined to represent the upper bound on the jobs' total energy consumption in each metering interval $I_i \in \mathcal{I}$.

Let $s_{k,j} \in \mathbb{Z}_{\geq 0}$ denote a *baseline start time* of job $J_{k,j} \in \mathcal{J}$ and $\boldsymbol{s} = (s_{k,j})_{J_{k,j} \in \mathcal{J}}$ to be *baseline start times*. Since unexpected events may happen during the execution of a schedule, the actual start times of the jobs may be delayed from their baseline start times; the actually carried out schedule is called *a realized schedule*. To formally define the realized schedules, the notion of scenarios and the maximum deviation has to be introduced. Let $\delta^{\max} \in \mathbb{Z}_{\geq 0}$ be a *maximum deviation* of any job. Then

$$\Delta = \{(\delta_{k,j})_{J_{k,j} \in \mathcal{J}} \mid \delta_{k,j} \in [0 \,..\, \delta^{\max}]\} \tag{2.1}$$

is *a set of all scenarios*, where $\delta_{k,j}$ represents *a deviation* of job $J_{k,j} \in \mathcal{J}$; notation $[a \,..\, b] \subset \mathbb{Z}$ is a shorthand for a set of integers $\{a, a+1, \ldots, b\}$. An element $\boldsymbol{\delta}^{(q)} \in \Delta$ is called *deviation scenario* that corresponds to some realized schedule (this claim is proven later in the thesis). By convention, deviation situation $\boldsymbol{\delta}^{(1)}$ corresponds to *zero deviation*, i.e., $\boldsymbol{\delta}^{(1)} = \mathbf{0}$. The maximum deviation is a user parameter, which can be set according to the required range of the covered realized schedules.

Let $\pi_k : [1\,..\,n_k] \to [1\,..\,n_k]$ be a bijective function representing a *permutation of the jobs* on machine $M_k \in \mathcal{M}$, i.e., function π_k maps position ℓ in the permutation to job index j. We will say that $(\pi_k)_{M_k \in \mathcal{M}}$ are the *corresponding* permutations of $\boldsymbol{s}$ if the ordering of the jobs in $\boldsymbol{s}$ on every machine $M_k \in \mathcal{M}$ is the same as in π_k.

Given baseline schedule $\boldsymbol{s}$, its corresponding permutations $(\pi_k)_{M_k \in \mathcal{M}}$, and arbitrary scenario $\boldsymbol{\delta} \in \Delta$, *a realized schedule* can be derived using recursive function RS as

$$RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,\pi_k(\ell)} = \begin{cases} s_{k,\pi_k(1)} + \delta_{k,\pi_k(1)} & \ell = 1 \\ \max\{s_{k,\pi_k(\ell)}, RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,\pi_k(\ell-1)} + p_{k,\pi_k(\ell-1)}\} + \delta_{k,\pi_k(\ell)} & \text{otherwise.} \end{cases} \tag{2.2}$$

Notice that for $\boldsymbol{\delta}^{(1)} \in \Delta$, the baseline start times are the same as the realized start times, i.e., $RS(\boldsymbol{s}, \boldsymbol{\delta}^{(1)})_{k,j} = s_{k,j}, \forall J_{k,j} \in \mathcal{J}$. Moreover, the definition of the realized schedules implies $\delta_{k,\pi_k(\ell)} \leq RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,\pi_k(\ell)} - s_{k,\pi_k(\ell)}$. To make the distinction clear, the value of $RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,\pi_k(\ell)} - s_{k,\pi_k(\ell)}$ is called a *delay*. Informally, deviation $\delta_{k,j}$ of job $J_{k,j}$ is independent of the deviations of the other jobs, whereas the delay is not.

To ensure that no job completes outside of the horizon, even if all jobs are delayed, a *maximum start time* for job $J_{k,j} \in \mathcal{J}$ is defined as

$$s^{\max}_{k,j} = H - (n_k \cdot \delta^{\max} + p_{k,j})\,, \tag{2.3}$$

where value $n_k \cdot \delta^{\max}$ represents the worst case delay of any job on machine M_k in any realized schedule.

Baseline start times $\boldsymbol{s}$ are *feasible* if:

1. The baseline start time of every job $J_{k,j} \in \mathcal{J}$ is at most $s^{\max}_{k,j}$.
2. The jobs scheduled on the same machine are not overlapping in time.
3. The energy constraints are not violated in any realized schedule, i.e.,

$$\sum_{J_{k,j} \in \mathcal{J}} \text{Overlap}(RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,j}, p_{k,j}, i) \cdot P_{k,j} \leq E^{\max}_i\,, \quad \forall \boldsymbol{\delta} \in \Delta; \forall I_i \in \mathcal{I}, \tag{2.4}$$

   where $\text{Overlap}(RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,j}, p_{k,j}, i)$ represents the overlap length of job $J_{k,j} \in \mathcal{J}$ with metering interval I_i if the job starts at time $RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,j}$.

If $\delta^{\max} \geq 1$, then the feasible baseline start times are called *robust*.

The goal of this scheduling problem is to find feasible baseline start times while optimizing some scheduling objective function γ. This work focuses on the minimization of *makespan* of the baseline schedule

$$C_{\max} = \max_{J_{k,j} \in \mathcal{J}} s_{k,j} + p_{k,j}\,. \tag{2.5}$$

This scheduling problem is denoted in Graham's scheduling notation [23] as $\text{PD}m|E^{\max}_i, \delta^{\max}_{k,j} = \delta^{\max}|\gamma$, where PD$m$ stands for the parallel dedicated machines [33], $E^{\max}_i$ corresponds to the maximum energy consumption limits in the metering intervals, and $\delta^{\max}_{k,j} = \delta^{\max}$ represents the maximum deviation of the jobs.

The rationale for assuming integer start times In practical applications, assuming integer start times is not restrictive since the granularity of the scheduling start times can always be scaled to the desired precision (e.g., minutes, seconds). Moreover, some of the state-of-the-art solvers that are used in this thesis (such as IBM CP Optimizer), can handle only integer variables.

The rationale for studying parallel dedicated machines problems The first reason is technological, i.e., the machines may not be identical (tempering furnaces, melting furnaces, drilling machines, etc.). The jobs represent different operations within the production process, which can be processed only by a specific machine type. The second reason is exploiting of the algorithms within decomposition methods, such as Logic-based Benders decomposition (LBBD), that can tackle the parallel machines environment [48, 64]. Here, the scheduling problem is usually decomposed into two phases that are performed iteratively: (i) a *master problem*, which finds an assignment of the jobs to the machines, and (ii) a *subproblem*, which sequences the jobs according to the fixed jobs' assignment obtained by solving the master problem. If the fixed jobs' assignment is not feasible in the subproblem, a cutting constraint is generated for the master problem to forbid the infeasible assignment. In such a case, the subproblem corresponds to the dedicated parallel machines problem. Therefore, efficient algorithms for the scheduling problem with the dedicated parallel machines are stepping stones towards solving the identical parallel machines problem using decomposition methods. Finally, in comparison to the single machine problems, the study of the multi-machine environment is more practically relevant since it provides a holistic perspective on the energy consumption of a factory [28].

2.1 Example

The following example illustrates the studied scheduling problem. The example considers two machines $\mathcal{M} = \{M_1, M_2\}$ with jobs set $\mathcal{J}_1^M = \{J_{1,1}, J_{1,2}, J_{1,3}, J_{1,4}\}$, and $\mathcal{J}_2^M = \{J_{2,1}, J_{2,2}\}$, respectively. Let $H = 30$, $D = 5$, $\mathcal{I} = \{I_1, I_2, I_3, I_4, I_5, I_6\}$, and $E_i^{\max} = E^{\max} = 60; \forall I_i \in \mathcal{I}$. The parameters of the jobs are provided in Table 2.1. A feasible baseline schedule $\boldsymbol{s}$ with one possible realized schedule $\boldsymbol{rs}$ are given in Table 2.2; the visualization of these schedules is shown in Fig. 2.1 and Fig. 2.2, respectively. The figures show both the schedules and the energy consumption in the metering intervals. The energy consumption is represented as a barplot, where the height of each colored box is the energy consumption of the corresponding job in a metering interval. The energy limits are represented by horizontal dashed lines.

Table 2.1: Parameters of the jobs in the example.

Job	$p_{k,j}$	$P_{k,j}$	$s_{k,j}^{\max}$
$J_{1,1}$	4	12	18
$J_{1,2}$	2	20	20
$J_{1,3}$	2	12	20
$J_{1,4}$	2	6	20
$J_{2,1}$	3	14	23
$J_{2,2}$	3	8	23

Table 2.2: Baseline start times $\boldsymbol{s}$ of one possible feasible solution and the corresponding realized start times $\boldsymbol{rs}$ for $\boldsymbol{\delta}$.

Job	$s_{k,j}$	$\delta_{k,j}$	$rs_{k,j} = RS(\boldsymbol{s}, \boldsymbol{\delta})$
$J_{1,1}$	0	2	2
$J_{1,2}$	10	2	12
$J_{1,3}$	15	0	17
$J_{1,4}$	12	1	15
$J_{2,1}$	5	1	6
$J_{2,2}$	14	0	14

The energy consumption in the metering intervals of the baseline and the realized schedule is computed in Table 2.3 and Table 2.4, respectively. The tables show that both schedules do not violate the energy consumption limits.

Table 2.3: Energy consumption in metering intervals in $\boldsymbol{s}$.

Metering interval	$\sum_{J_{k,j} \in \mathcal{J}} \text{Overlap}(s_{k,j}, p_{k,j}, i) \cdot P_{k,j}$
I_1	$4 \cdot 12 = 48$
I_2	$3 \cdot 14 = 42$
I_3	$2 \cdot 20 + 2 \cdot 6 + 1 \cdot 8 = 60$
I_4	$2 \cdot 12 + 2 \cdot 8 = 40$
I_5	0
I_6	0

Table 2.4: Energy consumption in metering intervals in $\boldsymbol{rs}$.

Metering interval	$\sum_{J_{k,j} \in \mathcal{J}} \text{Overlap}(rs_{k,j}, p_{k,j}, i) \cdot P_{k,j}$
I_1	$3 \cdot 12 = 36$
I_2	$1 \cdot 12 + 3 \cdot 14 = 54$
I_3	$2 \cdot 20 + 1 \cdot 8 = 48$
I_4	$2 \cdot 6 + 2 \cdot 12 + 2 \cdot 8 = 52$
I_5	0
I_6	0

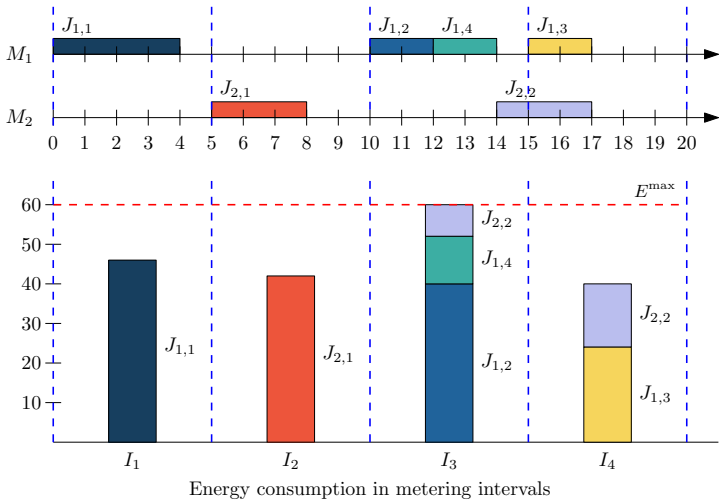

Figure 2.1: Baseline schedule $\boldsymbol{s}$ and the corresponding energy consumption in each metering interval.

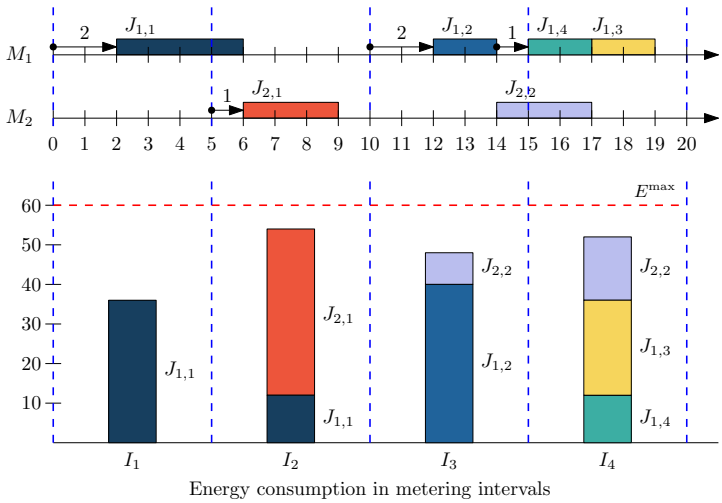

Figure 2.2: Realized schedule $\boldsymbol{rs}$ and the corresponding energy consumption in each metering interval.

2.2 Notation of the Scheduling Problem Variants

Since this thesis investigates different variants of the scheduling problem stated above, the variants need to be categorized. The following notation is used to describe each problem variant:

- $E_i^{\max} = E^{\max}$: Denotes a group of scheduling problems where the maximum energy consumption limit in every metering interval equals $E^{\max}$.
- $p_{k,j} = 1$: The processing time of all the jobs is 1.

- $|\mathcal{I}| = 2$: The number of metering intervals is 2.
- π_k: Order of the jobs on each machine $M_k \in \mathcal{M}$ is fixed, and it is given by function π_k, which maps position ℓ in the fixed order to the job index j. To ease the notation, it is assumed that the jobs' indices are sorted according to π_k, i.e., $\forall M_k \in \mathcal{M}; \forall \ell, \ell' \in [1 \,..\, n_k] : \ell < \ell' \implies \pi_k(\ell) < \pi_k(\ell')$.
- no-crossing: Any job in a feasible schedule $\mathbf{s}$ cannot cross the boundary of two metering intervals, i.e.,

  $$\text{Overlap}(s_{k,j}, p_{k,j}, i) = 0 \ \vee \ \text{Overlap}(s_{k,j}, p_{k,j}, i) = p_{k,j} \,, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I}. \tag{2.6}$$

  Notice that the no-crossing requirement implies that the processing time of each job is at most D.
- $\gamma = f$: Represents a *regular objective*, i.e., an objective function that is non-decreasing in the completion times of the jobs [55].
- $\gamma = \emptyset$: Represents the decision variant of a scheduling problem.

Chapter 3

Preliminaries

Before exploring the structure of the scheduling problem with the energy consumption limits, foundations need to be laid out. The first one is two modeling formalisms used thorough the whole thesis for formulating different scheduling problems. The second one is an adaptive local search framework which is used to design heuristic algorithms.

3.1 Modeling Formalisms

When a researcher or a practitioner is faced with the necessity to solve some optimization problem, there are two possible approaches to choose from. The first one is to spend some time researching the problem, and based on the obtained insight, to develop an algorithm that can solve it. The researcher may discover clever tricks or a combinatorial structure that lead to a very efficient algorithm. However, developing an algorithm is a long, tedious, and costly process with no guarantees on the final result.

The second approach is to model the problem in a formalism that is based on an existing theoretical framework. The modeling formalisms allow the researchers to describe the problem in a declarative way, which is then solved using some off-the-shelf solver. Therefore, the researcher can exploit the highly optimized state-of-the-art commercial solvers that usually find good solutions in a reasonable time. In many cases, such solutions are even better than what would be found by a simple algorithm developed by the researcher. However, the disadvantage of these solvers is that they mostly behave as black-boxes, i.e., different models of the same problem may yield different performance. Still, creating multiple models and comparing them usually takes less time than designing a problem-specific algorithm.

This thesis uses *Integer Linear Programming* and *Constraint Programming* modeling formalisms, which are described shortly in the following text.

3.1.1 Integer Linear Programming

ILP is an optimization problem expressed as [34]

$$\max_{\boldsymbol{x} \in \mathbb{Z}^n} \boldsymbol{c}^T \cdot \boldsymbol{x} \tag{3.1}$$

$$\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}\,, \tag{3.2}$$

where $\boldsymbol{A} \in \mathbb{Z}^{m \times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n$. Inequalities (3.2) are called *constraints* and they define a set of *feasible solutions* $\{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}\}$. The task of ILP is to (i) find *an optimal* feasible solution $\boldsymbol{x}^*$ that maximizes *objective function* $\boldsymbol{c}^T \cdot \boldsymbol{x}^*$, (ii) decide that there is no feasible solution, i.e., $\{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}\} = \emptyset$, or (iii) determine that the program is *unbounded*, i.e., $\sup\{\boldsymbol{c}^T \cdot \boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{Z}^n, \boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}\} = \infty$.

ILP is a generalization of Linear Programming (LP) problem that assumes $\boldsymbol{x} \in \mathbb{Q}^n$. This seemingly small change has a huge impact on the computational

complexities of these problems: whereas LP is known to be solvable in polynomial time, ILP is $\mathcal{NP}$-hard. When only a subset of $\boldsymbol{x} \in \mathbb{Q}^n$ are required to be integers, the problem is called Mixed Integer Linear Programming (MILP). Unfortunately, MILP is still an $\mathcal{NP}$-hard problem.

A common way to solve general MILP problems is a Branch-and-Bound (BAB) algorithm, which is essentially a guided enumeration method that uses a *relaxation* of the original problem (usually LP relaxation). Commercial state-of-the-art MILP solvers, such as Gurobi [24] and IBM CPLEX [31], implement BAB algorithm while adding additional improvements (e.g., model presolving, primal heuristics, cutting planes, parallelism, etc.) to increase the algorithm's efficiency.

To ease the formulation of MILP problems, the solvers provide various user-friendly modeling functions. Moreover, in some cases using such functions also allows us to avoid issues with numerical instability. In this thesis, the following modeling constructs are used.

1. Special-Ordered Set (SOS) constraint of type 2: Given an ordered list of variables $(x_{i_1}, x_{i_2}, \ldots, x_{i_q})$, SOS constraint of type 2 ensures that at most two consecutive variables in the list can have non-zero values. This constraint will be written as
$$\mathrm{SOS}_2(x_{i_1}, x_{i_2}, \ldots, x_{i_q})\,. \tag{3.3}$$

2. *Indicator* constraint: Quite often, we want some constraint $\boldsymbol{a}_k^T \cdot \boldsymbol{x} \leq b_k$ to hold if binary variable x_i takes specific value v. A common way how to model such a requirement is using *big M method*
$$\boldsymbol{a}_k^T \cdot \boldsymbol{x} \leq b_k + M(v - x_i)\,, \tag{3.4}$$
where M is a sufficiently large number. The disadvantages of the big M method are its poor linear relaxation and possible numerical instability caused by large M. Alternatively, modern solvers provide *indicator* constraints
$$x_i = v \implies \boldsymbol{a}_k^T \cdot \boldsymbol{x} \leq b_k \tag{3.5}$$
that are more numerically stable.

3. max constraint: This constraint sets variable x_r to be the maximum value of variables $\{x_{i_1}, x_{i_2}, \ldots, x_{i_q}\}$, i.e.,
$$x_r = \max\{x_{i_1}, x_{i_2}, \ldots, x_{i_q}\}\,. \tag{3.6}$$
To simplify the notation of the models in this thesis, we generalize the notation of the max constraint to arbitrary linear expressions instead of just variables.

An important note is that all of the modeling constructs mentioned above can be expressed equivalently as linear constraints (3.2).

3.1.1.1 Modeling of Scheduling Problems in MILP

Scheduling problems are commonly modeled using one of the following three approaches [35]

1. disjunctive (or relative order) modeling: the start times of the jobs are modeled as integer variables. The non-overlap between each pair of jobs $J_{k,j}, J_{k,j'}$ is enforced by a constraint requiring that one job completes before the other starts. The non-overlap constraint can be formulated using either indicator constraints or big M method, e.g.,

$$s_{k,j} + p_{k,j} \leq s_{k,j'} + M \cdot x \tag{3.7}$$
$$s_{k,j'} + p_{k,j'} \leq s_{k,j} + M \cdot (1 - x)\,, \tag{3.8}$$

   where x is a binary variable and M is a sufficiently large number.

2. time-indexed modeling: the scheduling horizon is divided into atomic time units of an equal length. For each job $J_{k,j}$ and each time unit t, a new binary variable is created denoting whether job $J_{k,j}$ starts at time t. Obviously, the performance of the time-indexed models depends on the length of the scheduling horizon, which may exclude them from being used in practice.

3. rank-based modeling: here, the job ordering π_k is modeled explicitly. A new binary variable is created for each $J_{k,j}$ and position ℓ representing whether the job is in ℓ-th position in the ordering. Moreover, the start times variables are related to the positions instead of jobs, allowing us to easily model the non-overlap constraint by requiring that job on position $(\ell - 1)$-th position completes before job on ℓ-th position starts.

3.1.2 Constraint Programming

CP [58] deals with solving combinatorial problems that are described declaratively using *constraints*. A constraint is a relation on a set of variables, and the goal is to assign values to the variables so that all the constraints are satisfied; this is the so-called Constraint Satisfaction Problem (CSP).

Formally, a CSP is given as a triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ is a set of *variables*, $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ is a set of *variable domains*, and $\mathcal{C}$ is the set of *constraints*. Each constraint $C_j \in \mathcal{C}$ is represented by a pair (R_j, S_j), where R_j is a relation on the corresponding domains of the variables in *scope* S_j. A *solution* to the given $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ is a *complete assignment* represented by a tuple $A = (a_1, a_2, \ldots, a_n)$, where $a_i \in D_i$ is an assignment of variable x_i to some value of its corresponding domain D_i. Complete assignment A is *feasible* if for every $C_j \in \mathcal{C}$ the projection of A onto scope S_j is in R_j. Similarly as in MILP, the task of CSP is to find a feasible complete assignment or to decide that no such assignment exists. CSP can be extended to the combinatorial optimization problems, which leads to Constraint Optimization Problem (COP); both of them are branches of the CP field.

To find a feasible optimal assignment, the general-purpose CSP solvers employ various tricks such as *inference techniques* that reduce the variable domains according to the incumbent partial assignment, which would lead to an infeasible solution. Another trick is the use of *global constraints* for which the solvers implement specialized algorithms. For example, a constraint requiring that a set of variables are assigned with pair-wise different values can be satisfied by finding a maximal matching in a graph.

The advantage of CP over MILP is in easier modeling. For modeling and solving CP models in this thesis, state-of-the-art IBM CP Optimizer [36] is used that provides specific constructs for scheduling problems. The following text shortly introduces some of these scheduling constructs.

The jobs in IBM CP Optimizer are usually modeled using *interval variables*. The interval variables represent time intervals during which something is being done. Each interval variable has a *start time*, an *end time*, and a *length*, all being variables by themselves. If a total ordering of a set of interval variables is required for modeling purposes (such as obtaining a preceding interval variable to another one), a *sequence variable* can be used.

The most common requirement in scheduling problems is that the jobs must not overlap each other for which IBM CP Optimizer provides global constraint NoOverlap. Formally, given some set of interval variables $\{x_{i_1}, x_{i_2}, \ldots, x_{i_q}\}$, the non-overlap constraint is denoted as

$$\text{NoOverlap}(x_{i_1}, x_{i_2}, \ldots, x_{i_q})\,. \tag{3.9}$$

Alternatively, the non-overlap constraint also accepts a sequence variable. In this case, the non-overlap is forced on the set of interval variables associated with the sequence variable.

The last construct from IBM CP Optimizer is function Overlap, which computes the overlap length between given interval variable x_i and fixed time interval having start s and end e, i.e.,

$$\text{Overlap}(x_i, s, e)\,. \tag{3.10}$$

3.2 General Adaptive Local Search Heuristic

In general, a local search is a heuristic approach for solving optimization problems. In each iteration, a local search generates a set of *neighbor* solutions by perturbing the current *incumbent* solution. If the best neighbor (w.r.t. the optimized objective function) is not worse than the incumbent solution, it becomes the incumbent solution for the next iteration. The whole procedure is repeated until the specified time-limit is not reached.

In the general local search framework used for solving scheduling problems in this thesis, each solution is represented by an *all-jobs ordering* denoted as λ, which is a permutation of all the jobs from $\mathcal{J}$. The start times for the given all-jobs ordering λ are obtained from a *scheduling operator algorithm*, which is specific to each optimization problem. Multiple neighborhood generators are implemented, and in each iteration, the local search adaptively selects the one which has the highest potential to generate good solutions.

The above-mentioned local search procedure is shown in Algorithm 1. The local search starts from a solution provided by an *initial heuristic*. The all-jobs ordering of the incumbent solution is denoted as λ^{best}, and its corresponding start times as $\mathbf{s}^{\text{best}}$. Finally, a generated neighbor is denoted as λ^{nh}.

The rest of this chapter explaons how the neighborhood is generated.

```
1  Function AdaptiveLocalSearch:
2      λ^best ← FindInitialOrder();
3      s^best ← SchedulingOperator(λ^best);
4      while time-limit not reached do
5          nhgen ← SelectNeighborhoodGenerator() ;
6          foreach (λ^nh, s^nh) ∈ GenerateNeighborhood(nhgen, λ^best, s^best) do
7              /* Diversification of the search:  allow solutions with the
                  same makespan to replace the incumbent.  */
8              if Makespan(s^nh) ≤ Makespan(s^best) then
9                  λ^best ← λ^nh;
10                 s^best ← s^nh;
11     return s^best;
```

Algorithm 1: The general local search algorithm with an adaptive selection of the neighborhood generator.

3.2.1 Adaptive Generation of the Solution Neighborhood

To diversify the search, the following six neighborhood generators were implemented:

- NHGEN-SWAP: swapping the positions of two random jobs;
- NHGEN-INS: inserting a random job into another random position;
- NHGEN-MKSP-SWAP: taking a random job completing at the makespan and swapping its position with a random job;
- NHGEN-MKSP-INS: taking a random job completing at the makespan and inserting it into another random position;
- NHGEN-BLOCK-SWAP: selecting a random block of consecutive jobs in λ and swapping it with another randomly selected block of the same length;
- NHGEN-BLOCK-INS: selecting a random block of consecutive jobs in λ and inserting it into another random position.

The performance of each generator may vary for different instance classes and regions of the solution space. To avoid using unpromising generators during the search, the generator with the highest chance of finding a good solution is selected in each iteration.

For each generator, two values are recorded: (i) *hits*, i.e., how many times the generator produced a solution with better makespan than the current best solution and (ii) *fails*, i.e., how many times the generator failed to produce a better solution. The generator that has the smallest value of $fails - hits$ is selected in each iteration.

Chapter

4 Scheduling with Energy Consumption Limits in Deterministic Environments

Enforcing robustness on the production schedules w.r.t. the energy consumption limits may lead to undesirable deterioration of the objective value. Moreover, in environments where the responsible workers are highly penalized for causing the delays, the schedules are implicitly robust. For such cases, it makes sense to study the scheduling problem introduced in Chapter 2 for $\delta^{\max} = 0$, i.e., when the jobs' start times do not deviate. Therefore, Chapter 4 explores different variants of the scheduling problem with the energy consumption limits without taking the robustness into account.

For the purpose of this chapter, the notation laid out in Chapter 2 is simplified. Since $\delta^{\max} = 0$, there is only one realized schedule, and that is the baseline schedule. Therefore, notation $\mathbf{s}$ will be used instead of the function RS. Moreover, we will simply say "start times" instead of "baseline start times".

4.1 Problem $1|p_{1,j} = 1, E_i^{\max} = E^{\max}|\emptyset$

The fundamental reason why the problem PD$m|E_i^{\max}|\emptyset$ is hard to solve is the underlying problem $1|p_{1,j} = 1, E_i^{\max} = E^{\max}|\emptyset$, which is $\mathcal{NP}$-complete. This also implies that restricting the multi-machine problem to a single machine one does not yield a polynomial algorithm (unless $\mathcal{NP}$ equals $\mathcal{P}$).

Lemma 1. *Problem* $1|E_i^{\max} = E^{\max}|\emptyset$ *is in* $\mathcal{NP}$.

Proof. This can be proven by showing that the start times' feasibility is verifiable in polynomial time w.r.t. the length of the input instance. This is easy to see since checking that the jobs are not overlapping can be done in $\mathcal{O}(n^2)$ time, and checking the energy consumption constraints can be performed in $\mathcal{O}(n \cdot |\mathcal{I}|)$ time. □

Theorem 1. *Problem* $1|p_{1,j} = 1, E_i^{\max} = E^{\max}|\emptyset$ *is* $\mathcal{NP}$*-complete.*

Proof. Due to Lemma 1, the problem $1|p_{1,j} = 1, E_i^{\max} = E^{\max}|\emptyset$ is in $\mathcal{NP}$. It remains to show that 3-PARTITION, which is $\mathcal{NP}$-complete problem in the strong sense, reduces to $1|p_{1,j} = 1, E_i^{\max} = E^{\max}|\emptyset$.

Problem:	3-PARTITION
Input:	Bound $B \in \mathbb{Z}_{>0}$ and finite set A of $3 \cdot q$ positive integers such that $\frac{B}{4} < a_j < \frac{B}{2}$ for every $a_j \in A$ and $\sum_{a_j \in A} a_j = q \cdot B$.
Question:	Can A be partitioned into disjoint sets $A_1, \ldots, A_q$, each having 3 integers and $\sum_{a_j \in A_i} a_j = B$ for all i?

Given an instance for 3-PARTITION, construct an instance for $1|p_{1,j}=1,E_i^{\max}=E^{\max}|\emptyset$ as follows. Set $\mathcal{M}=\{M_1\}$, $\mathcal{J}=\mathcal{J}_1^M=\{J_{1,1},J_{1,2},\ldots,J_{1,|A|}\}$, $\mathcal{I}=\{I_1,I_2,\ldots,I_q\}$, $D=3$, $H=3\cdot q$, $E^{\max}=B$, and define each job $J_{1,j}\in\mathcal{J}$ as $p_{1,j}=1,\ P_{1,j}=a_j$.

The proof by reduction requires that given a feasible solution of 3-PARTITION, it must be possible to construct a feasible solution of $1|p_{1,j}=1,E_i^{\max}=E^{\max}|\emptyset$ problem in polynomial time (and vice versa)

1. 3-PARTITION $\implies$ $1|p_{1,j}=1,E_i^{\max}=E^{\max}|\emptyset$: From the feasible solution of 3-PARTITION, construct a schedule in which the jobs corresponding to the integers of set A_i are scheduled within interval I_i (the ordering of the jobs in the interval is arbitrary). The jobs do not overlap in any interval since there are exactly three integers in each set A_i, and the length of each metering interval equals 3. Finally, for every metering interval I_i holds that $E^{\max}=B=\sum_{a_j\in A_i}a_j=\sum_{a_j\in A_i}P_{1,j}$, i.e., the energy consumption limit is not violated in any metering interval. Therefore, the schedule is feasible.

2. $1|p_{1,j}=1,E_i^{\max}=E^{\max}|\emptyset$ $\implies$ 3-PARTITION: Construct a partition where the integers corresponding to the jobs scheduled in interval I_i belong to set A_i. Due to $D=3$ and $H=3\cdot q=n$, any feasible schedule has exactly three jobs scheduled in every metering interval; thus, every set A_i has exactly three integers. Since the total energy consumption of all the jobs equals $q\cdot E^{\max}$, the total energy consumption in every metering interval of a feasible schedule must be exactly $E^{\max}$. Therefore, the sum of integers in any set A_i equals $E^{\max}=B$.

□

4.2 Problem $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$

This section's result is that the problem $1|p_{1,j}=1,E_i^{\max}=E^{\max}|\emptyset$ remains $\mathcal{NP}$-complete even for $|\mathcal{I}|=2$. Unfortunately, this shows that if the problem $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$ is tackled using a *rolling-horizon* based method [55], where the horizon window is limited to only two metering intervals, it is unlikely that there exists a polynomial algorithm solving one iteration exactly (unless $\mathcal{NP}=\mathcal{P}$).

Theorem 2. *Problem* $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$ *is* $\mathcal{NP}$*-complete.*

Proof. Due to Lemma 1, the problem $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$ is in $\mathcal{NP}$.

It remains to show that PARTITION, which is $\mathcal{NP}$-complete problem, reduces to $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$.

Problem: PARTITION
Input: Finite set A of positive integers a_j.
Question: Is there subset $A'\subseteq A$ such that $\sum_{a_j\in A'}a_j=\sum_{a_j\in A\setminus A'}a_j=\frac{\sum_{a_j\in A}a_j}{2}$?

Given an instance for PARTITION, construct an instance for $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$ as follows. Set $\mathcal{M}=\{M_1\}$, $\mathcal{J}=\mathcal{J}_1^M=\{J_{1,1},J_{1,2},\ldots,J_{1,|A|}\}$, $\mathcal{I}=\{I_1,I_2\}$, $D=|A|$, $H=2\cdot|A|$, $E^{\max}=\frac{\sum_{a_j\in A}a_j}{2}$, and define each job $J_{1,j}\in\mathcal{J}$ as $p_{1,j}=1,\ P_{1,j}=a_j$.

The proof by reduction requires that given a feasible solution of PARTITION, it must be possible to construct a feasible solution of $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$ problem in polynomial time (and vice versa).

1. PARTITION $\implies$ $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$: From the feasible solution of PARTITION, construct a schedule in which the jobs corresponding to the integers from set A' are scheduled within interval I_1. The rest of the jobs are scheduled within interval I_2 (ordering of the jobs in both intervals is not important). The jobs do not overlap in the metering intervals since $D=|A|=n$. Moreover, the energy consumption limit is not violated in such a schedule since $E^{\max}=\frac{\sum_{a_j\in A}a_j}{2}$. Therefore, the constructed schedule is feasible.

2. $1|p_{1,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$ $\implies$ PARTITION: Construct a partition where all the integers corresponding to the jobs scheduled in interval I_1 belong to set A'. Given a feasible schedule, the total energy consumption in each metering interval I_1,I_2 must be $\frac{\sum_{a_j\in A}a_j}{2}$. This is easy to see due to $2\cdot E^{\max}=\sum_{J_{1,j}\in\mathcal{J}}P_{1,j}$: if the total energy consumption in one metering interval would be less than $E^{\max}$, then the total energy consumption in another metering interval would have to be greater than $E^{\max}$. Therefore, the sum of the integers A' equals $E^{\max}$, and the constructed partition is feasible.

□

4.3 Problem $\mathrm{PD}m|\pi_k,p_{k,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$

As already mentioned, tackling a scheduling problem using global optimization methods, such as genetic algorithms, may involve solving a constrained sub-problem of the original problem. For example, if a fixed ordering of the jobs represents an individual in the population, the actual start times are obtained by solving a sub-problem in which the order of the jobs is fixed. If the sub-problem can be solved exactly in polynomial time, the genetic algorithm may converge to a very good solution in a short time.

Unfortunately, this section shows that approaching the problem $\mathrm{PD}m|E_i^{\max}=E^{\max}|C_{\max}$ in such a manner requires a heuristic algorithm for the sub-problem constraining the ordering of the jobs since it is unlikely that it can be solved optimally in a reasonable time.

Theorem 3. *Problem* $\mathrm{PD}m|\pi_k,p_{k,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$ *is* $\mathcal{NP}$*-complete.*

Proof. The complexity proof is analogous to problem $1|p_{k,j}=1,|\mathcal{I}|=2,E_i^{\max}=E^{\max}|\emptyset$, which used PARTITION problem for the reduction. The only difference is that each machine is dedicated to only one job

in problem PD$m|\pi_k, p_{k,j} = 1, |\mathcal{I}| = 2, E_i^{\max} = E^{\max}|\emptyset$, i.e., the fixed order on each machine consists of a single job and $n = m$. Therefore, each job $J_{k,1}$ on its machine M_k corresponds to integer a_k. □

4.4 Problem PD$m|\pi_k$, no-crossing, $E_i^{\max} = E^{\max}|C_{\max}$

This section studies the scheduling problem with the no-crossing constraint requiring that each job must have non-zero overlap with exactly one metering interval. The no-crossing requirement is widely considered in the domain of scheduling with *periodic maintenance activities* [32, 54], where the machines must be periodically repaired or inspected. During this time, the machines cannot process any production jobs. Such requirement has applications in the assembly of wiring harness for the aerospace industry, where each task is complex and must be performed within one working shift by the same team. By extending the periodic intervals with a maximum capacity constraint, where the capacity represents available material (i.e., the energy consumption limit), additional applications such as metal casting can be modeled.

For the problem PD$m|\pi_k$, no-crossing, $E_i^{\max} = E^{\max}|C_{\max}$, a shortest paths based algorithm is provided with complexity $\mathcal{O}(n^{2 \cdot m})$, that is, the algorithm becomes polynomial if the number of machines is fixed. Therefore, problem PD$m|\pi_k$, no-crossing, $E_i^{\max} = E^{\max}|C_{\max}$ is in XP complexity class [17] for parameter m.

Let us start by defining a set of *slices*, each representing a set of jobs that can be scheduled within the same metering interval without violating the energy consumption limit.

Definition 1 (Slices)**.** *Let $\boldsymbol{l}, \boldsymbol{r}$ be two integer vectors such that $\boldsymbol{r} \in [0\,..\,n_1] \times \cdots \times [0\,..\,n_m]$, and $\mathbf{1} \leq \boldsymbol{l} \leq \boldsymbol{r} + \mathbf{1}$. Define $\boldsymbol{l}, \boldsymbol{r}$ to be compatible iff*

$$\sum_{j \in [l_k\,..\,r_k]} p_{k,j} \leq D\,, \quad \forall M_k \in \mathcal{M} \tag{4.1}$$

$$\sum_{M_k \in \mathcal{M}} \sum_{j \in [l_k\,..\,r_k]} p_{k,j} \cdot P_{k,j} \leq E^{\max}\,. \tag{4.2}$$

A pair of compatible vectors $(\boldsymbol{l}, \boldsymbol{r})$ is called a slice, and the set of all slices is denoted as Ψ.

Notice that the definition of a slice allows some machine M_k to have an empty set of jobs, which is the case for $l_k = r_k + 1$.

Any feasible schedule can be understood as a sequence of slices, where each slice represents the jobs scheduled in a particular metering interval. An example of a schedule with the slices is shown in Fig. 4.1. The example reveals an important pattern $(\boldsymbol{l}, \boldsymbol{r}), (\boldsymbol{r} + \mathbf{1}, \boldsymbol{r'})$ between the slices of two consecutive metering intervals. Informally, if some metering interval I_i is labeled by slice $(\boldsymbol{l}, \boldsymbol{r})$, all the jobs with indices smaller or equal to $\boldsymbol{r}$ were scheduled till τ_i^{end}; on the other hand, all the jobs with indices greater or equal to $\boldsymbol{l}$ were not scheduled before τ_i^{start}.

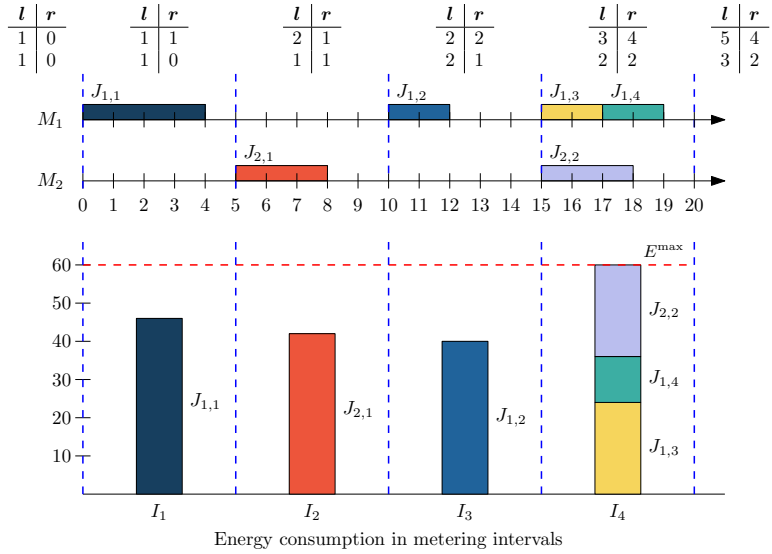

Figure 4.1: The slices of a two-machine feasible schedule for the example instance from the problem statement (see Section 2.1).

Notice that all the slices but the last one contribute equally to the makespan by D. Therefore, the makespan of a feasible schedule is defined by the number of slices minus one plus the maximum sum of the processing times over all machines in the last slice. This observation is based on an easy fact that the makespan of a feasible schedule is not increased if the jobs in the last slice are shifted to the left as much as possible. That is, the jobs in the last slice can be processed without any idle time between two consecutive jobs.

Lastly, some of the slices can be ignored in further computation since they cannot occur in optimal solutions. These ignored slices represent metering intervals in which no jobs are scheduled. By removing such slices from a feasible schedule, the makespan does not increase. The set of slices, in which at least one job is scheduled in a metering interval, is $\{(\boldsymbol{l},\boldsymbol{r}) \in \Psi \mid \exists k : l_k \leq r_k\}$.

+By linking the ideas mentioned above, the problem $\mathrm{PD}m|\pi_k, \text{no-crossing}, E_i^{\max} = E^{\max}|C_{\max}$ can be solved by finding the shortest path in a graph, where the vertices represent the slices and the edges are defined by the pattern $(\boldsymbol{l},\boldsymbol{r}),(\boldsymbol{r}+\mathbf{1},\boldsymbol{r}')$ connecting two consecutive slices. Formally, let us construct the following directed graph $G = (V, E, w)$, where V is the set of *vertices*, E is the set of *edges*, and w is the *weight* function of the edges

$$V = \{v_{\boldsymbol{l},\boldsymbol{r}} \mid (\boldsymbol{l},\boldsymbol{r}) \in \Psi; \exists k : l_k \leq r_k\} \cup \{v^{\mathrm{s}}, v^{\mathrm{t}}\} \tag{4.3}$$

$$E = \{(v_{\boldsymbol{l},\boldsymbol{r}}, v_{\boldsymbol{r}+\mathbf{1},\boldsymbol{r}'}) \in V \times V\}\,. \tag{4.4}$$

Vertices $\{v^{\mathrm{s}}, v^{\mathrm{t}}\}$ have special meaning. Vertex v^{s} is the *source* vertex $v^{\mathrm{s}} = v_{\mathbf{1},\mathbf{0}}$, representing the beginning of the schedule, i.e., no jobs are scheduled yet. Vertex v^{t} is the *target* vertex $v^{\mathrm{t}} = v_{\boldsymbol{l},\boldsymbol{r}}$, where $\boldsymbol{l} = (n_1 + 1, n_2 + 1, \ldots, n_m + 1)$ and $\boldsymbol{r} = (n_1, n_2, \ldots, n_m)$. The target vertex represents the end of the schedule, i.e., all the jobs were scheduled.

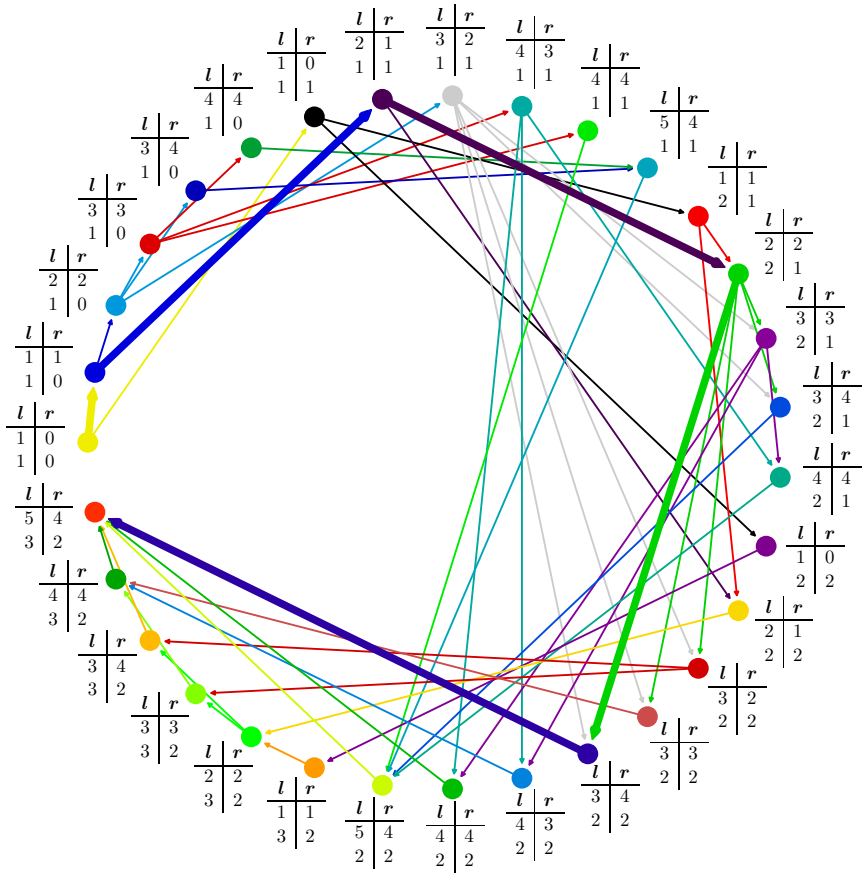

Figure 4.2: The constructed graph for the example instance from the problem statement (see Section 2.1); the weights of the edges are omitted. The highlighted path corresponds to the schedule from Fig. 4.1.

To model the makespan of a feasible schedule, the weight of edge $(v_{\boldsymbol{l},\boldsymbol{r}}, v_{\boldsymbol{l}',\boldsymbol{r}'})$ is defined as

$$w(v_{\boldsymbol{l},\boldsymbol{r}}, v_{\boldsymbol{l}',\boldsymbol{r}'}) = \begin{cases} 0 & v_{\boldsymbol{l},\boldsymbol{r}} = v^{\mathrm{s}} \\ \max\limits_{M_k \in \mathcal{M}} \sum\limits_{j \in [l_k \,..\, r_k]} p_{k,j} & v_{\boldsymbol{l}',\boldsymbol{r}'} = v^{\mathrm{t}} \\ D & \text{otherwise} \end{cases} \tag{4.5}$$

The shortest path from vertex v^{s} to vertex v^{t} represents a feasible schedule with the minimum makespan (or, if the path does not exist, the corresponding instance is infeasible). An example of the constructed graph for the example instance from the problem statement (see Section 2.1) is shown in Fig. 4.2.

It remains to show that the algorithm is polynomial for a fixed number of machines.

Theorem 4. *Problem* $\mathrm{PD}m|\pi_k, \text{no-crossing}, E_i^{\max} = E^{\max}|C_{\max}$ *is in XP complexity class for parameter* m.

Proof. Consider the above-described graph $G = (V, E, w)$. Since the shortest path algorithm in the graph is polynomial in the number of vertices, it suffices to show that the number of vertices (or equivalently, slices) is polynomial for a fixed number of machines.

Consider machine M_k and $r_k \in [0\,..\,n_k]$. There are at most r_k corresponding values of l_k. Therefore, the total number of possible slices on machine M_k is

$$\sum_{r_k \in [0\,..\,n_k]} r_k = \frac{n_k \cdot (n_k + 1)}{2}\,. \tag{4.6}$$

Thus, the upper bound on the number of slices is

$$\prod_{M_k \in \mathcal{M}} \frac{n_k \cdot (n_k + 1)}{2}\,, \tag{4.7}$$

which is polynomial for a fixed number of machines. □

4.5 Problem $\mathrm{PD}m|E_i^{\max}|C_{\max}$

The last problem $\mathrm{PD}m|E_i^{\max}|C_{\max}$ investigated in this chapter is the most general one within the deterministic setting. However, it is also the most practical one since it is the closest to the motivational production processes (steel hardening and tempered glass production). This scheduling problem is $\mathcal{NP}$-hard since it is a generalization of $1|p_{1,j} = 1, E_i^{\max} = E^{\max}|C_{\max}$ that was studied in the earlier sections.

To solve the problem optimally, we propose one CP model and various MILP models. Moreover, to solve large real-world instances, we also design a simple yet effective local search heuristic algorithm. All the methods are experimentally evaluated in Section 4.5.4.

4.5.1 CP Model

The proposed CP model has one interval variable $x_{k,j}$ for each job $J_{k,j} \in \mathcal{J}$ that denotes the time interval in which the job is scheduled. The main idea behind the model is to use CP function Overlap, which computes the length of the overlap between the job's interval variable and a metering interval that is given by its start and end. The overlaps in a metering interval are multiplied by the corresponding job's power consumption, added together and constrained to be within the energy consumption limits.

Note that the *cumulative functions* cannot be used to model the energy consumption since the *height* of an elementary cumulative function is a constant. In contrast, our scheduling problem would require the height to be a function of the overlap length with a metering interval.

The complete model follows

$$\min \max_{J_{k,j} \in \mathcal{J}} \text{EndOf}(x_{k,j}) \tag{4.8}$$

$$\text{EndOf}(x_{k,j}) \leq H, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.9}$$

$$\text{LengthOf}(x_{k,j}) = p_{k,j}, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.10}$$

$$\text{NoOverlap}(\{x_{k,j} \mid J_{k,j} \in \mathcal{J}_k^M\}), \quad \forall M_k \in \mathcal{M} \tag{4.11}$$

$$\sum_{J_{k,j} \in \mathcal{J}} \text{Overlap}(x_{k,j}, \tau_i^{\text{start}}, \tau_i^{\text{end}}) \cdot P_{k,j} \leq E_i^{\max}, \quad \forall I_i \in \mathcal{I} \tag{4.12}$$

Objective (4.8) minimizes the maximum completion time of all the jobs. Constraint (4.9) guarantees that the completion time of every job is bounded by the scheduling horizon H. The following constraint (4.10) sets the length of an interval variable $x_{k,j}$ corresponding to some job $J_{k,j}$ to be $p_{k,j}$. Global constraint NoOverlap in (4.11) ensures that the jobs on each dedicated machine are not overlapping. Finally, the energy limits in every metering interval are enforced by constraint (4.12).

4.5.2 MILP Models

The modeling of the jobs' overlap with the metering intervals is crucial for problem PD$m|E_i^{\max}|C_{\max}$. This section formulates the problem as a disjunctive and time-indexed model. Moreover, the section investigates an alternative approach called *an implicit model* developed in the field of the energy-aware scheduling with RTP.

4.5.2.1 Disjunctive MILP Model

The disjunctive MILP model uses the formulation for computing the jobs' overlaps with the metering intervals presented in [26, 27]. We adapted the model proposed in [26, 27] to consider the dedicated machines environment, energy consumption limits, and makespan as an objective.

The disjunctive MILP model uses the following variables: (i) $C_{\max} \in \mathbb{Z}_{\geq 0}$ is the makespan of the schedule, (ii) $s_{k,j} \in \mathbb{Z}_{\geq 0}$ represents the start time of job $J_{k,j}$, (iii) $x_{k,j,j'} \in \{0,1\}$ denotes whether job $J_{k,j}$ is scheduled before job $J_{k,j'}$, (iv) $d_{k,j,i} \in \mathbb{Z}_{\geq 0}$ expresses the length of the overlap between job $J_{k,j}$ and metering interval I_i, (v) $z_{k,j,i}^{\text{start}} \in \{0,1\}$ denotes whether $s_{k,j} \in [0, \tau_i^{\text{end}} - 1]$, and (vi) $z_{k,j,i}^{\text{end}} \in \{0,1\}$ denotes whether $s_{k,j} + p_{k,j} \in [0, \tau_i^{\text{start}}]$. The complete model follows

$$\min C_{\max} \tag{4.13}$$

$$s_{k,j} + p_{k,j} \leq C_{\max}, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.14}$$

$$x_{k,j,j'} = 1 \implies s_{k,j} + p_{k,j} \leq s_{k,j'}, \quad \forall M_k \in \mathcal{M}; \forall J_{k,j}, J_{k,j'} \in \mathcal{J}_k^M : j < j' \tag{4.15}$$

$$x_{k,j,j'} = 0 \implies s_{k,j'} + p_{k,j'} \leq s_{k,j}, \quad \forall M_k \in \mathcal{M}; \forall J_{k,j}, J_{k,j'} \in \mathcal{J}_k^M : j < j' \tag{4.16}$$

$$s_{k,j} \geq \tau_i^{\text{end}} \cdot (1 - z_{k,j,i}^{\text{start}}), \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \setminus \{I_{|\mathcal{I}|}\} \tag{4.17}$$

$$z_{k,j,i}^{\text{start}} = 1 \implies s_{k,j} \leq \tau_i^{\text{end}} - 1, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \setminus \{I_{|\mathcal{I}|}\} \tag{4.18}$$

$$z_{k,j,i+1}^{\text{start}} \geq z_{k,j,i}^{\text{start}}, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \setminus \{I_{|\mathcal{I}|}\} \tag{4.19}$$

$$z^{\text{start}}_{k,j,|\mathcal{I}|} = 1, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.20}$$

$$s_{k,j} + p_{k,j} \geq \tau^{\text{start}}_i \cdot (1 - z^{\text{end}}_{k,j,i}) + 1, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \tag{4.21}$$

$$z^{\text{end}}_{k,j,i} = 1 \implies s_{k,j} + p_{k,j} \leq \tau^{\text{start}}_i, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \tag{4.22}$$

$$z^{\text{end}}_{k,j,i+1} \geq z^{\text{end}}_{k,j,i}, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \setminus \{I_{|\mathcal{I}|}\} \tag{4.23}$$

$$d_{k,j,i} \leq D \cdot (z^{\text{start}}_{k,j,i} - z^{\text{end}}_{k,j,i}), \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \tag{4.24}$$

$$\sum_{I_i \in \mathcal{I}} d_{k,j,i} = p_{k,j}, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.25}$$

$$d_{k,j,i} \geq D \cdot (z^{\text{start}}_{k,j,i-1} - z^{\text{end}}_{k,j,i+1}), \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \setminus \{I_1, I_{|\mathcal{I}|}\} \tag{4.26}$$

$$d_{k,j,i} \geq \tau^{\text{end}}_i \cdot (1 - z^{\text{start}}_{k,j,i-1}) - s_{k,j} - D \cdot z^{\text{end}}_{k,j,i+1}, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \setminus \{I_1, I_{|\mathcal{I}|}\} \tag{4.27}$$

$$z^{\text{end}}_{k,j,i+1} = 1 \implies d_{k,j,i} \geq s_{k,j} + p_{k,j} - \tau^{\text{end}}_i + D \cdot z^{\text{start}}_{k,j,i-1} \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \setminus \{I_1, I_{|\mathcal{I}|}\} \tag{4.28}$$

$$d_{k,j,1} \geq D \cdot z^{\text{start}}_{k,j,1} - s_{k,j} - D \cdot z^{\text{end}}_{k,j,2}, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.29}$$

$$d_{k,j,|\mathcal{I}|} \geq s_{k,j} + p_{k,j} - H + D \cdot z^{\text{start}}_{k,j,|\mathcal{I}|-1}, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.30}$$

$$\sum_{J_{k,j} \in \mathcal{J}} d_{k,j,i} \cdot P_{k,j} \leq E^{\max}_i, \quad \forall I_i \in \mathcal{I} \tag{4.31}$$

Objective (4.13) minimizes the makespan, which is the maximum of the jobs' completion times, see constraint (4.14). Ensuring that the jobs are not overlapping is modeled by constraints (4.15) and (4.16). Constraints (4.17)-(4.30) models the overlaps of the jobs with the metering intervals; see [26] for details. The last constraint (4.31) enforces the energy limit.

4.5.2.2 Time-indexed MILP Model

The time-indexed model uses two variables: (i) $C_{\max} \in \mathbb{Z}_{\geq 0}$ representing the makespan of the solution and (ii) $x_{k,j,t} \in \{0,1\}$ denoting whether job $J_{k,j}$ starts at time t. The model uses the following *allocation* expression to model the non-overlap constraint and the energy consumption in the metering intervals

$$\sum_{t'=\max\{0,t-p_{k,j}+1\}}^{t} x_{k,j,t'} \,. \tag{4.32}$$

The allocation expression denotes whether job $J_{k,j}$ overlaps time unit t in the solution. The time-indexed model for problem PD$m|E^{\max}_i|C_{\max}$ can be formulated as follows

$$\min C_{\max} \tag{4.33}$$

$$\sum_{t \in [0\,..\,H-1]} t \cdot x_{k,j,t} + p_{k,j} \leq C_{\max}, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.34}$$

$$\sum_{t \in [0\,..\,H-1]} x_{k,j,t} = 1, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.35}$$

$$\sum_{t\in[0\,..\,H-1]} t\cdot x_{k,j,t} + p_{k,j} \leq H, \quad \forall J_{k,j}\in\mathcal{J} \tag{4.36}$$

$$\sum_{J_{k,j}\in\mathcal{J}_k^M} \sum_{t'=\max\{0,t-p_{k,j}+1\}}^{t} x_{k,j,t'} \leq 1, \quad \forall M_k\in\mathcal{M}; \forall t\in[0\,..\,H-1] \tag{4.37}$$

$$\sum_{t\in[\tau_i^{\text{start}}\,..\,\tau_i^{\text{end}}-1]} \sum_{J_{k,j}\in\mathcal{J}} \sum_{t'=\max\{0,t-p_{k,j}+1\}}^{t} P_{k,j}\cdot x_{k,j,t'} \leq E_i^{\max}, \quad \forall I_i\in\mathcal{I} \tag{4.38}$$

Objective (4.33) minimizes the makespan, which is the maximum completion time of all the jobs (4.34). Constraint (4.35) ensures that each job is started, and constraint (4.36) guarantees that the completion time of every job is bounded by the scheduling horizon H. The non-overlap constraint is satisfied by requiring that each time unit is overlapped by at most job on each machine; see (4.37). Finally, the energy consumption limits in every metering interval are modeled using constraint (4.12).

4.5.2.3 Implicit MILP Model

In this section, a MILP formulation called *implicit model* is presented. The implicit model's main property is that it does not model the start times of the jobs explicitly. Such an approach is suitable for the scheduling problems where the actual sequencing of the jobs within each metering interval is not important. The implicit model was presented in [70] for a scheduling problem with energy cost minimization and RTP. We adapted the implicit model from [70] to problem PD$m|E_i^{\max}|C_{\max}$ to consider the makespan minimization. Moreover, we improved the model by observing that *boundary constraints* are only necessary for a subset of jobs, thus reducing the implicit model's size.

For modeling purposes, the maximum number of consecutive metering intervals in which job $J_{k,j}$ can have a non-zero overlap for any feasible start times $\mathbf{s}$ has to be determined; this number will be denoted as $\Omega_{k,j}$.

Lemma 2. *The maximum number of consecutive metering intervals that can have non-zero overlap with job $J_{k,j}$ for any feasible start times $\mathbf{s}$ is*

$$\Omega_{k,j} = \left\lceil \frac{p_{k,j}}{D} \right\rceil + 1\,. \tag{4.39}$$

Proof. Let I_{i_1} be the metering interval, where the job starts and let I_{i_2} be the metering interval, where the job completes; thus, $\text{Overlap}(s_{k,j}, p_{k,j}, i_1) \in \mathbb{Z}_{>0}, \text{Overlap}(s_{k,j}, p_{k,j}, i_2) \in \mathbb{Z}_{>0}$. Consider the following two cases

1. $I_{i_1} = I_{i_2}$: This implies that the job has non-zero overlap with only one metering interval in $\mathbf{s}$. Since $\Omega_{k,j} \geq 1$, the lemma holds.

2. $I_{i_1} < I_{i_2}$: Since the job is processed without preemption, the job completely fills up each metering interval between I_{i_1}, I_{i_2}, i.e., the overlap length of the job with each metering interval between I_{i_1}, I_{i_2} is D. The number of such

metering intervals is $q = i_2 - i_1 - 1$. The processing time can then be written as

$$p_{k,j} = q \cdot D + \text{Overlap}(s_{k,j}, p_{k,j}, i_1) + \text{Overlap}(s_{k,j}, p_{k,j}, i_2)\,. \tag{4.40}$$

To finish the proof it has to be shown that

$$\Omega_{k,j} \geq q + 2\,, \tag{4.41}$$

i.e., the number of metering intervals with a non-zero overlap in $\mathbf{s}$ is at most $\Omega_{k,j}$.

Notice that from the definition of the ceiling function, it holds that

$$\Omega_{k,j} = \left\lceil \frac{p_{k,j}}{D} \right\rceil + 1 = \underbrace{\frac{p_{k,j}}{D} + \epsilon}_{\in \mathbb{Z}_{>0}} + 1\,, \tag{4.42}$$

where $\epsilon \in [0, 1)$. Therefore,

$$\begin{aligned}
\Omega_{k,j} &= \frac{p_{k,j}}{D} + \epsilon + 1 \\
&= q + \underbrace{\frac{\text{Overlap}(s_{k,j}, p_{k,j}, i_1)}{D} + \frac{\text{Overlap}(s_{k,j}, p_{k,j}, i_2)}{D} + \epsilon}_{\in \mathbb{Z}_{>0}} + 1 \\
&\geq q + 2\,.
\end{aligned}$$

The last inequality follows from $q \in \mathbb{Z}_{\geq 0},$, $\Omega_{k,j} \in \mathbb{Z}_{>0}$, $\text{Overlap}(s_{k,j}, p_{k,j}, i_1) \in \mathbb{Z}_{>0}$, $\text{Overlap}(s_{k,j}, p_{k,j}, i_2) \in \mathbb{Z}_{>0}$.

□

Similarly to the CP model, the jobs' overlap with the metering intervals needs to be formulated. However, MILP does not provide any constructs such as Overlap; thus, it has to be modeled using linear expressions. We observed that for "shorter" jobs, the necessary constraints could be modeled in more straightforward way than for "longer" jobs; thus, the set of the jobs is split into *non-spannable* and *spannable* jobs, i.e., the set of spannable jobs is

$$\mathcal{J}^{\text{span}} = \{J_{k,j} \in \mathcal{J} \mid p_{k,j} > D\}\,, \tag{4.43}$$

while the set of non-spannable jobs is $\mathcal{J} \setminus \mathcal{J}^{\text{span}}$.

The implicit MILP model uses the following variables: (i) $d_{k,j,i} \in \mathbb{Z}_{\geq 0}$ denoting the length of the overlap between job $J_{k,j}$ and metering interval I_i, (ii) $x^s_{k,j,i} \in \{0, 1\}$ indicating whether spannable job $J_{k,j}$ starts in metering interval I_i, (iii) $x^+_{k,j,i} \in \{0, 1\}$ expressing whether job $J_{k,j}$ has non-zero overlap in metering interval I_i, (iv) $y^+_{k,i} \in \{0, 1\}$ denoting whether some job has non-zero overlap with metering interval I_i on machine M_k, and (v) $C_{\max} \in \mathbb{Z}_{\geq 0}$ representing the objective

value. The implicit model is then formulated as

$$\min \quad C_{\max} \tag{4.44}$$

$$\sum_{J_{k,j} \in \mathcal{J}_k^M} d_{k,j,i} \leq D \cdot y_{k,i}^+, \quad \forall M_k \in \mathcal{M}; \forall I_i \in \mathcal{I} \tag{4.45}$$

$$y_{k,i}^+ = 1 \implies \tau_i^{\text{start}} + \sum_{J_{k,j} \in \mathcal{J}_k^M} d_{k,j,i} \leq C_{\max}, \quad \forall M_k \in \mathcal{M}; \forall I_i \in \mathcal{I} \tag{4.46}$$

$$\sum_{I_i \in \mathcal{I}} d_{k,j,i} = p_{k,j}, \quad \forall J_{k,j} \in \mathcal{J} \tag{4.47}$$

$$\sum_{J_{k,j} \in \mathcal{J}_k^M} d_{k,j,i} \leq D, \quad \forall I_i \in \mathcal{I}; \forall M_k \in \mathcal{M} \tag{4.48}$$

$$\sum_{J_{k,j} \in \mathcal{J}} d_{k,j,i} \cdot P_{k,j} \leq E_i^{\max}, \quad \forall I_i \in \mathcal{I} \tag{4.49}$$

$$\begin{aligned} &x_{k,j_1,i}^+ + x_{k,j_1,i+1}^+ + x_{k,j_2,i}^+ + x_{k,j_2,i+1}^+ \leq 3, \\ &\qquad \forall M_k \in \mathcal{M}; \forall I_i \in \mathcal{I} \setminus \{I_{|\mathcal{I}|}\}; \\ &\qquad \forall J_{k,j_1}, J_{k,j_2} \in \mathcal{J}_k^M : J_{k,j_1} \neq J_{k,j_2}, p_{k,j_1} \geq 2, p_{k,j_2} \geq 2; \\ &\qquad p_{k,j_1} + p_{k,j_2} \leq 2 \cdot D \end{aligned} \tag{4.50}$$

$$d_{k,j,i} \leq \min(D, p_{k,j}) \cdot x_{k,j,i}^+, \quad \forall J_{k,j} \in \mathcal{J}; \forall I_i \in \mathcal{I} \tag{4.51}$$

$$\text{SOS}_2(d_{k,j,1}, \ldots, d_{k,j,|\mathcal{I}|}), \quad \forall J_{k,j} \in \mathcal{J} \setminus \mathcal{J}^{\text{span}} \tag{4.52}$$

$$\sum_{I_i \in \mathcal{I}} x_{k,j,i}^s = 1, \quad \forall J_{k,j} \in \mathcal{J}^{\text{span}} \tag{4.53}$$

$$d_{k,j,i} \leq D \cdot \sum_{i'=\max(1,i-\Omega_{k,j}+1)}^{i} x_{k,j,i'}^s, \quad \forall J_{k,j} \in \mathcal{J}^{\text{span}}; \forall I_i \in \mathcal{I} \tag{4.54}$$

$$D \cdot (x_{k,j,i-1}^+ + x_{k,j,i+1}^+ - 1) \leq d_{k,j,i}, \quad \forall J_{k,j} \in \mathcal{J}^{\text{span}}; \forall I_i \in \mathcal{I} \setminus \{I_1, I_{|\mathcal{I}|}\} \tag{4.55}$$

In the following text, the model is explained in more detail.

Objective (4.44) of the implicit model is the minimization of the makespan that is modeled using constraints (4.45)-(4.46). Constraint (4.45) ensures that if any job on machine M_k has non-zero overlap with I_i, the value of variable $y_{k,i}^+$ is 1. This variable is then used as an indicator in the following indicator constraint (4.46), which bounds the makespan from below by the total overlap in the last metering interval. Constraint (4.46) exploits a simple observation that the jobs in the metering interval, where the schedule completes, can be processed without idle time.

The following constraint (4.47) guarantees that each job's total overlap with the metering intervals is exactly its processing time. Bounding the total overlap on every machine and every metering interval to be at most D is enforced in constraint (4.48). Finally, constraint (4.49) models the energy limits.

Up to this point, the introduced constraints do not guarantee non-preemption of the jobs, i.e., the model allows a job to be interrupted by another job. To fix this, three groups of constraints are added to the model: *boundary constraints* (4.50)-(4.51), *continuity constraints* (4.52)-(4.54), and *fill constraints* (4.55).

The boundary constraints (4.50) enforce that at most one job can cross a boundary of two consecutive metering intervals on a machine. The constraint is better illustrated in Fig. 4.3 showing the infeasible case when two jobs J_{k,j_1}, J_{k,j_2} cross the boundary of metering intervals I_i, I_{i+1} on the same machine. We can see that this is the only case when all variables $x^+_{k,j_1,i}, x^+_{k,j_1,i+1}, x^+_{k,j_2,i}, x^+_{k,j_2,i+1}$ equal 1. Therefore, the infeasible case can be cut by enforcing that at least one of those variables equals 0. The semantics of variables $x^+_{k,j,i}$ themselves is modeled in constraint (4.51), which pushes $x^+_{k,j,i}$ to be 1 if job $J_{k,j}$ has non-zero overlap in metering interval I_i.

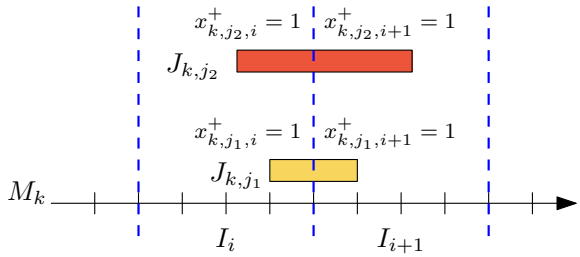

Figure 4.3: Values of $x^+_{k,j,i}$ in the infeasible case when two jobs are crossing the boundary of two same consecutive metering intervals.

We observed that the boundary constraints are not necessary for pairs of jobs, whose sum of processing times are greater than $2 \cdot D$. Using this observation, the number of constraints in the model can be reduced.

Lemma 3. *Let $M_k \in \mathcal{M}$ and let $J_{k,j_1}, J_{k,j_2} \in \mathcal{J}_k^M$ be two jobs such that $j_1 \neq j_2, p_{k,j_1} \geq 2, p_{k,j_2} \geq 2$ and $p_{k,j_1} + p_{k,j_2} \geq 2 \cdot D + 1$. Let $\boldsymbol{d} = (d_{k,j,i})_{J_{k,j} \in \mathcal{J}, I_i \in \mathcal{I}}$ be the values of some feasible solution to the implicit model. Then Eq. (4.50) holds for J_{k,j_1}, J_{k,j_2} in any metering interval $I_i \in \mathcal{I} \setminus \{I_{|\mathcal{I}|}\}$.*

Proof. By contradiction. Assume that $I_i \in \mathcal{I} \setminus \{I_{|\mathcal{I}|}\}$ is a metering interval such that Eq. (4.50) is violated, i.e., $d_{k,j_1,i} \geq 1, d_{k,j_1,i+1} \geq 1, d_{k,j_2,i} \geq 1, d_{k,j_2,i+1} \geq 1$. Since Eq. (4.48) holds, it also holds that

$$\begin{aligned} d_{k,j_1,i} + d_{k,j_2,i} &\leq D \\ d_{k,j_1,i+1} + d_{k,j_2,i+1} &\leq D\,, \end{aligned} \tag{4.56}$$

thus

$$d_{k,j_1,i} + d_{k,j_1,i+1} + d_{k,j_2,i} + d_{k,j_2,i+1} \leq 2 \cdot D\,. \tag{4.57}$$

For $J_{k,j} \in \{J_{k,j_1}, J_{k,j_2}\}$, one of the following two cases is possible

1. $d_{k,j,i} + d_{k,j,i+1} = p_{k,j}$
2. $d_{k,j,i} + d_{k,j,i+1} < p_{k,j}$. This means that the job has non-zero overlap with either I_{i-1} or I_{i+2}. Due to Eq. (4.55), either $d_{k,j,i}$ or $d_{k,j,i+1}$ is pushed to be at least D, i.e., either $d_{k,j,i} \geq D, d_{k,j,i+1} \geq 1$ or $d_{k,j,i} \geq 1, d_{k,j,i+1} \geq D$.

The proof is now split according to combinations of those cases

- Case 1 holds for J_{k,j_1} and Case 1 holds for J_{k,j_2}: By substitution, we obtain
$$p_{k,j_1} + p_{k,j_2} \leq 2 \cdot D\,, \tag{4.58}$$
which is a contradiction with the initial assumption $p_{k,j_1} + p_{k,j_2} \geq 2 \cdot D + 1$.
- Case 2 holds for J_{k,j_1} and Case 2 holds for J_{k,j_2}: By substitution, we obtain
$$D + 1 + D + 1 \leq 2 \cdot D\,, \tag{4.59}$$
which is an contradiction.
- Case 1 holds for J_{k,j_1} and Case 2 holds for J_{k,j_2}: W.l.o.g., assume that $d_{k,j_1,i} \geq D$. Since $d_{k,j_2,i} \geq 1$, it holds that $d_{k,j_1,i} + d_{k,j_2,i} \geq D + 1$, which is a contradiction with $d_{k,j_1,i} + d_{k,j_2,i} \leq D$.
- Case 2 holds for J_{k,j_1} and Case 1 holds for J_{k,j_2}: Proof is analogous to the previous one.

□

The second type of constraints necessary for the non-preemption of the jobs is the continuity constraints. These constraints impose the property from Lemma 2, i.e., a job can have a non-zero overlap with at most $\Omega_{k,j}$ consecutive metering intervals. We can easily model such a requirement for non-spannable jobs (i.e., $\Omega_{k,j} \leq 2$) using SOS_2 constraints; see Eq. (4.52). However, for jobs $\mathcal{J}^{\text{span}}$, the constraints are more involved.

The idea behind the continuity constraints for jobs $\mathcal{J}^{\text{span}}$ is very similar to how the non-preemption is modeled in the time-indexed model using allocation expression Eq. (4.32). First, constraint (4.53) ensures that each job starts in some metering interval. Then, the following constraint (4.54) guarantees that if job $J_{k,j}$ has a non-zero overlap in some metering interval I_i, the job had to start in one of $\{I_{i-\Omega_{k,j}+1}, I_{i-\Omega_{k,j}+2}, \ldots, I_i\}$ metering intervals.

Finally, the last constraints that must be included in the model to guarantee the jobs' non-preemption are the fill constraints (4.55). Informally, fill constraint ensures that if job $J_{k,j}$ has a non-zero overlap with two metering intervals I_{i-1}, I_{i+1}, it must also have a non-zero overlap with interval I_i. Moreover, the job "fills" metering interval I_i, i.e., its overlap length is D.

As was noted above, the model does not include variables that represent the start times explicitly. The rest of this section explains how to construct start times $\mathbf{s}$ from the values of feasible overlap variables $\boldsymbol{d}$. The full pseudocode of the algorithm is listed in Algorithm 2.

The algorithm first assigns the start times to the jobs crossing a boundary of at least two metering intervals. For these jobs, the start time is unambiguous since it equals $\tau_{i_1}^{\text{end}} - d_{k,j,i_1}$, where I_{i_1} is the first metering interval in which job $J_{k,j}$ has non-zero overlap.

Afterward, the start times of the remaining jobs, i.e., having a non-zero overlap in only one metering interval, are assigned. Since the order of such jobs in a particular metering interval is not important, their start times are assigned in an arbitrary order. The jobs are assigned at the earliest start time in the corresponding metering interval on a machine. After each assignment, the earliest start time is increased by the processing time of the assigned job.

Function `ConstructStartTimesFromImplicitMilpModel`($\boldsymbol{d}$):
  `/* The start times. */`
  $\mathbf{s} \leftarrow \mathbf{0}$;
  `/* Earliest start time on each machine and metering interval. */`
  $\mathrm{est} \leftarrow \mathbf{0}$;
  foreach $M_k \in \mathcal{M}$ **do**
    foreach $I_i \in \mathcal{I}$ **do**
      $\mathrm{est}_{k,i} \leftarrow \tau_i^{\mathrm{start}}$;
  `/* The jobs crossing a boundary of metering intervals. */`
  foreach $\left\{ J_{k,j} \in \mathcal{J} \;\middle|\; \left|\{I_i \in \mathcal{I} \mid d_{k,j,i} > 0\}\right| \geq 2 \right\}$ **do**
    $i_1 \leftarrow \min \left\{ i \in [1 \,..\, |\mathcal{I}|] \;\middle|\; d_{k,j,i} > 0 \right\}$;
    $i_2 \leftarrow \max \left\{ i \in [1 \,..\, |\mathcal{I}|] \;\middle|\; d_{k,j,i} > 0 \right\}$;
    $s_{k,j} \leftarrow \tau_{i_1}^{\mathrm{end}} - d_{k,j,i_1}$;
    $\mathrm{est}_{k,i_2} \leftarrow \mathrm{est}_{k,i_2} + d_{k,j,i_2}$;
  `/* The jobs scheduled within a single metering interval. */`
  foreach $\left\{ J_{k,j} \in \mathcal{J} \;\middle|\; \left|\{I_i \in \mathcal{I} \mid d_{k,j,i} > 0\}\right| = 1 \right\}$ **do**
    $I_{i_1} \leftarrow I_i \in \mathcal{I} : d_{k,j,i} > 0$;
    $s_{k,j} \leftarrow \mathrm{est}_{k,i_1}$;
    $\mathrm{est}_{k,i_1} \leftarrow \mathrm{est}_{k,i_1} + p_{k,j}$;
  return s;

Algorithm 2: The construction of start times $\mathbf{s}$ from overlap variables $\boldsymbol{d}$, which represent a feasible solution to the implicit MILP model.

4.5.2.4 Iterative Implicit MILP Model

The disadvantage of the implicit MILP model is the formulation of the objective, which uses the indicator constraints. An alternative, more efficient approach is described in this section.

The idea is to "wrap" the implicit model into an algorithm that iteratively decreases the number of metering intervals in the scheduling horizon. Each iteration re-solves the implicit model in metering intervals $I_1, I_2, \ldots, I_{i^{\max}}$, where $I_{i^{\max}}$ denotes the scheduling horizon's last metering interval. Moreover, the original implicit model's objective is replaced with a simpler one, i.e., constraints (4.45)-(4.46) are replaced with

$$\tau_{i^{\max}}^{\text{start}} + \sum_{J_{k,j} \in \mathcal{J}_k^M} d_{k,j,i} \leq C_{\max}\,, \quad \forall M_k \in \mathcal{M}\,. \tag{4.60}$$

The new objective minimizes the maximum total overlap in $I_{i^{\max}}$ per machine. A simple observation is that no idle time between two consecutive jobs on the same machine is necessary within $I_{i^{\max}}$, since the subsequent job is fully contained in $I_{i^{\max}}$, and shifting it to the completion time of the preceding job has no effect on the consumed energy in $I_{i^{\max}}$. Therefore, minimization of the maximum total overlap in $I_{i^{\max}}$ also minimizes the makespan in $I_{i^{\max}}$. The full pseudocode of the iterative algorithm is listed in Algorithm 3.

The algorithm starts by checking the feasibility of the given problem instance by trying to find any feasible solution within the whole scheduling horizon H. If the instance is infeasible, function `SolveImplicitMilp` returns an empty set instead of a vector of start times. If the solution is feasible, the algorithm starts iterating until the optimal solution is found.

To determine the optimality of a feasible solution, the algorithm tests its objective value $C_{\max}$. If the makespan is greater than $\tau_{i^{\max}}^{\text{start}}$, the found solution is optimal to the original problem. On the other hand, if the objective is $\tau_{i^{\max}}^{\text{start}}$, a shorter solution ending in the previous metering interval might exist to the original problem. Therefore, $i^{\max}$ is decreased by one, and the implicit model is resolved with the smaller horizon.

Notice that in every iteration, the previously found start times are passed to `SolveImplicitMilp`, which are used as a warm start (by transformation to the overlap variables) to reduce the implicit model's running time.

4.5.3 Adaptive Local Search Heuristic

Since the exact methods can solve only small instances in a reasonable time, heuristic algorithms are necessary to solve (at least sub-optimally) industrial-size instances with thousands of jobs. This section describes a heuristic algorithm for problem PD$m|E_i^{\max}|C_{\max}$ that is based on the adaptive local search framework from Section 3.2. The specific parts of the general adaptive local search algorithm for this problem are the scheduling operator and the method for finding the initial incumbent solution. Both are described in the following text.

```
1  Function IterativeImplicitMilpModel():
2      i^max ← |I| ;
3      /* Solve the implicit model for metering intervals I_1, I_2, ..., I_{i^max}.
          */
4      (s, C_max) ← SolveImplicitMilp(i^max, ∅) ;
5      if s = ∅ then
6          /* The original problem is infeasible.  */
7          return ∅;
8      while C_max = τ^start_{i^max} do
9          i^max ← i^max − 1 ;
10         /* Solve the implicit model for metering intervals
              I_1, I_2, ..., I_{i^max}.  */
11         s, C_max ← SolveImplicitMilp(i^max, s) ;
12     /* Found optimal solution to the original problem.  */
13     return s;
```

Algorithm 3: Iterative implicit MILP model.

4.5.3.1 The Scheduling Operator

The scheduling operator algorithm is responsible for finding the feasible start times from the given all-jobs order. Formally, the scheduling operator solves the problem $\mathrm{PD}m|\pi_k, E_i^{\max}|C_{\max}$. However, as shown in Section 4.3, this scheduling problem is $\mathcal{NP}$-hard. Moreover, a high computational complexity for the variant with the fixed number of machines and no-cross constraint (see Section 4.4) gives a pessimistic insight that even the problem $\mathrm{PD}m|\pi_k, E_i^{\max}|C_{\max}$ with a fixed number of machines could not be solved to optimality with an efficient algorithm. Since the operator is called for every generated neighbor, its implementation must be efficient so that the local search can provide solutions in a short time. Thus, we opt to use a heuristic scheduling operator that might find sub-optimal solutions but has a small computational complexity. The pseudocode of the used scheduling operator is listed in Algorithm 4.

The scheduling operator schedules each job in the order given by all-jobs ordering $\lambda : [1\,..\,n] \to \mathcal{J}$ at their *earliest* feasible start time w.r.t. the currently scheduled jobs. The pseudocode of the earliest start time computation is given in Algorithm 5. The algorithm keeps the current *candidate start time* $s_{k,j}$ of job $J_{k,j}$ to be scheduled (initialized by *available time* c_k of the corresponding machine M_k). The algorithm iterates over the metering intervals, and in every metering interval, the feasibility of the current candidate start time is tested by comparing the maximum possible overlap of the job within the tested metering interval with the overlap computed from the current candidate start time. If the overlap is greater than the maximum possible overlap, the current candidate start time is not feasible, and the next candidate start time is tried (the next one is computed from the maximum possible overlap in the tested metering interval). The earliest feasible candidate start time is then returned by the algorithm.

Note that the scheduling operator respects the ordering of the jobs scheduled on the same machine; however, the ordering over different machines is not guaranteed.

```
1  Function NonRobustSchedulingOperator(λ):
2      /* The current schedule.  */
3      s ← 0;
4      /* The consumed energy in the metering intervals in the current
          schedule.  */
5      e ← 0^{|I|};
6      /* The earliest time where each machine is available for
          processing the next job.  */
7      c ← 0^m;
8      foreach ℓ ∈ [1 .. n] do
9          J_{k,j} ← λ(ℓ);
10         s_{k,j} ← ComputeEarliestNonRobustStartTime(J_{k,j}, e, c);
11         e, c ← ScheduleJob(J_{k,j}, s_{k,j}, e, c);
12     return s;
13 Function ScheduleJob(J_{k,j}, s_{k,j}, e, c):
14     c_k ← max{s_{k,j} + p_{k,j}, c_k};
15     foreach I_i ∈ I do
16         e_i ← e_i + Overlap(s_{k,j}, p_{k,j}, i) · P_{k,j};
17     return e, c;
```

Algorithm 4: The non-robust scheduling operator for problem PD$m|\pi_k, E_i^{\max}|C_{\max}$.

```
1  Function ComputeEarliestNonRobustStartTime(J_{k,j}, e, c):
2      /* The candidate for the earliest feasible start time.  */
3      s_{k,j} ← c_k;
4      /* The index of a metering interval to check for feasibility.  */
5      i ← ⌊s_{k,j} / D⌋;
6      while TRUE do
7          if s_{k,j} + p_{k,j} > H then
8              return ∅;
9          if i > |I| ∨ Overlap(s_{k,j}, p_{k,j}, i) = 0 then
10             /* All metering intervals having non-zero overlap with I_i
                  were tested successfully.  */
11             return s_{k,j};
12         maxPossibleOverlap ← ⌊max{0, E_i^max − e_i} / P_{k,j}⌋;
13         if
               maxPossibleOverlap < p_{k,j} ∧ maxPossibleOverlap < Overlap(s_{k,j}, p_{k,j}, i)
               then
14             s_{k,j} ← τ_i^end − maxPossibleOverlap;
15         i ← i + 1;
```

Algorithm 5: Computation of the earliest feasible non-robust start time.

4.5.3.2 Finding the Initial Incumbent Solution: a Constructive Heuristic

The local search algorithm has to start from some initial all-jobs ordering λ. For this purpose, a constructive heuristic is designed. Besides the constructive heuristic, 100 random all-jobs orderings are sampled. The best found all-jobs ordering (w.r.t. the makespan obtained by the scheduling operator) is selected as the initial incumbent solution.

The constructive heuristic creates the all-jobs ordering iteratively one-by-one using *priority rules*; the complete pseudocode is shown in Algorithm 6. To select the next job in the ordering, the constructive heuristic first computes the earliest start time for all the unscheduled jobs using Algorithm 5. Then, the jobs are sorted using the following priority rules, and the job with the highest priority is scheduled next:

- total remaining unscheduled processing time on the machines (higher value has higher priority);
- the metering interval in which the jobs starts (earliest metering interval has higher priority);
- the processing overlap of a job with the metering interval in which the job starts (higher overlap has higher priority);
- the energy consumption within the metering interval in which the jobs starts (higher energy consumption has higher priority).

The preliminary experiments have shown that the order of the priority rules, according to which the jobs are sorted, is essential; different orderings work well for different classes of instances. Therefore, the priority rules' order is configurable, and the constructive heuristic tries all the possible orderings of the priority rules ($4 \cdot 3 \cdot 2 \cdot 1 = 24$ in total). That is, the constructive heuristic is given an ordered sequence of the priority rules. When the tested values by a priority rule are the same, the next priority rule in the sequence is used until either the corresponding tested values are different or there are no more priority rules to try (in that case, the job with a smaller index has larger priority).

4.5.4 Experiments

The experiments in this section focus on PD$m|E_i^{\max}|C_{\max}$ scheduling problem. The experiments evaluate the performance of the presented methods on small and large instances (in terms of the number of jobs).

4.5.4.1 Benchmark Instances

The instances for the experiments were generated as follows. The maximum energy limit $E^{\max}$ was fixed to 1000[1], and the number of jobs n_k dedicated to each machine M_k is the same. To generate the processing times, parameter $\alpha_1 \in \mathbb{R}_{>0}$ is used,

[1]The limit is the same since one energy limit is contracted for a longer period of time (months) in practice.

```
1  Function NonRobustConstructiveHeuristic(priorityRules):
2      /* The unscheduled jobs.  */
3      J' ← J;
4      /* The constructed all-jobs ordering.  */
5      λ ← ∅;
6      /* The current position into the all-jobs ordering.  */
7      ℓ ← 1;
8      /* The consumed energy in the metering intervals.  */
9      e ← 0^|I|;
10     /* The earliest time where each machine is available for
          processing the next job.  */
11     c ← 0^m;
12     /* The start times.  */
13     s ← 0;
14     while J' ≠ ∅ do
15         /* Select the next unscheduled job using the priority rules.
              */
16         J_{k,j} ← GetNextJob(J', e, c, priorityRules);
17         J' ← J' \ {J_{k,j}};
18         s_{k,j} ← ComputeEarliestNonRobustStartTime(J_{k,j}, e, c);
19         e, c ← ScheduleJob(J_{k,j}, s_{k,j}, e, c);
20         λ(ℓ) ← J_{k,j};
21         ℓ ← ℓ + 1;
22     return λ;
```

Algorithm 6: Constructive heuristic for finding the initial incumbent solution for problem PD$m|E_i^{\max}|C_{\max}$.

which links the maximum allowed processing time with the length of the metering intervals. The processing times are then sampled from discrete uniform distribution $\mathcal{U}\{1, \lceil D \cdot \alpha_1 \rceil\}$.

The power consumption of the jobs is generated using parameter α_2 that controls the energy limit's tightness. Given fixed $\alpha_2 \in \mathbb{R}_{\geq 0}$, the power consumption of each job is then sampled from continuous uniform distribution

$$\mathcal{U}\left(\alpha_2 \cdot \frac{E^{\max}}{m \cdot D}, 2 \cdot \frac{E^{\max}}{m \cdot D}\right). \tag{4.61}$$

To ensure that no job can violate the energy limit by itself, the power consumption is clamped, i.e., if $\min(D, p_{k,j}) \cdot P_{k,j} > E^{\max}$ for some job $J_{k,j}$, its power consumption is modified to $\frac{E^{\max}}{\min(D, p_{k,j})}$.

The length of the horizon influences the performance of the CP and MILP models, since the number of the constraints and variables increases. On the other hand, the local search heuristic is insensitive to the horizon length. To have a fair comparison of the CP and MILP models with the local search heuristic algorithm, the scheduling horizon of every instance is shortened in the following way: a feasible schedule is found from a random all-jobs ordering using Algorithm 4, and the horizon is set to the end of the metering interval in which the last job completes.

For each $n_k \in \{15, 50, 150, 350\}, m \in \{2, 5, 10\}$, $D \in \{15, 60\}$[2], $\alpha_1 \in \{1, 3\}$, and $\alpha_2 \in \{0.8, 1.2, 1.6\}$, seven random instances were generated using the scheme above. Therefore, the generated dataset has 1008 instances in total. Notice that the generated dataset contains only feasible instances due to the clamping of the jobs' power consumption and the heuristically-obtained horizons.

The dataset is divided into two parts: small ($n_k \in \{15, 50\}$) and large instances ($n_k \in \{150, 350\}$). The separation is necessary since different MILP models dot not fit into the available memory when solving either medium or large instances.

4.5.4.2 Experiment Setting and Remarks

The following abbreviations of the evaluated method are used: (i) NR-CONS: constructive heuristic (see Algorithm 6), (ii) NR-ALS: adaptive local search algorithm (see Algorithm 4.5.3) initialized with the solution found by NR-CONS, (iii) NR-CP: reference CP model (see Section 4.5.1), (iv) NR-MILP-DIS: the disjunctive MILP model (see Section 4.5.2.1), (v) NR-MILP-TIME: the time-indexed MILP model (see Section 4.5.2.2), (vi) NR-MILP-ITER: the iterative implicit MILP model (see Section 4.5.2.4), and (vii) NR-MILP-ZENG: the implicit MILP model from the literature [70] adapted to the scheduling problem with the energy consumption limits and a makespan as objective (see Section 4.5.2.3 for the description how the adaptation is performed).

The time-limit given to each method on each instance is 10 minutes (the time-limit for NR-ALS is decreased by the running time of NR-CONS). Five neighbors were generated in each iteration of the local search. The maximum possible length of a random block selected in NHGEN-BLOCK-SWAP, NHGEN-BLOCK-INS neighborhood generators is $0.01 \cdot n$.

The experiment result tables report the following values:

[2] Acoording to [66], the length of the metering intervals is usually between 15 to 60 minutes.

- # opt. proofs: the number of instances for which a method found the optimal feasible solution and was able to prove its optimality within the given time-limit.

- # best: the number of best solutions found by a method for a group of instances with the same parameters, e.g., number of the jobs. If more methods found a solution having the best objective, then the table values are increased for all these methods.

- # no sol.: the number of instances for which a method could not find any feasible solution within the given time-limit.

- worse-to-best ratio: shows how far are the solutions from the best-found solutions. Formally, let BEST be a best-found solution objective to a given instance and let UB be an objective of another solution of the same instance such that UB > BEST. Then, a *worse-to-best ratio* is computed as $\frac{\text{UB}}{\text{BEST}}$. The tables report the median and the maximum of the worse-to-best ratios for each instance group.

  If a method found the best solution for all the instances, the value of corresponding table cell will be BEST.

The best values for a group of instances with the same parameters are shown in bold.

The experiments were executed on 2x Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz with 180GB of RAM and Debian 10 operating system. The MILP and CP models were implemented in Gurobi 8.1 and IBM CP Optimizer 12.9 solvers, respectively. All the algorithms and solvers were running as a single thread and were programmed in C#.

4.5.4.3 Experiment Results: Comparison of MILP-based Methods on Small Instances, $n_k \in \{15, 50\}$

Table 4.1 contains the results for the MILP-based methods on the small instances. For $n_k = 15$, method NR-MILP-ITER performs the best w.r.t. all the performance indicators. The second best method is NR-MILP-ZENG (for $n_k = 15, m = 5$ it even solved more instances to optimality than NR-MILP-ITER), although this model has a high chance of not finding any feasible solution within the time-limit.

However, the results for $n_k = 50$ are substantially different. NR-MILP-TIME and NR-MILP-ITER are the only MILP-based methods that can find feasible solutions, with NR-MILP-TIME being the better of the two. The reason is that NR-MILP-ITER has poor lower bound, which is non-zero only if the optimal makespan is within $I_{i^{\max}}$.

In conclusion, NR-MILP-ITER performs the best for $n_k = 15$, whereas NR-MILP-TIME is better for $n_k = 50$. The performance of both models could be potentially increased by providing a warm start. In the rest of the experiments, only NR-MILP-ITER and NR-MILP-TIME will be compared with the rest of the non-MILP methods.

Table 4.1: The experiment results for the MILP-based methods on the small instances. Group parameters: n_k, m.

n_k	m	method	# opt. proofs	# best	# no sol.
15	2	NR-MILP-DIS	11	19	18
		NR-MILP-ITER	**34**	**81**	**0**
		NR-MILP-ZENG	30	37	38
		NR-MILP-TIME	4	7	2
	5	NR-MILP-DIS	2	13	26
		NR-MILP-ITER	39	**70**	**0**
		NR-MILP-ZENG	**42**	44	31
		NR-MILP-TIME	17	30	1
	10	NR-MILP-DIS	1	11	41
		NR-MILP-ITER	**55**	**77**	**0**
		NR-MILP-ZENG	39	43	40
		NR-MILP-TIME	18	26	3
50	2	NR-MILP-DIS	**0**	0	84
		NR-MILP-ITER	**0**	20	47
		NR-MILP-ZENG	**0**	0	82
		NR-MILP-TIME	**0**	**41**	**30**
	5	NR-MILP-DIS	**0**	0	84
		NR-MILP-ITER	**0**	13	70
		NR-MILP-ZENG	**0**	0	84
		NR-MILP-TIME	**0**	**29**	**44**
	10	NR-MILP-DIS	**0**	0	84
		NR-MILP-ITER	**0**	13	71
		NR-MILP-ZENG	**0**	0	84
		NR-MILP-TIME	**0**	**35**	**38**

4.5.4.4 Experiment Results: Small Instances, $n_k \in \{15, 50\}$

The results for the small instances are presented in Table 4.2, Table 4.3, Table 4.4, Table 4.5, and Table 4.6. From the results the following can be concluded:

- In total, NR-MILP-ITER method solved more instances to the optimality than NR-CP; 25.4% of the small instances were solved optimally by NR-MILP-ITER, 7.74% by NR-MILP-TIME, whereas NR-CP solved 6.35% of the small instances. However, note that for $n_k = 50$, only NR-CP was able to solve some instances optimally.
- The NR-ALS's performance degrades with the increasing number of machines, i.e., NR-CP finds better solutions more often than NR-ALS if the number of machines is 10.
- Overall, the median of the worse-to-best ratio of NR-ALS is better than of other methods. This means that solutions found by NR-ALS, which are not

the best one, tend to be better than the non-best solutions found by other methods.

- The maximum processing time of the jobs has a large impact on the performance of NR-MILP-ITER and NR-MILP-TIME, which find the most optimal solutions when the processing times are short.
- The vast majority of the optimally solved instances by NR-CP are for $\alpha_2 = 0.8$, that is, when the energy limit is not tight w.r.t. the power consumption of the jobs. Other exact methods are not significantly influenced by this parameter.
- Both NR-CP and NR-MILP-ITER find more optimal solutions when the metering interval is longer.
- In total, NR-CONS finds the best solution in 13.29% cases, NR-ALS in 78.57%, NR-CP in 65.28%, NR-MILP-ZENG in 27.18%, and NR-MILP-TIME in 8.33% cases. For $n_k = 15$, NR-CP performs the best, whereas for instances with $n_k = 50$ the NR-ALS begins to outperform the other methods.

Table 4.2: The experiment results for the small instances. Group parameters: D.

D	method	# opt. proofs	# best	# no sol.
15	NR-CONS	0	41	**0**
	NR-ALS	0	**202**	**0**
	NR-CP	1	152	**0**
	NR-MILP-ITER	**51**	58	101
	NR-MILP-TIME	24	27	45
60	NR-CONS	0	26	**0**
	NR-ALS	0	**194**	**0**
	NR-CP	31	177	**0**
	NR-MILP-ITER	**77**	79	87
	NR-MILP-TIME	15	15	73

Table 4.3: The experiment results for the small instances. Group parameters: α_1.

α_1	method	# opt. proofs	# best	# no sol.
1	NR-CONS	0	35	**0**
	NR-ALS	0	**207**	**0**
	NR-CP	16	206	**0**
	NR-MILP-ITER	**112**	120	97
	NR-MILP-TIME	39	42	32
3	NR-CONS	0	32	**0**
	NR-ALS	0	**189**	**0**
	NR-CP	**16**	123	**0**
	NR-MILP-ITER	**16**	17	91
	NR-MILP-TIME	0	0	86

Table 4.4: The experiment results for the small instances. Group parameters: α_2.

α_2	method	# opt. proofs	# best	# no sol.
0.8	NR-CONS	0	32	**0**
	NR-ALS	0	**128**	**0**
	NR-CP	31	125	**0**
	NR-MILP-ITER	**49**	52	50
	NR-MILP-TIME	17	18	38
1.2	NR-CONS	0	20	**0**
	NR-ALS	0	**125**	**0**
	NR-CP	1	108	**0**
	NR-MILP-ITER	**43**	45	60
	NR-MILP-TIME	10	11	31
1.6	NR-CONS	0	15	**0**
	NR-ALS	0	**143**	**0**
	NR-CP	0	96	**0**
	NR-MILP-ITER	**36**	40	78
	NR-MILP-TIME	12	13	49

Table 4.5: The experiment results for the small instances. Group parameters: n_k, m.

n_k	m	method	# opt. proofs	# best	# no sol.	worse-to-best ratio median	worse-to-best ratio max
15	2	NR-CONS	0	1	**0**	1.01310	1.06062
		NR-ALS	0	**74**	**0**	1.00389	**1.00959**
		NR-CP	1	49	**0**	**1.00301**	1.03226
		NR-MILP-ITER	**34**	38	**0**	1.02784	1.07566
		NR-MILP-TIME	4	5	2	1.04155	1.12693
	5	NR-CONS	0	7	**0**	1.00733	1.04522
		NR-ALS	0	62	**0**	**1.00156**	**1.00714**
		NR-CP	5	**75**	**0**	1.00510	1.03431
		NR-MILP-ITER	**39**	41	**0**	1.00694	1.02368
		NR-MILP-TIME	17	19	1	1.01871	1.20688
	10	NR-CONS	0	13	**0**	1.00865	1.04536
		NR-ALS	0	39	**0**	**1.00196**	1.00735
		NR-CP	14	**78**	**0**	1.00487	**1.00595**
		NR-MILP-ITER	**55**	57	**0**	1.00919	1.02430
		NR-MILP-TIME	18	18	3	1.01698	1.21916
50	2	NR-CONS	**0**	6	**0**	1.00705	1.03727
		NR-ALS	**0**	**82**	**0**	**1.00181**	**1.00183**
		NR-CP	**0**	20	**0**	1.01428	1.03235
		NR-MILP-ITER	**0**	0	47	1.14371	1.19807
		NR-MILP-TIME	**0**	0	30	1.03967	1.25255
	5	NR-CONS	**0**	14	**0**	1.00161	1.01351
		NR-ALS	**0**	**76**	**0**	**1.00050**	**1.00188**
		NR-CP	**0**	41	**0**	1.00144	1.01146
		NR-MILP-ITER	**0**	1	70	1.04466	1.11765
		NR-MILP-TIME	**0**	0	44	1.04182	1.14242
	10	NR-CONS	0	26	**0**	1.00164	1.03052
		NR-ALS	0	63	**0**	**1.00049**	**1.00195**
		NR-CP	**12**	**66**	**0**	1.00062	1.01146
		NR-MILP-ITER	0	0	71	1.03448	1.09646
		NR-MILP-TIME	0	0	38	1.05122	1.13553

Table 4.6: The experiment results for the small instances. Overall median of the worst-to-best ratio.

Method	Median worst-to-best ratio
NR-CONS	1.00581
NR-ALS	**1.00163**
NR-CP	1.00290
NR-MILP-ITER	1.02313
NR-MILP-TIME	1.03984

4.5.4.5 Experiment Results: Large Instances, $n_k \in \{150, 350\}$

The results for the large instances are presented in Table 4.7, Table 4.8, Table 4.9, and Table 4.10. The results show that NR-ALS clearly outperforms NR-CP and NR-CONS w.r.t. the number of best solutions found. For the largest instances ($n_k = 350, m = 5$), NR-CP cannot find feasible solutions for 3 instances. In total, NR-CONS finds the best solution in 17.86% cases, NR-ALS in 97.62%, and NR-CP in 5.36% cases.

Table 4.7: The experiment results for the large instances. Group parameters: D.

D	method	# opt. proofs	# best	# no sol.
15	NR-CONS	**0**	81	**0**
	NR-ALS	**0**	**250**	**0**
	NR-CP	**0**	8	3
60	NR-CONS	**0**	9	**0**
	NR-ALS	**0**	**242**	**0**
	NR-CP	**0**	19	**0**

Table 4.8: The experiment results for the large instances. Group parameters: α_1.

α_1	method	# opt. proofs	# best	# no sol.
1	NR-CONS	**0**	48	**0**
	NR-ALS	**0**	**248**	**0**
	NR-CP	**0**	19	**0**
3	NR-CONS	**0**	42	**0**
	NR-ALS	**0**	**244**	**0**
	NR-CP	**0**	8	3

Table 4.9: The experiment results for the large instances. Group parameters: α_2.

α_2	method	# opt. proofs	# best	# no sol.
0.8	NR-CONS	**0**	23	**0**
	NR-ALS	**0**	**156**	**0**
	NR-CP	**0**	22	**0**
1.2	NR-CONS	**0**	38	**0**
	NR-ALS	**0**	**168**	**0**
	NR-CP	**0**	5	**0**
1.6	NR-CONS	**0**	29	**0**
	NR-ALS	**0**	**168**	**0**
	NR-CP	**0**	0	3

Table 4.10: The experiment results for the large instances. Group parameters: n_k, m.

n_k	m	method	# opt. proofs	# best	# no sol.	worse-to-best ratio median	worse-to-best ratio max
150	2	NR-CONS	**0**	3	**0**	1.00314	1.01951
		NR-ALS	**0**	**84**	**0**	**BEST**	**BEST**
		NR-CP	**0**	1	**0**	1.01246	1.03688
	5	NR-CONS	**0**	18	**0**	1.00041	1.00380
		NR-ALS	**0**	**80**	**0**	**1.00039**	**1.00062**
		NR-CP	**0**	14	**0**	1.00162	1.00617
	10	NR-CONS	**0**	27	**0**	**1.00057**	1.02634
		NR-ALS	**0**	**82**	**0**	1.00114	**1.00168**
		NR-CP	**0**	6	**0**	1.00139	1.01140
350	2	NR-CONS	**0**	8	**0**	1.00077	1.01471
		NR-ALS	**0**	**84**	**0**	**BEST**	**BEST**
		NR-CP	**0**	0	**0**	1.01292	1.05578
	5	NR-CONS	**0**	18	**0**	1.00023	1.00177
		NR-ALS	**0**	**84**	**0**	**BEST**	**BEST**
		NR-CP	**0**	0	**0**	1.00305	1.01231
	10	NR-CONS	**0**	16	**0**	**1.00028**	1.02221
		NR-ALS	**0**	**78**	**0**	1.00339	**1.00735**
		NR-CP	**0**	6	3	1.00359	1.01598

4.5.4.6 Experiment Results: Additional Analysis

Fig. 4.4 shows the total number of hits per neighborhood generator used by NR-ALS. The neighborhood generators that try to move the jobs defining the makespan provide the most number of hits. However, this does not mean that ALS should not use the other neighborhood generators since they may provide additional diversification of the search (recall that any solution with better or the same makespan as the incumbent is accepted as the new incumbent, see Algorithm 1).

The last Fig. 4.5 compares NR-CP and NR-ALS w.r.t. the best-found objective

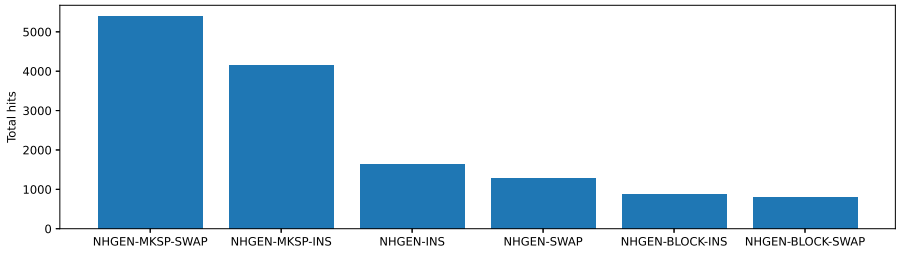

Figure 4.4: NR-ALS, the total number of hits per neighborhood generator over all the instances.

in time represented by best-so-far curves. *A best-so-far curve* plots the makespan of the best solution found over time. Fig. 4.5 averages all such curves to show an aggregated execution of each method over all instances. Since the instances' optimal objective may differ, the curves' objective values are normalized by the best-found solution. The average best-so-far curves reveal that NR-CP starts with low-quality solutions that are progressively improved. On the other hand, NR-CONS already finds high-quality solutions, which are slightly improved by NR-ALS. Moreover, NR-ALS tends to find the best solution earlier than NR-CP.

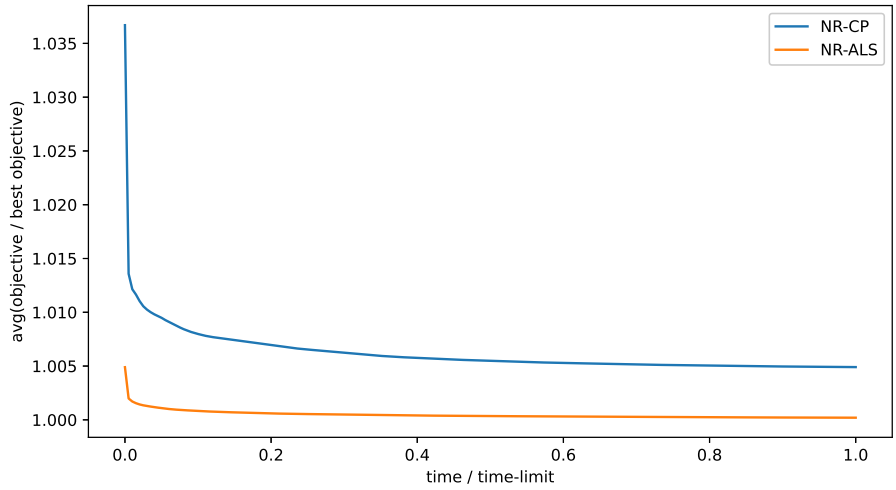

Figure 4.5: The average of the best-so-far curves over all the instances (normalized to the time-limit and the best-found solution by NR-CP and NR-ALS).

Chapter

Scheduling with Energy Consumption Limits under Disturbances

5

As discussed in the introduction, considering disturbances in many environments is necessary for designing practical schedules. Therefore, this chapter is devoted to methods that can guarantee the satisfaction of the energy consumption limits even if some jobs are delayed.

The most important result of this chapter is the pseudo-polynomial algorithm for problem $1|\pi_1, E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|f$; see Section 5.2. Thanks to this efficient algorithm, we can design various methods for solving the single machine scheduling problem without the fixed permutation; see Section 5.3.

5.1 Problem $\mathrm{PD}m|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$

Due to the power of CP formalism, we can easily model the general scheduling problem from Chapter 2 in CP. The idea behind the model is straightforward, i.e., for every deviation situation, its corresponding realized schedule is constructed. Then, for each realized schedule, we add constraints that enforce the energy limits in that realized schedule. The disadvantage of such a model is that it contains an exponential number of variables and constraints; thus, it is suitable only for very small instances.

For every job $J_{k,j} \in \mathcal{J}$ and deviation situation $\boldsymbol{\delta}^{(q)} \in \Delta$, the model uses one interval variable $x_{k,j}^{(q)}$ representing the time interval in which the job is scheduled in the corresponding realized schedule (recall that $q = 1$ corresponds to the baseline schedule). Since the definition of the realized schedules is based on the permutation of the jobs π corresponding to the baseline schedule, we may use sequence variables to construct the realized schedules. Unfortunately, IBM CP Optimizer formalism does not allow to specify one sequence variable that could be used over all realized schedules, i.e., a sequence variable is tied to specific interval variables. Instead, a new sequence variable $y_k^{(q)}$ is created for each deviation situation $\boldsymbol{\delta}^{(q)} \in \Delta$ and machine $M_k \in \mathcal{M}$, and the sequences are enforced to be the same using the SameSequence constraint.

The complete model follows

$$\min \max_{J_{k,j} \in \mathcal{J}} \mathrm{EndOf}(x_{k,j}^{(1)}) \tag{5.1}$$

$$\mathrm{StartOf}(x_{k,j}^{(1)}) \leq s_{k,j}^{\max}, \quad \forall J_{k,j} \in \mathcal{J} \tag{5.2}$$

$$\mathrm{LengthOf}(x_{k,j}^{(q)}) = p_{k,j}, \quad \forall J_{k,j} \in \mathcal{J}; \forall \boldsymbol{\delta}^{(q)} \in \Delta \tag{5.3}$$

$$\mathrm{Sequence}(y_k^{(q)}, \{x_{k,j}^{(q)} \mid J_{k,j} \in \mathcal{J}_k^M\}), \quad \forall \boldsymbol{\delta}^{(q)} \in \Delta; \forall M_k \in \mathcal{M} \tag{5.4}$$

$$\text{SameSequence}(y_k^{(1)}, y_k^{(q)}), \quad \forall \boldsymbol{\delta}^{(q)} \in \Delta : q \geq 2; \forall M_k \in \mathcal{M} \tag{5.5}$$

$$\text{NoOverlap}(y_k^{(q)}), \quad \forall \boldsymbol{\delta}^{(q)} \in \Delta; \forall M_k \in \mathcal{M} \tag{5.6}$$

$$\text{StartOf}(x_{k,j}^{(q)}) = \max\{\text{StartOf}(x_{k,j}^{(1)}), \text{EndOfPrev}(y_k^{(q)}, x_{k,j}^{(q)}, 0)\} + \delta_{k,j}^{(q)}, \quad \forall \boldsymbol{\delta}^{(q)} \in \Delta; \forall J_{k,j} \in \mathcal{J} \tag{5.7}$$

$$\sum_{J_{k,j} \in \mathcal{J}} \text{Overlap}(x_{k,j}^{(q)}, \tau_i^{\text{start}}, \tau_i^{\text{end}}) \cdot P_{k,j} \leq E_i^{\max}, \quad \forall \boldsymbol{\delta}^{(q)} \in \Delta; \forall I_i \in \mathcal{I} \tag{5.8}$$

The objective (5.1) minimizes the maximum completion time of all the jobs in the baseline schedule. Constraint (5.2) guarantees that every job is scheduled within the time window bounded above by the job's maximum baseline start time. The following constraint (5.3) sets the length of interval variable $x_{k,j}^{(q)}$ corresponding to job $J_{k,j}$ and deviation situation $\boldsymbol{\delta}^{(q)} \in \Delta$ to be its processing time $p_{k,j}$. The sequencing of interval variables $x_{k,j}^{(q)}$ in a realized schedule corresponding to deviation situation $\boldsymbol{\delta}^{(q)} \in \Delta$ is modeled in constraint (5.4). Next, those sequence variables are forced to represent the same sequence using constraint (5.5). To ensure that the jobs are not overlapping in any realized schedule, their interval variables are linked with global constraint NoOverlap, see (5.6). The starting time of each job in every deviation situation $\boldsymbol{\delta}^{(q)} \in \Delta$ is formulated in constraint (5.7) (the third argument in EndOfPrev is the default value for the first position in a sequence). Finally, the energy limits in every metering interval and every realized schedule are enforced by constraint (5.8).

In some cases, customizing the search priority of the variables may increase the efficiency of the models (both CP and MILP). IBM CP Optimizer allows specifying such search priority using *search phases*. In this case, we first search on the baseline sequence $y_k^{(1)}$ and then on the baseline start times $x_{k,j}^{(1)}$. With the fixed baseline sequence and start times, all the other variables' values can be inferred without a search.

5.2 Problem $1|\pi_1, E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|f$

This section presents the main result of this chapter: the pseudo-polynomial algorithm for problem $1|\pi_1, E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|f$ (recall that f represents any regular objective). We start first by introducing some preliminaries, i.e., the latest start time schedules, right-shift schedules, and earliest robust baseline schedules. Based on these foundations, we provide the algorithm, prove its correctness and asymptotic time complexity.

5.2.1 Latest Start Time and Right-shift Schedules

Before we explain the algorithm, the notions of *latest start times* and *right-shift schedules* have to be defined. Both definitions assume given baseline schedule $\boldsymbol{s}$ and its corresponding permutation π.

Latest start time schedule $\boldsymbol{ls}$ is defined using vector function LS as

$$LS(\boldsymbol{s})_{1,\pi_1(\ell)} = RS(\boldsymbol{s}, (\delta^{\max}, \delta^{\max}, \ldots, \delta^{\max}))_{1,\pi_1(\ell)}, \quad \forall J_{1,j} \in \mathcal{J}. \tag{5.9}$$

Informally, the latest start time of a job represents the maximum possible starting time over all realized schedules of a given baseline schedule $\boldsymbol{s}$.

Lemma 4. *Let $\boldsymbol{s}^1, \boldsymbol{s}^2$ be two baseline schedules with the same permutation π such that for some position $\bar{\ell} \in [1\,..\,n]$, the following holds*

$$s^1_{1,\pi_1(\ell)} \leq s^2_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}]\,. \tag{5.10}$$

Then

$$ls^1_{1,\pi_1(\ell)} \leq ls^2_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}]\,, \tag{5.11}$$

where $\boldsymbol{ls}^1 = LS(\boldsymbol{s}^1)$ and $\boldsymbol{ls}^2 = LS(\boldsymbol{s}^2)$.

Proof. Proof by induction on ℓ

1. basis, $\ell = 1$: $ls^1_{1,\pi_1(1)} = s^1_{1,\pi_1(1)} + \delta^{\max} \leq s^2_{1,\pi_1(1)} + \delta^{\max} = ls^2_{1,\pi_1(1)}$.
2. induction step, $1 < \ell \leq \bar{\ell}$:

$$\begin{aligned} ls^1_{1,\pi_1(\ell)} &= \max\{s^1_{1,\pi_1(\ell)}, ls^1_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} + \delta^{\max} \\ &\leq \max\{s^2_{1,\pi_1(\ell)}, ls^2_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} + \delta^{\max} \\ &= ls^2_{1,\pi_1(\ell)}\,. \end{aligned} \tag{5.12}$$

□

Let $J_{1,\pi_1(\bar{\ell})} \in \mathcal{J}$ be some job and $t \in [s_{1,\pi_1(\bar{\ell})}\,..\,LS(\boldsymbol{s})_{1,\pi_1(\bar{\ell})}]$ its arbitrary realized start time. Then *right-shift schedule* $\boldsymbol{rss}$ is defined using recursive vector function RSS as

$$RSS(\boldsymbol{s}, \bar{\ell}, t)_{1,\pi_1(\ell)} = \begin{cases} t & \ell = \bar{\ell} \\ \min\{LS(\boldsymbol{s})_{1,\pi_1(\ell)}, RSS(\boldsymbol{s}, \bar{\ell}, t)_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)}\} & \ell \in [1\,..\,\bar{\ell}-1]\,. \end{cases} \tag{5.13}$$

Informally, a right-shift schedule is obtained from $\boldsymbol{s}$ by fixing the start time of job $J_{1,\pi_1(\bar{\ell})}$ to t and shifting all the jobs on positions $\ell < \bar{\ell}$ to the right as much as possible while respecting the latest start times and the non-overlap constraint. Notice that a right-shift schedule defines starting times only for the jobs on positions $[1\,..\,\bar{\ell}]$.

An important property of the right-shift schedules is that they are also realized schedules, i.e., for each right-shift schedule $\boldsymbol{rss}$, there exists deviation scenario $\boldsymbol{\delta} \in \Delta$, whose corresponding realized schedule $\boldsymbol{rs}$ is the same as $\boldsymbol{rss}$.

Lemma 5. *Let $\boldsymbol{s}$ be a baseline schedule and π be its corresponding permutation. Let $RSS(\boldsymbol{s}, \bar{\ell}, t)$ be the right-shift schedule for some job $J_{1,\pi_1(\bar{\ell})} \in \mathcal{J}$ and time $t \in [s_{1,\pi_1(\bar{\ell})}\,..\,LS(\boldsymbol{s})_{1,\pi_1(\bar{\ell})}]$. Then there exists scenario $\boldsymbol{\delta} \in \Delta$ such that*

$$RS(\boldsymbol{s}, \boldsymbol{\delta})_{1,\pi_1(\ell)} = RSS(\boldsymbol{s}, \bar{\ell}, t)_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}]\,. \tag{5.14}$$

Proof. Let $\boldsymbol{rss} = RSS(\boldsymbol{s}, \bar{\ell}, t)$, $\boldsymbol{rs} = RS(\boldsymbol{s}, \boldsymbol{\delta})$, and $\boldsymbol{ls} = LS(\boldsymbol{s})$. First, we prove that

$$rss_{1,\pi_1(\ell)} \in [s_{1,\pi_1(\ell)}\,..\,ls_{1,\pi_1(\ell)}]\,, \quad \forall \ell \in [1\,..\,\bar{\ell}]\,. \tag{5.15}$$

The property holds trivially from the definition for $\ell = \bar{\ell}$. To prove the property for $\ell < \bar{\ell}$, assume by contradiction that $\ell < \bar{\ell}$ is the largest position such that $rss_{1,\pi_1(\ell)} \notin [s_{1,\pi_1(\ell)} \,..\, ls_{1,\pi_1(\ell)}]$. Since $rss_{1,\pi_1(\ell)} \leq ls_{1,\pi_1(\ell)}$ holds from the definition of the right-shift start time, it must $rss_{1,\pi_1(\ell)} < s_{1,\pi_1(\ell)} \leq ls_{1,\pi_1(\ell)}$. Therefore,

$$\begin{aligned} rss_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)} &\geq s_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)} \\ &\geq s_{1,\pi_1(\ell)} \\ &> rss_{1,\pi_1(\ell)} \\ &= \min\{ls_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)}\} \\ &= rss_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)}\,, \end{aligned} \tag{5.16}$$

which is a contradiction.

Now we prove that if $\boldsymbol{\delta}$ is defined as

$$\delta_{1,\pi_1(\ell)} = \begin{cases} rss_{1,\pi_1(1)} - s_{1,\pi_1(1)} & \ell = 1 \\ rss_{1,\pi_1(\ell)} - \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} & \ell \in [2\,..\,\bar{\ell}]\,, \end{cases} \tag{5.17}$$

then for each position $\ell \in [1\,..\,\bar{\ell}]$ holds that $\delta_{1,\pi_1(\ell)} \in [0\,..\,\delta^{\max}]$, and $rs_{1,\pi_1(\ell)} = rss_{1,\pi_1(\ell)}$. Proof by induction on ℓ

1. basis, $\ell = 1$:

   - $\delta_{1,\pi_1(1)} \in [0\,..\,\delta^{\max}]$:

   $$\delta_{1,\pi_1(1)} = rss_{1,\pi_1(1)} - s_{1,\pi_1(1)} \geq 0 \tag{5.18}$$

   $$\begin{aligned} \delta_{1,\pi_1(1)} &= rss_{1,\pi_1(1)} - s_{1,\pi_1(1)} \\ &\leq ls_{1,\pi_1(1)} - s_{1,\pi_1(1)} \\ &= s_{1,\pi_1(1)} + \delta^{\max} - s_{1,\pi_1(1)} \\ &= \delta^{\max} \end{aligned} \tag{5.19}$$

   - $rs_{1,\pi_1(1)} = rss_{1,\pi_1(1)}$:

   $$rs_{1,\pi_1(1)} = s_{1,\pi_1(1)} + \delta_{1,\pi_1(1)} = s_{1,\pi_1(1)} + rss_{1,\pi_1(1)} - s_{1,\pi_1(1)} = rss_{1,\pi_1(1)} \tag{5.20}$$

2. induction step, $1 < \ell \leq \bar{\ell}$:

   - $\delta_{1,\pi_1(\ell)} \geq 0$: consider cases

     (a) $s_{1,\pi_1(\ell)} \geq rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}$:

     $$\begin{aligned} \delta_{1,\pi_1(\ell)} &= rss_{1,\pi_1(\ell)} - \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} \\ &= rss_{1,\pi_1(\ell)} - s_{1,\pi_1(\ell)} \\ &\geq 0 \end{aligned} \tag{5.21}$$

(b) $s_{1,\pi_1(\ell)} < rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}$:

$$\begin{aligned}\delta_{1,\pi_1(\ell)} &= rss_{1,\pi_1(\ell)} - \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} \\ &= rss_{1,\pi_1(\ell)} - p_{1,\pi_1(\ell-1)} - rss_{1,\pi_1(\ell-1)} \\ &\geq \min\{ls_{1,\pi_1(\ell-1)}, rss_{1,\pi_1(\ell)} - p_{1,\pi_1(\ell-1)}\} - rss_{1,\pi_1(\ell-1)} \\ &= rss_{1,\pi_1(\ell-1)} - rss_{1,\pi_1(\ell-1)} \\ &= 0\end{aligned} \tag{5.22}$$

- $\delta_{1,\pi_1(\ell)} \leq \delta^{\max}$: consider cases

  (a) $rss_{1,\pi_1(\ell-1)} = ls_{1,\pi_1(\ell-1)}$: since

$$\begin{aligned}ls_{1,\pi_1(\ell)} &= \max\{s_{1,\pi_1(\ell)}, ls_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} + \delta^{\max} \\ &= \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} + \delta^{\max}\end{aligned} \tag{5.23}$$

  therefore

$$\begin{aligned}\delta_{1,\pi_1(\ell)} &= rss_{1,\pi_1(\ell)} - \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} \\ &= rss_{1,\pi_1(\ell)} - ls_{1,\pi_1(\ell)} + \delta^{\max} \\ &\leq \delta^{\max}\end{aligned} \tag{5.24}$$

  (b) $rss_{1,\pi_1(\ell-1)} = rss_{1,\pi_1(\ell)} - p_{1,\pi_1(\ell-1)}$:

$$\begin{aligned}\delta_{1,\pi_1(\ell)} &= rss_{1,\pi_1(\ell)} - \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} \\ &= rss_{1,\pi_1(\ell)} - \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell)}\} \\ &\leq 0 \\ &\leq \delta^{\max}\end{aligned} \tag{5.25}$$

- $rs_{1,\pi_1(\ell)} = rss_{1,\pi_1(\ell)}$:

$$\begin{aligned}rs_{1,\pi_1(\ell)} &= \max\{s_{1,\pi_1(\ell)}, rs_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} + \delta_{1,\pi_1(\ell)} \\ &= \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} + \delta_{1,\pi_1(\ell)} \\ &= \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} \\ &\quad + rss_{1,\pi_1(\ell)} - \max\{s_{1,\pi_1(\ell)}, rss_{1,\pi_1(\ell-1)} + p_{1,\pi_1(\ell-1)}\} \\ &= rss_{1,\pi_1(\ell)}\end{aligned} \tag{5.26}$$

□

5.2.2 Earliest Robust Baseline Schedule

The algorithm for finding the optimal robust schedule is based on the iterative computation of the *earliest robust baseline start time* for each job in the order given by π. A robust baseline start time of $J_{1,\pi_1(\bar{\ell})}$ is a baseline start time such that there is no realized schedule of jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell})}$ in which some energy

consumption limit is violated. More formally, $s_{1,\pi_1(\bar{\ell})}$ is *robust relative to* baseline start times $s_{1,\pi_1(1)}, s_{1,\pi_1(2)}, \ldots, s_{1,\pi_1(\bar{\ell}-1)}$ if

$$\sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(RS(\boldsymbol{s},\boldsymbol{\delta})_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} \leq E_i^{\max}, \quad \forall \boldsymbol{\delta} \in \Delta; \forall I_i \in \mathcal{I}\,. \tag{5.27}$$

The *earliest robust baseline start time* is simply a robust baseline start time that is the smallest possible (relative to the preceding jobs' baseline start times).

In the rest of this section, we prove a theorem (see Theorem 5) that is central to the algorithm constructing the robust baseline start times; it proves that baseline schedule $\boldsymbol{s}$ is robust and optimal w.r.t. the regular objective function if every job starts at its earliest robust time in $\boldsymbol{s}$. However, before formally proving it, we need to lay some theoretical groundwork.

Lemma 6. *Let* $\mathbf{s}^1, \mathbf{s}^2$ *be two schedules (not necessarily baseline, realised, etc.) with the same permutation* π *such that for some position* $\bar{\ell} \in [1\,..\,n]$

$$\begin{aligned} s^1_{1,\pi_1(\bar{\ell})} &= s^2_{1,\pi_1(\bar{\ell})} \\ s^1_{1,\pi_1(\ell)} &\leq s^2_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}-1]\,. \end{aligned} \tag{5.28}$$

Let $I_i \in \mathcal{I}$ *be an arbitrary metering interval such that* $\mathrm{Overlap}(s^1_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) > 0$. *Then*

$$\sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(s^1_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} \leq \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(s^2_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)}\,. \tag{5.29}$$

Proof. The Lemma obviously holds for metering intervals I_i such that $s^1_{1,\pi_1(\bar{\ell})} \leq \tau_i^{\mathrm{start}}$ because only job $J_{1,\pi_1(\bar{\ell})}$ may have non-zero overlap with I_i in both schedules, i.e.,

$$\begin{aligned} \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(s^1_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} &= \mathrm{Overlap}(s^1_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) \cdot P_{1,\pi_1(\bar{\ell})} \\ &= \mathrm{Overlap}(s^2_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) \cdot P_{1,\pi_1(\bar{\ell})} \\ &= \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(s^2_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)}\,. \end{aligned} \tag{5.30}$$

Now, assume the case $\tau_i^{\mathrm{start}} < s^1_{1,\pi_1(\bar{\ell})}$. We prove the Lemma by showing

$$\mathrm{Overlap}(s^1_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \leq \mathrm{Overlap}(s^1_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i)\,, \quad \forall \ell \in [1\,..\,\bar{\ell}]\,. \tag{5.31}$$

Due to assumption $s^1_{1,\pi_1(\bar{\ell})} = s^2_{1,\pi_1(\bar{\ell})}$, the inequality holds for $\ell = \bar{\ell}$; thus, let us consider case $\ell < \bar{\ell}$. Since $\mathrm{Overlap}(s^1_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) > 0$, it must $s^1_{1,\pi_1(\bar{\ell})} < \tau_i^{\mathrm{end}}$, which in combination with the Lemma assumptions gives

$$s^1_{1,\pi_1(\ell)} + p_{1,\pi_1(\ell)} \leq s^2_{1,\pi_1(\ell)} + p_{1,\pi_1(\ell)} \leq s^1_{1,\pi_1(\bar{\ell})} < \tau_i^{\mathrm{end}}\,. \tag{5.32}$$

We use this knowledge to simplify the overlap length of the jobs in I_i

$$\begin{aligned}
\text{Overlap}(s^1_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) &= \max\{0, \min\{s^1_{1,\pi_1(\bar{\ell})}, s^1_{1,\pi_1(\ell)} + p_{1,\pi_1(\ell)}\} \\
&\quad - \max\{s^1_{1,\pi_1(\ell)}, \tau_i^{\text{start}}\}\} \\
&= \max\{0, s^1_{1,\pi_1(\ell)} + p_{1,\pi_1(\ell)} - \max\{s^1_{1,\pi_1(\ell)}, \tau_i^{\text{start}}\}\} \\
\text{Overlap}(s^2_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) &= \max\{0, \min\{s^1_{1,\pi_1(\bar{\ell})}, s^2_{1,\pi_1(\ell)} + p_{1,\pi_1(\ell)}\} \\
&\quad - \max\{s^2_{1,\pi_1(\ell)}, \tau_i^{\text{start}}\}\} \\
&= \max\{0, s^2_{1,\pi_1(\ell)} + p_{1,\pi_1(\ell)} - \max\{s^2_{1,\pi_1(\ell)}, \tau_i^{\text{start}}\}\}\,.
\end{aligned} \tag{5.33}$$

Now, consider function $g(x) = x + p_{1,\pi_1(\ell)} - \max\{x, \tau_i^{\text{start}}\}$. It is easy to see that $g(x)$ is non-decreasing in x, and since from the Lemma assumption we know that $s^1_{1,\pi_1(\ell)} \le s^2_{1,\pi_1(\ell)}$, the following holds

$$\begin{aligned}
\text{Overlap}(s^1_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) &= \max\{0, g(s^1_{1,\pi_1(\ell)})\} \\
&\le \max\{0, g(s^2_{1,\pi_1(\ell)})\} \\
&= \text{Overlap}(s^2_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i)\,.
\end{aligned} \tag{5.34}$$

□

Lemma 7. *Let $\boldsymbol{s}^1, \boldsymbol{s}^2$ be two baseline schedules with the same permutation π such that for some position $\bar{\ell} \in [1\,..\,n]$*

$$\begin{aligned}
&s^1_{1,\pi_1(\bar{\ell})} = s^2_{1,\pi_1(\bar{\ell})} \\
&s^1_{1,\pi_1(\ell)} \le s^2_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell} - 1]\,.
\end{aligned} \tag{5.35}$$

Moreover, let $\boldsymbol{ls}^1 = LS(\boldsymbol{s}^1)$ and $\boldsymbol{ls}^2 = LS(\boldsymbol{s}^2)$. Then for every $t \in [s^1_{1,\pi_1(\bar{\ell})}\,..\,ls^1_{1,\pi_1(\bar{\ell})}]$ and every $\boldsymbol{\delta} \in \Delta$ such that $t = RS(\boldsymbol{s}^1, \boldsymbol{\delta})_{1,\pi_1(\bar{\ell})}$ it holds that

$$\begin{aligned}
&RS(\boldsymbol{s}^1, \boldsymbol{\delta})_{1,\pi_1(\bar{\ell})} = RSS(\boldsymbol{s}^2, \bar{\ell}, t)_{1,\pi_1(\bar{\ell})} \\
&RS(\boldsymbol{s}^1, \boldsymbol{\delta})_{1,\pi_1(\ell)} \le RSS(\boldsymbol{s}^2, \bar{\ell}, t)_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell} - 1]\,.
\end{aligned} \tag{5.36}$$

Proof. Let $\boldsymbol{rs}^1 = RS(\boldsymbol{s}^1, \boldsymbol{\delta})$ and $\boldsymbol{rss}^2 = RSS(\boldsymbol{s}^2, \bar{\ell}, t)$. We prove this lemma by induction on ℓ

1. basis, $\ell = \bar{\ell}$: holds from the assumptions.
2. induction step, $1 \le \ell < \bar{\ell}$: consider the following two cases
   (a) $ls^2_{1,\pi_1(\ell)} > rss^2_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)}$: then

$$rs^1_{1,\pi_1(\ell)} \le rs^1_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)} \le rss^2_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)} = rss^2_{1,\pi_1(\ell)}\,. \tag{5.37}$$

   (b) $ls^2_{1,\pi_1(\ell)} \le rss^2_{1,\pi_1(\ell+1)} - p_{1,\pi_1(\ell)}$: then from Lemma 4

$$rs^1_{1,\pi_1(\ell)} \le ls^1_{1,\pi_1(\ell)} \le ls^2_{1,\pi_1(\ell)} = rss^2_{1,\pi_1(\ell)}\,. \tag{5.38}$$

□

Theorem 5. *Let $\boldsymbol{s}$ be a baseline schedule with the corresponding permutation π. If every job starts at its earliest robust time in $\boldsymbol{s}$, then $\boldsymbol{s}$ is robust and optimal w.r.t. any regular objective f for permutation π.*

Proof.

1. *Robustness*: The robustness of $\boldsymbol{s}$ follows from the definition of the robust baseline start time; see Eq. (5.27).

2. *Optimality*: Let $\mathbf{s}^\star$ be the optimal and robust baseline schedule w.r.t. f for permutation π. We show that $\forall J_{1,j} \in \mathcal{J} : s_{1,j} \leq s^\star_{1,j}$, which implies $f(\boldsymbol{s}) \leq f(\mathbf{s}^\star)$.

   Assume by contradiction that $\bar{\ell} \in [1\,..\,n]$ is the first position in π such that

   $$\begin{aligned} &s_{1,\pi_1(\bar{\ell})} > s^\star_{1,\pi_1(\bar{\ell})} \\ &s_{1,\pi_1(\ell)} \leq s^\star_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}-1]\,. \end{aligned} \tag{5.39}$$

   Construct schedule $\mathbf{s}'$ such that

   $$\begin{aligned} &s'_{1,\pi_1(\bar{\ell})} = s^\star_{1,\pi_1(\bar{\ell})} \\ &s'_{1,\pi_1(\ell)} = s_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}-1]\,, \end{aligned} \tag{5.40}$$

   i.e., $\mathbf{s}'$ is the same as $\boldsymbol{s}$, except for job $J_{1,\pi_1(\bar{\ell})}$ that starts at time $s^\star_{1,\pi_1(\bar{\ell})}$. Since $s'_{1,\pi_1(\bar{\ell})} = s^\star_{1,\pi_1(\bar{\ell})}$ is not the earliest robust time in $\boldsymbol{s}$, there exists some realized schedule $\boldsymbol{rs}'$ of $\mathbf{s}'$ such that

   $$E_i^{\max} < \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(rs'_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} \tag{5.41}$$

   in some metering interval $I_i \in \mathcal{I}$. Since jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell}-1)}$ start at their earliest robust time in both schedules $\boldsymbol{s}$ and $\mathbf{s}'$, metering interval I_i must have a non-zero overlap with $J_{1,\pi_1(\bar{\ell})}$ in $\boldsymbol{rs}'$. Construct right-shift schedule $\boldsymbol{rss}^\star = RSS(\mathbf{s}^\star, \bar{\ell}, rs'_{1,\pi_1(\bar{\ell})})$; the construction is possible since from Lemma 4 it holds that $rs'_{1,\pi_1(\bar{\ell})} \in [s^\star_{1,\pi_1(\bar{\ell})}\,..\,LS(\mathbf{s}^\star)_{1,\pi_1(\bar{\ell})}]$. By applying Lemma 7 ($\boldsymbol{s}^1 = \mathbf{s}', \boldsymbol{s}^2 = \mathbf{s}^\star$), and Lemma 6 ($\mathbf{s}^1 = \boldsymbol{rs}', \mathbf{s}^2 = \boldsymbol{rss}^\star$), we conclude that

   $$\begin{aligned} E_i^{\max} &< \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(rs'_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} \\ &\leq \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(rss^\star_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)}\,, \end{aligned} \tag{5.42}$$

   which is a contradiction with the assumption that schedule $\mathbf{s}^\star$ is robust.

□

5.2.3 Algorithm for Finding the Optimal Robust Schedule

From the definition, the computation of the earliest robust start time of any job $J_{1,\pi_1(\bar{\ell})} \in \mathcal{J}$ depends only on the baseline start times of $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell}-1)}$. Therefore, the earliest robust start times can be computed one-by-one according to the ascending order of the positions in the given permutation π. Such an algorithm terminates either with computing the earliest robust baseline schedule or concluding that permutation π is infeasible, i.e., for the given permutation π, it is not possible to find a robust baseline start time of some job.

The algorithm for computing the earliest robust baseline start time of $J_{1,\pi_1(\bar{\ell})}$ is presented in the remainder of this section. First, we show a naïve version of the algorithm, which illustrates the basic structure, albeit with exponential complexity. Then, we gradually improve this complexity by introducing the concepts behind the more efficient algorithm.

5.2.3.1 Naïve Algorithm for Computing the Earliest Robust Baseline Start Time of $J_{1,\pi_1(\bar{\ell})}$

Let $\boldsymbol{s}$ be a baseline schedule where jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell}-1)}$ start at their earliest robust baseline start time, and we want to find the earliest robust baseline start time for $J_{1,\pi_1(\bar{\ell})}$ (all other jobs are not yet assigned to any start time). A naïve algorithm (see its pseudo-code in Algorithm 7) directly applies the definition of the earliest robust baseline start time: iterate over every possible baseline start time $s_{1,\pi_1(\bar{\ell})} \in [s_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell}-1)} \mathinner{..} s^{\max}_{1,\pi_1(\bar{\ell})}]$ in increasing order, and select the earliest baseline start time such that Eq. (5.27) is not violated. Unfortunately, such an algorithm is inefficient since the number of realized schedules of $\boldsymbol{s}$ is exponential in n.

```
1  Function ComputeEarliestRobustStartTime(π, J_{1,π1(ℓ̄)}, s):
2      if ℓ̄ = 1 then
3          s_{1,π1(ℓ̄)} ← 0;
4      else
5          s_{1,π1(ℓ̄)} ← max s_{1,π1(ℓ̄−1)} + p_{1,π1(ℓ̄−1)};
6      while s_{1,π1(ℓ̄)} ≤ s^max_{1,π1(ℓ̄)} do
7          energyLimitViolated ← FALSE;
8          foreach (δ, I_i) ∈ Δ × I do
9              rs ← RS(s, δ);
10             if Σ_{ℓ=1}^{ℓ̄} Overlap(rs_{1,π1(ℓ)}, p_{1,π1(ℓ)}, i) · P_{1,π1(ℓ)} > E_i^max then
11                 energyLimitViolated ← TRUE;
12                 break;
13         if energyLimitViolated = FALSE then
14             return s_{1,π1(ℓ̄)}, FEASIBLE;
15         else
16             s_{1,π1(ℓ̄)} ← s_{1,π1(ℓ̄)} + 1;
17     return ∅, INFEASIBLE_PERMUTATION;
```

Algorithm 7: Earliest robust baseline start time of $J_{1,\pi_1(\bar{\ell})}$ for fixed permutation π, naïve version.

5.2.3.2 Increasing the Efficiency of the Naïve Algorithm - Energy Consumption Dominance of the Right-shift Schedules

The first key observation for obtaining an efficient algorithm is the "energy consumption dominance" of the right-shift schedules. Let $\boldsymbol{rs}$ be some realized schedule of jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell})}$. Then, it can be proven that the energy consumption in the metering intervals intersected by $J_{1,\pi_1(\bar{\ell})}$ in $\boldsymbol{rss} = RSS(\boldsymbol{s}, \bar{\ell}, rs_{1,\pi_1(\bar{\ell})})$ is not less than in $\boldsymbol{rs}$.

Lemma 8. *Let $\boldsymbol{s}$ be a baseline schedule with the corresponding permutation π. Let $\ell \in [1\,..\,n]$, and $\boldsymbol{rs}$ be some realised schedule of jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell})}$, and $\boldsymbol{rss} = RSS(\boldsymbol{s}, \bar{\ell}, rs_{1,\pi_1(\bar{\ell})})$. Then,*

$$\forall I_i \in \mathcal{I} : \mathrm{Overlap}(rs_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) > 0 \implies$$
$$\sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(rs_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} \leq \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(rss_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)}\,. \tag{5.43}$$

Proof. From Lemma 7 it holds that

$$\begin{aligned} rs_{1,\pi_1(\bar{\ell})} &= rss^2_{1,\pi_1(\bar{\ell})} \\ rs^1_{1,\pi_1(\ell)} &\leq rss^2_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}-1]\,. \end{aligned} \tag{5.44}$$

By applying Lemma 6 on $\mathbf{s}^1 = \boldsymbol{rs}$ and $\mathbf{s}^2 = \boldsymbol{rss}$, the Lemma is proven. □

Therefore, it suffices to consider only the right-shift schedules since if some energy limit is violated in $\boldsymbol{rs}$, it will also be violated in $\boldsymbol{rss}$. Lines 8-12 in Algorithm 7 can be replaced with the following pseudo-code

```
1  if ℓ̄ = 1 then
2      foreach (δ_{1,π1(ℓ̄)}, I_i) ∈ [0 .. δ^max] × 𝓘 do
3          if Overlap(s_{1,π1(ℓ̄)} + δ_{1,π1(ℓ̄)}, p_{1,π1(ℓ̄)}, i) · P_{1,π1(ℓ̄)} > E_i^max then
4              energyLimitViolated ← TRUE;
5              break;
6  else
7      foreach (t, δ_{1,π1(ℓ̄)}, I_i) ∈ [s_{1,π1(ℓ̄−1)} .. LS(s)_{1,π1(ℓ̄−1)}] × [0 .. δ^max] × 𝓘 do
8          rs ← RSS(s, ℓ̄ − 1, t);
9          rs_{1,π1(ℓ̄)} ← max{s_{1,π1(ℓ̄)}, rs_{1,π1(ℓ̄−1)} + p_{1,π1(ℓ̄−1)}} + δ_{1,π1(ℓ̄)} ;
10         if Σ_{ℓ=1}^{ℓ̄} Overlap(rs_{1,π1(ℓ)}, p_{1,π1(ℓ)}, i) · P_{1,π1(ℓ)} > E_i^max then
11             energyLimitViolated ← TRUE;
12             break;
```

Although this modified algorithm does not have exponential complexity anymore, it is still inefficient since it asymptotically depends on the length of the horizon.

5.2.3.3 Increasing the Efficiency of the Naïve Algorithm - Maximum Possible Overlap of the Jobs with the Metering Intervals

As was noted at the end of the previous section, the complexity of the naïve algorithm depends on the length of the horizon since whenever a realized schedule violating any energy consumption limit is found, the baseline start time of $J_{1,\pi_1(\bar{\ell})}$ is increased by 1; see Line 16 in Algorithm 7. The question is whether we can identify a range of non-robust baseline start times of $J_{1,\pi_1(\bar{\ell})}$ so that it is possible to "jump" by more than 1 time unit on Line 16 in Algorithm 7. This is possible by considering *a maximum possible overlap* of job $J_{1,\pi_1(\bar{\ell})}$ with the metering intervals.

Lemma 9. *Let $\boldsymbol{s}$ be a baseline schedule with the corresponding permutation π. Assume that the baseline start times of jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell}-1)}$ are robust for some $\bar{\ell} \in [1\,..\,n]$. Moreover, assume that there exists some realized schedule $\boldsymbol{rs}$ of $\boldsymbol{s}$ such that for some metering interval $I_i \in \mathcal{I}$ the following holds*

$$\begin{aligned} &E_i^{\max} < \sum_{\ell=1}^{\bar{\ell}} \mathrm{Overlap}(rs_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} \\ &\mathrm{Overlap}(rs_{1,\pi_1(\bar{\ell}-1)}, p_{1,\pi_1(\bar{\ell}-1)}, i) > 0\,. \end{aligned} \tag{5.45}$$

Then all baseline start times $[s_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell}-1)}\,..\,rs_{1,\pi_1(\bar{\ell})}]$ of $J_{1,\pi_1(\bar{\ell})}$ are not robust.

Proof. Assume by contradiction that there is a robust baseline schedule $\mathbf{s}'$ such that

$$\begin{aligned} &s'_{1,\pi_1(\bar{\ell})} \in [s_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell}-1)}\,..\,rs_{1,\pi_1(\bar{\ell})}] \\ &s'_{1,\pi_1(\ell)} = s_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}-1]\,. \end{aligned} \tag{5.46}$$

Construct realized schedule $\boldsymbol{rs}'$ of $\mathbf{s}'$ such that

$$\begin{aligned} &rs'_{1,\pi_1(\bar{\ell})} = \max\{s'_{1,\pi_1(\bar{\ell})}, rs'_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell}-1)}\} \\ &rs'_{1,\pi_1(\ell)} = rs_{1,\pi_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}-1]\,. \end{aligned} \tag{5.47}$$

It is easy to see that $\boldsymbol{rs}'$ is a realized schedule in which the deviation of $J_{1,\pi_1(\bar{\ell})}$ is 0. Since

$$\begin{aligned} &rs_{1,\pi_1(\bar{\ell})} \geq s'_{1,\pi_1(\bar{\ell})} \\ &rs_{1,\pi_1(\bar{\ell})} \geq rs_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell}-1)} = rs'_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell}-1)}\,, \end{aligned} \tag{5.48}$$

it must $rs_{1,\pi_1(\bar{\ell})} \geq rs'_{1,\pi_1(\bar{\ell})}$. Moreover, due to $\mathrm{Overlap}(rs_{1,\pi_1(\bar{\ell}-1)}, p_{1,\pi_1(\bar{\ell}-1)}, i) = \mathrm{Overlap}(rs'_{1,\pi_1(\bar{\ell}-1)}, p_{1,\pi_1(\bar{\ell}-1)}, i) > 0$, it holds that $rs'_{1,\pi_1(\bar{\ell})} \geq \tau_i^{\mathrm{start}}$. Therefore,

$$\mathrm{Overlap}(rs_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) \leq \mathrm{Overlap}(rs'_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i)\,. \tag{5.49}$$

However, this means that

$$\begin{aligned} E_i^{\max} &< \sum_{\ell=1}^{\bar{\ell}} \text{Overlap}(rs_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)} \\ &\leq \sum_{\ell=1}^{\bar{\ell}} \text{Overlap}(rs'_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)}\,, \end{aligned} \tag{5.50}$$

i.e., $\boldsymbol{rs'}$ is a realized schedule in which a energy limit is violated; thus, $\mathbf{s}'$ cannot be robust. □

Assume that $\boldsymbol{rs}$ is some realized schedule of jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell})}$ such that the energy consumption limit is violated in some metering interval $I_i \in \mathcal{I}$. One of the following two cases occurs:

Case 1 $\text{Overlap}(rs_{1,\pi_1(\bar{\ell}-1)}, p_{1,\pi_1(\bar{\ell}-1)}, i) > 0$: Since the baseline start times of $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell}-1)}$ are robust, it must $\text{Overlap}(rs_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) > 0$. We may ask what is the maximum possible overlap of $J_{1,\pi_1(\bar{\ell})}$ in I_i w.r.t. to realized start times $\boldsymbol{rs}$ of $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell}-1)}$ without violating the energy limit

$$\mathit{maxPossibleOverlap}_i = \left\lfloor \frac{E_i^{\max} - \sum_{\ell=1}^{\bar{\ell}-1} \text{Overlap}(rs_{1,\pi_1(\ell)}, p_{1,\pi_1(\ell)}, i) \cdot P_{1,\pi_1(\ell)}}{P_{1,\pi_1(\bar{\ell})}} \right\rfloor . \tag{5.51}$$

Lemma 9 implies that all baseline start times $[s_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell})} \mathinner{..} \tau_i^{\text{end}} - \mathit{maxPossibleOverlap}_i - 1]$ of $J_{1,\pi_1(\bar{\ell})}$ are not robust, i.e., $\tau_i^{\text{end}} - \mathit{maxPossibleOverlap}_i \leq s_{1,\pi_1(\bar{\ell})}$ must hold; otherwise, $E_i^{\max}$ is violated.

Case 2 $\text{Overlap}(rs_{1,\pi_1(\bar{\ell}-1)}, p_{1,\pi_1(\bar{\ell}-1)}, i) = 0$: In this case, $J_{1,\pi_1(\bar{\ell})}$ is the only job having a non-zero overlap with I_i in $\boldsymbol{rs}$. Therefore, it holds that $\text{Overlap}(rs_{1,\pi_1(\bar{\ell})}, p_{1,\pi_1(\bar{\ell})}, i) \cdot P_{1,\pi_1(\bar{\ell})} > E_i^{\max}$. We can compute the maximum possible overlap of $J_{1,\pi_1(\bar{\ell})}$ with I_i

$$\mathit{maxPossibleOverlap}_i = \left\lfloor \frac{E_i^{\max}}{P_{1,\pi_1(\bar{\ell})}} \right\rfloor , \tag{5.52}$$

which represents the maximum overlap between I_i and $J_{1,\pi_1(\bar{\ell})}$ without violating the energy consumption limit. Assuming that all baseline start times $[s_{1,\pi_1(\bar{\ell}-1)} + p_{1,\pi_1(\bar{\ell}-1)} \mathinner{..} s_{1,\pi_1(\bar{\ell})} - 1]$ of $J_{1,\pi_1(\bar{\ell})}$ are not robust, it is easy to see that $\tau_i^{\text{end}} - \mathit{maxPossibleOverlap}_i \leq s_{1,\pi_1(\bar{\ell})}$ must hold to assure that the overlap of $J_{1,\pi_1(\bar{\ell})}$ with I_i in any realized schedule is not larger than $\mathit{maxPossibleOverlap}_i$.

Due to these two cases, the computation of the earliest robust baseline start time can be split into two consecutive steps (corresponding to the cases described above)

Step 1 The earliest robust baseline start time relative to $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell}-1)}$: for each right-shift schedule $RSS(\boldsymbol{s}, \bar{\ell}-1, t)$, where $t \in [s_{1,\pi_1(\bar{\ell}-1)} \,..\, LS(\boldsymbol{s})_{1,\pi_1(\bar{\ell}-1)}]$, find the earliest baseline start time of $J_{1,\pi_1(\bar{\ell})}$ using the maximum possible overlap which does not violate $E_i^{\max}$, where $I_i \in \mathcal{I}$ is the last metering interval having a non-zero overlap with $J_{1,\pi_1(\bar{\ell}-1)}$. Notice that due to Lemma 9 it is efficient to check t in decreasing order since if some energy consumption limit is violated for some $rs_{1,\pi_1(\bar{\ell})}$, then all baseline start times $s_{1,\pi_1(\bar{\ell})} \leq rs_{1,\pi_1(\bar{\ell})}$ cannot be robust, and the algorithm can continue directly with the second step.

Step 2 The earliest robust baseline start time relative to only $J_{1,\pi_1(\bar{\ell})}$: After the first step, it is easy to see that for every baseline start time $[s_{1,\pi_1(\bar{\ell})} \,..\, s^{\max}_{1,\pi_1(\bar{\ell})}]$ of $J_{1,\pi_1(\bar{\ell})}$, there is no realized schedule of jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell})}$ in which both $J_{1,\pi_1(\bar{\ell}-1)}, J_{1,\pi_1(\bar{\ell})}$ have a non-zero overlap with some metering interval $I_i \in \mathcal{I}$, and $E_i^{\max}$ would be violated. However, the energy limits can still be violated in the metering intervals in which only $J_{1,\pi_1(\bar{\ell})}$ has a non-zero overlap.

For each $t \in [s_{1,\pi_1(\bar{\ell})} \,..\, LS(\boldsymbol{s})_{1,\pi_1(\bar{\ell})}]$, there exists a realized schedule $\boldsymbol{rs}$ of jobs $J_{1,\pi_1(1)}, J_{1,\pi_1(2)}, \ldots, J_{1,\pi_1(\bar{\ell})}$ such that $rs_{1,\pi_1(\bar{\ell})} = t$. Therefore, the maximum overlap of $J_{1,\pi_1(\bar{\ell})}$ with every metering interval $I_i \in \mathcal{I}$, such that $i \geq \lfloor \frac{s_{1,\pi_1(\bar{\ell})}}{D} \rfloor$, is

$$\begin{aligned} \mathit{maxOverlap}_i = \min\{p_{1,\pi_1(\bar{\ell})}, \min\{\tau_i^{\text{end}}, LS(\boldsymbol{s})_{1,\pi_1(\bar{\ell})} + p_{1,\pi_1(\bar{\ell})}\} \\ - \max\{\tau_i^{\text{start}}, s_{1,\pi_1(\bar{\ell})}\}\}\,. \end{aligned} \tag{5.53}$$

Then, $s_{1,\pi_1(\bar{\ell})}$ is not robust if there exists metering interval I_i such that the maximum overlap is larger than the maximum possible overlap in I_i, i.e.,

$$\mathit{maxPossibleOverlap}_i = \left\lfloor \frac{E_i^{\max}}{P_{1,\pi_1(\bar{\ell})}} \right\rfloor < \mathit{maxOverlap}_i\,. \tag{5.54}$$

If this is the case, then $\tau_i^{\text{end}} - \mathit{maxOverlap}_i$ is the earliest baseline start time that can be robust.

The complete algorithm for computing the robust baseline start time that combines all the discussed ideas is shown in Algorithm 8.

```
1  Function ComputeEarliestRobustStartTime(π, J_{1,π1(ℓ̄)}, s):
2      /* Computes latest start time for jobs J_{1,π1(1)}, J_{1,π1(2)}, ..., J_{1,π1(ℓ̄−1)}
          */
3      ls ← LS(s);
4      s_{1,π1(ℓ̄)} ← rs_{1,π1(ℓ̄)};
5      /* Step 1: Earliest robust baseline start time relative to
          J_{1,π1(1)}, J_{1,π1(2)}, ..., J_{1,π1(ℓ̄−1)}.  */
6      if ℓ̄ > 1 then
7          s_{1,π1(ℓ̄)} ← max{s_{1,π1(ℓ̄)}, s_{1,π1(ℓ̄−1)} + p_{1,π1(ℓ̄−1)}} ;
8          t ← ls_{1,π1(ℓ̄−1)};
9          while t ≥ min{ls_{1,π1(ℓ̄−1)}, s_{1,π1(ℓ̄−1)}} do
10             rss ← RSS(s, ℓ̄ − 1, t);
11             i ← ⌊(rss_{1,π1(ℓ̄−1)} + p_{1,π1(ℓ̄−1)} − 1)/D⌋;
12             maxPossibleOverlap_i =
                   ⌊(E_i^max − Σ_{ℓ=1}^{ℓ̄−1} Overlap(rss_{1,π1(ℓ)}, p_{1,π1(ℓ)}, i)·P_{1,π1(ℓ)})
                   / P_{1,π1(ℓ̄)}⌋;
13             if p_{1,π1(ℓ̄−1)} ≤ maxPossibleOverlap_i then
14                 /* Small optimization: Entire job fits into I_i without
                      violating E_i^max; therefore, continue with metering
                      interval I_{i−1}.  */
15                 t ← τ_i^start − p_{1,π1(ℓ̄−1)} − 1;
16             else if maxPossibleOverlap_i ≥ τ_i^end − (rss_{1,π1(ℓ̄−1)} + p_{1,π1(ℓ̄−1)}) then
17                 /* E_i^max is not violated.  */
18                 t ← t − 1;
19             else
20                 /* E_i^max is violated.  */
21                 s_{1,π1(ℓ̄)} ← τ_i^end − maxPossibleOverlap_i;
22                 break;
23     /* Step 2: Earliest robust baseline start time relative to only
          J_{1,π1(ℓ̄)}.  */
24     ls_{1,π1(ℓ̄)} ← LS(s)_{1,π1(ℓ̄)};
25     i ← ⌊(s_{1,π1(ℓ̄)} − 1)/D⌋;
26     while i ≤ |ℐ| do
27         maxPossibleOverlap_i ← ⌊E_i^max / P_{1,π1(ℓ̄)}⌋;
28         maxOverlap_i =
             min{p_{1,π1(ℓ̄)}, min{τ_i^end, ls_{1,π1(ℓ̄)} + p_{1,π1(ℓ̄)}} − max{τ_i^start, s_{1,π1(ℓ̄)}}};
29         if maxOverlap_i = 0 then
30             break;
31         else if maxPossibleOverlap_i < maxOverlap_i then
32             ls_{1,π1(ℓ̄)} ← τ_i^end − maxPossibleOverlap_i;
33             ls_{1,π1(ℓ̄)} ← LS(s)_{1,π1(ℓ̄)};
34         i ← i + 1;
35     if s_{1,π1(ℓ̄)} > s^max_{1,π1(ℓ̄)} then
36         return s_{1,π1(ℓ̄)}, FEASIBLE;
37     else
38         return ∅, INFEASIBLE_PERMUTATION;
```

Algorithm 8: Earliest robust baseline start time of $J_{1,\pi_1(\bar{\ell})}$ for fixed permutation π, optimized version.

5.2.4 Time Complexity of the Algorithm for Computing the Robust Baseline Schedule

The complexity of *Step 1* of Algorithm 8 is $\mathcal{O}(n^2 \cdot \delta^{\max})$ since the number of unique right-shift schedules of $J_{1,\pi_1(\bar{\ell}-1)}$ is at most $n \cdot \delta^{\max}$, and the computation of a right-shift schedule can be done in $\mathcal{O}(n)$. The complexity of *Step 2* is $\mathcal{O}(|\mathcal{I}|)$. Since Algorithm 8 is repeated for each position in the fixed permutation, the complexity of computing the optimal robust schedule for the given fixed permutation is $\mathcal{O}(n^3 \cdot \delta^{\max} + n \cdot |\mathcal{I}|)$; the pseudo-polynomiality of the algorithm arises due to term $\delta^{\max}$. Notice that if the jobs cannot violate the energy consumptions limits by themselves, i.e.,

$$\min\{p_{1,j}, D\} \cdot P_{1,j} \leq E_i^{\max}, \quad \forall J_{1,j} \in \mathcal{J}; \forall I_i \in \mathcal{I}, \tag{5.55}$$

then *Step 2* is not necessary, and the complexity decreases to $\mathcal{O}(n^3 \cdot \delta^{\max})$.

5.3 Problem $1|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$

To solve the single machine problem $1|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$, we propose various methods (both exact and heuristic) that utilize the pseudo-polynomial algorithm for the fixed permutation (see Section 5.2).

5.3.1 Logic-based Benders Decomposition

LBBD [30] is a generalization of the classical Benders decomposition that is used for solving large-scale optimization problems. In the classical Benders decomposition, the subproblem must be a continuous linear or non-linear problem, whereas in LBBD, the subproblem may have an arbitrary form.

The idea of LBBD is to decompose the original problem into two parts: (i) a *master problem*, which is a relaxation of the original problem, and (ii) a *subproblem*. After the master problem is solved to optimality, its solution is verified by the subproblem, whether it is feasible in the original problem or not. If yes, then the decomposition algorithm finishes since an optimal feasible solution for the original problem has been found. If not, a *no-good cut* is generated, which is a constraint such that the found solution violates it. The new no-good cut is added to the master problem, and the whole procedure is repeated; see Fig. 5.1.

In our case, the master problem is essentially a MILP model of $1|E_i^{\max}|C_{\max}$. The master problem's solution, i.e., baseline start times $\boldsymbol{s}$, is checked in the subproblem, whether it is robust or not. If schedule $\boldsymbol{s}$ is not robust, a no-good cut is generated for the master problem.

The modern LBBD implementations [38] add the no-good cuts gradually while solving the master problem. This approach is more integrated into MILP solvers and therefore more efficient since the master problem does not need to be resolved from scratch every time a new no-good cut is generated. State-of-the-art solvers, such as Gurobi or CPLEX, support adding the no-good cuts on-the-fly using mechanism called *lazy constraints generation*.

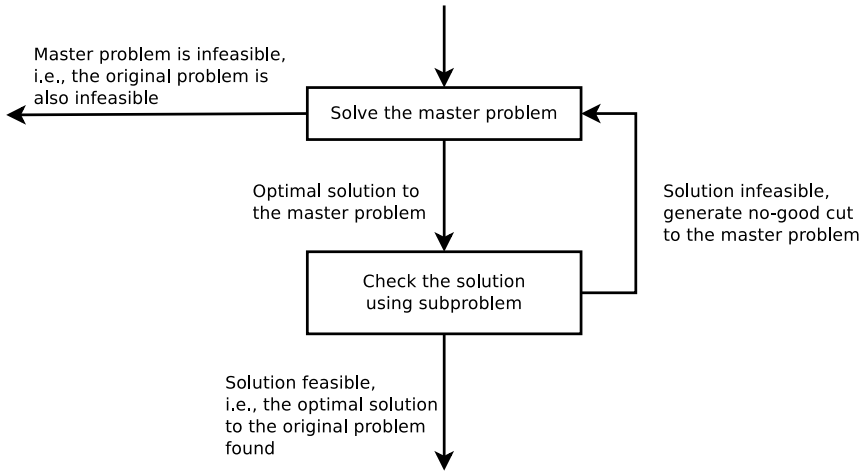

Figure 5.1: General schema of logic-based Benders decomposition.

5.3.1.1 Master Problem

The master problem is modeled as a time-indexed MILP, which is suitable for generating the no-good cuts. There are three types of variables in the program: (i) a *binary baseline start time* defined as $x_{j,t} = 1$ iff $J_{1,j}$ starts at t in the baseline schedule, (ii) the *energy consumed in time* t denoted as E_t^{time}, and (iii) $C_{\max}$ denoting the makespan. The master problem formulation follows.

$$\min C_{\max} \tag{5.56}$$

$$\sum_{t \in [0\,..\,H-1]} t \cdot x_{j,t} + p_{1,j} \leq C_{\max}, \quad \forall J_{1,j} \in \mathcal{J} \tag{5.57}$$

$$\sum_{t=0}^{s_{1,j}^{\max}} x_{j,t} = 1, \quad \forall J_{1,j} \in \mathcal{J} \tag{5.58}$$

$$\sum_{J_{1,j} \in \mathcal{J}} \sum_{t'=\max\{0, t-p_{1,j}+1\}}^{\min\{s_{1,j}^{\max}, t\}} x_{j,t'} \leq 1, \quad \forall t \in [0\,..\,H-1] \tag{5.59}$$

$$\sum_{J_{1,j} \in \mathcal{J}} \sum_{t'=\max\{0, t-p_{1,j}+1\}}^{\min\{s_{1,j}^{\max}, t\}} x_{j,t'} \cdot P_{1,j} \leq E_t^{\text{time}}, \quad \forall t \in [0\,..\,H-1] \tag{5.60}$$

$$\sum_{t=\tau_i^{\text{start}}}^{\tau_i^{\text{end}}} E_t^{\text{time}} \leq E_i^{\max}, \quad \forall I_i \in \mathcal{I} \tag{5.61}$$

The objective (5.56) of the master problem minimizes the makespan, which is the maximum completion time over all jobs (5.57). Constraint (5.58) ensures that each job starts in some time that is at most its maximum start time $s_{1,j}^{\max}$. Constraint (5.59) enforces that each time can be occupied by at most one job. Computation of consumed energy in time t is in Constraint (5.60). The last

constraint (5.61) ensures that the energy consumption limit in each metering interval is not violated for the baseline schedule.

The master model mentioned above does not take the deviations of the start times into account. We can strengthen the model by including the following constraint

$$\sum_{t=\max\{0,\tau_i^{\text{start}}-\delta\}}^{\tau_i^{\text{end}}-1-\delta} E_t^{\text{time}} \leq E_i^{\max}, \quad \forall I_i \in \mathcal{I}; \forall \delta \in [1\,..\,\delta^{\max}] \tag{5.62}$$

that guarantees the energy consumption limits for a subset of realized schedules, in which only a single job deviated. The idea is illustrated in Fig. 5.2 and explained below. Assume that $J_{1,j}$ is the job that occupies time $\tau_i^{\text{start}} - \delta$ in baseline schedule $\mathbf{s}$, where $\delta \in [1\,..\,\delta^{\max}]$. Let $\boldsymbol{rs}$ be a realized schedule corresponding to deviation situation $\boldsymbol{\delta}$ defined as $\delta_{k,j} = \delta$ and $\delta_{k,j'} = 0$ for $J_{1,j'} \neq J_{1,j}$. Due to the deviation of $J_{1,j}$, all unit parts of the jobs that are allocated within time interval $[\tau_i^{\text{start}} - \delta\,..\,\tau_i^{\text{end}} - \delta]$ in the baseline schedule will be right-shifted into interval $[\tau_i^{\text{start}}\,..\,\tau_i^{\text{end}}]$ in realized schedule $\boldsymbol{rs}$. The constraint is also valid when there are idle times between the jobs, although the constraint is weaker in such case because only a subset of jobs' unit parts in $[\tau_i^{\text{start}}\,..\,\tau_i^{\text{end}}]$ is taken into account when computing the total energy consumption in I_i.

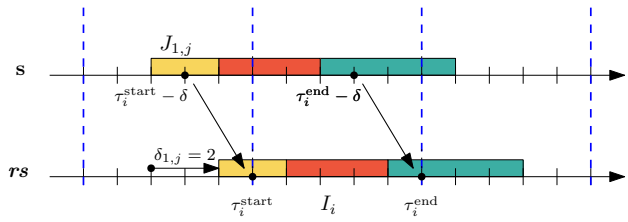

Figure 5.2: Illustration of strengthening cut (5.62) for the master problem.

5.3.1.2 Subproblem - Robustness Cuts

When integer baseline schedule $\mathbf{s}'$ is found for the master problem, it is checked for robustness, i.e., whether the energy consumption limits are satisfied in every realized schedule. The robustness check is performed by extracting the jobs permutation π_1' from $\mathbf{s}'$ and finding the optimal robust start times $\mathbf{s}^\star$ for π_1' using the procedure explained in Section 5.2. The next action depends on the makespan of both solutions. If the makespan is equal, no cut is generated. If makespan $\mathbf{s}'$ is larger than of $\mathbf{s}^\star$, the following simple no-good cut is generated

$$C_{\max} \leq \max_{J_{1,j} \in \mathcal{J}} s_{1,j}^\star + p_{1,j}\,. \tag{5.63}$$

On the other hand, if makespan $\mathbf{s}'$ is smaller than of $\mathbf{s}^\star$, then schedule $\mathbf{s}'$ is not robust, and no-good cuts based on *cutting intervals* are generated. In general, no-good cuts with cutting intervals have a form of

$$\sum_{J_{1,j} \in \mathcal{J}} \sum_{t \in T_j} x_{j,t} \leq n - 1\,, \tag{5.64}$$

where T_j is a cutting interval of job $J_{1,j}$ for schedule $\mathbf{s}'$. Informally, a no-good cut based on cutting intervals enforces that at least one job starts outside its cutting interval.

Simple cutting intervals that forbid one particular schedule $\mathbf{s}'$ can be defined as $\forall J_{1,j} \in \mathcal{J} : T_j^1 = \{s'_{1,j}\}$. However, such no-good cuts are weak since only one non-robust schedule is forbidden. In the following text, we introduce cutting intervals T_j^2 that exploit the knowledge of the optimal robust schedule $\mathbf{s}^\star$ for permutation π'_1. Informally, the cutting intervals are defined in such a way that the start times of the jobs in the baseline schedules having the same order as π'_1 are "pushed" towards $\mathbf{s}^\star$.

The cutting intervals T_j^2 depend on whether π'_1 is infeasible (i.e., no robust baseline schedule exists with permutation π'_1) or whether optimal robust schedule exists.

1. Infeasible permutation: this means that any baseline schedule having the same permutation of the jobs as π'_1 cannot be robust. Therefore, a no-good cut must be generated that "forbids" π'_1. Such no-good cut can be formulated by introducing disjunctive modeling into the master problem using big M method, which has poor relaxation. Alternatively, we can forbid only a subset of all the schedules having the same permutation of the jobs as π'_1 with a simpler constraint described below.

   Consider two jobs $J_{1,\pi'_1(\ell)}, J_{1,\pi'_1(\ell+1)}$ that are executed consecutively in $\mathbf{s}'$. Consider another baseline schedule $\mathbf{s}''$ in which the start time of $J_{1,\pi'_1(\ell+1)}$ is at least $s'_{1,\pi'_1(\ell+1)}$. To guarantee that $J_{1,\pi'_1(\ell)}$ starts before $J_{1,\pi'_1(\ell+1)}$ in $\mathbf{s}''$, the completion time of $J_{1,\pi'_1(\ell)}$ must be at most $s'_{1,\pi'_1(\ell+1)}$. Based on this idea, we can find the cutting interval for every job

   $$T^2_{\pi'_1(\ell)} = \begin{cases} [s'_{1,\pi'_1(\ell)} \,..\, \min\{s^{\max}_{1,\pi'_1(\ell)}, s'_{1,\pi'_1(\ell+1)} - p_{1,\pi'_1(\ell)}\}] & \ell \in [1\,..\,n-1] \\ [s'_{1,\pi'_1(\ell)} \,..\, s^{\max}_{1,\pi'_1(\ell)}] & \ell = n \end{cases} \tag{5.65}$$

   It is guaranteed that if all the jobs start anywhere in these cutting intervals, the order of jobs is the same as infeasible permutation π'_1.

2. Feasible permutation: since $\mathbf{s}' \neq \mathbf{s}^\star$ (otherwise $\mathbf{s}'$ would be robust), there exists a position $\bar{\ell}$ in permutation π'_1 such that

   $$\begin{aligned} &s'_{1,\pi'_1(\ell)} = s^\star_{1,\pi'_1(\ell)}\,, \quad \forall \ell \in [1\,..\,\bar{\ell}-1] \\ &s'_{1,\pi'_1(\bar{\ell})} \neq s^\star_{1,\pi'_1(\bar{\ell})}\,, \end{aligned} \tag{5.66}$$

   i.e., job $J_{1,\pi'_1(\bar{\ell})}$ is the earliest job with a different start time than the optimal robust one. One of the following two cases occurs.

   (a) $s'_{1,\pi'_1(\bar{\ell})} < s^\star_{1,\pi'_1(\bar{\ell})}$: consider Fig. 5.3 illustrating this case. The idea of this no-good cut is that if in any baseline schedule $\boldsymbol{s}$ the jobs on positions $[1\,..\,\bar{\ell}]$ are the same as in π'_1, then the start time of $J_{1,\pi'_1(\bar{\ell})}$ must be pushed towards $s^\star_{1,\pi'_1(\bar{\ell})}$ since any earlier baseline start time is not robust for $J_{1,\pi'_1(\bar{\ell})}$. However, special care needs to be taken for the case when job $J_{1,\pi'_1(\bar{\ell}+1)}$ starts before $s^\star_{1,\pi'_1(\bar{\ell})}$ in $\mathbf{s}'$; to make sure that no feasible

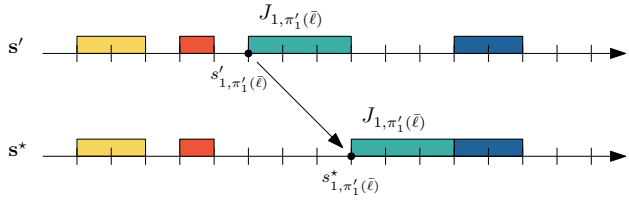

Figure 5.3: Cutting interval T_j^2 for a feasible permutation, case $s'_{1,\pi'_1(\bar{\ell})} < s^\star_{1,\pi'_1(\bar{\ell})}$.

baseline schedule in which job $J_{1,\pi'_1(\bar{\ell}+1)}$ starts before $s^\star_{1,\pi'_1(\bar{\ell})}$ is not cut out, the cutting interval of $J_{1,\pi'_1(\bar{\ell})}$ is bounded above by $s'_{1,\pi'_1(\bar{\ell}+1)}$.

The last thing to notice is that the permutation of the jobs on positions $\ell > \bar{\ell}$ is not important, since the earliest robust baseline start time of $s^\star_{1,\pi'_1(\bar{\ell})}$ is dependent only on the jobs on positions $\ell \leq \bar{\ell}$; see Eq. (5.27). Therefore, the no-good cut only needs to guarantee that they are not executed before $\pi'_1(\bar{\ell})$.

Based on this, the cutting intervals for this case can be formulated as

$$T^2_{\pi'_1(\ell)} = \begin{cases} [s'_{1,\pi'_1(\ell)} \,..\, \min\{s^{\max}_{1,\pi'_1(\ell)}, s'_{1,\pi'_1(\ell+1)} - p_{1,\pi'_1(\ell)}\}] & \ell \in [1\,..\,\bar{\ell}-1] \\ [s'_{1,\pi'_1(\bar{\ell})} \,..\, s^\star_{1,\pi'_1(\bar{\ell})} - 1] & \ell = \bar{\ell} \wedge \bar{\ell} = n \\ [s'_{1,\pi'_1(\bar{\ell})} \,..\, \min\{s^\star_{1,\pi'_1(\bar{\ell})} - 1, s'_{1,\pi'_1(\bar{\ell}+1)} - p_{1,\pi'_1(\bar{\ell})}\}] & \ell = \bar{\ell} \wedge \bar{\ell} < n \\ [\min\{s^\star_{1,\pi'_1(\bar{\ell})}, s'_{1,\pi'_1(\bar{\ell}+1)}\} \,..\, s^{\max}_{1,\pi'_1(\ell)}] & \ell \in [\bar{\ell}+1\,..\,n] \end{cases} \tag{5.67}$$

(b) $s'_{1,\pi'_1(\bar{\ell})} > s^\star_{1,\pi'_1(\bar{\ell})}$: this case is analogous to the previous one with the difference that job $J_{1,\pi'_1(\bar{\ell})}$ is being pushed to the left to $s^\star_{1,\pi'_1(\bar{\ell})}$

$$T^2_{\pi'_1(\ell)} = \begin{cases} [s'_{1,\pi'_1(\ell)} \,..\, \min\{s^{\max}_{1,\pi'_1(\ell)}, s'_{1,\pi'_1(\ell+1)} - p_{1,\pi'_1(\ell)}\}] & \ell \in [1\,..\,\bar{\ell}-1] \\ [s^\star_{1,\pi'_1(\bar{\ell})} + 1 \,..\, s'_{1,\pi'_1(\bar{\ell})}] & \ell = \bar{\ell} \\ [s'_{1,\pi'_1(\bar{\ell})} \,..\, s^{\max}_{1,\pi'_1(\ell)}] & \ell \in [\bar{\ell}+1\,..\,n] \end{cases} \tag{5.68}$$

Notice that even though the cutting intervals T_j^2 may cut out robust baseline schedules in which the jobs are not starting at their earliest robust start times, they do not cut out the optimal schedule $\mathbf{s}^\star$. Therefore, LBBD of the scheduling problem remains optimal.

5.3.2 Branch-and-Bound Based Algorithm

A Branch-and-Bound method is a common way how to devise an exact algorithm for combinatorial problems, which is based on a systematic search space enumeration. The method has two key ingredients: (i) branching, i.e., how the search space is partitioned and explored, and (ii) bounding, i.e., a mechanism for discarding unpromising parts of the solution space.

As with the LBBD approach, the proposed Branch-and-Bound based algorithm exploits the fact that the earliest robust start time of a job can be found in pseudo-polynomial time using Algorithm 8. The proposed algorithm constructs the jobs permutation one-by-one in a depth-first search manner. In each node of the search tree, the position of one job is fixed. The algorithm branches on every unscheduled job so far, and whenever a job is appended at the end of the current schedule, its start time is fixed to its earliest robust start time.

The lower bound for bounding is simple: we add the sum of the processing times of the unscheduled jobs to the Branch-and-Bound node's current makespan. Each node having a lower bound larger or equal to the makespan of the best feasible solution found so far is not explored further.

5.3.3 Adaptive Local Search Heuristic

Similarly, as in the non-robust scheduling, we propose a heuristic algorithm for the problem $1|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$ so that large instances can be handled. The heuristic is based on the adaptive local search framework from Section 3.2.

5.3.3.1 The Scheduling Operator

Given an all-jobs ordering, the scheduling operator computes the start times using Algorithm 8 on a one-by-one basis. This way, we obtain the optimal robust baseline start times for the given all-jobs ordering.

5.3.3.2 Finding the Initial Incumbent Solution: a Constructive Heuristic

The adaptive local search is initialized with an incumbent solution found by the following constructive heuristic algorithm. In each iteration, the algorithm selects the currently unscheduled job that minimizes the makespan, if scheduled at its earliest robust start time. The full pseudocode is given in Algorithm 9.

5.3.4 Experiments

The proposed methods for the $1|E^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$ scheduling problem are evaluated in this section. Since the are no existing algorithms for this scheduling problem in the related literature, the proposed methods are only compared to each other.

5.3.4.1 Benchmark Instances

The dataset for the experiments is generated similarly as in Section 4.5.4.1. The only two differences are in the setting of the maximum deviation and the size of the instances. To evaluate how instances' complexity changes with increasing maximum deviation, each generated instance is copied while differing only in the maximum deviation parameter $\delta^{\max} \in \{0, 2, 4\}$.

For each $n \in \{5, 10, 15, 50, 150\}$, $m \in \{1\}$, $D \in \{15, 60\}$, $\alpha_1 \in \{1, 2, 4\}$, $\alpha_2 \in \{0.8, 1.2, 1.6\}$, and $\delta^{\max} \in \{0, 2, 4\}$ we randomly generated 10 instances using

```
1  Function RobustConstructiveHeuristic():
2      /* The unscheduled jobs.  */
3      J' ← J;
4      /* The constructed all-jobs ordering.  */
5      π_1 ← ∅;
6      /* The current position into the all-jobs ordering.  */
7      ℓ ← 1;
8      /* The baseline start times.  */
9      s ← 0;
10     while J' ≠ ∅ do
11         /* Select the next unscheduled job that minimizes the current
              makespan.  */
12         (j*, C_max) ← (∅, ∞);
13         for J_{1,j} ∈ J' do
14             π_1(ℓ) ← j;
15             s_{1,j} ← ComputeEarliestRobustStartTime(π_1, J_{1,j}, s);
16             if s_{1,j} ≤ s^max_{1,j} ∧ s_{1,j} + p_{1,j} < C_max then
17                 (j*, C_max) ← (j, s_{1,j} + p_{1,j});
18         if j* = ∅ then
19             /* Cannot construct the initial order.  */
20             return ∅;
21         J' ← J' \ {J_{1,j*}};
22         π_1(ℓ) ← j*;
23         s_{1,j*} ← ComputeEarliestRobustStartTime(π_1, J_{1,j*}, s);
24         ℓ ← ℓ + 1;
25     return π_1;
```

Algorithm 9: Constructive heuristic for finding the initial incumbent solution for problem $1|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$.

the scheme above. Therefore, the generated dataset has $5 \cdot 1 \cdot 2 \cdot 3 \cdot 3 \cdot 3 \cdot 10 = 2700$ instances in total. Notice that the generated dataset contains only feasible instances due to the clamping of the jobs' power consumption and the heuristically-obtained horizons.

5.3.4.2 Experiment Setting and Remarks

The following abbreviations of the evaluated method are used: (i) R-CONS: constructive heuristic (see Algorithm 9), (ii) NR-ALS: adaptive local search algorithm (see Algorithm 5.3.3) initialized with the solution found by R-CONS, (iii) R-BAB: Branch-and-Bound algorithm (see Section 5.3.2) initialized with the solution found by R-CONS, and (iv) R-LBBD: Logic-based Benders Decomposition (see Section 5.3.1) initialized with the solution found by R-CONS.

The rest of the experiment setting is the same as in Section 4.5.4.2.

5.3.4.3 Experiment Results

The results for the experiment are shown in tables 5.1, 5.2, 5.3, and 5.4. From the results, we may conclude the following:

- The small instances ($n \in \{5, 10\}$) can all be solved by R-BAB to optimality. This is not surprising since R-BAB is basically an exhaustive search through all the permutations of the jobs with additional pruning. The number of permutations for $n = 10$ is not large, and can be easily enumerated even by inexpensive computers. This is supported by the results for $n \geq 15$, where R-BAB solves significantly fewer instances to optimality.

  For R-LBBD, the breaking point for solving the instances to optimality is around $n = 10$. The reason is that R-LBBD works primarily with the start times of the jobs, not their permutation.

- R-ALS performs well both on small and large instances. Except for instances $n = 5$, its median of the worse-to-best ratio is at most 0.7%, which is smaller than what R-BAB achieves 1.6% on the larger instances.

- R-ALS and R-BAB are not affected by α_1, α_2, D. However, α_1 and D have a slight influence on the R-LBBD. With increasing values of these parameters, R-LBBD has a lesser chance of solving an instance to optimality.

Table 5.1: The experiment results. Group parameters: D.

D	method	# opt. proofs	# best
15	R-CONS	0	121
	R-ALS	0	**1335**
	R-BAB	**554**	608
	R-LBBD	300	450
60	R-CONS	0	83
	R-ALS	0	**1317**
	R-BAB	**551**	585
	R-LBBD	234	324

Table 5.2: The experiment results. Group parameters: α_1.

α_1	method	# opt. proofs	# best
1	R-CONS	0	7
	R-ALS	0	**885**
	R-BAB	**360**	384
	R-LBBD	201	286
2	R-CONS	0	39
	R-ALS	0	**876**
	R-BAB	**362**	384
	R-LBBD	184	215
4	R-CONS	0	158
	R-ALS	0	**891**
	R-BAB	**383**	425
	R-LBBD	149	273

Table 5.3: The experiment results. Group parameters: α_2.

α_2	method	# opt. proofs	# best
0.8	R-CONS	0	68
	R-ALS	0	**873**
	R-BAB	**378**	396
	R-LBBD	177	238
1.2	R-CONS	0	73
	R-ALS	0	**891**
	R-BAB	**366**	397
	R-LBBD	178	266
1.6	R-CONS	0	63
	R-ALS	0	**888**
	R-BAB	**361**	400
	R-LBBD	179	270

Table 5.4: The experiment results. Group parameters: n.

n	method	# opt. proofs	# best	# no sol.	worse-to-best ratio median	worse-to-best ratio max
5	R-CONS	0	127	**0**	1.04865	1.48571
	R-ALS	0	535	**0**	1.03922	1.10294
	R-BAB	**540**	**540**	**0**	**BEST**	**BEST**
	R-LBBD	499	539	**0**	1.00413	1.00413
10	R-CONS	0	55	**0**	1.03468	1.36620
	R-ALS	0	501	**0**	1.00667	1.04478
	R-BAB	**540**	**540**	**0**	**BEST**	**BEST**
	R-LBBD	35	180	**0**	1.01881	1.23148
15	R-CONS	0	22	**0**	1.02607	1.31884
	R-ALS	0	**536**	**0**	**1.00226**	**1.05096**
	R-BAB	**24**	112	**0**	1.00808	1.10145
	R-LBBD	0	55	**0**	1.01818	1.31884
50	R-CONS	0	0	**0**	1.01898	1.29959
	R-ALS	0	**540**	**0**	**BEST**	**BEST**
	R-BAB	**1**	1	**0**	1.01589	1.24380
	R-LBBD	0	0	**0**	1.01776	1.29959
150	R-CONS	**0**	0	**0**	1.01213	1.15898
	R-ALS	**0**	**540**	**0**	**BEST**	**BEST**
	R-BAB	**0**	0	**0**	1.01161	1.15898
	R-LBBD	**0**	0	**0**	1.01213	1.15898

Chapter 6

Conclusion

This thesis investigated the production scheduling problem with energy consumption limits. The aim is to decrease the demand charge part of the electricity bill, which corresponds to penalty fees for violating the contracted limits. We studied this problem and developed multiple algorithms for deterministic and stochastic environments.

6.1 Fulfillment of the Goals

1. *Study the related energy-aware scheduling literature. Formalize the scheduling problem with the energy consumption limits in both deterministic and stochastic production environments.*

   The peak demand charge takes a considerable portion of the electricity bill. Despite that, most of the existing scheduling literature considers energy charge related objectives, e.g., energy cost minimization with real-time prices. Only a few works deal with the peak power demand charge, where the goal is to minimize the maximum peak power demand.

   In the Czech Republic, where we are based, the production companies must satisfy the contracted energy consumption limits. These limits are often violated if only traditional scheduling objectives are considered; thus, penalty fees must be paid. This was the case for the company producing tempered glass that motivated our research. Moreover, the company's production was subject to non-determinism in the production, so even if human schedulers designed feasible production schedules w.r.t. energy limits, random delays in start times of the jobs lead to violation of these limits anyway.

   The scheduling problem with the energy consumption limits can be seen as a generalization of a problem with the peak power limits. However, even fewer works deal with the energy limits topic. One of these works proposed a mixed-integer linear programming model, which has a limitation that a machine can process only one job within a metering interval. The closest scheduling problem is RCPSP/π; however, it does not deal with the robustness against uncertainty in the production, and the achieved algorithmic results are dependent on the length of the scheduling horizon.

2. *Formalize the scheduling problem with the energy consumption limits in both deterministic and stochastic production environments.*

   The scheduling problem was formalized in Chapter 2. The formalization captures multi-machine environments, possibly different energy limits in every metering interval, and delays of the jobs' start times.

3. *Investigate the proposed scheduling problem. Determine its computational complexity and design algorithms (both exact and heuristic) for solving the problem.*

The scheduling problem with the energy consumption limits was studied within a deterministic and stochastic environment. In the deterministic environment, hard problem variants were identified. An important theoretical result is that the problem with fixed ordering of the jobs, constant number of machines, the no-crossing assumption, and makespan as the objective can be solved in polynomial time (although this problem is in XP if the number of machines is variable). Due to the pessimistic complexity results, we designed an adaptive local search algorithm with a heuristic scheduling operator, alongside with CP and MILP-based models. We improved an existing MILP model from the literature to the related scheduling problem by finding a property that decreases the number of constraints in the model.

A significant result obtained within the stochastic environment is a pseudo-polynomial algorithm for the fixed order of the jobs on a single machine while minimizing a regular objective. This algorithm is then utilized in two algorithms (BAB and LBBD) and one heuristic algorithm.

4. *Propose a parametrized generator of the benchmark instances. Experimentally evaluate the performance of the developed algorithms.*

   We designed a generator of the benchmark instances parametrized by the energy-limit tightness, length of the jobs' processing times, metering intervals' length, maximum deviation, number of machines, and number of jobs.

   In the deterministic multi-machine environment, the proposed CP, MILP-based models, and the local search algorithm were compared against MILP models described in the existing literature that were adapted from the related scheduling problems. The experiment results show that our methods outperformed the existing ones from the literature. Among the proposed methods, CP performs the best up to 15 jobs per machine (up to 10 machines), whereas for 50 jobs the proposed heuristic local search algorithm outperforms CP model (except for the instances with 10 machines). For large scale instances (up to 350 jobs on each machine), the local search algorithm outperforms all the other methods.

   The algorithms designed for the robust scheduling problem were evaluated within a single-machine environment. For small instances (up to 10 jobs), BAB performed the best, although the proposed adaptive local search algorithm remained competitive. For the large instances (up to 150 jobs), the heuristic algorithm clearly outperforms the exact methods.

6.2 Future Work

The scheduling problem with energy consumption limits is far from being thoroughly researched. The following list presents possible directions, that would extend this work.

1. Investigation of the time complexity of the deterministic problem with a fixed order of the jobs and a fixed number of machines. A positive answer to this question could open a way for a more efficient scheduling operator.

2. Generalization of the achieved results for the robust scheduling problem to the environment with multiple machines. At the moment, the robust multi-machine scheduling problem can be tackled by splitting the available energy in every interval among the machines according to some heuristic rule. This would decompose the original multi-machine problem into many independent single-machine ones. The question remains on how to split the energy effectively.

3. The latest two trends in the energy-aware scheduling are real-time energy prices and power-saving states of the machines [7, 5]. By combing these aspects with the energy consumption limits, the resulting scheduling problem would consider all parts of the electricity bill, which could be very attractive for the energy-intensive manufacturing companies.

Bibliography

[1] Jose Batista Abikarram, Katie McConky, and Ruben 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: https://doi.org/10.1016/j.jclepro.2018.10.048.

[2] MohammadMohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. "Complexity analysis of energy-efficient single machine scheduling problems". In: *Operations Research Perspectives* 6 (2019), p. 100105. ISSN: 2214-7160. DOI: https://doi.org/10.1016/j.orp.2019.100105.

[3] Y. Alaouchiche, Y. Ouazene, and F. Yalaoui. "Economic and Energetic Performance Evaluation of Unreliable Production Lines: An Integrated Analytical Approach". In: *IEEE Access* 8 (2020), pp. 185330–185345. DOI: 10.1109/ACCESS.2020.3029761.

[4] R. Alvarez-Valdes et al. "GRASP and path relinking for project scheduling under partially renewable resources". In: *European Journal of Operational Research* 189.3 (2008), pp. 1153–1170. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2006.06.073.

[5] Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Constraints* (2020). ISSN: 1572-9354. DOI: https://doi.org/10.1007/s10601-020-09317-y.

[6] Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Proceedings of the 17th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*. 2020.

[7] 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.

[8] Konstantin Biel and Christoph H. Glock. "Systematic literature review of decision support models for energy-efficient production planning". In: *Computers & Industrial Engineering* 101 (2016), pp. 243–259. ISSN: 0360-8352.

[9] Michal Bouška et al. "Data-driven Algorithm for Scheduling with Total Tardiness". In: *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems (ICORES)*. INSTICC. SciTePress, 2020, pp. 59–68. ISBN: 978-989-758-396-4. DOI: 10.5220/0008915300590068.

[10] A.A.G. Bruzzone et al. "Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops". In: *CIRP Annals - Manufacturing Technology* 61.1 (2012), pp. 459–462. ISSN: 0007-8506. DOI: http://dx.doi.org/10.1016/j.cirp.2012.03.084.

[11] Ada Che, Yizeng Zeng, and Ke Lyu. "An efficient greedy insertion heuristic for energy-conscious single machine scheduling problem under time-of-use electricity tariffs". In: *Journal of Cleaner Production* 129 (2016), pp. 565–577. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2016.03.150.

[12] Ada Che, Shibohua Zhang, and Xueqi 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: https://doi.org/10.1016/j.jclepro.2017.04.018.

[13] Lu Chen, Jinfeng Wang, and Xianyang Xu. "An energy-efficient single machine scheduling problem with machine reliability constraints". In: *Computers & Industrial Engineering* 137 (2019), p. 106072. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106072.

[14] Azeddine Cheref, Christian Artigues, and Jean-Charles Billaut. "A new robust approach for a production scheduling and delivery routing problem". In: *IFAC-PapersOnLine* 49.12 (2016). 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, Troyes, France, 28—30 June 2016, pp. 886–891. ISSN: 2405-8963.

[15] Ywh-Leh Chou, Ju-Min Yang, and Cheng-Hung Wu. "An energy-aware scheduling algorithm under maximum power consumption constraints". In: *Journal of Manufacturing Systems* 57 (2020), pp. 182–197. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2020.09.004.

[16] Elvin Coban et al. "Robust Scheduling with logic-based Benders decomposition". In: *Operations Research Proceedings (OR 2014)*. Ed. by Marco Lübbecke et al. Springer International Publishing, 2016, pp. 99–105. ISBN: 978-3-319-28697-6.

[17] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer Publishing Company, Incorporated, 2013. ISBN: 1447155580, 9781447155584.

[18] Kan Fang et al. "Flow shop scheduling with peak power consumption constraints". In: *Annals of Operations Research* 206.1 (2013), pp. 115–145. ISSN: 0254-5330.

[19] 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.

[20] 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.

[21] Dragoljub Gajic et al. "Implementation of an integrated production and electricity optimization system in melt shop". In: *Journal of Cleaner Production* 155 (2017). Sustainable Development of Energy, Water and Environmental Systems, pp. 39–46. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2016.09.170.

[22] 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: https://doi.org/10.1016/j.jclepro.2018.10.289.

[23] R.L. Graham et al. "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". In: *Discrete Optimization II*. Ed. by P.L. Hammer, E.L. Johnson, and B.H. Korte. Vol. 5. Annals of Discrete Mathematics. Elsevier, 1979, pp. 287–326. DOI: https://doi.org/10.1016/S0167-5060(08)70356-X. URL: http://www.sciencedirect.com/science/article/pii/S016750600870356X.

[24] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2019. URL: http://www.gurobi.com.

[25] 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: http://dx.doi.org/10.1016/j.compchemeng.2015.02.004.

[26] A. Haït and C. Artigues. "A hybrid CP/MILP method for scheduling with energy costs". In: *European Journal of Industrial Engineering* 5.4 (2011), pp. 471–489.

[27] Alain Haït and Christian Artigues. "On electrical load tracking scheduling for a steel plant". In: *Computers & Chemical Engineering* 35.12 (2011), pp. 3044–3047. ISSN: 0098-1354.

[28] C. Herrmann et al. "Energy oriented simulation of manufacturing systems – Concept and application". In: *CIRP Annals* 60.1 (2011), pp. 45–48. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2011.03.127.

[29] Willy Herroelen. "Generating Robust Project Baseline Schedules". In: *Tutorials in Operations Research: OR Tools and Applications : Glimpses of Future Technologies*. TutORials in Operations Research. Institute for Operations Research and the Management Sciences (INFORMS), 2007. Chap. 8, pp. 124–144. ISBN: 9781877640223.

[30] J. N. Hooker. "Planning and Scheduling by logic-based Benders decomposition". In: *Operations Research* 55.3 (2007), pp. 588–602. ISSN: 0030-364X.

[31] IBM. *IBM CPLEX Optimization Studio Documentation*. 2019. URL: https://www.ibm.com/support/knowledgecenter/SSSA5P.

[32] Min Ji, Yong He, and T.C.E. Cheng. "Single-machine scheduling with periodic maintenance to minimize makespan". In: *Computers & Operations Research* 34.6 (2007). Part Special Issue: Odysseus 2003 Second International Workshop on Freight Transportation Logistics, pp. 1764–1770. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2005.05.034.

[33] H. Kellerer and V.A. Strusevich. "Scheduling parallel dedicated machines under a single non-shared resource". In: *European Journal of Operational Research* 147.2 (2003). Fuzzy Sets in Scheduling and Planning, pp. 345–364. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(02)00246-1.

[34] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. 5th. Springer Publishing Company, Incorporated, 2012. ISBN: 3642244874, 9783642244872.

[35] Wen-Yang Ku and J. Christopher Beck. "Mixed Integer Programming models for job shop scheduling: A computational analysis". In: *Computers & Operations Research* 73 (2016), pp. 165–173. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.04.006.

[36] 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. URL: https://doi.org/10.1007/s10601-018-9281-x.

[37] Olivier Lambrechts, Erik Demeulemeester, and Willy Herroelen. "Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities". In: *Journal of Scheduling* 11.2 (2007), pp. 121–136. ISSN: 1099-1425.

[38] Sifeng Lin, Gino J. Lim, and Jonathan F. Bard. "Benders decomposition and an IP-based heuristic for selecting IMRT treatment beam angles". In: *European Journal of Operational Research* 251.3 (2016), pp. 715–726. ISSN: 0377-2217.

[39] Hao Luo et al. "Hybrid flow shop scheduling considering machine electricity consumption cost". In: *International Journal of Production Economics* 146.2 (2013), pp. 423–439. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2013.01.028.

[40] Oussama Masmoudi, Xavier Delorme, and Paolo Gianessi. "Job-shop scheduling problem with energy consideration". In: *International Journal of Production Economics* 216 (2019), pp. 12–22. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2019.03.021.

[41] Lennart Merkert et al. "Scheduling and energy – Industrial challenges and opportunities". In: *Computers & Chemical Engineering* 72.0 (2015), pp. 183–198. ISSN: 0098-1354. DOI: http://dx.doi.org/10.1016/j.compchemeng.2014.05.024.

[42] István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "Algorithms for robust production scheduling with energy consumption limits". In: *Computers & Industrial Engineering* 112 (2017), pp. 391–408. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2017.08.011.

[43] István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "Energy-aware Robust Scheduling: Algorithm for Efficient Solution Space Search". In: *Booklet of Abstracts - The 29th Conference of the European Chapter on Combinatorial Optimization*. May 2016.

[44] István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "On parallel dedicated machines scheduling under energy consumption limit". In: *Computers & Industrial Engineering* (2021).

[45] István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "Robust scheduling for manufacturing with energy consumption limits". In: *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sept. 2016. ISBN: 978-1-5090-1314-2. DOI: http://dx.doi.org/10.1109/ETFA.2016.7733513.

[46] István Módos et al. "Adaptive online scheduling of tasks with anytime property on heterogeneous resources". In: *Computers & Operations Research* 76 (2016), pp. 95–117. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.06.008.

[47] István Módos et al. "Scheduling on Dedicated Machines with Energy Consumption Limit". In: *Proceedings of the 8th International Conference on Operations Research and Enterprise Systems (ICORES)*. INSTICC. SciTePress, 2019, pp. 53–62. ISBN: 978-989-758-352-0. DOI: 10.5220/0007307200530062.

[48] Bahman Naderi and Vahid Roshanaei. "Branch-Relax-and-Check: A tractable decomposition method for order acceptance and identical parallel machine scheduling". In: *European Journal of Operational Research* 286.3 (2020), pp. 811–827. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.10.014.

[49] Keisuke Nagasawa, Yuto Ikeda, and Takashi Irohara. "Robust flow shop scheduling with random processing times for reduction of peak power consumption". In: *Simulation Modelling Practice and Theory* 59 (2015), pp. 102–113. ISSN: 1569-190X.

[50] M. Nattaf et al. "A batch sizing and scheduling problem on parallel machines with different speeds, maintenance operations, setup times and energy costs". In: *2015 International Conference on Industrial Engineering and Systems Management (IESM)*. Oct. 2015, pp. 883–891. DOI: 10.1109/IESM.2015.7380260.

[51] Kristian Nolde and Manfred Morari. "Electrical load tracking scheduling of a steel plant". In: *Computers & Chemical Engineering* 34.11 (2010), pp. 1899–1903. ISSN: 0098-1354. DOI: http://dx.doi.org/10.1016/j.compchemeng.2010.01.011.

[52] Maroua Nouiri, Abdelghani Bekrar, and Damien Trentesaux. "An energy-efficient scheduling and rescheduling method for production and logistics systems". In: *International Journal of Production Research* 58.11 (2020), pp. 3263–3283. DOI: 10.1080/00207543.2019.1660826.

[53] Ruilin Pan et al. "Electrical load tracking scheduling of steel plants under time-of-use tariffs". In: *Computers & Industrial Engineering* 137 (2019), p. 106049. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106049.

[54] Paz Perez-Gonzalez and Jose M. Framinan. "Single machine scheduling with periodic machine availability". In: *Computers & Industrial Engineering* 123 (2018), pp. 180–188. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2018.06.025.

[55] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. 5th. Springer Publishing Company, 2016. ISBN: 978-3-319-26580-3.

[56] Stathis Plitsos et al. “Energy-aware decision support for production scheduling”. In: *Decision Support Systems* 93 (2017), pp. 88–97. ISSN: 0167-9236.

[57] Nicola Policella et al. “From Precedence Constraint Posting to Partial Order Schedules: A CSP Approach to Robust Scheduling”. In: *AI Commun.* 20.3 (Aug. 2007), pp. 163–180. ISSN: 0921-7126.

[58] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. New York, NY, USA: Elsevier Science Inc., 2006. ISBN: 0444527265.

[59] Saeed Rubaiee and Mehmet Bayram Yildirim. “An energy-aware multiobjective ant colony algorithm to minimize total completion time and energy cost on a single-machine preemptive scheduling”. In: *Computers & Industrial Engineering* 127 (2019), pp. 240–252. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2018.12.020.

[60] Timo Schudeleit et al. “The Total Energy Efficiency Index for machine tools”. In: *Energy* 102 (2016), pp. 682–693. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2016.02.126.

[61] Abhay Sharma, Fu Zhao, and John W. Sutherland. “Econological scheduling of a manufacturing enterprise operating under a time-of-use electricity tariff”. In: *Journal of Cleaner Production* 108 (2015), pp. 256–270. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2015.06.002.

[62] 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: https://doi.org/10.1016/j.jclepro.2013.12.024.

[63] Přemysl Šůcha et al. “Online Scheduling System for Server Based Personnel Rostering Applications”. In: *The 10th International Conference of the Practice and Theory of Automated Timetabling*. Aug. 2014, pp. 561–564. ISBN: 978-0-9929984-0-0.

[64] Giorgi Tadumadze, Simon Emde, and Heiko Diefenbach. “Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines”. In: *OR Spectrum* 42 (2020), pp. 461–497. ISSN: 1436-6304. DOI: https://doi.org/10.1007/s00291-020-00586-w.

[65] Junbo Tuo, Fei Liu, and Peiji Liu. “Key performance indicators for assessing inherent energy performance of machine tools in industries”. In: *International Journal of Production Research* 57.6 (2019), pp. 1811–1824. DOI: 10.1080/00207543.2018.1508904.

[66] U.S. Department of Energy. *Turn Motors Off When Not in Use*. https://www.energy.gov/sites/prod/files/2014/04/f15/motor_tip_sheet10.pdf. Accessed: 2019-06-02.

[67] Energetický regulační úřad. *Energetický regulační věstník 10/2018*. https://portal.gov.cz/obcan/vestniky/eeuaau7/626101795.pdf (in Czech). Accessed August 25, 2019.

[68] Jing-jing Wang and Ling Wang. "Decoding methods for the flow shop scheduling with peak power consumption constraints". In: *International Journal of Production Research* 57.10 (2019), pp. 3200–3218. DOI: 10.1080/00207543.2019.1571252.

[69] Ye Yang et al. "Robust optimization for integrated scrap steel charge considering uncertain metal elements concentrations and production scheduling under time-of-use electricity tariff". In: *Journal of Cleaner Production* 176 (2018), pp. 800–812. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2017.12.094.

[70] YiZeng Zeng, Ada Che, and Xueqi Wu. "Bi-objective scheduling on uniform parallel machines considering electricity cost". In: *Engineering Optimization* 50.1 (2018), pp. 19–36. DOI: 10.1080/0305215X.2017.1296437.

[71] Shengnan Zhao, Ignacio E. Grossmann, and Lixin Tang. "Integrated scheduling of rolling sector in steel production with consideration of energy consumption under time-of-use electricity prices". In: *Computers & Chemical Engineering* 111 (2018), pp. 55–65. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2017.12.018.

Chapter

Nomenclature

A

Problem Statement

M_k	Machine with index k.
$\mathcal{M}$	Set of machines.
$J_{k,j}$	Job with index j on machine M_k.
$\mathcal{J}_k^M$	Set of jobs on machine with index k.
$\mathcal{J}$	Set of all jobs over all machines.
n_k	Number of jobs on machine M_k.
n	Number of all jobs over all machines.
$p_{k,j}$	The processing time of job $J_{k,j}$.
$P_{k,j}$	The power consumption of job $J_{k,j}$.
H	The scheduling horizon.
I_i	Metering interval with index i.
$\mathcal{I}$	Set of all metering intervals.
D	Length of the metering intervals.
τ_i^{start}	The start of metering interval I_i.
τ_i^{end}	The end of metering interval I_i.
$E_i^{\max}$	The maximum energy limit in metering interval I_i.
$E^{\max}$	The maximum energy limit (same for every metering interval).
$s_{k,j}$	The baseline start time of job $J_{k,j}$.
$\mathbf{s}$	The baseline start times.
$\delta^{\max}$	The maximum deviation.
$\boldsymbol{\delta}^{(q)}$	The deviation scenario with index q.
$\delta_{k,j}^{(q)}$	The deviation of job $J_{k,j}$ in deviation scenario $\boldsymbol{\delta}^{(q)}$.
$\boldsymbol{\delta}^{(1)}$	Zero deviation scenario.
Δ	The set of all deviation scenarios.
π_k	Fixed order on machine M_k.
$\pi_k(\ell)$	The index of a job on machine M_k on ℓ-th position in the fixed order.

$RS(\boldsymbol{s}, \boldsymbol{\delta})_{k,j}$	The realized start time of the job with index j on machine M_k for baseline start times $\boldsymbol{s}$ and deviation scenario $\boldsymbol{\delta}$.
$\boldsymbol{rs}$	The realized start times.
$rs_{k,j}$	The realized start time of job $J_{k,j}$.
$s_{k,j}^{\max}$	The maximum baseline start time of job $J_{k,j}$.
$\text{Overlap}(s_{k,j}, p_{k,j}, i)$	The length of the overlap of starting at time $s_{k,j}$ and having processing time $p_{k,j}$ with metering interval I_i.
f	Any regular objective.
$C_{\max}$	The makespan of a schedule.
PDm	Parallel dedicated machines.

General Adaptive Local Search

λ	All-jobs ordering.
λ^{best}	All-jobs ordering of the incumbent solution.
$\mathbf{s}^{\text{best}}$	The jobs' start times of the incumbent solution.
λ^{nh}	All-jobs ordering of a neighbor.
$\mathbf{s}^{\text{nh}}$	The jobs' start times of a neighbor.
nhgen	The selected neighborhood generator.

3-PARTITION problem

A	The set of integers to partition.
a_j	An element of A indexed by j.
A_i	A subset of integers indexed by i.
q	The number of subsets.
B	The sum of the integers in any subset A_i.

PARTITION problem

A	The set of integers to partition.
a_k	An element of A indexed by k.
A'	A subset of integers.

Problem $\mathrm{PD}m|\pi_k, \text{no-crossing}, E_i^{\max} = E^{\max}|C_{\max}$

$(\boldsymbol{l}, \boldsymbol{r})$	A slice.
Ψ	Set of all slices.
V	Set of vertices.
E	Set of edges.
$w(v_{\boldsymbol{l},\boldsymbol{r}}, v_{\boldsymbol{l}',\boldsymbol{r}'})$	The weight for edge $(v_{\boldsymbol{l},\boldsymbol{r}}, v_{\boldsymbol{l}',\boldsymbol{r}'})$.
G	A graph.
v^{s}	Source vertex.
v^{t}	Target vertex.

Problem $\mathrm{PD}m|E_i^{\max}|C_{\max}$: CP model

$x_{k,j}$	An interval variable denoting where job $J_{k,j}$ is scheduled.

Problem $\mathrm{PD}m|E_i^{\max}|C_{\max}$: disjunctive MILP model

$C_{\max}$	Variable for the makespan.
$s_{k,j}$	The start time of job $J_{k,j}$.
$x_{k,j,j'}$	Variable denoting whether job $J_{k,j}$ is scheduled before job $J_{k,j'}$.
$d_{k,j,i}$	The length of the overlap between job $J_{k,j}$ and metering interval I_i.
$z_{k,j,i}^{\mathrm{start}}$	Variable that models $s_{k,j} \in [0, \tau_i^{\mathrm{end}} - 1]$.
$z_{k,j,i}^{\mathrm{end}}$	Variable that models $s_{k,j} + p_{k,j} \in [0, \tau_i^{\mathrm{start}}]$.

Problem $\mathrm{PD}m|E_i^{\max}|C_{\max}$: time-indexed MILP model

$C_{\max}$	Variable for the makespan.
t	A time.
$x_{k,j,t}$	Variable denoting whether job $J_{k,j}$ starts at time t.

Problem PD$m|E_i^{\max}|C_{\max}$: implicit MILP model

$C_{\max}$	Variable for the makespan.
$\Omega_{k,j}$	The maximum number of consecutive metering intervals that can have non-zero overlap with job $J_{k,j}$.
$\mathcal{J}^{\text{span}}$	Set of spannable jobs.
$d_{k,j,i}$	The length of the overlap between job $J_{k,j}$ and metering interval I_i.
$x^s_{k,j,i}$	Variable for whether spannable job $J_{k,j}$ starts in metering interval I_i.
$x^+_{k,j,i}$	Variable for whether job $J_{k,j}$ has non-zero overlap in metering interval I_i.
$y^+_{k,i}$	Variable for whether some job has non-zero overlap with metering interval I_i on machine M_k.
$\boldsymbol{d}$	All the overlap variables.
$\text{est}_{k,i}$	The earliest start time on machine M_k and in metering interval I_i.

Problem PD$m|E_i^{\max}|C_{\max}$: iterative implicit MILP model

$i^{\max}$	The index of the last metering interval within the considered scheduling horizon.

Problem PD$m|E_i^{\max}|C_{\max}$: adaptive local search

e_i	The consumed energy in metering interval $\|I_i$.
c_k	The earliest available time for machine M_k.
ℓ	Position in all-jobs ordering.
maxPossibleOverlap	The maximum possible overlap of a job with a metering interval.
priorityRules	The ordered list of priority rules.
$\mathcal{J}'$	The unscheduled jobs.

Problem PD$m|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$: CP model

$x^{(q)}_{k,j}$	The time interval within the realized schedule corresponding to deviation situation $\boldsymbol{\delta}^{(q)}$ in which job $J_{k,j}$ is scheduled.

$y_k^{(q)}$	The sequence on machine M_k in realized schedule corresponding to deviation situation $\boldsymbol{\delta}^{(q)}$.

Problem $1|\pi_1, E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|f$

$LS(\boldsymbol{s})_{1,j}$	The latest start time of job $J_{1,j}$ for baseline start times $\boldsymbol{s}$.
$\boldsymbol{ls}$	The latest start times.
t	A time.
$RSS(\boldsymbol{s}, \bar{\ell}, t)_{1,\pi_1(\ell)}$	The right-shift start time of job $J_{1,\pi_1(\ell)}$ for jobs' order π_1 corresponding to baseline start times $\boldsymbol{s}$, time t, and fixed job $J_{1,\pi_1(\bar{\ell})}$.
$\boldsymbol{rss}$	The right-shift start times.
$\mathbf{s}^\star$	The optimal and robust baseline start times.
$\boldsymbol{rss}^\star$	The right-shift start times obtained from $\mathbf{s}^\star$.

Problem $1|\pi_1, E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|f$: scheduling operator

energyLimitViolated	Value indicating whether the energy limit was violated in the tested metering interval.
$maxPossibleOverlap_i$	The maximum possible overlap in metering interval I_i.
$maxOverlap_i$	The maximum overlap of a specific job in metering interval I_i.

Problem $1|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$: LBBD

t	A time.
$x_{j,t}$	Variable denoting whether job $J_{1,j}$ starts at time t.
$C_{\max}$	Variable for the makespan.
E_t^{time}	Energy consumed in time t.
T_j^1	Cutting interval of type 1 for job $J_{1,j}$.
T_j^2	Cutting interval of type 2 for job $J_{1,j}$.

Problem $1|E_i^{\max}, \delta_{k,j}^{\max} = \delta^{\max}|C_{\max}$: adaptive local search

$\mathcal{J}'$	The set of unscheduled jobs.

π_1	The constructed all-jobs ordering.
ℓ	The current position into π_1.
$j^\star$	The unscheduled job that minimizes the makespan.

Experiments

α_1	The multiplier of metering intervals' length used for sampling the processing times.
α_2	The energy limit tightness.
BEST	The best-found solution objective for the given instance.
UB	A solution objective.

Chapter B

Curriculum Vitae

István Módos was born in Győr, Hungary, in 1990. He received his bachelor's degree in 2012 in the Open Informatics study program at Czech Technical University in Prague. He continued with the same study program for his master's studies, which he completed with honors in 2014. A half-year later, he started his journey towards a Ph.D. at the Department of Control Engineering on the topic of production scheduling with energy consumption limits.

István has published two papers in impacted journals during his research years, where he is listed as the first author. His third paper is currently under review in Computers & Industrial Engineering. Moreover, he collaborated with Ondřej Benedikt on a related topic of production scheduling with power-saving states and real-time energy prices, where Ondřej and István proposed an exact algorithm that significantly outperforms the state-of-the-art method. Their contributions were published in a journal paper, which won the best student paper award at the CPAIOR2020 conference.

István participated in two research projects: (i) Design, Monitoring, and Operation of Adaptive Networked Embedded Systems and (ii) Centrum aplikované kybernetiky. Moreover, he collaborated on several projects for industrial partners, for example, on an energy consumption minimization for Škoda Auto in Vrchlabí. Besides research work, he worked in a software company that develops advanced planning and scheduling system, where he utilized the obtained academic knowledge.

With respect to the teaching activities, he led the labs for Combinatorial Optimization and Parallel Algorithms courses. Moreover, he successfully supervised two master students, David Král and Kiryl Kalodkin.

István Módos
Prague, January 2021

Chapter C

List of Author's Publications

Publications in Journals with Impact Factor

István Módos, Přemysl Šůcha, Roman Václavík, Jan Smejkal, and Zdeněk Hanzálek. "Adaptive online scheduling of tasks with anytime property on heterogeneous resources". In: *Computers & Operations Research* 76 (2016), pp. 95–117. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.06.008
Coauthorship 20%, indexed in Web of Science and Scopus, 1 citation (Web of Science)

István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "Algorithms for robust production scheduling with energy consumption limits". In: *Computers & Industrial Engineering* 112 (2017), pp. 391–408. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2017.08.011
Coauthorship 33.3%, indexed in Web of Science and Scopus, 8 citations (Web of Science)

István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "On parallel dedicated machines scheduling under energy consumption limit". In: *Computers & Industrial Engineering* (2021)
Coauthorship 33.3%, under review (the decision will be known soon)

Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Constraints* (2020). ISSN: 1572-9354. DOI: https://doi.org/10.1007/s10601-020-09317-y
Coauthorship 33.3%

Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "A Fast Branch-and-Bound Algorithm for Scheduling with Variable Energy Pricing and Power-Saving States" (2020)
Coauthorship 33.3%, soon to be submitted to European Journal of Operational Research

International Conferences and Workshops

István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "Robust scheduling for manufacturing with energy consumption limits". In: *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sept. 2016. ISBN: 978-1-5090-1314-2. DOI: http://dx.doi.org/10.1109/ETFA.2016.7733513
Coauthorship 33.3%, indexed in Web of Science and Scopus

István Módos, Kiryl Kalodkin, Přemysl Šůcha, and Zdeněk Hanzálek. "Scheduling on Dedicated Machines with Energy Consumption Limit". In: *Proceedings of the 8th International Conference on Operations Research and Enterprise Systems (ICORES)*. INSTICC. SciTePress, 2019, pp. 53–62. ISBN: 978-989-758-352-0. DOI: 10.5220/0007307200530062
Coauthorship 25%, indexed in Scopus

István Módos, Přemysl Šůcha, and Zdeněk Hanzálek. "Energy-aware Robust Scheduling: Algorithm for Efficient Solution Space Search". In: *Booklet of Abstracts - The 29th Conference of the European Chapter on Combinatorial Optimization*. May 2016
Coauthorship 33.3%

Ondřej Benedikt, István Módos, and Zdeněk Hanzálek. "Power of Pre-Processing: Production Scheduling with Variable Energy Pricing and Power-Saving States". In: *Proceedings of the 17th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*. 2020
Coauthorship 33.3%, best student paper award, invitation to fast-track issue of Constraints journal

Ondřej Benedikt, Přemysl Šůcha, István Móds, 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
Coauthorship 20%, indexed in Web of Science and Scopus

Michal Bouška, Antonín Novák, Přemysl Šůcha, István Módos, and Zdeněk Hanzálek. "Data-driven Algorithm for Scheduling with Total Tardiness". In: *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems (ICORES)*. INSTICC. SciTePress, 2020, pp. 59–68. ISBN: 978-989-758-396-4. DOI: 10.5220/0008915300590068
Coauthorship 20%, indexed in Web of Science and Scopus

Přemysl Šůcha, István Módos, Roman Václavík, Jan Smejkal, and Zdeněk Hanzálek. "Online Scheduling System for Server Based Personnel Rostering Applications". In: *The 10th International Conference of the Practice and Theory of Automated Timetabling*. Aug. 2014, pp. 561–564. ISBN: 978-0-9929984-0-0
Coauthorship 20%

István Módos
Prague, January 2021

This thesis is focused on the scheduling problem with energy consumption limits. The problem was studied in both deterministic and stochastic environments.

The main contributions in the deterministic environment are:

- A complexity study of different variants of this scheduling problem.
- A polynomial algorithm finding the shortest schedule satisfying the maximum energy consumption limits for a fixed permutation of the jobs on a fixed number of machines with the assumption that the jobs cannot cross multiple metering intervals.
- An efficient CP model for the multi-machine scheduling under maximum energy consumption limits.
- Improvement of the existing ILP model from the literature.
- An adaptive local search algorithm with a heuristic scheduling operator for the large instances.

The main contributions in the stochastic environment are:

- A pseudo-polynomial algorithm for a single machine scheduling problem with a fixed permutation of the jobs, which finds an optimal schedule w.r.t. any regular objective function and which does not violate the energy consumption limits even if the jobs are delayed.
- The algorithm mentioned above for the fixed permutation of the jobs is exploited in exact methods (Branch-and-Bound and Logic-based Benders Decomposition) for finding the optimal and robust permutation of the jobs.
- An adaptive local search algorithm with a heuristic scheduling operator for the large instances.