Bernhard Primas BSc

New Scheduling Models for Cloud Computing Systems: Analysis, Algorithms and Applications

MASTER THESIS

written to obtain the academic degree of a

Diplom-Ingenieur

Master programme Technical Mathematics: Operations Research and Statistics

presented to

Graz University of Technology

Supervisors:
Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Bettina Klinz
Dr. Natasha Shakhlevich

Institute of Optimization and Discrete Mathematics (Math B)

Graz, July 2015

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Datum/Date	Unterschrift/Signature

Acknowledgement

There are so many people I would like to thank. First I would like to express my sincere gratitude to my supervisor Dr. Bettina Klinz. She has supported me with useful material and comments through the whole learning process of my thesis. Furthermore, thanks to her selfless commitment and perseverance, she has enabled me to go on a five-month academic research stay to the University of Leeds which has definitely opened many promising doors for my future life. By my opinion, this fact cannot be appreciated and emphasised enough.

My sincere gratitude goes, of course, also to Dr. Natasha Shakhlevich, who has been my external supervisor during my time in Leeds. I thank her for all her support, motivation and patience. Many thanks also for all the time she took for me and for all the experiences I have made in England. I also thank her very much for all her effort and time in order to get my research stay at the University of Leeds organised.

Furthermore I would like to warmly thank Dr. Peter Garraghan, who supported this thesis with his knowledge in cloud computing. An interdisciplinary thesis as the present thesis would not have been possible without his contribution. I am very grateful for all the time and all the interesting meetings and discussions with him and Dr. Natasha Shakhlevich.

My sincere thanks also goes to Prof. Behrndt, the chairman of the audit committee, and Dr. Eranda Dragoti-Çela, the second examiner in the audit committee.

I also thank all my fellow students for all the common time and support. In particular I thank Elisabeth Gaar for everything and especially for the fruitful cooperation in solving exercise sheets and preparing for exams during our studies.

Last but not least I would like to thank my mother, my father and my brother as well as all my friends for their comprehensive support and motivation during my education and in my private life. I do not go too far by saying that without their support I would not be where I am today.

Abstract

Scientific research in the area of mathematically orientated scheduling problems has started about sixty years ago by seminal papers by Johnson and Bellman. In the last decade developing scheduling algorithms for distributed computing systems has become a centre of interest for practical applications of scheduling problems. However, most of the applied scheduling models for distributed computing systems do not fully make use of existing mathematically orientated scheduling theory developed within the last sixty years. The goal of this thesis is to help bridging the gap between mathematically orientated scheduling theory and applied scheduling models for distributed computed systems.

In the thesis we first give a definition of cloud computing which is a specialised form of distributed computing. Then we present some important features that appear in cloud computing that are worth to be modelled in mathematically orientated scheduling models. These features include objectives like the energy consumption or the reliability as well as restrictions like the immediate start condition or the consideration of several different computational resources. We continue with a short review of the literature on cloud computing where we highlight differences in the literature on applied scheduling models and mathematically orientated scheduling models, respectively. Thereafter we present a new energy model in which the energy consumption is calculated based on the cores of a processor. Then we give an overview of existing literature on energy efficient scheduling algorithms. Finally, we propose a scheduling model which combines the immediate start condition with the objectives of minimising the total flow time and the energy consumption. For the special cases with a single processor or with equal-work jobs algorithms with running time $\mathcal{O}(n \log n)$ are suggested where n denotes the number of jobs. For the general case, the complexity status of minimising the total flow time while respecting an upper limit on the consumed energy remained open.

Kurzfassung

Wissenschaftliche Forschung im Bereich von mathematisch orientierten Schedulingproblemen wird seit den anfänglichen Arbeiten von Johnson und Bellman vor etwa sechzig Jahren betrieben. Innerhalb der letzten zehn Jahre rückte die Entwicklung von Scheduling Modellen für verteilte Rechensysteme in den Mittelpunkt des Interesses für praktische Anwendungen von Schedulingproblemen. Allerdings verwenden derzeit die meisten angewandten Schedulingmodelle für verteilte Rechensysteme nicht die gesamte zur Verfügung stehende mathematisch orientierte Schedulingtheorie, die innerhalb der letzten sechzig Jahre entwickelt wurde. Ziel dieser Arbeit ist es, einen Beitrag zu leisten angewandte Schedulingmodelle für verteilte Rechensysteme und mathematisch orientierte Schedulingtheorie näher zusammenzubringen.

Wir starten mit einer Definition von Cloud Computing, einem Spezialfall von verteiltem Rechnen. Danach führen wir einige wichtige Eigenschaften von Cloud Computing-Systemen an, für die eine Miteinbeziehung in mathematische Modelle wichtig erscheint. Beispiele für solche Eigenschaften sind etwa Zielfunktionen wie der Energieverbrauch oder die Zuverlässigkeit, als auch Restriktionen wie die unverzügliche Startbedingung oder die Berücksichtigung mehrerer verschiedener rechnerischer Ressourcen. Danach befassen wir uns mit Unterschieden zwischen Literatur über angewandte Schedulingmodelle bzw. über mathematisch orientierte Schedulingtheorie. Wir präsentieren ein neues Energiemodell, in dem der Energieverbrauch eines Prozessors auf Basis des Energiebedarfs der Kerne des Prozessors berechnet wird. Dann geben wir einen kurzen Literaturüberblick über existierende Literatur von energieeffizienten Schedulingalgorithmen. Schließlich schlagen wir ein neues Schedulingmodell vor, das die unverzügliche Startbedingung erfüllt und als Zielfunktionen den Total Flow sowie den Energieverbrauch berücksichtigt. Für den Spezialfall eines Einmaschinenproblems sowie für den Spezialfall für Jobs mit gleichem Bearbeitungsvolumen werden Algorithmen mit Laufzeit $\mathcal{O}(n \log n)$ angegeben wobei n die Anzahl der Jobs bezeichnet. Der Komplexitätsstatus des allgemeinen Problems, in dem der Total Flow minimiert werden soll sodass eine obere Schranke des Energieverbrauchs nicht überschritten wird, bleibt offen.

Contents

1 Introduction

Scheduling theory is about assigning limited resources to tasks over time with the goal of optimising one or more objective functions. Scientific research in this area started about sixty years ago by seminal papers by Johnson [28] and Bellman [13]. The first main applications of scheduling problems have stemmed from the area of manufacturing. For example machines in a workshop (resources) should be assigned to operations in a workshop (tasks) such that the time to complete all tasks is minimised. In the last decades a lot of research concerning scheduling theory has been done including topics as complexity theory, integer programming and stochastic programming.

By the time personal computers started to permeate, new applications in service arose. Managing take-offs and planning landings at an airport or timetabling are examples of applications in service. See for example the textbook of Pinedo [36] for scheduling with applications in both manufacturing and services.

With the rapid development of computers in the last decades many new application areas arose. One of them is scheduling in distributed computing systems. An example of a scheduling problem in this area is to assign processors to tasks to minimise the total completion time of the tasks or to minimise the required energy consumption of the processors.

To face the challenge of scheduling tasks in distributed computing systems, a lot of effort has been put into research and a huge number of new scheduling algorithms has been developed. However, there is still a big gap between research focussing on theoretical and mathematically orientated scheduling theory and research on applied scheduling models in distributed computing systems. This means that on the one hand researchers who work in the area of applied scheduling do not use all the theory developed by researchers in the area of theoretical scheduling models and on the other hand only very few theoretically orientated scheduling researchers work on scheduling models for distributed computing systems. The textbook of Drozdowski [21] is an example of how mathematically orientated scheduling theory can be used for scheduling models in parallel and distributed computing systems.

The aim of the present thesis is to make an attempt to help bridging the gap between the theoretical and the applied scheduling research. To that end we study and analyse several forms of the energy consumption as an objective function and deal with some features of scheduling problems in distributed computing. Afterwards classical mathematical scheduling problems are extended by these features and are then studied.

This thesis is organised as follows. In Section 2, we introduce the concept of *cloud computing*, which is a specialised form of distributed computing. We restrict our considerations to cloud computing in this thesis, which has become a very important area of distributed computing in the last years.

In Section 3 we provide some basic mathematical definitions which will be needed later in the thesis.

Section 4 gives a short overview of objective functions, job characteristics and other

restrictions which may appear in a model of a cloud computing system. We highlight their role in mathematical scheduling theory and cloud computing. Finally the commonly used $\alpha \mid \beta \mid \gamma$ notation is extended for our purposes.

In Section 5 we present a short review of differences between the literature in theoretical and in applied scheduling theory, respectively. We highlight possible difficulties that mathematicians have to face when studying literature on applied scheduling models for distributed computing systems. Finally we consider two simple energy models from the literature and point out that these models behave contradictory.

Having observed a contradicting behaviour with simple energy models, a more complex energy model is proposed in Section 6. In this model, the calculation of the energy consumption of a processor is based on the energy consumption of the cores of the processor. For the special case of identical jobs, the trade-off between the energy consumption and the total flow is studied. Furthermore we point out why this energy model might be to complex to be applied in distributed computing systems.

The rest of the thesis is focussed on scheduling problems for which both the energy consumption and the total flow time should be optimised. Section 7 gives an overview over existing mathematical literature.

In Section 8 we consider several scheduling models that focus on on-demand scheduling. On-demand scheduling is modelled by the *immediate start* condition, which forces each job to start its execution immediately when it becomes available. Considered scheduling problems are bi-criteria optimisation problems with the energy consumption and the total flow as objective functions. We study both the problem version of determining all Pareto optimal schedules as well as the problem version of minimising the total flow such that a given upper limit on the consumed energy is not exceeded. For the special cases with a single processor or with equal-work jobs algorithms with running time $\mathcal{O}(n \log n)$ are suggested for both problem versions where n denotes the number of jobs. The complexity status for the general case remained open.

Finally, in Section 9 we present our conclusions and some open problems.

2 Modelling: New Features in Distributed Computing

In this thesis we will focus uniquely on *cloud computing*, which is an important specialised form of distributed computing. Cloud computing had a major influence on both business and academic fields in the last few years and is therefore worth to be intensively studied. In this section we provide a description of those components of a cloud computing system, which are relevant for this thesis. It turns out that in the setting of distributed computing scheduling problems arise which differ from traditional mathematical scheduling problems.

2.1 About Cloud Computing

Cloud computing is still an emerging field. Its definition, attributes and characteristics will evolve over time. The currently most accepted definition of cloud computing systems was given by the National Institute of Standards and Technology (NIST) in [33]. It reads as follows:

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

In the following the essential characteristics, service models and deployment models mentioned above are explained in more detail. The explanations are again directly taken from the NIST document [33] as for definitions of this type rephrasing does not make sense.

Essential Characteristics (from [33]):

> "*On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
>
> *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g. mobile phones, tablets, laptops, and workstations).
>
> *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction

(e.g. country, state, or datacentre). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service. Cloud systems automatically control and optimise resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g. storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilised service."

Service Models (from [33]):

"*Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g. web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

Platform as a Service (PaaS). The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g. host firewalls)."

Deployment Models (from [33]):

"*Private cloud.* The cloud infrastructure is provisioned for exclusive use by a single organisation comprising multiple consumers (e.g. business units). It may be owned, managed, and operated by the organisation, a third party, or

some combination of them, and it may exist on or off premises.

Community cloud. The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organisations that have shared concerns (e.g. mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organisations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public cloud. The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organisation, or some combination of them. It exists on the premises of the cloud provider.

Hybrid cloud. The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardised or proprietary technology that enables data and application portability (e.g. cloud bursting for load balancing between clouds)."

The three service models differ in their degree of abstraction and their level of control a customer has. The IaaS service model is seen as the least abstract model but offers the highest level of control to customers. On the other hand, the SaaS service model is seen as the most abstract model, but offers the lowest level of control to customers. These aspects can be visualised in the so-called *cloud computing stack* as depicted in Figure 1 .

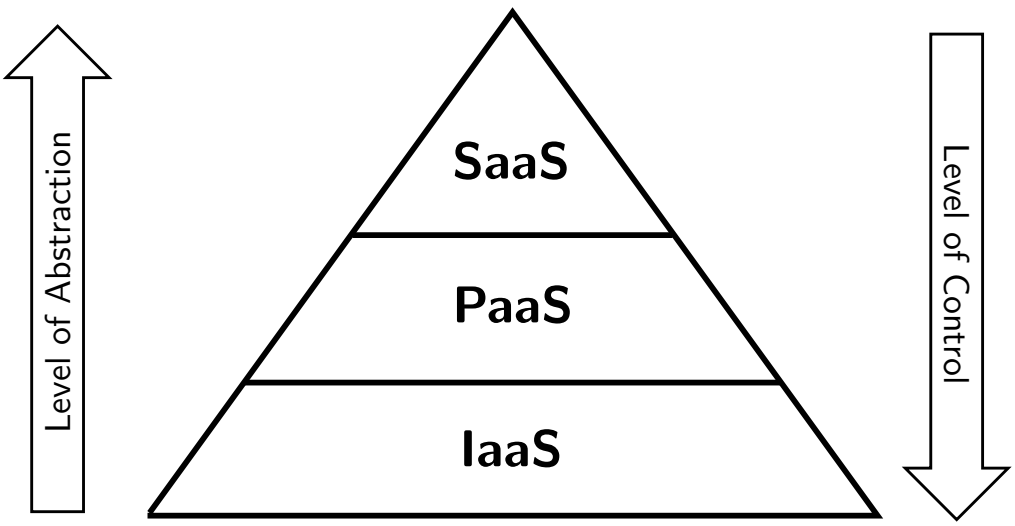

Figure 1: Cloud computing stack

Customers get access to their SaaS, PaaS or IaaS products through a network. For example, in case of SaaS a customer may use a software through a network when needed or in case of IaaS a customer may get access to a server that is located somewhere else through a network.

A cloud computing system has a lot of components and a lot of aspects play a role. In this thesis we will not explain all of these components and we will not mention all of the aspects which are relevant for cloud computing systems. Instead we will concentrate on features which appear in scheduling models for cloud computing systems, see Section 4. Some of these features have not been addressed in traditional mathematical scheduling theory so far. For further information about cloud computing see the textbook of Buyya et al. [18] for more information.

The classification of the service models into SaaS, PaaS and IaaS is widespread. The SaaS service model is that service model out these three, where a customer has the least opportunities to manage or control the underlying cloud infrastructure. As a consequence all of this is done by the cloud computing system itself, so there is the greatest potential for optimisation by the cloud computing system in the SaaS service model [24]. It is important to emphasise that general statements about any of these three service models are difficult and should be treated with caution since these statements depend also on the particular situation and the boundaries between these models are often fluid. Nevertheless we will restrict our considerations to the SaaS service model in this thesis.

Cloud computing offers many advantages for customers. Companies can run their programs and can store a huge amount of data in the cloud. The usage of pay-as-you-go options are much cheaper for companies than to provide and maintain all the software and hardware resources in their own buildings. In addition, since all the data is stored in the *cloud*, backing up and restoring is much easier in the cloud setting than on physical devices. On the other hand, companies should be aware that they may get vulnerable if they store sensitive data in the cloud computing network.

There is a lot of research ongoing in the wide field of cloud computing by applied computer scientists. Current research includes energy efficient scheduling, fault-tolerance as well as economic and ecological aspects of clouds. Leading researchers in this area are for example Buyya, Zomaya, Xu or Li. On the other hand, there exists far less literature containing mathematical models of cloud computing systems. However, researchers have studied important mathematical models which cover features of cloud computing systems. These features include widespread and important objective functions for clouds such as energy consumption, reliability or the total flow time. See Section 4 for more information. Contributions to the study of such theoretical mathematical scheduling models have been given by researchers including Albers, Trystram, Pruhs, Bansal and others.

2.2 From the Cloud Computing System to the Mathematical Model

We consider the situation when customers make use of SaaS products. Examples for SaaS products are Google Apps, Office 365 or salesforce.com. Note that for Office 365 it is not one-hundred-percent clear if they offer a SaaS or a PaaS product. The boundaries are, as mentioned before, often fluid.

Throughout the usage of SaaS-products many *jobs* occur that have to be executed on

a computer (which is also called a *server*) in order to serve the customers. These jobs are typically any types of processes, which may occur for instance while downloading a video, saving data or using a search engine. These jobs require hardware where they can be executed on. The hardware is provided by a cloud computing provider, for example through a *data centre*. A data centre is a facility that is used to house servers and associated components. We consider these hardware components abstractly as *machines*. Jobs are executed in the data centre on these machines. After the execution of these jobs the customers can finally be served.

In general these jobs arrive at random times at the data centre, since customers consume SaaS products at random times. In addition job characteristics such as the processing volume (number of instructions which have to be done to finish the job) or the due date (point of time at which the customer wants the job to be finished) are also not known in advance. However, through historical analysis, predicting techniques or a combination of both of them it is possible to obtain estimators of these job characteristics, see for example [34]. Due to this we assume that job parameters such as the processing volume, the release time and the due date are given. Of course it is important that the job characteristics are well estimated, since all further calculations depend on the used estimators. In this thesis we assume that we are given sufficiently good estimators and therefore we will not consider this aspect in the rest of this thesis.

The general mathematical scheduling model will be of the following form. Imagine the situation for a scheduler in a data centre as described above. We assume that we are given n jobs to be executed on m machines. For each job j, job characteristics such as the release time r_j, the due date d_j and/or the processing volume w_j are assumed to be given (due to available historical analysis/ predicting techniques). The scheduling problem then consists of choosing an execution speed s_j for each job, such that all problem specific restrictions are satisfied and such that a given objective function is optimised. We will provide more details on an mathematical scheduling model for cloud computing in Section 3 and the sections to follow.

Differences that Occur for the PaaS and IaaS Service Models

Finally, we highlight some differences for a scheduler when the underlying service model is either PaaS or IaaS. In the SaaS service model there are more variables in the scheduling model than in the PaaS or the IaaS service model. As an example we consider a customer consuming a SaaS product and assume that the customer is downloading a video through a cloud computing network. To download the video a number of jobs have to be executed in a data centre. In this case, the scheduler can decide which job is executed on which machine at which speed (as long all as restrictions on the jobs are satisfied). Consider now a customer consuming a PaaS or IaaS product. As an example we assume that a customer is renting some CPUs through a cloud computing network. In this case there is significantly less room for optimisation for the scheduler. The scheduler has to choose suitable servers of the data centre to serve the customer. But besides this decision there are not many other decisions to make. In order to compare this example with the example of the SaaS

product above, assume that the customer runs some applications on these servers. There arise jobs which need to be executed in order to run the applications. These jobs have now to be scheduled on the corresponding CPUs. This means that the scheduler does not make a decision on which machine the jobs are executed. Furthermore the scheduler has very limited control (or mostly even no control at all) at which speed the jobs are executed, since the customer has almost all the control of the rented CPUs, see also Figure 1.

As a consequence, there is the most space for optimisation in the SaaS model. The resulting drawback is that these scheduling problems get more complex since there are more decisions to make. On the other hand, an advantage is that good scheduling algorithms may lead to significantly better results than straightforward algorithms achieve.

3 Basic Definitions

The purpose of this section is to introduce basic definitions and notations for scheduling problems which are relevant for this thesis.

3.1 Mathematical Scheduling Model for Scheduling Problems in Cloud Computing Systems

In general n jobs $j = 1, \ldots, n$ have to be executed on m machines M_i for $i = 1, \ldots, m$. To execute a job, a scheduler has to choose on which machine the job has to be executed. Furthermore a scheduler has to choose the time when the execution of a job should be started and at which speed the job should be executed. The following definitions are based on [1], [17] and [39].

Definition 1. For each job j, we define the following parameters:

- The *release time* r_j denotes the time at which job j becomes available.
- The processing volume w_j denotes the amount of work that has to be done to finish job j. In cloud computing the unit of the processing volume of a job is measured in *million instructions* (MI) [24].
- The *due date* d_j denotes the time at which job j should be finished.

In this thesis we consider a machine as a processor that can run at different *execution speeds*. We assume that the execution speed is constant throughout the execution of each job, but different jobs can have different execution speeds. Note that this means in particular that it is not allowed to stop the execution of a job and continue it at a later point of time (in other words we do not allow *preemption*).

Definition 2. For each job j, we introduce the following variables:

- The *execution speed* s_j denotes how much work can be done per time unit. In cloud computing the unit of the execution speed of a job is measured in *million instructions per seconds* (MIPS) [24].
- The *execution time* t_j denotes the time needed to execute the job. In cloud computing the unit of the execution time of a job is measured in seconds [24]. The connection between w_j, s_j and t_j is then defined by

$$t_j = \frac{w_j}{s_j}. \tag{3.1}$$

Note that equation (3.1) also makes sense in terms of the measurement units:

$$t_j = \frac{w_j}{s_j} \qquad \longleftrightarrow \qquad \text{seconds} = \frac{\text{million instructions}}{\text{million instructions per second}}$$

- The *completion time* C_j denotes the time when the job is finished.
- The *unit penalty* U_j is defined by

$$U_j \coloneqq \begin{cases} 0 & \text{if } C_j \leq d_h \\ 1 & \text{otherwise} \end{cases}. \tag{3.2}$$

- The *tardiness* T_j is defined by $T_j \coloneqq \max\{0, C_j - d_j\}$.

Definition 3. We define the following objective functions:

- The *makespan* $C_{\max} \coloneqq \max\{C_1, \ldots, C_n\}$
- The *total completion time* $\sum_{j=1}^{n} C_j$
- The *total flow time* $\sum_{j=1}^{n} F_j$, for $F_j \coloneqq C_j - r_j$
- The *number of delayed jobs* $\sum_{j=1}^{n} U_j$
- The *total tardiness* $\sum_{j=1}^{n} T_j$

Definition 4. A *schedule* is for each job an allocation of a time interval to a machine. A schedule is called *feasible* if no two time intervals overlap and if it meets a number of problem-specific characteristics. A schedule is called *optimal* if it is feasible and minimises a given objective criterion.

Definition 5. A *Gantt chart* is a graphical representation of a feasible schedule. Each job will be represented by a box. The length of the box corresponds to the execution time, the height of the box corresponds to the execution speed and the area of the box corresponds to the processing volume. Each machine is represented by a horizontal band. The Gantt chart is two-dimensional. The time horizon (measured in seconds) is presented in x-direction and the computation speed (measured in MIPS) is presented in (measured in seconds) y-direction. Note that each machine has its own execution speed scale. If a job is allocated to a certain machine, the box of this job is drawn on the horizontal band of the machine in the Gantt chart.

An example of a Gantt chart is depicted in Figure 2.

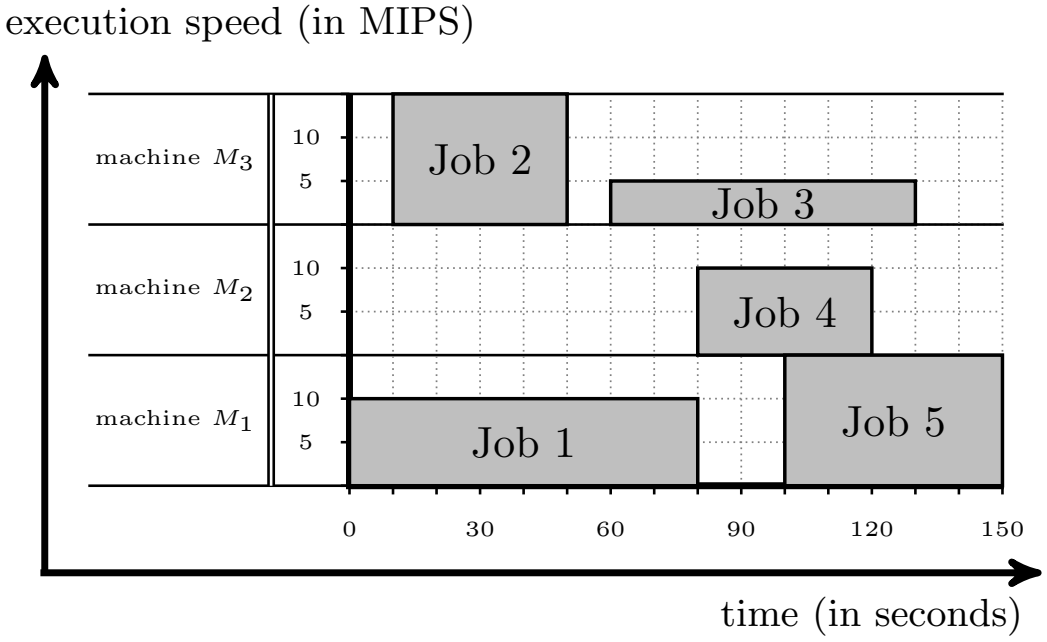

Figure 2: Example of a Gantt chart with five jobs. The release times are given by $r_1 = 0, r_2 = 10, r_3 = 60, r_4 = 80, r_5 = 100$ and the processing volumes (area of the boxes) are given by $w_1 = 800, w_2 = 600, w_3 = 350, w_4 = 400, w_5 = 750$. It is assumed that the executions of all jobs are started exactly at the corresponding release times. The execution times are chosen as $t_1 = 80, t_2 = 40, t_3 = 70, t_4 = 40, t_5 = 50$ and the execution speeds are chosen as $s_1 = 10, s_2 = 15, s_3 = 5, s_4 = 10, s_5 = 15$.

We will also consider scheduling problems with two objective functions in this thesis. Therefore we give some basic definitions on *bi-criteria optimisation problems* (optimisation problems with two objective functions). The definitions and notations are taken from [22].

Definition 6 (Pareto optimality)**.** Consider a bi-criteria optimisation problem with objective functions f_1 and f_2. The set of feasible solutions is denoted by $\mathcal{X}$. A feasible solution $x \in \mathcal{X}$ is called *Pareto optimal* or *efficient* if there is no other $x' \in \mathcal{X}$ such that $f_1(x') \leq f_1(x)$ and $f_2(x') \leq f_2(x)$ holds with at least one inequality being strict. The set of all Pareto optimal solutions is called the *efficient set*.

In what follows we denote by f_1 and f_2 two given objective functions and by $\mathcal{X}$ the *feasible set*, which is the set of feasible solutions.

Since we consider two objective functions, we can study the following four types of optimisation problems:

(i) Given two parameters γ_1 and γ_2, find an $x \in \mathcal{X}$ that minimises $\gamma_1 f_1 + \gamma_2 f_2$.

(ii) Given an upper bound $\overline{U}_2$ on f_2, find an $x \in \mathcal{X}$ that minimises f_1 such that $f_2 \leq \overline{U}_2$.

(iii) Given an upper bound $\overline{U}_1$ on f_1, find an $x \in \mathcal{X}$ that minimises f_2 such that $f_1 \leq \overline{U}_1$.

(iv) Find the Pareto optimal set.

3.2 Extension of the Classical $\alpha \mid \beta \mid \gamma$ Notation

In scheduling literature, the $\alpha \mid \beta \mid \gamma$ notation (introduced by Lawler et al. in [25]) is widely used to classify scheduling problems. We briefly describe below the parts of this notation which are needed in this thesis.

- The $\alpha-$ field specifies the machine environment. We will include $\alpha = 1$ if we consider a scheduling problem with one machine. If we include $\alpha = Pm$ for $m \in \mathbb{N}$ and $m \geq 2$, then we consider a scheduling problems with m identical machines. The symbol P highlights that all considered machines are identical.

- The $\beta-$ field specifies the job environment. This field contains information about the values w_j, d_j and r_j. If this field contains the string „$w_j = 1$“, then all jobs have the same processing volume which is assumed to be equal to 1 without loss of generality. On the other hand, if the field contains the string „w_j“, the processing volumes of the jobs are arbitrary. Similar notation is used for d_j and r_j. The only exception is that if neither the string „d_j“ nor a string of the form „$d_j = d$“ appears in this field, this indicates that no due dates are considered in the corresponding problem. This then corresponds to $d_j = \infty$ for all jobs.

- The γ- field specifies which objective function is considered. For example, we include „$\gamma = C_{\max}$“ or „$\gamma = \sum_{j=1}^{n} U_j$“, if we want to minimise the makespan or the number of delayed jobs respectively.

  In this thesis we will also consider bi-criteria scheduling problems of the types (i) - (iv) introduced above. To have a compact notation for these problems, we extend the $\alpha \mid \beta \mid \gamma$ notation from above. To denote a problem of type (i), we include a string of the form „$(\gamma_1 f_1 + \gamma_2 f_2)$“ into field γ. A problem of type (ii) or (iii) is denoted by including the correspond upper-bound restriction in field β and the corresponding objective function into field γ. Finally, to denote a problem of type (iv), we include a string of the form „(f_1, f_2)“ into field γ.

We will consider new properties of scheduling problems in Section 4. To include this new features in the $\alpha \mid \beta \mid \gamma$ notation we will extend the $\alpha \mid \beta \mid \gamma$ classification further in Section 4.3.

4 Features in Distributed Computing

A cloud computing system is very complex and involves a lot of technologies. Furthermore it may need to satisfy several goals at once. This means that in order to formulate a complete mathematical model of a cloud computing system there are a huge number of possible restrictions and many different objective functions to consider. Many of these restrictions and objective functions have not been studied from a mathematical point of view so far.

In this section we describe several classes of objective functions and restrictions that are typical for cloud computing systems.

4.1 Objective Functions in the Context of Cloud Computing

4.1.1 Energy Consumption

One of the most important objective functions to consider in the context of cloud computing is the energy consumption. Providers of large cloud services consume many megawatts to operate their data centres. Examples are Google with over $1,120$ GWh or Microsoft with over 600 GWh. This energy usage results in electricity bills of about $67 million and $36 million for Google and Microsoft for 2010 [38]. The energy usage is expected to increase further in the next years. By 2012, data centres that power internet-scale applications consumed about 1.3% of the worldwide electricity supply and this fraction is predicted to reach 8% by 2020 [23]. These aspects highlight the importance of the criterion of energy consumption.

In the literature there is no generally accepted energy model that describes the energy usage of a server or a whole data centre in mathematical terms, see Section 5 for a further discussion. A number of different energy models have been proposed in the literature. So far no consensus has been obtained on a common or best model. In the following we discuss two models from the literature. The first one is more mathematically orientated and the second one comes from the applied computer science literature.

Model 1: The first energy model has mostly been studied by mathematicians, see for example [1]. Given a processor that can run at different speeds, we consider n jobs $1, \dots, n$ with processing volumes $w_j > 0$ for $1 \leq j \leq n$. If the processor runs at execution speed $s_j > 0$ to execute job j, then the execution time t_j is given by $t_j = \frac{w_j}{s_j}$. The power consumption of processor running at speed s is given by s_j^α for $\alpha > 1$. The energy consumption of job j is then calculated by integrating the power consumption over time. Therefore the energy consumption can be calculated as

$$\sum_{j=1}^{n} \int_0^{t_j} s_j^\alpha \, \mathrm{d}\tau = \sum_{j=1}^{n} t_j s_j^\alpha = \sum_{j=1}^{n} w_j s_j^{\alpha-1}. \tag{4.1}$$

Model 2: The second energy model is a linear model considered mainly by computer scientists, see for example [35]. For a given server, the *CPU utilisation level* is denoted

by $U \in [0,1]$, the power consumed by a fully utilised CPU is denoted by $P_{\max}$ and the fraction of consumed power when the CPU is idle is denoted by k. Then the power consumption $P(U)$ is calculated as

$$P(U) = kP_{\max} + (1-k)P_{\max}U. \tag{4.2}$$

To obtain the energy consumption, we have to integrate $P(U)$ over time. This model is motivated according to experiments that indicates that the power consumption depends mainly and linearly on the utilisation level U, see [14], [19] and [26] for example.

4.1.2 Reliability

Another important objective function in cloud computing systems is the *reliability*. This objective function measures how „reliable“ the considered cloud service for the customers is. For example, it can happen in real world cloud computing systems, that the execution of a job gets aborted due to hardware failures, software faults or resources removal. As a consequence of these incidents, the customer will not be satisfied since the submitted jobs may finish delayed or even not at all. This is of course also a problem for the cloud service provider, because unsatisfied customers are not good for the business.

There exist several mathematical reliability models in the literature. One of them [27] is presented in the following.

We are given a set of jobs $\mathcal{J} = \{1, \dots, n\}$ and a set of m *uniform* machines $\mathcal{M} = \{M_1, \dots, M_m\}$. Each job j requires w_j instructions to be completed. Machine M_i computes $\tau_{M_i}^{-1}$ operations per time unit and has a probability of failures according to an exponential distribution with failure rate λ_{M_i}. In this model it is assumed that failures on different machines occur independently from each other. The execution time of job j on machine M_i is given by $t_{j,M_i} = w_j\tau_{M_i}$. The mapping $\pi : \mathcal{J} \to \mathcal{M}$ assigns each job to a machine on which the job should be executed. Let $\mathcal{J}(M_i, \pi) \coloneqq \{j \mid \pi(j) = M_j\}$ denote the set of jobs assigned to machine M_j under mapping π. The *execution time of machine* M_i, which is the amount of time the machine is executing a job, is then given by $t_{M_i}(\pi) \coloneqq \sum_{j \in \mathcal{J}(M_i,\pi)} t_{j,M_i}$. Then the probability that machine M_i executes all its jobs without failure is given by $\mathbb{P}_{M_i}(\pi) \coloneqq \exp\left(-\lambda_{M_i} t_{M_i}(\pi)\right)$ because we assume an exponential distribution of the failure rate. Since failures occur independently on different machines, the probability that all jobs on all machines execute without failure is given by $\mathbb{P}(\pi) \coloneqq \exp\left(-\sum_{i=1}^{m} \lambda_{M_i} t_{M_i}(\pi)\right)$. In this model, optimising the reliability corresponds to maximising $\mathbb{P}(\pi)$ which corresponds to minimising the reliability index $\mathrm{rel}(\pi) \coloneqq \sum_{i=1}^{m} \lambda_{M_i} t_{M_i}(\pi)$.

4.1.3 Other Objective Functions

Applied computer scientists use the term *performance* when they classify how „good“ the cloud computing systems operates. A closer look at papers in this area shows that many authors do not specify what performance means exactly in their work. Performance can

be a single objective or include several aspects as reliability, energy efficiency or scalability at once [29]. We now give a list of objective functions which can be considered as a performance measurement.

- Total number of delayed jobs $\sum_{j=1}^{n} U_j$
- Makespan $C_{\max}$
- Total completion time $\sum_{j=1}^{n} C_j$
- Total flow time $\sum_{j=1}^{n} F_j$
- Total tardiness $\sum_{j=1}^{n} T_j$

All of these objective functions lead to optimisation problems in minimisation form. The objective functions listed above have already been investigated in mathematical scheduling theory. The following objective functions are typical for applied literature on cloud computing and are taken from [12].

- Performance efficiency: The amount of resources used under stated conditions. Resources can include software products or hardware components.
- Time behaviour: The degree to which the response, processing times and throughput rates of a product or system, when performing its functions, meet requirements.
- Capacity: The degree to which the maximum limits of a product or system parameter meet requirements.

4.2 Typical Restrictions for Models of Cloud Computing Systems

4.2.1 Different Computational Resources

In order to execute a job in a cloud computing system, jobs require different kinds of computational resources. In this thesis we just consider the CPU as a computational resource. The performance of a CPU is measured in *million instructions per second* (MIPS). From a computer science point of view, we can interpret the formula $t_j s_j = w_j$ which was introduced in (3.1) as follows. The execution time t_j measures the time in seconds which is needed to execute job j. The executing speed s_j measures the performance of the processor respectively the CPU and is measured in MIPS. Finally, the processing volume w_j denotes how much CPU resource is required to finish job j or equivalently how many instructions have to be done to finish job j. The processing volume w_j can therefore be measured in *million instructions* (MI).

Besides CPU there are many other computational resources which can be included instead or in addition. Examples for other types of computational resources are memory,

disk or network. As mentioned before we do not consider other computational resources than the CPU in this thesis. We just point out that including further types of computational resources is one option to obtain more general models.

4.2.2 Immediate Start in the Presence of Abundant Resources

In a cloud computing system, a huge number of computational resources are brought together through a network. As a consequence, the cloud computing provider can offer big amounts of computational resources to the customers at almost any point of time. We say that there are *abundant computational* resources available in the system (see for example [30]). There are options for customers to monitor the execution process. Customers get unsatisfied if their execution takes too much time or if the start of the execution is postponed by the system. So it is in the service provider's interest to make use of the available abundant resources and to start the execution of arriving jobs immediately. This motivates the following definition.

Definition 7 (Immediate Start Condition)**.** We say that a scheduling problem takes into account the *immediate start condition*, if the execution of every job is started immediately at its release time.

4.2.3 Virtual Machines

Virtualisation is one important technological concept of cloud computing. This concept plays a role when jobs have to be executed on machines. Virtualisation allows to build *virtual machines* on a physical server. This means that a virtual machine occupies a part of the computational resources of the underlying server and can then itself be considered as a machine. There can be several virtual machines on one physical machine (occupying distinct computational resources), but it is not possible that one virtual machine occupies computational resources of different servers. When a job needs to be executed to serve a customer, the job is executed on a virtual machine which itself is running on a physical machine in the data centre.

If we consider the aspects of building virtual machines, the model becomes more complex. The scheduler then in addition has to create virtual machines on the physical machines.

4.3 Further Extension of the $\alpha \mid \beta \mid \gamma$ Notation to the Cloud Computing Case

Having already introduced and extended the classical $\alpha \mid \beta \mid \gamma$ notation in Section 3, we further extend our $\alpha \mid \beta \mid \gamma$ notation from Section 3 to the cloud computing case.

The following notation goes beyond the classical notation, but will be needed in the context of this thesis.

- **imst**: If the field β includes the string „imst“, a scheduling problem with the immediate start condition is studied. This means that the execution of every job has to be started at its release time.

- **energy**: The string „energy“ stands for the total energy consumption of all jobs. We will use the following energy formula in the rest of the thesis:

$$\text{energy} = \sum_{j=1}^{n} w_j s_j^{\alpha-1}$$

  - If a string of the form „energy $\leq \overline{E}$“ appears in the β- field, the notation indicates that the total energy consumption of all jobs is restricted to an upper bound $\overline{E}$.
  - If a string of the form „energy“ appears in the γ- field, the notation indicates that the total energy consumption is the objective function in this problem.

5 Short Review of the Literature on Cloud Computing

The purpose of this section is to highlight the differences between the literature on mathematically oriented scheduling theory and on applied scheduling models for cloud computing systems, respectively. Furthermore some possible difficulties a mathematician has to face when studying literature on applied scheduling models in cloud computing systems are described.

- **Lower degree of abstraction:** Scheduling models in the literature on mathematically orientated scheduling are usually of high abstraction. However, scheduling models in the literature on applied scheduling models for cloud computing systems are usually significantly less abstract. For example, applied scheduling models for cloud computing systems take into account technical aspects of hardware or network systems and often do not propose a mathematically well-defined scheduling model. A mathematician might not be familiar with these technical aspects and this therefore makes it more difficult for a mathematician to get familiar with the literature on applied scheduling models for cloud computing systems. As a consequence it becomes difficult to make an attempt to provide an accurate mathematical model for the scenario the applied people are interested into.

- **Definitions:** Mathematicians are used to work with exact and well defined definitions. However it turned out that it is very often the case in literature on applied scheduling models for cloud computing systems that not all terms are defined in a way a mathematician is used to. For example, the term „performance“ appears in almost every paper dealing with applied scheduling models for cloud computing systems as a measurement of how good scheduling a cloud computing system works, but the term „performance“ has different meanings in different situations and only a few papers give a definition what „performance“ means exactly in the corresponding situation.

- **Energy model:** As already mentioned in Section 4.1.1, the energy consumption is one of the most important objective functions to consider in cloud computing. However there is no general agreement about a formula for the energy consumption. In many papers that consider the energy consumption as an objective function, no formulas for the calculation of the energy consumption are presented. It is not always clear how exactly the authors of these papers get to numbers for the energy consumption in the evaluation part of these papers.

5.1 Comments on the Energy Consumption

We now compare the two models for the energy consumption that are briefly introduced in Section 4.1.1. Our aim is to highlight the difficulty of choosing a proper energy

consumption formula. To that end we will consider versions of the two models and compare the optimal strategies that lead to optimal solutions depending on the chosen energy-consumption-objective function. We will see that one optimal strategy consists of executing all jobs at the lowest possible speed whereas the other optimal strategy consists of executing all jobs at the highest possible speed. So the optimal strategies behave completely contrary to each other.

Model 1: We consider the energy model as proposed in the first model in Section 4.1.1. We are given n jobs j for $j = 1, \ldots, n$ with processing volume w_j for each job j. The execution speed of job j is given by $s_j = \frac{w_j}{t_j}$. The power consumption for job j at a given time is given by s_j^α and the energy consumption is obtained by integrating the power consumption over time. No restrictions are considered in this model. The total energy consumption is given by the formula

$$\sum_{j=1}^{n} \left(\int_0^{t_j} s_j^\alpha \, \mathrm{d}\tau \right) = \sum_{j=1}^{n} t_j s_j^\alpha = \sum_{j=1}^{n} w_j s_j^{\alpha-1}. \tag{5.1}$$

It is clear that the two strategies are completely contradictory.

Model 2: Now we consider a linear energy model as proposed in the second model in Section 4.1.1. In this model jobs should be executed on a machine and the energy consumption is a linear function of the utilisation level. The power consumption of job j for $j = 1, \ldots, n$ is given by

$$P(U_j) = \alpha + \gamma_j U_j,$$

where $U_j \in [0, 1]$ denotes the CPU utilisation level while job j is executed and α and γ_j are system dependent parameters. The processing volume of job j is denoted by w_j (measured in MI) and the *capacity* of the machine by ψ (measured in MIPS). The capacity of a machine denotes how many instructions per second the machine is able to perform if the CPU utilisation level is 100%. In this model the execution time of job j is given by

$$t_j = \frac{w_j}{\psi U_j}.$$

Again, the energy consumption is calculated by integrating the power consumption over time:

$$\begin{aligned} \sum_{j=1}^{n} \left(\int_0^{t_j} (\alpha + \gamma_j U_j) \, \mathrm{d}\tau \right) &= \sum_{j=1}^{n} \left(t_j \alpha + \frac{\gamma_j w_j}{\psi} \right) = \alpha \sum_{j=1}^{n} t_j + \sum_{j=1}^{n} \frac{\gamma_j w_j}{\psi} \\ &= \alpha \sum_{j=1}^{n} \frac{w_j}{s_j} + \sum_{j=1}^{n} \frac{\gamma_j w_j}{\psi} \end{aligned}$$

It is clear that an optimal strategy consists of executing jobs at highest possible speed.

We have now presented two simple energy models with a different approach of modelling the energy consumption. In the first model it is optimal to execute jobs at their lowest speed, but in the second model it is optimal to execute them at their highest speed. We note again that both formulas for the energy consumption have been considered in the literature (see [1] for the first model and [14], [19], [26] for the second model). The fact that the two strategies are completely contradicting is an indication that none of the two formulas for the energy consumption is complex enough to describe the energy consumption sufficiently good. This is one reason why empirical energy consumption models are often preferred to to theoretically justified energy consumption models [24].

6 Core Based Energy Models

In Section 5.1 we have seen that there exist different energy models in the literature that can lead to quite different scheduling strategies. This fact is taken as a motivation to consider more complex energy models in order to obtain a more realistic energy model. We consider a new and more complex energy model in the following. This model was developed with the help of Peter Garraghan [24].

We consider a processor consisting of a given number γ of identical cores. In what follows we introduce the notation needed to formulate core based energy models. In this section, we just consider problems with one processor. For this reason the considered machine is denoted by M (so we omit the index). The cores of processor M are denoted by $\Gamma_1, \ldots, \Gamma_\gamma$. We consider in the following a fixed job j with release time r_j and processing volume w_j. In order to execute j, each core Γ_k completes a certain fraction of job j. We call the fraction of job j that is executed on core Γ_k the *job-fraction* of j on Γ_k. We denote the processing volume of the job-fraction of j on Γ_k by $w_{j,k}$ and we require $0 \leq w_{j,k} \leq w_j$ for $1 \leq k \leq \gamma$ and $\sum_{k=1}^{\gamma} w_{j,k} = w_j$. The job-fraction of j on Γ_k is assumed to be executed at constant speed $s_{j,k}$ on core Γ_k (but job-fractions of different jobs on Γ_k may be executed at different execution speeds). In particular we do not allow preemption, so the execution of a job-fraction cannot be suspended and resumed later. We allow a core to execute job-fractions of several jobs at the same time. At every point of time, the *execution speed of a core* is given by the sum of all execution speeds of job-fractions executed at this time. It is assumed that an upper bound $s_{\max}$ of the execution speed of a core is given in this model. For a fixed point of time t, the execution speed of j on M at time t is denoted by $s_j(t)$ and the execution speed of the job-fraction of j on Γ_k at time t is denoted by $s_{j,k}(t)$. The relation between the execution speed on M and the execution speed on the cores $\Gamma_1, \ldots, \Gamma_o$ is given by

$$s_j(t) \quad = \sum_{\substack{\text{cores } \Gamma_k \text{ that execute} \\ \text{a job-fraction of } j \\ \text{at time } t}} s_{j,k}(t). \tag{6.1}$$

It also assumed that a common lower bound on the execution speeds of the jobs is given by $s_{\min}$. Note that $s_{\max}$ is a bound on the execution speeds of the cores, whereas $s_{\min}$ is a lower bound on the execution speeds of jobs. We denote the execution time and completion time for job-fraction j on core Γ_k by $t_{j,k} = \frac{w_{j,k}}{s_{j,k}}$ and $C_{j,k}$. The relation between the completion time of job j on M and the completion time of the job-fractions on the cores is given by $C_j = \max_{1 \leq k \leq \gamma} C_{j,k}$. The execution time for job j is given by the difference between C_j and the first time at which a job-fraction of j is executed on a core. We assume that for each job j all job-fractions become available at time r_j. In this section we will consider only models that include the immediate start condition, so we have

$$r_j + t_{j,k} = C_{j,k} \qquad 1 \leq j \leq n, 1 \leq k \leq \gamma \qquad \text{and} \qquad r_j + t_j = C_j \qquad 1 \leq j \leq n.$$

If at a certain point of time a core is executed at execution speeds s, then power consumption of the core at this time is given by s^3. The energy consumption of a core is

calculated by integrating the power consumption over time. The energy consumption of M is given as the sum of the energy consumption of all cores.

Before we move on, we consider a simple example.

Example 1. *Consider a machine M with two cores Γ_1 and Γ_2. The following three jobs have to be executed:*

job	*release time*	*processing volume*
1	$r_1 = 0$	$w_1 = 4$
2	$r_2 = 4$	$w_2 = 12$
3	$r_3 = 5$	$w_3 = 9$

We execute jobs $1, 2$ and 3 as depicted in the following:

job	*executing cores*	*processing volume of job fractions*	*execution speed of job fractions*	*execution time of job fractions*
1	Γ_1	$w_{1,1} = 4$	$s_{1,1} = 0.5$	$t_{1,1} = 8$
2	Γ_1, Γ_2	$w_{2,1} = 8, w_{2,2} = 4$	$s_{2,1} = 1, s_{2,2} = 2$	$t_{2,1} = 8, t_{2,2} = 2$
3	Γ_1, Γ_2	$w_{3,1} = 5, w_{3,2} = 4$	$s_{3,1} = 0.5, s_{3,2} = 0.5$	$t_{3,1} = 10, t_{3,2} = 8$

A graphical representation in a Gantt chart of this solution is depicted in Figure 3.

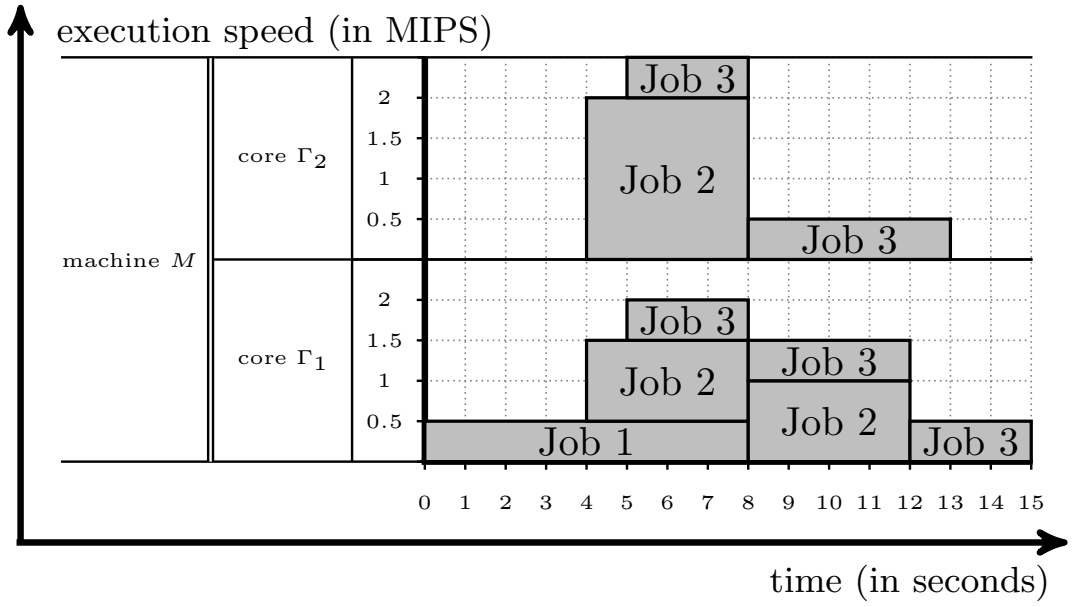

Figure 3: Gantt chart of an instance for a core based scheduling problem

The proposed core based energy model is quite general. In order to study the trade-off between energy consumption and performance in more detail it is natural to consider a simplified model at first. This is the purpose of the following section.

6.1 A Core Based Energy Model with Identical Jobs

6.1.1 Model assumptions

We consider a machine M consisting of γ identical cores. There are given n jobs with $r_j = 0$ and $w_j = 1$ for $1 \leq j \leq n$. We will present a model that minimises the energy consumption such that a given bound $\overline{R}$ on the response time is not exceeded. In order to execute a job j we need to define a strategy of how job j is divided into job-fractions to be executed on cores $\Gamma_1, \ldots, \Gamma_k$. In this model, we choose all job-fractions to have equal size, so we require

$$w_{j,k} = \frac{w_j}{\gamma} = \frac{1}{\gamma} \qquad 1 \leq j \leq n, 1 \leq k \leq \gamma. \tag{6.2}$$

Now we can formulate the model as follows:

Problem 1.

$$T(\overline{R}) \coloneqq \text{minimise} \quad \sum_{\ell=1}^{\gamma} \sum_{j=1}^{n} (C_j - C_{j-1}) \left(\frac{1}{\gamma} \sum_{k=j}^{n} \frac{1}{C_k} \right)^3 \tag{6.3}$$

$$\text{subject to} \quad \sum_{j=1}^{n} C_j \leq \overline{R} \tag{6.4}$$

$$s_{\min} \leq \frac{1}{C_j} \qquad j = 1, \ldots, n \tag{6.5}$$

$$\frac{1}{\gamma} \sum_{j=1}^{n} \frac{1}{C_j} \leq s_{\max} \qquad j = 1, \ldots, n \tag{6.6}$$

$$0 = C_0 \leq C_1 \leq \ldots \leq C_n \tag{6.7}$$

Since all jobs are identical, we can consider jobs in increasing order of their completion time, which is done in constraint (6.7). The variable C_0 is introduced to formulate the objective function in (6.3) in a more convenient way. Since $r_j = 0$ and according to the immediate start property, we have $t_j = C_j$. Furthermore the execution speed of a job-fraction of job j executed on a core is given by $\frac{1}{\gamma C_j}$. The execution of all job-fractions is started at time 0, so at the beginning there is the most work to do. So constraint (6.6) ensures that at each point of time each core is not executed at higher speed than $s_{\max}$. The execution speed of a job is given by $\frac{1}{C_j}$ according to (6.1) and therefore constraint (6.5) ensures that each job is executed at a minimum speed of $s_{\min}$. constraint (6.4) bounds the response time $\sum_{j=1}^{n} C_j$ from above by $\overline{R}$. Next we explain the objective function (energy consumption) in (6.3). Between times $C_0 = 0$ and C_1 all jobs are executed. Therefore the execution speed of each core in this time interval is given by $\frac{1}{\gamma} \sum_{k=1}^{n} \frac{1}{C_j}$. As a consequence the power consumption in this time interval on one core is given by $\left(\frac{1}{\gamma} \sum_{k=1}^{n} \frac{1}{C_j} \right)^3$ and the

energy consumed in this time interval on all cores is therefore given by

$$\sum_{\ell=1}^{\gamma} (C_1 - C_0) \left(\frac{1}{\gamma} \sum_{k=1}^{n} \frac{1}{C_j} \right)^3.$$

Between times C_1 and C_2 all jobs but job 1 are executed. Therefore the execution speed of each core in this time interval is given by $\frac{1}{\gamma} \sum_{k=2}^{n} \frac{1}{C_j}$. As a consequence the power consumption in this time interval on one core is given by $\left(\frac{1}{\gamma} \sum_{k=2}^{n} \frac{1}{C_j} \right)^3$. Therefore the total energy consumption between times $C_0 = 0$ and C_2 on all cores is therefore given by

$$\sum_{\ell=1}^{\gamma} \left[(C_1 - C_0) \left(\frac{1}{\gamma} \sum_{k=1}^{n} \frac{1}{C_j} \right)^3 + (C_2 - C_1) \left(\frac{1}{\gamma} \sum_{k=2}^{n} \frac{1}{C_j} \right)^3 \right].$$

Iterating this argument over all of these time intervals leads to the formula given in (6.3).

6.1.2 Trade-Off between Energy Consumption and Performance

In this section we will study the trade-off in Problem 1 between the energy consumption and the performance. Therefore we analyse the trade-off function T defined in (6.3) on the feasible set

$$\mathcal{X} \coloneqq \{(C_0, C_1, \ldots, C_n) \mid \text{ constraints (6.4) to (6.7) are satisfied}\}.$$

We note that Problem 1 is a special convex parametric optimisation problem. There exist general results on convex parametric optimisation problems in the literature (see for example [6]). In the following we will analyse the trade-off between energy consumption and performance in detail by making use of the special properties of the problem under investigation and not by relying on the general theory of parametric programming as in this manner a better insight will be provided. Theorem 6.1 summarises the results obtained from our analysis.

Theorem 6.1. *The set*

$$I \coloneqq \{\overline{R} \in \mathbb{R} \mid \textit{ Problem 1 has an optimal solution for the response time bound } \overline{R}\}$$

is non-empty if and only if

$$\gamma s_{\max} \geq n s_{\min}. \tag{6.8}$$

If (6.8) *holds then we have* $I = [\overline{R}_{\min}, \infty)$ *and* $\overline{R}_{\min} \leq \overline{R}_{\max}$ *with*

$$\overline{R}_{\min} \coloneqq \frac{n^2}{\gamma s_{\max}} \qquad \textit{and} \qquad \overline{R}_{\max} \coloneqq \frac{n}{s_{\min}}.$$

Function T is strictly monotonic decreasing and strictly convex on $[\overline{R}_{\min}, \overline{R}_{\max}]$, continuous in $\overline{R}_{\max}$ and constant with value

$$T(\overline{R}_{\max}) = \frac{n^3 \left(s_{\min}\right)^2}{\gamma^2}$$

on $[\overline{R}_{\max}, \infty)$.

An optimal solution $C_0^*, C_1^*, \ldots, C_n^*$ for Problem 1 with response bound $R \in I$ satisfies

$$\sum_{j=1}^{n} C_j^* = \begin{cases} R & \text{for } R \in \left[\overline{R}_{\min}, \overline{R}_{\max}\right] \\ \overline{R}_{\max} & \text{for } R \in \left[\overline{R}_{\max}, \infty\right) \end{cases}.$$

An optimal solution for Problem 1 with response bound $\overline{R}_{\min}$ is obtained by

$$C_0 = 0 \qquad \text{and} \quad C_j = \frac{n}{\gamma s_{\max}} \quad \text{for} \quad 1 \leq j \leq n.$$

The corresponding objective function value is given by

$$T(\overline{R}_{\min}) = n \left(s_{\max}\right)^2.$$

Remark 1 (Interpretation of $\gamma s_{\max} \geq n s_{\min}$)**.** The condition $\gamma s_{\max} \geq n s_{\min}$ in Lemma 6.3 for the non-emptiness of I can be interpreted as follows. The left side $\gamma s_{\max}$ denotes the maximal available CPU resources on the machine including all cores. The right side $n s_{\min}$ is the value of minimal required CPU resources to execute all n jobs when they are executed at their lowest execution speed. The inequality therefore ensures that there are enough resources available to execute all jobs.

The proof of Theorem 6.1 is divided into several lemmas that are presented in the following.

Lemma 6.1. *Function $\sum_{\ell=1}^{\gamma} \sum_{j=1}^{n} (C_j - C_{j-1}) \left(\sum_{k=j}^{n} \frac{1}{C_j}\right)^3$ is strictly convex on $\mathcal{X}$ and can be written as a sum of terms of the form*

$$\frac{\gamma}{C_{j_1} C_{j_2}} \qquad \text{with} \quad j_1, j_2 \in \{1, \ldots, n\},$$

such that each variable $C_1, \ldots, C_n$ is contained in the denominator of at least one term.

Proof. Obviously it is sufficient to show the convexity of $\sum_{j=1}^{n} (C_j - C_{j-1}) \left(\sum_{k=j}^{n} \frac{1}{C_j}\right)^3$ since

$$\sum_{\ell=1}^{\gamma} \sum_{j=1}^{n} (C_j - C_{j-1}) \left(\sum_{k=j}^{n} \frac{1}{C_j}\right)^3 = \gamma \sum_{j=1}^{n} (C_j - C_{j-1}) \left(\sum_{k=j}^{n} \frac{1}{C_j}\right)^3.$$

By use of $C \coloneqq (C_0, C_1, \ldots, C_n)$ we define $f(C) \coloneqq \sum_{j=1}^{n} (C_j - C_{j-1}) \left(\sum_{k=j}^{n} \frac{1}{C_k}\right)^3$. For $C \in \mathcal{X}$ we have $C_0 = 0$ and we can therefore write $f(C)$ as

$$f(C) = C_1 \left(\left(\sum_{k=1}^{n} \frac{1}{C_k} \right)^3 - \left(\sum_{k=2}^{n} \frac{1}{C_k} \right)^3 \right) + C_2 \left(\left(\sum_{k=2}^{n} \frac{1}{C_k} \right)^3 - \left(\sum_{k=3}^{n} \frac{1}{C_k} \right)^3 \right) + \ldots + C_n \left(\frac{1}{C_n} \right)^3.$$

We consider now a fixed summand of the form

$$C_\ell \left(\left(\sum_{k=\ell}^{n} \frac{1}{C_k} \right)^3 - \left(\sum_{k=\ell+1}^{n} \frac{1}{C_k} \right)^3 \right) \tag{6.9}$$

for $\ell \in \{1, \ldots, n-1\}$ and define $a_\ell \coloneqq \left(\sum_{k=\ell}^{n} \frac{1}{C_k}\right)^3$ and $b_\ell \coloneqq \left(\sum_{k=\ell+1}^{n} \frac{1}{C_k}\right)^3$. By expanding we see that a_ℓ and b_ℓ are sums of terms of the form $\frac{1}{C_f C_g C_h}$ for, without loss of generality, $C_f \leq C_g \leq C_h$. Each term that appears in b_ℓ also appears in a_ℓ, therefore $a_\ell - b_\ell$ is a sum of terms of the form $\frac{1}{C_\ell C_f C_g}$ with $C_\ell \leq C_f \leq C_g$. Note that the variable C_ℓ must be contained in the denominator. Note also that all the signs of these terms are positive. As a consequence the summand $C_\ell(a_\ell - b_\ell)$ in (6.9) is a sum of terms of the form $\frac{1}{C_f C_g}$ with $C_\ell \leq C_f \leq C_g$. Note that the last summand in $f(C)$, namely $C_n \left(\frac{1}{C_n}\right)^3$, is also of this form (for $\ell = n$) and therefore the objective function $f(C)$ is a sum of terms of this form too. All terms $\frac{1}{C_1^2}, \ldots, \frac{1}{C_n^2}$ are contained in the sum, which proves the second statement of the lemma.

Functions $\frac{1}{C_f C_g}$ are convex for $0 < C_f \leq C_g$. Note that the strictly convex function $\frac{1}{C_n^2}$ is contained in the sum. Therefore $f(C)$ is strictly convex on $\mathcal{X}$ as a sum of convex functions with at least one strictly convex summand. □

Lemma 6.2. *Consider the optimisation problem*

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} C_j & & \\
\text{s.t.} \quad & \sum_{j=1}^{n} \tfrac{1}{\gamma C_j} & \leq s_{\max} & \\
& s_{\min} & \leq \tfrac{1}{C_j} & \qquad 1 \leq j \leq n \\
& C_j & \leq C_{j+1} & \qquad 1 \leq j \leq n-1,
\end{aligned}$$

with $\gamma s_{\max} \geq n s_{\min}$. Then an optimal solution is given by

$$C_j = \frac{n}{\gamma s_{\max}} \qquad 1 \leq j \leq n. \tag{6.10}$$

Proof. Due to $\gamma s_{\max} \geq n s_{\min}$, the solution in (6.10) is feasible. Since the objective function is continuous and the feasible set is non-empty and compact, there is an optimal solution.

We consider an optimal solution $C_1^*, \ldots, C_n^*$. If this solution is not as in (6.10), then there are j_1, j_2 such that

$$C_{j_1}^* < \frac{n}{\gamma s_{\max}} < C_{j_2}^*,$$

because otherwise constraint $\sum_{j=1}^{n} \frac{1}{\gamma C_j^*} \leq s_{\max}$ is not satisfied or $C_1^*, \ldots, C_n^*$ is not an optimal solution. It is easy to see that there exists values δ_1 and δ_2 with $\delta_2 > \delta_1$ such that

$$C_{j_1}^* < C_{j_1}^* + \delta_1 < C_{j_2}^* - \delta_2 < C_{j_2}^*$$

and

$$\frac{1}{C_{j_1}^*} + \frac{1}{C_{j_2}^*} = \frac{1}{C_{j_1}^* + \delta_1} + \frac{1}{C_{j_2}^* - \delta_2}$$

holds. But then solution $C_1', \ldots, C_n'$ with

$$C_j' \coloneqq \begin{cases} C_j^* & \text{for } j \neq j_1, j_2 \\ C_{j_1}^* + \delta_1 & \text{for } j = j_1 \\ C_{j_2}^* - \delta_2 & \text{for } j = j_2 \end{cases}$$

is also feasible with a smaller objective function value which contradicts the optimality of $C_1^*, \ldots, C_n^*$. □

Lemma 6.3. *The set*

$$I \coloneqq \{\overline{R} \in \mathbb{R} \mid \textit{ Problem 1 has an optimal solution for the response time bound } \overline{R}\}$$

is a left-bounded and right-unbounded interval $[\overline{R}_{\min}, \infty)$*. The interval is non empty if and only if* $\gamma s_{\max} \geq n s_{\min}$*. In this case we have*

$$\overline{R}_{\min} = \frac{n^2}{\gamma s_{\max}}$$

and the objective function value for Problem 1 with response $\overline{R}_{\min}$ *is given by*

$$T_{\overline{R}}(\overline{R}_{\min}) = n \left(s_{\max}\right)^2.$$

The optimal solutions for $T(\overline{R}_{\min})$ *is obtained for*

$$C_j = \frac{n}{\gamma s_{\max}}$$

with $1 \leq j \leq n$ *respectively.*

Proof. The feasible set $\mathcal{X}$ is compact and the objective function in (6.3) is continuous. As a consequence Problem 1 has a feasible solution if and only if there is an optimal solution for Problem 1. Considering constraint (6.5) and (6.6), it is easy to see that $\gamma s_{\max} \geq n s_{\min}$ is necessary for feasibility. From now on we assume $\gamma s_{\max} \geq n s_{\min}$. It is obvious that it

follows for $R_1, R_2 \in I$ with $R_1 < R_2$ that Problem 1 with response bound $R_3 \in (R_1, R_2)$ has a feasible (and therefore also an optimal) solution. As a consequence I is an interval.

Lemma 6.2 implies that $I \neq \emptyset$ and

$$\min_{\overline{R} \in \mathbb{R}} I = \frac{n^2}{\gamma s_{\max}} = \overline{R}_{\min}.$$

Furthermore it follows from Lemma 6.2 that for Problem 1 with response bound $\overline{R}_{\min}$ an optimal solution is given by

$$C_j = \frac{n}{\gamma s_{\max}} \quad 1 \leq j \leq n \qquad \text{and} \qquad C_0 = 0.$$

The optimal objective function value for Problem 1 with response bound $\overline{R}_{\min}$ is given by

$$T(\overline{R}_{\min}) = \gamma \left(\frac{n}{\gamma s_{\max}} \right) \left(\frac{n}{\gamma} \frac{\gamma s_{\max}}{n} \right)^3 = n \left(s_{\max} \right)^2.$$

It is easy to see that I is right-unbounded which completes the proof. □

Lemma 6.4. *We assume $\gamma s_{\max} \geq n s_{\min}$. Let*

$$\overline{R}_{\max} = \frac{n}{s_{\min}}$$

as before. Then the following statements hold:

1. *We have $\overline{R}_{\min} \leq \overline{R}_{\max}$. So $\overline{R}_{\max} \in I = [\overline{R}_{\min}, \infty)$ holds.*
2. *An optimal solution $C_0^*, C_1^*, \ldots, C_n^*$ for Problem 1 with response bound $R \in [\overline{R}_{\min}, \overline{R}_{\max}]$ satisfies $\sum_{j=1}^n C_j^* = R$.*
3. *T is strictly monotonic decreasing on $[\overline{R}_{\min}, \overline{R}_{\max}]$.*
4. *An optimal solution $C_0^*, C_1^*, \ldots, C_n^*$ for Problem 1 with response bound $R \in [\overline{R}_{\max}, \infty)$ satisfies $\sum_{j=1}^n C_j^* = \overline{R}_{\max}$. T is constant on $[\overline{R}_{\max}, \infty)$, more precisely*

$$T(R) = T(\overline{R}_{\max}) = \frac{n^3 \left(s_{\min} \right)^2}{\gamma^2} \qquad \text{for} \qquad R \in [\overline{R}_{\max}, \infty)$$

holds.

Proof. Due to $\gamma s_{\max} \geq n s_{\min}$ we obtain

$$\overline{R}_{\min} = \frac{n^2}{\gamma s_{\max}} \leq \frac{n}{s_{\min}} = \overline{R}_{\max},$$

which proves Statement 1.

In order to prove Statement 2, we consider an optimal solution $C_0^*, C_1^*, \ldots, C_n^*$ for Problem 1 with response bound $R \in [\overline{R}_{\min}, \overline{R}_{\max}]$ and assume $\sum_{j=1}^n C_j^* < R$. If $C_j^* = \frac{1}{s_{\min}}$ for $1 \leq j \leq n$ (i.e. all constraints in (6.5) are actually equalities), then we have $\sum_{j=1}^n C_j^* = \overline{R}_{\max}$ which contradicts our assumption. As a consequence, there exists an index j' with

$$j' = \operatorname*{arg\,max}_{1 \leq j \leq n} \{ j \mid C_j^* < \frac{1}{s_{\min}} \}.$$

We define a new solution $\tilde{C}_0, \tilde{C}_1, \ldots, \tilde{C}_n$ by

$$\tilde{C}_j := \begin{cases} C_j^* & \text{for } j \neq j' \\ C_{j'}^* + \min\{\frac{1}{s_{\min}} - C_{j'}^*, R - \sum_{h=1}^n C_h^*\} & \text{for } j = j'. \end{cases}$$

Note that $\tilde{C}_{j'} > C_{j'}^*$ and that the solution $\tilde{C}_0, \tilde{C}_1, \ldots, \tilde{C}_n$ is feasible by construction. Lemma 6.1 states that the objective function for Problem 1 can be written as a sum of terms of the form $\frac{\gamma}{C_{j_1} C_{j_2}}$ such that each index out of $\{1, \ldots, n\}$ appears in at least one denominator. In particular, the objective function is strictly monotonic decreasing in each component. As a consequence the objective function value for $\tilde{C}_0, \tilde{C}_1, \ldots, \tilde{C}_n$ is strictly better than the objective function value for $C_0^*, C_1^*, \ldots, C_n^*$ which yields a contradiction. This proves Statement 2.

Now we consider $R_1, R_2 \in [\overline{R}_{\min}, \overline{R}_{\max}]$ with $R_1 < R_2$. We denote an optimal solution for Problem 1 with response bound R_1 by $C_0^{(1)}, C_1^{(1)}, \ldots, C_n^{(1)}$. As before, there exists j'' with

$$j'' = \operatorname*{arg\,max}_{1 \leq j \leq n} \{ j \mid C_j^{(1)} < \frac{1}{s_{\min}} \}.$$

We define a new solution $C_0^{(2)}, C_1^{(2)}, \ldots, C_n^{(2)}$ by

$$C_j^{(2)} := \begin{cases} C_j^{(1)} & \text{for } j \neq j'' \\ C_{j''}^{(1)} + \min\{\frac{1}{s_{\min}} - C_{j''}^{(1)}, R_2 - \sum_{h=1}^n C_h^{(1)}\} & \text{for } j = j''. \end{cases}$$

Inequality $C_{j''}^{(2)} > C_{j''}^{(1)}$ is satisfied and $C_0^{(2)}, C_1^{(2)}, \ldots, C_n^{(2)}$ is feasible for Problem 1. In Problem 1 with response bound R_2, the objective function value for $C_0^{(1)}, C_1^{(1)}, \ldots, C_n^{(1)}$ is larger than the objective function value for $C_0^{(2)}, C_1^{(2)}, \ldots, C_n^{(2)}$. This proves $T(R_1) > T(R_2)$ for $R_1 < R_2$ and $R_1, R_2 \in [\overline{R}_{\min}, \overline{R}_{\max}]$, so Statement 3 is proven.

Finally, we consider $R \in [\overline{R}_{\max}, \infty)$. We denote an optimal solution for Problem 1 with response bound R by $C_0^{(3)}, C_1^{(3)}, \ldots, C_n^{(3)}$. Due to the constraint in (6.5), we obtain $\sum_{j=1}^n C_j^{(3)} \leq \frac{n}{s_{\min}} = \overline{R}_{\max}$. It follows that T is constant on $[\overline{R}_{\max}, \infty)$. Using the same argument as above it can be shown that $\sum_{j=1}^n C_j^{(3)} = \overline{R}_{\max}$. As a consequence, $C_j^{(3)} = \frac{1}{s_{\min}}$ holds for $1 \leq j \leq n$ and the objective function value is given by

$$T(R) = T(\overline{R}_{\max}) = \gamma \frac{1}{s_{\min}} \left(\frac{1}{\gamma} n s_{\min} \right)^3 = \frac{n^3 \left(s_{\min}\right)^2}{\gamma^2},$$

which shows Statement 4. □

Lemma 6.5. *We consider positive real numbers $a_1, \ldots, a_n, b_1, \ldots, b_n$, a positive constant K and $\lambda \in [0,1]$. We assume that $\sum_{j=1}^{n} \frac{1}{a_j} \leq K$ and $\sum_{j=1}^{n} \frac{1}{b_j} \leq K$. Then the inequality*

$$\sum_{j=1}^{n} \frac{1}{\lambda a_j + (1-\lambda) b_j} \leq K$$

holds.

Proof. Due to the convexity of the mapping $x \mapsto \frac{1}{x}$, we obtain for $j \in \{1, \ldots, n\}$ the inequality

$$\frac{1}{\lambda a_j + (1-\lambda) b_j} \leq \frac{\lambda}{a_j} + \frac{1-\lambda}{b_j}.$$

By summing up we get

$$\sum_{j=1}^{n} \frac{1}{\lambda a_j + (1-\lambda) b_j} \leq \lambda \sum_{j=1}^{n} \frac{1}{a_j} + (1-\lambda) \sum_{j=1}^{n} \frac{1}{b_j} \leq \lambda K + (1-\lambda) K = K.$$

□

Lemma 6.6. *The trade-off function T is strictly convex on $[\overline{R}_{\min}, \overline{R}_{\max}]$ and continuous in $\overline{R}_{\max}$.*

Proof. If $\gamma s_{\max} < n s_{\min}$ holds, then $I = \emptyset$ and there is nothing to show. So we assume $\gamma s_{\max} \geq n s_{\min}$ in the following. We consider $R_1, R_2 \in [\overline{R}_{\min}, \overline{R}_{\max}]$ with $R_1 < R_2$ and $\lambda \in (0,1)$ and prove $T(\lambda R_1 + (1-\lambda) R_2) < \lambda T(R_1) + (1-\lambda) T(R_2)$. To prove this, we consider optimal solutions $C^{(1)} \coloneqq (C_0^{(1)}, C_1^{(1)}, \ldots, C_n^{(1)})$ and $C^{(2)} \coloneqq (C_0^{(2)}, C_1^{(2)}, \ldots, C_n^{(2)})$ for Problem 1 with response bounds R_1 and R_2 respectively. Since T is strictly monotonic decreasing on $[\overline{R}_{\min}, \overline{R}_{\max}]$, the considered optimal solutions $C^{(1)}$ and $C^{(2)}$ differ in at least one component. The solution $\lambda C^{(1)} + (1-\lambda) C^{(2)}$ is feasible for Problem 1 with response bound $\lambda R_1 + (1-\lambda) R_2$ due to the following:

- Considering constraint (6.4), if $\sum_{j=1}^{n} C_j^{(1)} \leq R_1$ and $\sum_{j=1}^{n} C_j^{(2)} \leq R_2$ then we have

$$\sum_{j=1}^{n} \lambda C_j^{(1)} + (1-\lambda) C_j^{(2)} \leq \lambda R_1 + (1-\lambda) R_2.$$

- Considering constraint (6.5), if $C_j^{(1)} \leq \frac{1}{s_{\min}}$ and $C_j^{(2)} \leq \frac{1}{s_{\min}}$ for $1 \leq j \leq n$, then we also have

$$\lambda C_j^{(1)} + (1-\lambda) C_j^{(2)} \leq \frac{1}{s_{\min}} \quad \text{for} \quad 1 \leq j \leq n.$$

- Considering constraint (6.6), if $\sum_{j=1}^{n} \frac{1}{C_j^{(1)}} \leq \gamma s_{\max}$ and $\sum_{j=1}^{n} \frac{1}{C_j^{(2)}} \leq \gamma s_{\max}$, then by Lemma 6.5 we also have

$$\sum_{j=1}^{n} \frac{1}{\lambda C_j^{(1)} + (1-\lambda) C_j^{(2)}} \leq \gamma s_{\max}.$$

- Considering constraint (6.7), if $C_0^{(1)} \leq C_1^{(1)} \leq \ldots \leq C_n^{(1)}$ and $C_0^{(2)} \leq C_1^{(2)} \leq \ldots \leq C_n^{(2)}$, then we also have

$$\alpha C_0^{(1)} + (1-\alpha)C_0^{(2)} \leq \alpha C_1^{(1)} + (1-\alpha)C_1^{(2)} \leq \ldots \leq \alpha C_n^{(1)} + (1-\alpha)C_n^{(2)}.$$

So $\lambda C^{(1)} + (1-\lambda)C^{(2)}$ is indeed feasible for Problem 1 with response bound $\lambda R_1 + (1-\lambda)R_2$ and therefore we have

$$\begin{aligned} T(\lambda R_1 + (1-\lambda)R_2) &\leq f(\lambda C^{(1)} + (1-\lambda)C^{(2)}) \\ &< \lambda f(C^{(1)}) + (1-\lambda) f(C^{(2)}) = \lambda T(R_1) + (1-\lambda)T(R_2). \end{aligned}$$

In the second inequality we have used Lemma 6.1 and the fact that $C^{(1)}$ and $C^{(2)}$ differ in at least one component. This proves the strict convexity of T on $[\overline{R}_{\min}, \overline{R}_{\max}]$.

In order to show that T is continuous in $\overline{R}_{\max}$ we show that T is convex on $[\overline{R}_{\min}, \infty)$. Therefore we consider $\tilde{\lambda} \in [0,1]$ and $\tilde{R}_1, \tilde{R}_2 \in [\overline{R}_{\min}, \infty)$ with $\tilde{R}_1 \leq \tilde{R}_2$. Using the same arguments as above we obtain

$$T(\tilde{\lambda}\tilde{R}_1 + (1-\tilde{\lambda})\tilde{R}_2) \leq \tilde{\lambda} T(\tilde{R}_1) + (1-\tilde{\lambda})T(\tilde{R}_2),$$

which proves the convexity of T on $[\overline{R}_{\min}, \infty)$ and as a consequence the continuity of T in $\overline{R}_{\max}$. Note that we do not obtain strict convexity since the optimal solutions for Problem 1 with response bounds $\tilde{R}_1$ and $\tilde{R}_2$ might be identical. □

Putting all lemmas together we obtain a proof for Theorem 6.1.

6.2 Some Notes on Core Based Energy Models for More Complex Job Instances

In Section 6.1 we have studied the trade-off between the energy consumption and the total flow time in detail for the special case of identical jobs. However, the assumption that all jobs are identical is normally too restrictive for applications in cloud computing.

In order to obtain a core based energy model that is suitable to be applied in cloud computing systems, the model should model the following features:

- The release times and the processing volumes of the jobs should be arbitrary.
- The immediate start condition should be satisfied. In order to make it possible to start jobs almost at their release time, a scheduling algorithm is required to be very fast.
- The model should take into account at least one further objective (for example the total flow time). The reason for that is that a scheduling model that minimises only the energy consumption tends to execute jobs as slowly as possible in order to save energy which is typically not wanted in practical applications.

We now consider again the core based energy model proposed in Section 6.1. By considering the corresponding convex programming formulation in Problem 1, it seems to be unlikely that it is possible to obtain a more general core based energy model that meets the features from above. So the contribution of this section is the insight that analytically more complex energy models may lead to insurmountable difficulties. This impression got reinforced by a discussion with Peter Garraghan [24] who confirmed that the energy consumption is seen as difficult to handle analytically and that for this reason empiric energy models are often preferred in the literature on applied scheduling models for cloud computing systems.

For this reason we put our focus back on Model 1 for the energy consumption as proposed in Section 5.1. In Section 7 we give a short literature overview on energy efficient scheduling algorithms and in Section 8 we propose scheduling models that take into account (some of) the features from above.

7 Known Results on Energy Efficient Scheduling Algorithms

In Section 4.1.1 the high importance of the energy consumption as an objective function for distributed computing systems has already been pointed out. The aim of this section is to present an overview of *existing energy efficient algorithms*. There are several survey papers by Albers [1], [2] and [3]. The last one is an introductory survey which is less formal than the others. In the context of energy efficient scheduling, there are basically two techniques to reduce the energy consumption:

- *Power-down mechanisms*: A system is considered to have one active state and several low-power states (for example standby or sleeping mode). When an active system is idle, one can decide whether to transit the system into a low-power state (for example standby or sleeping mode) or to keep the system active. The energy costs for transitioning the system into a low-power state are usually negligible, but transitioning the system back into the active state incurs a significant amount of energy. The goal is to find a state transition schedule that minimises the energy consumption.

- *Speed scaling:* Nowadays sold microprocessors are usually able to operate at variable speed. Typically, higher speed results in better performance but requires more energy. The goal is to use the full speed spectrum of a processor in order to save energy.

7.1 Results for Speed Scaling Problems with a Single Objective Function

Due to the focus of this thesis, we will only list papers which deal with the speed scaling issue. An algorithm for a speed scaling problem must make two decisions at any given time:

- A *job scheduling policy* must determine which job to be scheduled next.

- A *speed scaling policy* must determine how fast to execute the selected job.

In practical applications, one is usually interested in minimising the energy consumption together with a certain Quality of Service (QoS) measure. In general these objectives are in opposition. This means that a higher QoS requires a higher energy consumption in most applications.

Early theoretical investigations into this class of problems were initiated by Yao F., Demers and Shenker in their initial paper [40] in 1995. They considered a scheduling problem with n jobs on a single processor. For each job j a release time r_j, a due date d_j and a processing volume w_j are given. If a job is executed at speed s, it takes $\frac{w_j}{s}$ time units to complete the jobs. If the processor is executed at speed s, the power consumption is assumed to be given by $P(s) = s^\alpha$ (they assume $\alpha \geq 2$) and the energy

consumption is then given by integrating the power consumption over time. They allow preemption, so the execution of a job can be suspended and resumed later. The goal is to minimise energy consumption such that every job meets its deadline constraint (so the QoS measure is deadline feasibility). For the offline version of this problem, their presented algorithm YDS (named after the three authors) yields an optimal solution. A straightforward implementation of YDS requires $\mathcal{O}(n^3)$ time. In 2005, Li, Yao A. and Yao F. [31] presented a new algorithm for the same problem with running time $\mathcal{O}(n^2)$.

The online version of this problem was also studied in [40] by Yao F., Demers and Shenker. In the online version, one learns about w_j and d_j when job j becomes available. The *competitive ratio* of an online algorithm is defined as the worst-case ratio between the cost of the solution found by the online algorithm and the cost of an optimal solution. For a solid introduction to the area of online algorithms see e.g. the text book by Borodin and El-Yaniv [15]. Two simple online heuristics, Average Rate and Optimal Available, were proposed. In [40], they show that Average Rate computes a feasible schedule and the competitive ratio r of Average Rate is bounded by $\alpha^\alpha \leq r \leq 2^{\alpha-1}\alpha^\alpha$ for any $\alpha \geq 2$.

In [9], Bansal, Kimbrel and Pruhs showed that Optimal Available has a competitive ratio of exactly α^α. It can be concluded that Average Rate typically outperforms Optimal Available in terms of running time, but Optimal Available outperforms Average Rate in terms of competitiveness. Also in [9], a new online heuristic BKP (named after the three authors) was proposed. They showed that a competitive ratio of $2\left(\frac{\alpha}{\alpha-1}\right)^\alpha e^\alpha$ can be achieved for algorithm BKP. It can be concluded that algorithm BKP outperforms algorithm Optimal Available in terms of competitiveness for sufficiently large α.

7.2 Bi-Criteria Problems of Minimising Total Flow and Energy

Early research on energy efficient algorithms typically used QoS measures involving deadlines. In [37], Pruhs, Uthaisombut and Woeginger initiated the study of the bi-criteria problem of minimising average response time (the QoS measure in this case) and energy consumption. They consider the offline problem of scheduling n jobs on one machine with given release dates r_i, processing volumes w_i and a given energy bound $\overline{E}$. Execution speed, power consumption and energy consumption are modelled in the same way as above. Preemption is assumed to be allowed. The goal is to minimise the total response time such that the total energy consumption does not exceed $\overline{E}$.

They first studied the equal work case, where $w_j = 1$ for all j can be assumed without loss of generality. They considered the two job scheduling policies First-Come-First-Serve (FCFS) and Shortest-Remaining-Processing-Time (SRPT). The job scheduling policy FCFS non-preemptively schedules jobs according to the order that they arrive. SRPT is a preemptive job scheduling policy that always runs the job with the least remaining work that has been released but not completed. We note that the problem for the unit work case can be denoted by $1 \mid \text{pmtn}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$ using our notation.

For the equal work case, it is easy to see that both scheduling policies First-Come-First-Serve (FCFS) and Shortest-Remaining-Processing-Time (SRPT) behave identically and are optimal job scheduling policies. As a consequence, one can restrict the

consideration to schedules where no jobs are preempted and such that $C_1 < \ldots < C_n$ is satisfied. This observation is used in [37] to find an optimal solution for the unit work problem by determining an optimal solution for the following convex programming problem:

$$\begin{array}{lll} \min & \sum_{j=1}^{n} C_j & \\ \text{s.t.} & \sum_{j=1}^{n} \frac{1}{t_j^{\alpha-1}} & \leq \overline{E} \\ & C_{j-1} + t_j & \leq C_j \qquad 1 \leq j \leq n \\ & r_j + t_j & \leq C_j \qquad 1 \leq j \leq n \\ & -t_j & \leq -\frac{1}{\overline{E}^{\frac{1}{\alpha-1}}} \qquad 1 \leq j \leq n. \end{array} \tag{7.1}$$

We note that the last restriction in (7.1) follows from the first restriction and is used to ensure $t_j > 0$ for all j. By use of the Karush-Kuhn-Tucker conditions (see for example [32] or [16]) they developed an efficient algorithm that determines an optimal solution. In their approach, they first determine an optimal schedule for the case of a sufficiently large energy bound. Then they decrease the energy bound and show that there are only polynomially many energy levels at which the corresponding schedule changes and explain how to find the new optimal schedule. When the energy bound is decreased to $\overline{E}$, the optimal schedule is determined. The algorithm has a running time of $\mathcal{O}(n^2 \log L)$, where L is the range of possible energies divided by the precision that they desire. They also point out that the approach of their algorithm is problematic for the generalised problem with arbitrary processing volumes as the energy optimal schedule is not a smooth function of the energy bound $\overline{E}$. Thus a new approach is required for the general case.

Very recently Barcelo showed in his PhD thesis [11] that both the problems $1 \mid \text{pmtn}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ as well as $1 \mid \text{pmtn}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum v_j C_j$ are NP-hard.

In contrast it was noticed in [37] that the algorithm for the unit work case can be used to obtain an approximate algorithm for the general problem $1 \mid \text{pmtn}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ that is $\mathcal{O}(\frac{1}{\varepsilon})$-approximate given an additional factor of $(1+\epsilon)^\alpha$ energy.

Albers and Fujiwara [4] initiated combining the objectives energy consumption and flow time into the single objective of energy consumption plus flow time. They considered problem $1 \mid w_j = 1 \mid (\sum F_j + \text{energy})$ in both online and offline version. For the offline version of problem $1 \mid w_j = 1 \mid (\sum F_j + \text{energy})$ they obtain an algorithm that solves the problem in $\mathcal{O}(n^3 \log(\rho))$, where ρ is the inverse of the desired precision. For the online version of problem $1 \mid w_j = 1 \mid (\sum F_j + \text{energy})$ they obtain an approximation algorithm with a competitive ratio that is bounded by $8.22e \left(\frac{3+\sqrt{5}}{2}\right)^\alpha$. They also showed that there is no online algorithm that achieves a bounded competitive ratio for the online version of problem $1 \mid w_j \mid (\sum F_j + \text{energy})$. In more detail, they showed that for the online version of problem $1 \mid w_j \mid (\sum F_j + \text{energy})$, the competitive ratio of any deterministic online algorithm is $\Omega(n^{1-\frac{1}{\alpha}})$.

Bansal, Pruhs and Stein [10] continued this line of research and considered scheduling models that allow preemption. They proposed an algorithm that achieves a competitive

ratio of 4 for the online version of problem $1 \mid \text{pmtn}, w_j = 1 \mid (\sum F_j + \text{energy})$. They also considered the problem with arbitrary work and weighted flow time plus energy. For $\varepsilon > 0$, they define

$$\tilde{\gamma} := \max\left(2, \frac{2(\alpha-1)}{\alpha - (\alpha-1)^{1-\frac{1}{\alpha-1}}}\right) \qquad \text{and} \qquad \mu_\varepsilon := \max\left(1+\frac{1}{\varepsilon}, (1+\varepsilon)^\alpha\right).$$

In [10] they provide an online algorithm for the online version of problem $1 \mid \text{pmtn}, w_j \mid (\sum v_j F_j + \text{energy})$ with competitive ratio $\tilde{\gamma}\mu_\varepsilon$.

A big step forward was achieved by Bansal, Chan and Pruhs in 2009. In [7] they initiated the study of abitrary power functions. (See also [8] for the extended and improved journal version.) Work previously to [7] and [8] focused mainly on power consumption functions of the form s^α for $\alpha > 1$ (often $\alpha = 3$ was chosen according to the well known cube-root rule). In [7] and [8] they consider far more general power consumption functions. We briefly summarize the results of the more general paper [8] in the following. In [8], the power consumption function P is defined on its domain D and is required to satisfy just the following conditions:

- The domain D of P consists of an arbitrary collection of discrete points (possibly empty) and a collection of disjoint intervals.
- Power consumption function P is nonnegative on D.

Using a power consumption function that satiesfies the conditions from above, Bansal et al. proposed an algorithm for the online version of problem $1 \mid \text{pmtn}, w_j \mid (\sum F_j + \text{energy}_\text{g})$ that is 3-competitive. The string "energy_g" points out that a general power consumption function (that meets the conditions from above) is used to calculate the energy consumption. The obtained 3-competitiveness is remarkable since previously no guarantee independent of α was known even for the special case of $P(s) = s^\alpha$.

Results that further improve the work of Bansal, Chan and Pruhs from above were established by Andrew, Wierman and Tang [5]. In [5] they propose an algorithm that is $(2+\varepsilon)$-competitive under arbitrary non-negative and unbounded power consumption functions for arbitrary $\varepsilon > 0$. Further, [5] highlight that their competitive ratio is tight in two senses:

- For any power function, there is an instance of arriving jobs such that the algorithm proposed in [5] is a factor of arbitrarily close to 2 worse than optimal.
- For any scaling algorithm, there is a power function such that the competitive ratio under that power function is arbitrarily close to 2.

We note that in particular the tightness-result above makes the work of [5] very important. However, there exist other papers that beat the competitive ratio of [5] for some special values of α (see [20] for example).

8 Optimising Energy Consumption and Total Flow Time including On-Demand Scheduling

This section contains the main original contribution of this thesis from the algorithmic point of view.

We consider the scheduling problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ where for each job j there is given its work amount w_j. The jobs arrive over time and have to be started immediately at their release times. The goal is to minimise the total flow time such that a given energy limit $\overline{E}$ is not exceeded.

We have to schedule the jobs in order of their release times. So the only remaining decision a scheduler has to make is to choose the execution time t_j for every job j, or equivalently, to choose the speed s_j of processing job j,

$$s_j = \frac{w_j}{t_j}. \tag{8.1}$$

As we work with Model 1 as the energy model in this thesis (see Section 5.1), the energy E_j consumed by job j is

$$E_j = \int_{r_j}^{r_j+t_j} s_j^\alpha \,\mathrm{d}\tau = t_j s_j^\alpha = \frac{w_j^\alpha}{t_j^{\alpha-1}} = w_j s_j^{\alpha-1}$$

for $\alpha > 1$. We then get

$$\sum_{j=1}^{n} E_j = \sum_{j=1}^{n} w_j s_j^{\alpha-1}$$

for the total energy consumed by all jobs. An energy limit for the total energy consumed by all jobs is given by $\overline{E}$, so that

$$\sum_{j=1}^{n} w_j s_j^{\alpha-1} \leq \overline{E}$$

has to be satisfied. An upper bound on the job execution time is incurred by the length of the available interval that cannot be exceeded as this would postpone the next job:

$$t_j \leq r_{j+1} - r_j,$$

where for notational convenience we set $r_{n+1} = \infty$. By use of (8.1), we obtain a lower bound ℓ_j on the execution speed:

$$\frac{w_j}{s_j} \leq r_{j+1} - r_j \qquad \text{or equivalently} \qquad s_j \geq \frac{w_j}{r_{j+1} - r_j} =: \ell_j$$

We assume

$$\sum_{j=1}^{n} w_j \ell_j^{\alpha-1} \leq \overline{E},$$

because otherwise the problem would be infeasible. We start with the single machine case.

8.1 $1 \mid \textbf{imst}, w_j, \textbf{energy} \leq \overline{E} \mid \sum C_j$

For the case of different processing volumes w_j and on a single machine we may assume $r_1 < r_2 < \ldots < r_n$. If there are two jobs with equal release times, then the problem is infeasible since it would not be possible to satisfy the immediate start condition. The mathematical programming formulation for problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ is given by

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} C_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} w_j s_j^{\alpha-1} \leq \overline{E} \\ & s_j \geq \ell_j \qquad\qquad 1 \leq j \leq n. \end{aligned} \tag{8.2}$$

According to the immediate start condition we have $C_j = r_j + t_j$. The objective function $\sum_{j=1}^{n} C_j = \sum_{j=1}^{n} t_j + \sum_{j=1}^{n} r_j$ is equivalent to $\sum_{j=1}^{n} t_j = \sum_{j=1}^{n} \frac{w_j}{s_j}$ since $\sum_{j=1}^{n} r_j$ is a constant. Therefore the formulation in (8.2) is equivalent to

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} \frac{w_j}{s_j} \\ \text{s.t.} \quad & \sum_{j=1}^{n} w_j s_j^{\alpha-1} \leq \overline{E} \\ & s_j \geq \ell_j \qquad\qquad 1 \leq j \leq n. \end{aligned} \tag{8.3}$$

In order to solve problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ we present an algorithm that solves the optimisation problem in (8.3). The following lemma is essential for the algorithm.

Lemma 8.1. *The execution speeds $s_1^*, \ldots, s_n^*$ are optimal for problem (8.3) if and only if there exist sets $\mathcal{J}_1, \mathcal{J}_2 \subseteq \{1, \ldots, n\}$ with $\mathcal{J}_1 \cup \mathcal{J}_2 = \{1, \ldots, n\}$ and $\mathcal{J}_1 \cap \mathcal{J}_2 \neq \emptyset$ such that the following conditions are satisfied:*

(i) $\forall j \in \mathcal{J}_1 : s_j^* = \ell_j$

(ii) $\forall j \in \mathcal{J}_2 : s_j^* > \ell_j$

(iii) $\forall j_1, j_2 \in \mathcal{J}_2 : s_{j_1}^* = s_{j_2}^*$

(iv) $\forall j_1 \in \mathcal{J}_1, \forall j_2 \in \mathcal{J}_2 : s_{j_1}^* \geq s_{j_2}^*$

(v) $\sum_{j=1}^{n} w_j s_j^{*\alpha-1} = \overline{E}$

We will prove Lemma 8.1 by use of the Karush-Kuhn-Tucker conditions (KKT). For a general introduction to the KKT-conditions see the widely spread textbooks [32] and [16] for example. In the following we present related results that can directly applied to the convex programming formulation in (8.3). These results were taken from [16].

Definition 8 (KKT conditions)**.** Let f_0 and $f_1, \ldots, f_m$ be convex and differentiable functions. We consider the problem

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f_0(x) \\ &\text{s.t. } f_i(x) \leq 0 \qquad\qquad i = 1, \ldots, m. \end{aligned}$$

For this problem, the *Karush-Kuhn-Tucker conditions* hold if there exists $(\tilde{x}, \tilde{\lambda}) \in \mathbb{R}^n \times \mathbb{R}^m$ with $\tilde{\lambda}^T = (\tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ such that

$$\begin{aligned} \nabla f_0(\tilde{x}) + \sum_{i=1}^{m} \tilde{\lambda}_i \nabla f_i(\tilde{x}) &= 0 \\ f_i(\tilde{x}) &\leq 0 \qquad i = 1, \ldots, m \qquad (8.4) \\ \tilde{\lambda}_i &\geq 0 \qquad i = 1, \ldots, m \\ \tilde{\lambda}_i f_i(\tilde{x}) &= 0 \qquad i = 1, \ldots, m \qquad (8.5) \end{aligned}$$

are satisfied. The values $\tilde{\lambda}_i$ for $i = 1, \ldots, m$ are called *Lagrange multipliers*. Condition (8.4) ensures feasibility and condition (8.5) is called *complementary slackness condition*.

Theorem 8.1 (Sufficient KKT conditions)**.** *Let f_0 and $f_1, \ldots, f_m$ be convex and differentiable functions. Consider the problem*

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f_0(x) \\ &\text{s.t. } f_i(x) \leq 0 \qquad\qquad i = 1, \ldots, m. \end{aligned}$$

If there exists $(x^*, \lambda^*) \in (\mathbb{R}^n, \mathbb{R}^m)$ such that (x^*, λ^*) satisfies the KKT conditions, then x^* is an optimal solution.

Theorem 8.2. *Let f_0 and $f_1, \ldots, f_m$ be convex and differentiable functions and let x^* be an optimal solution for the problem*

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f_0(x) \\ &\text{s.t. } f_i(x) \leq 0 \qquad\qquad i = 1, \ldots, m. \end{aligned}$$

If there exist $\tilde{x} \in \mathbb{R}^n$ with $f_i(\tilde{x}) < 0$ for $1 \leq i \leq m$, then there exists $(x^*, \lambda^*) \in (\mathbb{R}^n, \mathbb{R}^m)$ such that (x^*, λ^*) satisfies the KKT conditions. The condition $f_i(\tilde{x}) < 0$ for $1 \leq i \leq m$ is known as Slater's condition in the literature.

Now we prove Lemma 8.1.

Proof of Lemma 8.1. At first we assume that we are given $\mathcal{J}_1, \mathcal{J}_2 \subseteq \{1, \ldots, n\}$ with $\mathcal{J}_1 \cup \mathcal{J}_2 = \{1, \ldots, n\}$ and $\mathcal{J}_1 \cap \mathcal{J}_2 = \emptyset$ such that $s_1^*, \ldots, s_n^*$ satisfy the conditions (i) - (v) of

Lemma 8.1. We will construct Lagrange multipliers $\lambda_0^*, \lambda_1^*, \ldots, \lambda_n^*$ such that $\lambda_0^*, \lambda_1^*, \ldots, \lambda_n^*$ and $s_1^*, \ldots, s_n^*$ satisfy the KKT conditions. Then by Theorem 8.1 the optimality of $s_1^*, \ldots, s_n^*$ will follow.

For problem (8.3) the KKT conditions are given by

$$-\frac{w_j}{s_j^2} + \lambda_0 \left(w_j(\alpha-1)s_j^{\alpha-2}\right) - \lambda_j = 0 \qquad j = 1, \ldots, n \tag{8.6}$$

$$\sum_{j=1}^{n} w_j s_j^{\alpha-1} - \overline{E} \leq 0 \tag{8.7}$$

$$\ell_j - s_j \leq 0 \qquad j = 1, \ldots, n \tag{8.8}$$

$$\lambda_0, \lambda_j \geq 0 \qquad j = 1, \ldots, n \tag{8.9}$$

$$\lambda_0 \left(\sum_{j=1}^{n} w_j s_j^{\alpha-1} - \overline{E} \right) = 0 \tag{8.10}$$

$$\lambda_j(\ell_j - s_j) = 0 \qquad j = 1, \ldots, n \tag{8.11}$$

Because of conditions (i), (ii) and (v) the execution speeds $s_1^*, \ldots, s_n^*$ satisfy the feasibility conditions (8.7) and (8.8). According to (v), we have $\sum_{j=1}^{n} w_j s_j^{\alpha-1} - \overline{E} = 0$ and therefore also condition (8.10) is satisfied. In order to show that the other conditions are satisfied as well, we make a case distinction between jobs in $\mathcal{J}_1$ and $\mathcal{J}_2$.

Case job $j \in \mathcal{J}_2$: Since all jobs in $\mathcal{J}_2$ have the same execution speed according to (iii), we can define $\tilde{s} := s_j^*$ for $j \in \mathcal{J}_2$. We have $\tilde{s} > \ell_j$, so we have to set $\lambda_j^* := 0$ for $j \in \mathcal{J}_2$ in order to satisfy condition (8.11). Then condition (8.6) reads for $j \in \mathcal{J}_2$ as

$$-\frac{w_j}{\tilde{s}^2} + \lambda_0 \left(w_j(\alpha-1)\tilde{s}^{\alpha-2}\right) = \underbrace{\frac{w_j}{\tilde{s}^2}}_{>0} \left(-1 + \lambda_0 \underbrace{(\alpha-1)\tilde{s}^{\alpha}}_{>0} \right) = 0.$$

Therefore we have to set $\lambda_0^* := \frac{1}{(\alpha-1)\tilde{s}^{\alpha}} > 0$ in order to satisfy (8.6). By now condition (8.9) for $j \in \mathcal{J}_2$ is already satisfied. Note that $r_{n+1} = \infty$ implies $n \in \mathcal{J}_2$, so λ_0^* is well-defined in any case.

Case job $j \in \mathcal{J}_1$: We already know that conditions (8.7),(8.8) and (8.10) are satisfied. Using $\lambda_0^* = \frac{1}{(\alpha-1)\tilde{s}^{\alpha}}$ from the previous case, condition (8.6) reads for $j \in \mathcal{J}_1$ as

$$\begin{aligned} -\frac{w_j}{\ell_j^2} + \lambda_0^* \left(w_j(\alpha-1)\ell_j^{\alpha-2}\right) - \lambda_j &= -\frac{w_j}{\ell_j^2} + \frac{1}{(\alpha-1)\tilde{s}^{\alpha}} \left(w_j(\alpha-1)\ell_j^{\alpha-2}\right) - \lambda_j \\ &= \underbrace{\frac{w_j}{\ell_j^2}}_{>0} \left(-1 + \left(\frac{\ell_j}{\tilde{s}}\right)^{\alpha} \right) - \lambda_j = 0. \end{aligned}$$

According to (i) and (iv) we have $\left(\frac{\ell_j}{\tilde{s}}\right)^\alpha \geq 1$ and therefore we can choose $\lambda_j^* \coloneqq \frac{w_j}{\ell_j^2}\left(-1+\left(\frac{\ell_j}{\tilde{s}}\right)^\alpha\right) \geq 0$ which satisfies conditions (8.6) and (8.9). Since $\ell_j - s_j^* = 0$, also the last condition (8.11) is satisfied for $j \in \mathcal{J}_1$.

This means that $s_1^*, \ldots, s_n^*$ together with $\lambda_0^*, \lambda_1^*, \ldots, \lambda_n^*$ satisfy the KKT conditions and therefore $s_1^*, \ldots, s_n^*$ is an optimal solution for problem (8.3).

Now we prove the other direction and assume that $s_1^*, \ldots, s_n^*$ is an optimal solution for problem (8.3). We define the sets $\mathcal{J}_1 \coloneqq \{j \in \{1, \ldots, n\} \mid s_j^* = \ell_j\}$ and $\mathcal{J}_2 \coloneqq \{j \in \{1, \ldots, n\} \mid s_j^* > \ell_j\}$. It follows that $\mathcal{J}_1 \cup \mathcal{J}_2 = \{1, \ldots, n\}$ and $\mathcal{J}_1 \cap \mathcal{J}_2 \neq \emptyset$ holds. We have to show that also the properties (i) - (v) are satisfied. By definition of $\mathcal{J}_1$ and $\mathcal{J}_2$ properties (i) and (ii) are satisfied. Furthermore it is easy to see that $\sum_{j=1}^n w_j {s_j^*}^{\alpha-1} = \overline{E}$ holds. If to the contrary $\sum_{j=1}^n w_j {s_j^*}^{\alpha-1} < \overline{E}$, we can increase the value of an arbitrary s_j^* by a sufficiently small value such that we stay feasible and obtain a strictly better objective function which contradicts the optimality of $s_1^*, \ldots, s_n^*$. As a consequence also property (v) holds.

Next we consider the case if $s_j^* = \ell_j$ holds for $j = 1, \ldots, n$. In this case we have $\mathcal{J}_1 = \{1, \ldots, n\}$ and $\mathcal{J}_2 = \emptyset$ which clearly satisfy conditions (i) - (v) since there is nothing to show for (iii) and (iv). For this reason we can assume from now on that there exists $j' \in \{1, \ldots, n\}$ with $s_{j'}^* > \ell_{j'}$.

Since $\sum_{j=1}^n w_j {s_j^*}^{\alpha-1} = \overline{E}$ and $s_{j'}^* > \ell_{j'}$ we have $\sum_{j=1}^n w_j {\ell_j^*}^{\alpha-1} < \overline{E}$. As a consequence there exists $\varepsilon > 0$ such that, by defining $s_j' \coloneqq \ell_j + \varepsilon$, the inequality $\sum_{j=1}^n w_j {s_j'}^{\alpha-1} < \overline{E}$ holds. As a consequence $s_j', \ldots, s_j'$ is strictly feasible and Slater's condition is satisfied. Theorem 8.2 states that there exists $\lambda_0^*, \lambda_1^*, \ldots, \lambda_n^*$ such that $\lambda_0^*, \lambda_1^*, \ldots, \lambda_n^*$ together with $s_1^*, \ldots, s_n^*$ satisfy the KKT conditions. Now we need again a case distinction between jobs in $\mathcal{J}_1$ and $\mathcal{J}_2$.

Case job $j \in \mathcal{J}_2$: Since $\ell_j^* - s_j < 0$, it follows from (8.11) that $\lambda_j^* = 0$ for $j \in \mathcal{J}_2$. Using $\lambda_j^* = 0$ in (8.6), we know that there exists $\lambda_0^* \geq 0$ such that

$$-\frac{w_j}{{s_j^*}^2} + \lambda_0^*\left(w_j(\alpha-1){s_j^*}^{\alpha-2}\right) = \underbrace{\frac{w_j}{{s_j^*}^2}}_{>0}\left(-1+\lambda_0^*(\alpha-1){s_j^*}^{\alpha}\right) = 0$$

holds for all $j \in \mathcal{J}_2$. This equation can only be satisfied if all values s_j^* for $j \in \mathcal{J}_2$ have the same value $\tilde{s}$. Therefore property (iii) is satisfied and we have

$$\lambda_0^* = \frac{1}{(\alpha-1)\tilde{s}^\alpha}. \tag{8.12}$$

Case job $j \in \mathcal{J}_1$: Considering (8.6) for $j \in \mathcal{J}_1$, we get

$$-\frac{w_j}{{\ell_j}^2} + \lambda_0^* \left(w_j(\alpha-1){\ell_j}^{\alpha-2}\right) - \lambda_j^* = -\frac{w_j}{\ell_j^2} + \frac{1}{(\alpha-1)\tilde{s}^\alpha}\left(w_j(\alpha-1){\ell_j}^{\alpha-2}\right) - \lambda_j^*$$
$$= \underbrace{\frac{w_j}{\ell_j^2}}_{>0}\left(-1 + \left(\frac{\ell_j}{\tilde{s}}\right)^\alpha\right) - \lambda_j^* = 0.$$

Since $\lambda^* \geq 0$ according to (8.9) it follows that

$$\left(\frac{\ell_j}{\tilde{s}}\right)^\alpha \geq 0$$

and since $s_j^* = \ell_j$ for $j \in \mathcal{J}_1$ we obtain $s_j^* \geq \tilde{s}$ for $j \in \mathcal{J}_1$. This implies property (iv).

We have now shown that all properties (i) - (v) are satisfied which finishes the proof. □

Let us apply Lemma 8.1 to see how an optimal solution of problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ looks like in a Gantt chart. For an optimal solution $s_1^*, \ldots, s_n^*$, all jobs $j \in \mathcal{J}_1$ satisfy $s_j^* = \ell_j$. According to (8) and by use of $t_j^* = \frac{w_j}{s_j^*}$ we have

$$\ell_j = \frac{w_j}{r_{j+1} - r_j} = s_j^* \quad \Leftrightarrow \quad t_j^* = r_{j+1} - r_j.$$

This means that in an optimal solution all jobs $j \in \mathcal{J}_1$ fully occupy the available time window $[r_j, r_{j+1}]$. We note that job n is always in set $\mathcal{J}_2$ since $r_{n+1} = \infty$. On the other hand, all jobs $j \in \mathcal{J}_2$ do not occupy their available time window $[r_j, r_{j+1}]$, but are all executed at the same speed according to Lemma 8.1.(iii). Jobs in $\mathcal{J}_1$ are executed at higher speed than jobs in $\mathcal{J}_2$. See Figure 4 for an example.

The characterisation of Lemma 8.1 gives us the main tools to present an algorithm for solving problem (8.3). The algorithm starts by checking $\sum_{j=1}^n w_j \ell_j^{\alpha-1} \leq \overline{E}$. If this inequality is satisfied, there exists a feasible and therefore also an optimal solution. If this is not satisfied, the problem is infeasible. After the feasibility check, all jobs are declared as non-frozen and the execution speeds are all set to infinity $s_1 = \ldots = s_n = \infty$. These values are not feasible, since $\sum_{j=1}^n w_j s_j^{\alpha-1} \leq \overline{E}$ is not satisfied. The algorithm then decreases the values s_j of all non-frozen jobs at the same rate. If for a job j' the equation $s_{j'} = \ell_{j'}$ gets satisfied, job j' will be frozen and the value $s_{j'}$ remains fixed until the end of the algorithm. This procedure is repeated until $\sum_{j=1}^n w_j s_j^{\alpha-1} = \overline{E}$ holds (in this case $s_1, \ldots, s_n$ is optimal according to Lemma 8.1). Pseudocode can be found in Algorithm 1.

It is easy to see that Algorithm 1 can be implemented to run in $\mathcal{O}(n \log(n))$ time. Lines 1 to 5 can be done in $\mathcal{O}(n)$. Before entering the loop in line 6, we introduce a new list of jobs ordered decreasingly according to the lower bounds ℓ_j which requires $\mathcal{O}(n \log(n))$ time. By use of this list lines 7 to 9 can be done in $\mathcal{O}(1)$ time in one iteration. Since there are at most n iterations, lines 6 to 10 can be done in $\mathcal{O}(n)$ time. Everything together leads to a running time of $\mathcal{O}(n \log n)$ for Algorithm 1.

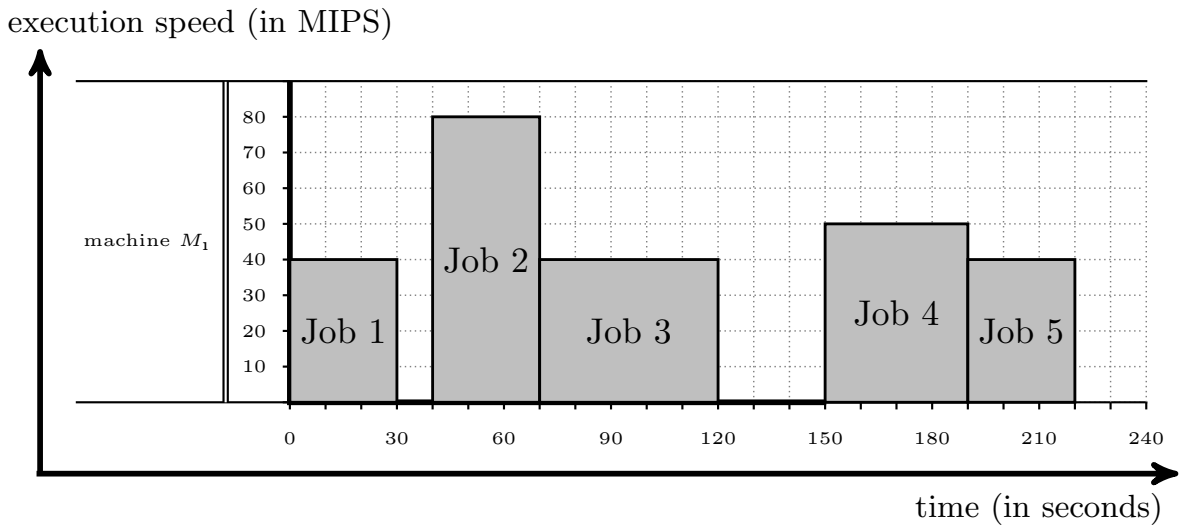

Figure 4: Example of a Gantt chart of an optimal schedule with sets $\mathcal{J}_1 = \{2, 4\}$ and $\mathcal{J}_2 = \{1, 3, 5\}$. Jobs in $\mathcal{J}_1$ fully occupy their available time window and have a higher execution speed than jobs in $\mathcal{J}_2$. All jobs in $\mathcal{J}_2$ are executed at the same speed.

Algorithm 1 Optimal execution speeds for problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$

Input: processing volumes $w_1, \ldots, w_n$, release times $r_1 < \ldots < r_n$, energy bound $\overline{E}$
Output: optimal execution speeds $s_1, \ldots, s_n$ or message that the problem is infeasible

Set $\ell_j \coloneqq \frac{w_j}{r_{j+1} - r_j}$ for $j = 1, \ldots, n$
if $\sum_{j=1}^{n} w_j \ell_j^{\alpha-1} > \overline{E}$ **then** // Check feasibility
  Return: "problem infeasible"
end if
Set $s_1 = \ldots = s_n = \infty, \mathcal{J}_1 \coloneqq \emptyset$ and $\mathcal{J}_2 \coloneqq \{1, \ldots, n\}$ // Initialisation
loop // Iteratively determine values s_j
  Reduce the values s_j for $j \in \mathcal{J}_2$ until event A or event B occurs.
  Event A: There is a job $j' \in \mathcal{J}_2$ with $s_{j'} = \ell_{j'}$. Move job j' from $\mathcal{J}_2$ to $\mathcal{J}_1$. **Repeat**.
  Event B: $\sum_{j=1}^{n} w_j s_j^{\alpha-1} = \overline{E}$**. Return:** $s_1, \ldots, s_n$.
end loop

8.2 $Pm \mid \textbf{imst}, w_j = 1, \textbf{energy} \leq \overline{E} \mid \sum C_j$

Now we are going to study the problem with $m \geq 2$ machines and equal processing volumes. We assume $n > m$, because if $n \leq m$ we can execute each job on a different machine and the problem becomes trivial. Renumber the jobs such that $r_1 \leq \ldots \leq r_n$ is satisfied. We assume that there is no $j \in \{1, \ldots, n-m\}$ such that $r_j = r_{j+1} = \ldots = r_{j+m}$, because otherwise the immediate start condition cannot be satisfied and the problem is infeasible.

In order to solve the problem $Pm \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$, we take a two step approach. In the first step, we decide which job should be executed on which machine. Afterwards, in the second step, we decide at which execution speeds the jobs are executed. Clearly, the optimal strategy in the second step depends on the realisation of the first step. We start with introducing a suitable notation.

Definition 9. Given jobs $j_1, \ldots, j_\ell$ which are executed on machine M_i one after the other in this order, we write $(j_1, j_2, \ldots, j_\ell)_{M_i}$. The tuple $(j_1, j_2, \ldots, j_\ell)_{M_i}$ is called *job sequence on* M_i or *just job sequence*.

Recall that in the first step we have to assign the jobs $1, \ldots, n$ to the machines $M_1, \ldots, M_m$ in a suitable way. Consider such an assignment and denote the jobs assigned to machine M_i by $j_1^{(i)}, \ldots, j_{k_i}^{(i)}$ (with $j_1^{(i)} \leq \ldots \leq j_{k_i}^{(i)}$) respectively. Then we can identify the assignment of jobs $1, \ldots, n$ to machines $M_1, \ldots, M_m$ by the m-tuple of job sequences $\left((j_1^{(1)}, j_2^{(1)}, \ldots, j_{k_1}^{(1)})_{M_1}, \ldots, (j_1^{(m)}, j_2^{(m)}, \ldots, j_{k_m}^{(m)})_{M_m}\right)$.

Definition 10. Let $\left((j_1^{(1)}, j_2^{(1)}, \ldots, j_{k_1}^{(1)})_{M_1}, \ldots, (j_1^{(m)}, j_2^{(m)}, \ldots, j_{k_m}^{(m)})_{M_m}\right)$ denote an m-tuple of job sequences that corresponds to an assignment of jobs $1, \ldots, n$ to machines $M_1, \ldots, M_m$. We call such a m-tuple of job sequences *optimal*, if an optimal solution in the second step based on this m-tuple of job sequences is an overall optimal solution for problem $Pm \mid \text{imst}, w_j = 1, \text{energy} \leq E \mid \sum C_j$ over all m-tuples of job sequences that can be chosen in the first step.

Therefore we need to find an optimal m-tuple of job sequences in the first step and to choose, based on the optimal m-tuple of job sequences, the optimal execution speeds in the second step to solve problem $Pm \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$.

We first explain how to solve the problem in step two given an optimal m-tuple of job sequences.

Lemma 8.2. *Let* $\left((j_1^{(1)}, j_2^{(1)}, \ldots, j_{k_1}^{(1)})_{M_1}, \ldots, (j_1^{(m)}, j_2^{(m)}, \ldots, j_{k_m}^{(m)})_{M_m}\right)$ *be an optimal m-tuple of job sequences which has been chosen in the first step. Then the problem in the second step, calculating the optimal execution times, can be formulated by*

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} \tfrac{1}{s_j} \\ \text{s.t.} \quad & \sum_{j=1}^{n} s_j^{\alpha-1} \leq \overline{E} \\ & s_j \geq \ell_j \qquad 1 \leq j \leq n. \end{aligned} \tag{8.13}$$

The lower bounds on the execution speeds are given as follows

$$\ell_{j_h^{(i)}} \coloneqq \frac{1}{r_{j_{h+1}^{(i)}} - r_{j_h^{(i)}}} \quad \text{for} \quad 1 \leq h < k_i \qquad \text{and} \qquad \ell_{j_{k_i}^{(i)}} \coloneqq 0.$$

Proof. An optimal m-tuple of job sequences specifies which jobs are executed on which machines. This specifies for every job a time window in which the job has to be executed. The stated lower bounds on the execution speeds as well as formulation (8.13) follow. □

It follows from Lemma 8.2 that the problem in the second step is actually a problem of the form $1 \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$. As a consequence the problem in the second step can be solved with the same techniques used in Section 8.1. Algorithm 2 is a modification of Algorithm 1 and solves the problem in the second step.

Algorithm 2 Optimal execution speeds for the second step problem of $Pm \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$

Input: processing volumes $w_1, \ldots, w_n$,
release times $r_1 \leq \ldots \leq r_n$ with $r_{j+m} - r_j > 0$ for $j = 1, \ldots, n-m$,
optimal m-tuple of job sequences $\left((j_1^{(1)}, j_2^{(1)}, \ldots, j_{k_1}^{(1)})_{M_1}, \ldots, (j_1^{(m)}, j_2^{(m)}, \ldots, j_{k_m}^{(m)})_{M_m}\right)$

Output: optimal execution speeds $s_1, \ldots, s_n$ or message that the problem is infeasible

1: Set $\ell_{j_h^{(i)}} \coloneqq \frac{1}{r_{j_{h+1}^{(i)}} - r_{j_h^{(i)}}}$ for $1 \leq i \leq m$ and $1 \leq h < k_i$ and $\ell_{j_{k_i}^{(i)}} \coloneqq 0$
2: **if** $\sum_{j=1}^{n} \ell_j^{\alpha-1} > \overline{E}$ **then** // Check feasibility
3: **Return:** "problem infeasible"
4: **end if**
5: Set $s_1 = \ldots = s_n = \infty, \mathcal{J}_1 \coloneqq \emptyset$ and $\mathcal{J}_2 \coloneqq \{1, \ldots, n\}$ // Initialisation
6: **loop** // Iteratively determine values s_j
7: Reduce the values s_j for $j \in \mathcal{J}_2$ until event A or event B occurs.
8: **Event A:** There is a job $j' \in \mathcal{J}_2$ with $s_{j'} = \ell_{j'}$. Move job j' from $\mathcal{J}_2$ to $\mathcal{J}_1$. **Repeat**.
9: **Event B:** $\sum_{j=1}^{n} s_j^{\alpha-1} = \overline{E}$**. Return:** $s_1, \ldots, s_n$.
10: **end loop**

Having solved the problem in the second step, we now consider the problem of how to determine an optimal m-tuple of job sequences. The answer is given by Theorem 8.3.

Theorem 8.3. *An m-tuple consisting of the job sequences*

$$\begin{array}{c} (1, m+1, 2m+1, 3m+1, \ldots)_{M_1} \\ (2, m+2, 2m+2, 3m+2, \ldots)_{M_2} \\ \vdots \\ (m, 2m, 3m, 4m, \ldots)_{M_m}. \end{array} \tag{8.14}$$

is an optimal m-tuple of job sequences.

Theorem 8.3 states that alternatively assigning the jobs to machines $M_1, \ldots, M_m$ respectively yields an optimal m-tuple of job sequences. In order to prove Theorem 8.3 we need three lemmas at first.

Lemma 8.3. *Consider problem* $P2 \mid imst, w_j = 1, energy \leq \overline{E} \mid \sum C_j$. *There exists at least one optimal* 2*-tuple of job sequences satisfying the following property:*

Job 1 *and job* 2 *are assigned to machine* M_1 *and machine* M_2 *respectively.* $(\ast)$

Proof. Consider an optimal 2-tuple of job sequences T that does not satisfy $(\ast)$. If in T job 1 is assigned to machine M_2 and job 2 is assigned to machine M_1, we can simply switch the roles of M_1 and M_2 to obtain an optimal 2-tuple of job sequences satisfying $(\ast)$. So we can assume that both jobs 1 and 2 are assigned to the same machine in T. We may assume without loss of generality that both jobs are assigned to machine M_2 (because otherwise we can again simply switch the roles of M_1 and M_2). But now we can simply move job 1 from machine M_2 to machine M_1 to obtain a new 2-tuple of job sequences T'. Denote the convex programming formulations as in (8.3) that correspond to T and T' by (P) and (P') respectively. It is clear that the formulations are identical except that the lower bound of job 1 in (P') is smaller or equal than the lower bound of job 1 in (P). As a consequence the optimal objective function value of (P') is smaller or equal than the optimal objective function value of (P). So T' is an optimal 2-tuple of job sequences with property $(\ast)$. □

Lemma 8.4 below will be used to prove Lemma 8.5. The same notation is used in the formulation of both lemmas.

Lemma 8.4. *Given* $r_{h_y} \leq r_{j_x} < r_{j_{x+1}} \leq r_{h_{y+1}}$ *and* $\tilde{E} > 0$. *Consider the following two convex programs:*

$$(P_1)\quad \begin{aligned} \min\ & \tfrac{1}{s_{j_x}} + \tfrac{1}{s_{h_y}} \\ \text{s.t.}\ & s_{j_x}^{\alpha-1} + s_{h_y}^{\alpha-1} = \tilde{E} \\ & s_{j_x} \geq \tfrac{1}{r_{j_{x+1}} - r_{j_x}} \\ & s_{h_y} \geq \tfrac{1}{r_{h_{y+1}} - r_{h_y}} \end{aligned} \qquad (P_2)\quad \begin{aligned} \min\ & \tfrac{1}{s_{j_x}} + \tfrac{1}{s_{h_y}} \\ \text{s.t.}\ & s_{j_x}^{\alpha-1} + s_{h_y}^{\alpha-1} \leq \tilde{E} \\ & s_{j_x} \geq \tfrac{1}{r_{h_{y+1}} - r_{j_x}} \\ & s_{h_y} \geq \tfrac{1}{r_{j_{x+1}} - r_{h_y}} \end{aligned}$$

We assume that (P_1) *has a feasible solution. Then the optimal objective function value of* (P_2) *is smaller or equal than the optimal objective function value of* (P_1).

Proof. Replace the variables s_{j_x} and s_{h_y} by $s'_{j_x} := \frac{1}{s_{j_x}}$ and $s'_{h_y} := \frac{1}{s_{h_y}}$ to obtain two equivalent convex programs:

$$(P'_1)\quad \begin{aligned} \min\ & s'_{j_x} + s'_{h_y} \\ \text{s.t.}\ & \left(s'_{j_x}\right)^{1-\alpha} + \left(s'_{h_y}\right)^{1-\alpha} = \tilde{E} \\ & 0 < s'_{j_x} \leq r_{j_{x+1}} - r_{j_x} =: u_{x,x+1} \\ & 0 < s'_{h_y} \leq r_{h_{y+1}} - r_{h_y} := u_{y,y+1} \end{aligned} \qquad (P'_2)\quad \begin{aligned} \min\ & s'_{j_x} + s'_{h_y} \\ \text{s.t.}\ & \left(s'_{j_x}\right)^{1-\alpha} + \left(s'_{h_y}\right)^{1-\alpha} \leq \tilde{E} \\ & 0 < s'_{j_x} \leq r_{h_{y+1}} - r_{j_x} =: u_{x,y+1} \\ & 0 < s'_{h_y} \leq r_{j_{x+1}} - r_{h_y} =: u_{y,x+1} \end{aligned}$$

Programme (P_1) is equivalent to (P_1') and (P_2) is equivalent to (P_2'). As a consequence it is sufficient to show that the optimal objective function value of (P_2') is smaller or equal to the optimal objective function value of (P_1'). By defining $\varepsilon \coloneqq r_{h_{y+1}} - r_{j_{x+1}}$ we have $u_{x,x+1} + \varepsilon = u_{x,y+1}$ and $u_{y,y+1} - \varepsilon = u_{y,x+1}$. The feasible set of (P_1') is compact and non-empty, so there exists an optimal solution for programme (P_1'). Let $s^*_{j_x}$ and $s^*_{h_y}$ denote an optimal solution for (P_1'). Since we have by assumption $u_{y,y+1} \geq u_{x,x+1}$, we can assume without loss of generality that $s^*_{j_x} \leq s^*_{h_y}$. By choice of

$$\delta \coloneqq \min\{\frac{s^*_{h_y} - s^*_{j_x}}{2}, \varepsilon\}$$

we obtain $\delta \geq 0$, $s^*_{j_x} \leq s^*_{j_x} + \delta \leq s^*_{h_y} - \delta \leq s^*_{h_y}$ as well as $s^*_{j_x} + \delta \leq u_{x,y+1}$ and $s^*_{h_y} - \delta \leq u_{y,x+1}$. Note that the objective function value for (P_2') for $s'_{j_x} = s^*_{j_x} + \delta$ and $s'_{h_y} = s^*_{h_y} - \delta$ is the same as the optimal objective function value for (P_1'). In order to show the feasibility of $s^*_{j_x} + \delta$ and $s^*_{h_y} - \delta$ for (P_2') it remains to show that

$$\left(s^*_{j_x} + \delta\right)^{1-\alpha} + \left(s^*_{h_y} - \delta\right)^{1-\alpha} \leq \tilde{E}$$

holds. This inequality follows, since the function $s \mapsto s^{1-\alpha}$ is convex and monotonic decreasing, from

$$\frac{(s^*_{j_x} + \delta)^{1-\alpha} - (s^*_{j_x})^{1-\alpha}}{(s^*_{j_x} + \delta) - s^*_{j_x}} \leq \frac{(s^*_{h_y})^{1-\alpha} - (s^*_{h_y} - \delta)^{1-\alpha}}{s^*_{h_y} - (s^*_{h_y} - \delta)}$$

and as a consequence from

$$\left(s^*_{j_x} + \delta\right)^{1-\alpha} + \left(s^*_{h_y} - \delta\right)^{1-\alpha} \leq s^*_{j_x} + s^*_{h_y} = \tilde{E}.$$

□

Lemma 8.5. *Consider problem $P2 \mid imst, w_j = 1, energy \leq E \mid \sum C_j$. A 2-tuple consisting of the job sequences*

$$(1, 3, 5, \ldots)_{M_1} \quad \textit{and} \qquad (2, 4, 6, \ldots)_{M_2} \tag{8.15}$$

is an optimal 2-tuple of job sequences.

Proof. Consider an optimal 2-tuple of job sequences

$$T \coloneqq ((j_1, \ldots, j_{k_1})_{M_1}, (h_1, \ldots, h_{k_2})_{M_2})$$

and denote the corresponding convex programming formulation from Section 8.1 by (P). According to $(\ast)$, we can assume that $j_1 = 1$ and $h_1 = 2$. To avoid technical case distinctions, we introduce „dummy jobs“ j_∞ and h_∞ with release times $r_{j_\infty} = r_{h_\infty} = \infty$ which are assigned to machines M_1 and M_2 respectively.

The extension of T by these dummy jobs is denoted by

$$T_\infty := ((j_1, \ldots, j_{k_1}, j_\infty)_{M_1}, (h_1, \ldots, h_{k_2}, h_\infty)_{M_2}) .$$

The reason why we introduced the dummy jobs is that now in T_∞ each job j for $1 \leq j \leq n$ has at least one successor on the corresponding machine. As a consequence there will be less technical cases to consider.

Assume that T has not the form as in (8.15). Then there exists a job j_x on a machine $M_{i'}$ such that $j_x + 1 = j_{x+1}$ holds. This means that jobs j_x and $j_x + 1$ are executed one after the other on machine $M_{i'}$. We choose j_x minimal with this property. Note that $j_x \geq 2$ holds. By the choice of j_x there exists a job h_y such that both h_{y+1} and h_y are executed on the other machine, i.e. machine $M_{i''}$ with $i'' \in \{1, 2\}$ and $i' \neq i''$ and such that $r_{h_y} \leq r_{j_x}$ and $r_{h_{y+1}} \geq r_{j_{x+1}}$ holds. Note that job h_{y+1} might be a dummy job.

We now construct a new 2-tuple of job sequences T'_∞. The job sequences in T'_∞ for machines $M_{i'}$ and $M_{i''}$ are changed from those in T_∞ according to the following scheme:

$$\begin{aligned} (\ldots, j_{x-1}, j_x, j_{x+1}, j_{x+2}, \ldots, j_\infty)_{M_{i'}} \quad &\text{changes to} \quad (\ldots, j_{x-1}, j_x, h_{y+1}, h_{y+2}, \ldots, h_\infty)_{M_{i'}} \\ (\ldots, h_{y-1}, h_y, h_{y+1}, h_{y+2}, \ldots, h_\infty)_{M_{i''}} \quad &\text{changes to} \quad (\ldots, h_{y-1}, h_y, j_{x+1}, j_{x+2}, \ldots, j_\infty)_{M_{i''}} \end{aligned}$$

Denote the m-tuple of job sequences that is obtained by removing all dummy jobs from T'_∞ by T'. So we have

$$\begin{aligned} T'_\infty &= \left((\ldots, j_{x-1}, j_x, h_{y+1}, h_{y+2}, \ldots, h_{k_2}, h_\infty)_{M_{i'}}, (\ldots, h_{y-1}, h_y, j_{x+1}, j_{x+2}, \ldots, j_{k_1}, j_\infty)_{M_{i''}} \right) \\ T' &= \left((\ldots, j_{x-1}, j_x, h_{y+1}, h_{y+2}, \ldots, h_{k_2})_{M_{i'}}, (\ldots, h_{y-1}, h_y, j_{x+1}, j_{x+2}, \ldots, j_{k_1})_{M_{i''}} \right) . \end{aligned}$$

We now prove that T' is also an optimal 2-tuple of job sequences. Denote the convex programming formulation obtained by T' by (P'). The only thing that differs between (P) and (P') are the lower bounds on the execution speeds of jobs j_x and h_y. In (P), we have

$$\ell_{j_x} = \frac{1}{r_{j_{x+1}} - r_{j_x}} \qquad \text{and} \qquad \ell_{h_y} = \frac{1}{r_{h_{y+1}} - r_{h_y}},$$

and in (P') these lower bounds are given by

$$\ell'_{j_x} = \frac{1}{r_{h_{y+1}} - r_{j_x}} \qquad \text{and} \qquad \ell'_{h_y} = \frac{1}{r_{j_{x+1}} - r_{h_y}}.$$

Lemma 8.4 implies that the objective function value of (P') is at least as good as the objective function value of (P). Therefore T' is also an optimal 2-tuple of job sequences. If T' has not the form as in the theorem, then again there exists a machine and a job j'_x, such that both j'_x and $j'_x + 1$ are both assigned to this machine. As a consequence of the construction of T' the inequality $j'_x > j_x$ is satisfied. That means that if we iteratively construct new optimal 2-tuples of job sequences as above, we obtain an optimal 2-tuple that is identical to the 2-tuple as stated in (8.15). □

Finally we can prove Theorem 8.3, the main result of this section.

Proof of Theorem 8.3. Consider an optimal m-tuple of job sequences T. If T is not as in (8.14) there exists a job j assigned to a machine $M_{i'}$ such that j is assigned to a different machine $M_{i''}$ in (8.14). We choose j minimal with this property. Consider the problem $P2 \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$ with machines $M_{i'}$ and $M_{i''}$ and with exactly the jobs which are in the job sequences of $M_{i'}$ and $M_{i''}$ in T. Lemma 8.5 states that assigning jobs to machines $M_{i'}$ and $M_{i''}$ alternatively is an optimal 2-tuple of job sequences. Denote these job sequences by $S_{M_{i'}}$ and $S_{M_{i''}}$. Replacing the job sequences of machines $M_{i'}$ and $M_{i''}$ in T by $S_{M_{i'}}$ and $S_{M_{i''}}$ respectively yields a new m-tuple of job sequences T' that is also optimal. If T' is not as in (8.14), then there exists a job j' assigned to a different machine than in (8.14). By construction of T' the inequality $j' > j$ holds. That means that if we iteratively construct new optimal m-tuples of job sequences as above, we obtain an optimal m-tuple of job sequences that is identical to (8.14). □

8.3 $Pm \mid \textbf{imst}, w_j, \textbf{energy} \leq \overline{E} \mid \sum C_j$

It turns out that assigning jobs according to the Round Robin technique as in Theorem 8.3 does not lead to an optimal m-tuple of job sequences for problem $Pm \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ in general. A counterexample is provided in Example 2.

Example 2. *We consider a problem of the form $Pm \mid imst, w_j, energy \leq \overline{E} \mid \sum C_j$ with $n = 4$ and $m = 2$. The release times and the processing volumes are given by $r_1 = 0, r_2 = 1, r_3 = 2, r_4 = 3$ and $w_1 = w_3 = w_4 = 6, w_2 = 1$ respectively. The energy bound is given by $\overline{E} = 37$. We consider the following 2-tuples of job sequences:*

$$T_1 \coloneqq ((1,3)_{M_1}, (2,4)_{M_2}) \qquad \text{and} \qquad T_2 \coloneqq ((1,4)_{M_1}, (2,3)_{M_2})$$

Note that T_1 has the same form as in (8.14). We now apply Algorithm 1 to T_2:

1. *We start with $s_1 = s_2 = s_3 = s_4 = \infty, \mathcal{J}_1 = \emptyset$ and $\mathcal{J}_2 = \{1,2,3,4\}$.*
2. *We have $s_1 = s_2 = s_3 = s_4 = \infty, \mathcal{J}_1 = \emptyset$ and $\mathcal{J}_2 = \{1,2,3,4\}$.*
   - ***Event A**:* $\max\{\frac{w_1}{r_4 - r_1}, \frac{w_2}{r_3 - r_2}, \frac{w_3}{\infty}, \frac{w_4}{\infty}\} = \{\frac{6}{3}, \frac{1}{1}, 0, 0\} = 2$
   - ***Event B**:* $37 = \sum_{j=1}^{4} w_j s^2 \Leftrightarrow 37 = 6 \cdot s^2 + 1 \cdot s^2 + 2 \cdot 6 \cdot s^2 \Rightarrow s = \sqrt{\frac{37}{19}}$

   Since $2 > \sqrt{\frac{37}{19}}$, event A occurs.
3. *We have $s_1 = s_2 = s_3 = s_4 = 2, \mathcal{J}_1 = \{1\}$ and $\mathcal{J}_2 = \{2,3,4\}$.*
   - ***Event A**:* $\max\{\frac{w_2}{r_3 - r_2}, \frac{w_3}{\infty}, \frac{w_4}{\infty}\} = \{\frac{1}{1}, 0, 0\} = 1$
   - ***Event B**:* $37 = w_1 \cdot 2^2 + \sum_{j=2}^{4} w_j s^2 \Leftrightarrow 37 = 6 \cdot 2^2 + 1 \cdot s^2 + 2 \cdot 6 \cdot s^2 \Rightarrow s = 1$

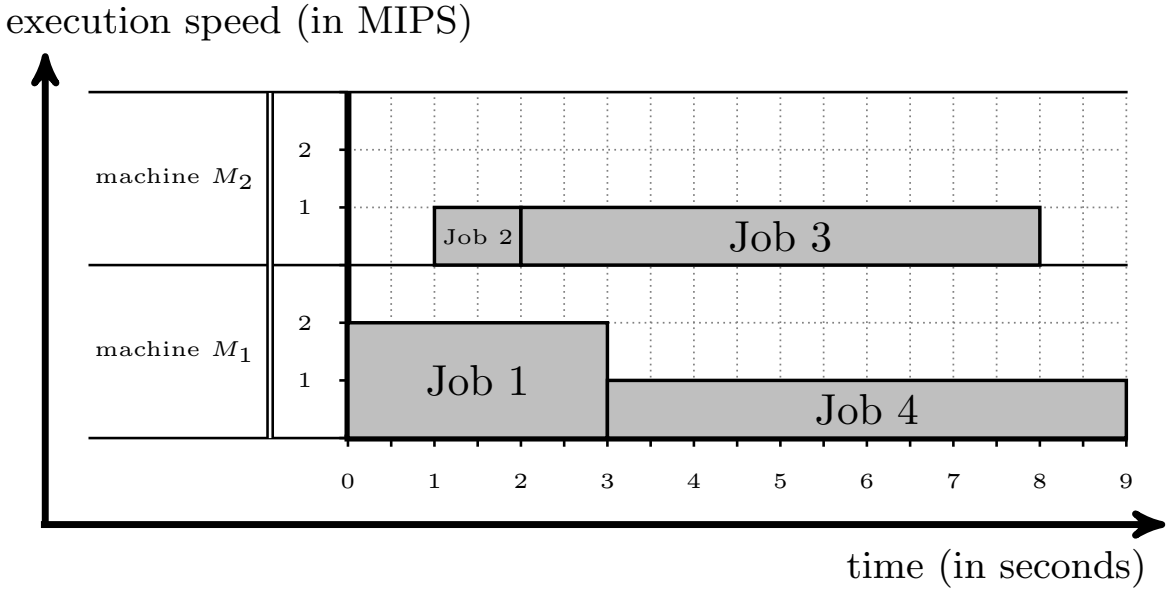

Figure 5: Gantt chart of an optimal solution for the problem in the second step based on T_2.

> *In this case both events occur and the algorithm is done. The optimal execution speeds are given as* $s_1^* = 2, s_2^* = s_3^* = s_4^* = 1$ *and the optimal objective function value is given by* $\sum_{j=1}^{4} \frac{w_j}{s_j} = \frac{6}{2} + \frac{1}{1} + \frac{6}{1} + \frac{6}{1} = 16$. *See Figure 5 for a graphical representation in a Gantt chart of this solution.*

Now we consider T_1. *In order to use the minimum amount of energy to execute job* 1, *we have to execute job* 1 *at the lowest possible execution speed. Since the jobs executed on machine* M_1 *are* 1 *and* 3, *a lower bound on the execution speed of job* 1 *is given by* $\frac{w_1}{r_3 - r_1} = \frac{6}{2} = 3$. *So we have to execute job* 1 *with an execution speed of at least* 3. *But the energy consumption for job* 1 *executed at speed* 3 *is already* $w_1 \cdot 3^2 = 6 \cdot 3^2 = 54$, *which is bigger than* $\overline{E} = 37$. *This means that the problem in the second step based on the* 2*-tuple of job sequences* T_1 *is infeasible. As a consequence* T_1 *cannot be an optimal* 2*-tuple of job sequences.*

8.4 $1 \mid \textbf{imst}, w_j \mid (\textbf{energy}, \sum C_j)$

In this section we study the trade-off between energy and the total completion time for the single machine problem with different processing volumes and the immediate start condition. As in Section 8.1 we assume $r_1 < r_2 < \ldots < r_n$.

At first we determine for which energy bounds $\overline{E}$ there is an optimal solution for problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$.

Lemma 8.6. *The set*

$$I \coloneqq \{E \in \mathbb{R} \mid \text{ problem } 1 \mid \textit{imst}, w_j, \textit{energy} \leq E \mid \sum C_j \text{ has an optimal solution}\} \quad (8.16)$$

is a closed, left-bounded and right-unbounded interval. Define $E_{\min} \coloneqq \min I$ *and denote by* $C(E_{\min})$ *the optimal objective function value for problem* $1 \mid \textit{imst}, w_j, \textit{energy} \leq E_{\min} \mid \sum C_j$. *Then the point* $(E_{\min}, C(E_{\min}))$ *is called extreme point and is given by*

$$(E_{\min}, C(E_{\min})) = \left(\sum_{j=1}^{n} w_j \ell_j^{\alpha-1}, \sum_{j=1}^{n} \frac{w_j}{\ell_j} \right).$$

Proof. Considering (8.3), the set of feasible solutions is closed and bounded (and therefore compact) and the objective function is convex. As a consequence problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ has a feasible solution if and only if it has an optimal solution. It follows that I is an interval and it is easy to see that I is also right-unbounded. Since the energy function $\sum_{j=1}^{n} w_j s_j^{\alpha-1}$ is strictly monotonic increasing in each argument, the lowest energy consumption is obtained by $s_j = \ell_j$. These execution speeds lead to the value $E_{\min} = \sum_{j=1}^{n} w_j \ell_j^{\alpha-1}$. With $E_{\min}$ as an upper bound, there is only one feasible solution that is obtained by $s_j = \ell_j$. As a consequence this solution is also optimal and we obtain $C(E_{\min}) = \sum_{j=1}^{n} \frac{w_j}{\ell_j}$. From above it is clear that I is left-bounded by $E_{\min}$ and closed. □

We explain in the following how Algorithm 1 can be modified to solve problem $1 \mid \text{imst}, w_j \mid (\text{energy}, \sum C_j)$.

Consider two given upper bounds of the energy consumption E_1 and E_2 with $E_1 < E_2$. Denote by S_1 (S_2) an instance for problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ with release times $r_1, \ldots, r_n$ and processing volumes $w_1, \ldots, w_n$ and upper bound for the energy consumption E_1 (E_2). Consider an optimal solution for instance S_2 determined by Algorithm 1. According to Lemma 8.1 the jobs in this optimal solution can be divided into two sets $\mathcal{J}_1$ and $\mathcal{J}_2$ such that the execution speeds of jobs in $\mathcal{J}_1$ reach their lower bounds and such that all execution speeds of jobs in $\mathcal{J}_2$ are equal. In order to solve the problem for instance S_1, we do not need to run Algorithm 1 from the very beginning. It is sufficient to enter the loop in Algorithm 1 with the optimal execution speeds for instance S_2, the sets $\mathcal{J}_1$ and $\mathcal{J}_2$, and the energy bound E_1. In other words, if we solve problem $1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ for a given instance with energy bound $\overline{E}$, then the algorithm solves all the problems with a higher energy bound "along the way". For this reason we are able to solve problem $1 \mid \text{imst}, w_j \mid (\text{energy}, \sum C_j)$ with the same techniques that are used in order to solve problem $1 \mid \text{imst}, w_j, \text{energy} \leq E_{\min} \mid \sum C_j$. In the following this process is explained in more detail.

In order to solve problem $1 \mid \text{imst}, w_j, \text{energy} \leq E_{\min} \mid \sum C_j$, Algorithm 1 initialises all execution speeds with infinity. We will use the notation $s_1^{(k)}, \ldots, s_n^{(k)}$ to denote the values of the execution speeds after k iterations of the loop in Algorithm 1. Similarly, we write $\mathcal{J}_1^{(k)}$ and $\mathcal{J}_2^{(k)}$ for the jobs which are in set $\mathcal{J}_1$ and $\mathcal{J}_2$ after iteration k. So at the moment

we have $s_1^{(0)} = \ldots = s_n^{(0)} = \infty$, $\mathcal{J}_1^{(0)} = \emptyset$ and $\mathcal{J}_2^{(0)} = \{1, \ldots, n\}$. Now we decrease the execution speeds until the highest lower bound $\ell^{(1)} := \max_{1 \leq j \leq n} \ell_j$ is reached. At this point in the algorithm, the execution speeds are given by $s_1^{(1)} = \ldots = s_n^{(1)} = \ell^{(1)}$. This is an optimal solution for problem $1 \mid \text{imst}, w_j, \text{energy} \leq E^{(1)} \mid \sum C_j$ with

$$E^{(1)} = \sum_{j=1}^{n} w_j {\ell^{(1)}}^{\alpha-1}.$$

It follows that for $E \in [E^{(1)}, \infty)$ in problem $1 \mid \text{imst}, w_j, \text{energy} \leq E \mid \sum C_j$, the sets $\mathcal{J}_1^{(0)}$ and $\mathcal{J}_2^{(0)}$ characterise an optimal solution (in the same way as $\mathcal{J}_1$ and $\mathcal{J}_2$ do in Lemma 8.1). As a consequence such an optimal solution $s_1, \ldots, s_n$ satisfies $s_1 = \ldots = s_n =: s$. Since the energy bound is always reached in an optimal solution, we obtain $\sum_{j=1}^{n} w_j s^{\alpha-1} = E$. As a consequence, we have

$$s = \left(\frac{E}{\sum_{j=1}^{n} w_j} \right)^{\frac{1}{\alpha-1}}$$

and the optimal objective function value $C^{(1)}(E)$ for $E \in [E^{(1)}, \infty)$ is given by

$$C^{(1)}(E) = \frac{1}{s} \sum_{j=1}^{n} w_j = \frac{\left(\sum_{j=1}^{n} w_j \right)^{1+\frac{1}{\alpha-1}}}{E^{\frac{1}{\alpha-1}}}. \tag{8.17}$$

Equation (8.17) implies that the trade-off curve is a strictly monotonic decreasing and convex function for $E \in [E^{(1)}, \infty)$.

Having decreased all execution speeds to $\ell^{(1)}$, we add all jobs j with $\ell_j = \ell^{(1)}$ to $\mathcal{J}_1^{(1)}$ and add all other jobs to $\mathcal{J}_2^{(1)}$. All execution speeds of jobs in set $\mathcal{J}_1^{(1)}$ remain the same until the end of the algorithm. The execution speeds of jobs in $\mathcal{J}_2^{(1)}$ however are decreased to the value $\ell^{(2)} := \max_{j \in \mathcal{J}_2^{(1)}} \ell_j$. So after the second iteration, we have $s_j^{(2)} = \ell^{(2)}$ for $j \in \mathcal{J}_2^{(1)}$ and $s_j^{(2)} = \ell^{(1)}$ for $j \in \mathcal{J}_1^{(1)}$. The execution speeds $s_1^{(2)}, \ldots, s_n^{(2)}$ are optimal for problem $1 \mid \text{imst}, w_j, \text{energy} \leq E^{(2)} \mid \sum C_j$ with

$$E^{(2)} := \sum_{j \in \mathcal{J}_1^{(1)}} w_j {\ell^{(1)}}^{\alpha-1} + \sum_{j \in \mathcal{J}_2^{(1)}} w_j {\ell^{(2)}}^{\alpha-1}.$$

An optimal solution $s_1, \ldots, s_n$ for $E \in [E^{(2)}, E^{(1)}]$ satisfies $s_j = \ell^{(1)}$ for $j \in \mathcal{J}_1^{(1)}$ and all s_j for $j \in \mathcal{J}_2^{(1)}$ are identical to a value $\tilde{s}$ such that

$$E = \sum_{j \in \mathcal{J}_1^{(1)}} w_j {\ell^{(1)}}^{\alpha-1} + \sum_{j \in \mathcal{J}_2^{(1)}} w_j \tilde{s}^{\alpha-1}.$$

We obtain

$$\tilde{s} = \left(\frac{E - \sum_{j \in \mathcal{J}_1^{(1)}} w_j {\ell^{(1)}}^{\alpha-1}}{\sum_{j \in \mathcal{J}_2^{(1)}} w_j} \right)^{\frac{1}{\alpha-1}}.$$

As a consequence the optimal objective function value $C^{(2)}(E)$ for $E \in [E^{(2)}, E^{(1)}]$ is given by

$$\begin{aligned} C^{(2)}(E) &= \frac{1}{\ell^{(1)}} \sum_{j \in \mathcal{J}_1^{(1)}} w_j + \frac{1}{\tilde{s}} \sum_{j \in \mathcal{J}_2^{(1)}} w_j \\ &= \frac{1}{\ell^{(1)}} \sum_{j \in \mathcal{J}_1^{(1)}} w_j + \left(\frac{\sum_{j \in \mathcal{J}_2^{(1)}} w_j}{E - \sum_{j \in \mathcal{J}_1^{(1)}} w_j \ell^{(1)\alpha-1}} \right)^{\frac{1}{\alpha-1}} \sum_{j \in \mathcal{J}_2^{(1)}} w_j. \end{aligned} \tag{8.18}$$

Similar as before, equation (8.18) implies that the trade-off curve is a strictly monotonic decreasing and convex function for $E \in [E^{(2)}, E^{(1)}]$. Note that the trade-off function is continuous in $E^{(1)}$ since

$$\begin{aligned} C^{(2)}(E^{(1)}) &= \frac{1}{\ell^{(1)}} \sum_{j \in \mathcal{J}_1^{(1)}} w_j + \left(\frac{\sum_{j \in \mathcal{J}_2^{(1)}} w_j}{\sum_{j=1}^{n} w_j \ell^{(1)\alpha-1} - \sum_{j \in \mathcal{J}_1^{(1)}} w_j \ell^{(1)\alpha-1}} \right)^{\frac{1}{\alpha-1}} \sum_{j \in \mathcal{J}_2^{(1)}} w_j \\ &= \frac{1}{\ell^{(1)}} \sum_{j \in \mathcal{J}_1^{(1)}} w_j + \left(\frac{\sum_{j \in \mathcal{J}_2^{(1)}} w_j}{\ell^{(1)\alpha-1} \sum_{j \in \mathcal{J}_2^{(1)}} w_j} \right)^{\frac{1}{\alpha-1}} \sum_{j \in \mathcal{J}_2^{(1)}} w_j \\ &= \frac{1}{\ell^{(1)}} \sum_{j=1}^{n} w_j = \frac{\left(\sum_{j=1}^{n} w_j \right)^{1+\frac{1}{\alpha-1}}}{\left(\sum_{j=1}^{n} w_j \ell^{(1)\alpha-1} \right)^{\frac{1}{\alpha-1}}} = C^{(1)}(E^{(1)}). \end{aligned}$$

These arguments can be repeated for each iteration in the loop of Algorithm 1. The obtained results are summarised in the following theorem.

Theorem 8.4. *Consider problem* $1 \mid imst, w_j \mid (energy, \sum C_j)$. *The domain*

$$I = \{E \in \mathbb{R} \mid \text{ problem } 1 \mid imst, w_j, energy \leq E \mid \sum C_j \text{ has an optimal solution}\}$$

of the trade-off function is a closed, left-bounded and right-unbounded interval. The trade-off function is a piecewise convex, strictly monotonic decreasing function with $k \in \{0, \ldots, n\}$ *breakpoints that is continuous in all breakpoints.*

8.5 $Pm \mid \mathbf{imst}, w_j = 1 \mid (\mathbf{energy}, \sum C_j)$

We know from Section 8.2 that once an optimal m-tuple of job sequences is chosen, problem $Pm \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$ becomes a problem of the form $1 \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$. For this reason problem $Pm \mid \text{imst}, w_j = 1 \mid (\text{energy}, \sum C_j)$ can be solved with the same techniques as problem $1 \mid \text{imst}, w_j \mid (\text{energy}, \sum C_j)$. Corollary 8.1 is therefore an immediate consequence of Theorem 8.4.

Corollary 8.1. *Consider problem* $Pm \mid imst, w_j = 1 \mid (energy, \sum C_j)$. *The domain*

$$I = \{E \in \mathbb{R} \mid \text{ problem } 1 \mid imst, w_j, energy \leq E \mid \sum C_j \text{ has an optimal solution}\}$$

of the trade-off function is a closed and right-unbounded interval. The trade-off function is a piecewise convex, strictly monotonic decreasing function with $k \in \{0, \ldots, n\}$ *breakpoints that is continuous in all breakpoints.*

9 Conclusions and Open Problems

In this thesis we considered scheduling problems that arise in cloud computing systems. We have introduced a new feature of scheduling models, the so-called immediate start property. This property is natural for cloud computing systems, as a customer usually expects that the execution of submitted jobs is started immediately.

We also studied various models of the energy consumption which belongs to the most important objectives in today's and future cloud computing systems. We highlighted that there exist different energy models in the literature which can lead to (totally) contrary optimal scheduling strategies. This suggests that the energy models considered in the literature are not capable of modelling all aspects of the energy consumption. Taking this as a motivation, we proposed a new and more complex energy model that is based on the energy consumption of each core of the processor. Using this energy model, we studied the scheduling problem of minimising the energy consumption such that a given bound on the total flow time is not exceeded. For the case of equal jobs, we have shown that the corresponding trade-off curve is convex and monotonic decreasing. However, the underlying convex programming formulation of this problem is already quite complex, even for the case of equal jobs. Keeping in mind that in cloud computing systems the executions of jobs should be started (almost) immediately, it is necessary that scheduling algorithms operate as fast as possible. As a consequence the core based energy approach might be unsuitable, since the higher complexity of the model may make it difficult to obtain fast scheduling algorithms.

Furthermore we provided a short literature overview of scheduling models that minimise total flow and energy consumption. We also considered scheduling problems with the immediate start property and with the goal of minimising the total flow time for a given amount of available energy. These scheduling models are more likely to meet practical requirements than the core based energy models from above. We studied both the problem version of determining all Pareto optimal schedules as well as the problem version of minimising the total flow such that a given upper limit on the consumed energy is not exceeded. We present an overview of obtained results in the table below.

problem	complexity	reference
$1 \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$	$\mathcal{O}(n \log n)$	Section 8.1
$Pm \mid \text{imst}, w_j = 1, \text{energy} \leq \overline{E} \mid \sum C_j$	$\mathcal{O}(n \log n)$	Section 8.2
$Pm \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$	open	-
$1 \mid \text{imst}, w_j \mid (\text{energy}, \sum C_j)$	$\mathcal{O}(n \log n)$	Section 8.4
$1 \mid \text{imst}, w_j \mid (\text{energy}, \sum C_j)$	$\mathcal{O}(n \log n)$	Section 8.5

Probably the most obvious open question that arises from this thesis is to decide for problem $Pm \mid \text{imst}, w_j, \text{energy} \leq \overline{E} \mid \sum C_j$ whether there exists a polynomial time algorithm or whether the problem is NP-hard. Another fruitful area for further research results by including the immediate start property into the speed scaling problems studied in the literature. This includes scheduling problems with other objective functions than

energy consumption or total flow (reliability or makespan for example). In addition one can consider scheduling models that include other computational resources than CPU (memory, disk or network for example). Finally, building virtual machines on a processor has not been modelled in a mathematical way before to the best of our knowledge and is therefore another new research direction.

References

[1] Susanne Albers. Algorithms for energy saving. In *Efficient Algorithms*, pages 173–186. Springer, 2009.

[2] Susanne Albers. Algorithms for energy management. In *Computer Science–Theory and Applications*, pages 1–11. Springer, 2010.

[3] Susanne Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, May 2010.

[4] Susanne Albers and Hiroshi Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms (TALG)*, 3(4):49, 2007.

[5] Lachlan L.H. Andrew, Adam Wierman, and Ao Tang. Optimal speed scaling under arbitrary power functions. *SIGMETRICS Perform. Eval. Rev.*, 37(2):39–41, October 2009.

[6] Bernd Bank, Jürgen Guddat, Diethard Klatte, Bernd Kummer, and Klaus Tammer. *Non-Linear Parametric Optimization*. Birkhäuser Basel, 1982.

[7] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 693–701, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[8] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. *ACM Trans. Algorithms*, 9(2):18:1–18:14, March 2013.

[9] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):3:1–3:39, March 2007.

[10] Nikhil Bansal, Kirk Pruhs, and Cliff Stein. Speed scaling for weighted flow time. *SIAM Journal on Computing*, 39(4):1294–1308, 2009.

[11] Neal Barcelo. *The Complexity of Speed-Scaling*. PhD thesis, University of Pittsburgh, 2015.

[12] Luis Bautista, Alain Abran, and Alain April. Design of a performance measurement framework for cloud computing. *Journal of Software Engineering and Applications*, 5(2):7, 2012.

[13] Richard Bellman. Mathematical aspects of scheduling theory. *Journal of the Society for Industrial & Applied Mathematics*, 4(3):168–205, 1956.

[14] Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 826–831. IEEE Computer Society, 2010.

[15] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005.

[16] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[17] Peter Brucker. *Scheduling Algorithms*. Springer, 5th edition, 2007.

[18] Rajkumar Buyya, Christian Vecchiola, and S Thamarai Selvi. *Mastering cloud computing: foundations and applications programming*. Newnes, 2013.

[19] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI*, volume 8, pages 337–350, 2008.

[20] Nikhil R Devanur and Zhiyi Huang. Primal dual gives almost optimal energy efficient online algorithms. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1123–1140. SIAM, 2014.

[21] Maciej Drozdowski. *Scheduling for parallel processing*. Springer, 2009.

[22] Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

[23] Peter Xiang Gao, Andrew R Curtis, Bernard Wong, and Srinivasan Keshav. It's not easy being green. *ACM SIGCOMM Computer Communication Review*, 42(4):211–222, 2012.

[24] Peter Garraghan. Private Communication, 2014.

[25] Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.

[26] Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. Power consumption of virtual machine live migration in clouds. In *Communications and Mobile Computing (CMC), 2011 Third International Conference on*, pages 122–125. IEEE, 2011.

[27] Emmanuel Jeannot, Erik Saule, and Denis Trystram. Bi-objective approximation scheme for makespan and reliability optimization on uniform parallel machines. In *Euro-Par 2008–Parallel Processing*, pages 877–886. Springer, 2008.

[28] Selmer Martin Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1):61–68, 1954.

[29] Niloofar Khanghahi and Reza Ravanmehr. Cloud computing performance evaluation: Issues and challenges. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 3(5), 2013.

[30] Kenji E Kushida, Jonathan Murray, John Zysman, Kenji E Kushida, Patrick Scaglia, Jamal Ibrahim Haidar, Takeo Hoshi, Takatoshi Ito, and Masahiko Aoki. Cloud computing: From scarcity to abundance. Technical report, BRIE Working Paper, 2014.

[31] Minming Li, Andrew C Yao, and Frances F Yao. Discrete and continuous min-energy schedules for variable voltage processors. *Proceedings of the National Academy of Sciences of the United States of America*, 103(11):3983–3987, 2006.

[32] David G Luenberger and Yinyu Ye. *Linear and nonlinear programming*, volume 116. Springer Science & Business Media, 2008.

[33] Peter Mell and Tim Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[34] Ismael Solis Moreno, Peter Garraghan, Paul Townend, and Jie Xu. Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. *Cloud Computing, IEEE Transactions on*, 2(2):208–221, 2014.

[35] Ismael Solis Moreno and Jie Xu. Neural network-based overallocation for improved energy-efficiency in real-time cloud environments. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2012 IEEE 15th International Symposium on*, pages 119–126. IEEE, 2012.

[36] Michael Pinedo and Xiuli Chao. *Operations scheduling with applications in manufacturing and services*. McGraw-Hill Companies, 1999.

[37] Kirk Pruhs, Patchrawat Uthaisombut, and Gerhard Woeginger. Getting the best response for your erg. *ACM Transactions on Algorithms (TALG)*, 4(3):38, 2008.

[38] Asfandyar Qureshi. *Power-demand routing in massive geo-distributed systems*. PhD thesis, Massachusetts Institute of Technology, 2010.

[39] Dvir Shabtay and George Steiner. A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666, 2007.

[40] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 374–382, Washington, DC, USA, 1995. IEEE Computer Society.