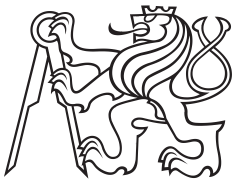
Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Resource-constrained project scheduling with machine states and variable energy prices

Jan Mandík

Supervisor: Ing. Antonín Novák, Ph.D.
Field of study: Open Informatics
Subfield: Software Engineering
January 2025

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mandík** Jméno: **Jan** Osobní číslo: **491961**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Rozvrhování projektů se stavy strojů a variabilní cenou energií

Název diplomové práce anglicky:

Resource-constrained project scheduling with machine states and variable energy prices

Pokyny pro vypracování:

Práce se zabývá formulací a řešením problému plynoucí z rozvrhování výrobních procesů s energeticky náročnou fází. Uvažujeme, že energeticky náročná fáze procesu je realizována na výrobním stroji který se může nacházet v různých stavech s rozdílnou spotřebou energie. Cena energie je variabilní a mění se ve stanovených fixních intervalech. Cílem je sestavit rozvrh který přepíná stavy stroje, balancuje délku rozvrhu a celkovou cenu spotřebované energie.
1. Seznamte se s problémy a metodami energeticky efektivního rozvrhování zohledňující tzv. Time-of-Use tarify a modelování stavovosti zdrojů se stand-by módy.
2. Formalizujte problém rozvrhování projektů ve výrobě s jedním energeticky náročným zdrojem popsaný stavovým diagramem a variabilní cenou energií jako úlohu bi-kriteriální optimalizace.
3. Navrhněte a implementujte modely Constraint Programming (CP) a Integer Linear Programming (ILP) pro řešení formulovaného problému.
4. Navrhněte a implementujete optimalizační algoritmus pro úlohu přerozvrhování sloužící pro reakci na změnu cen energií.
5. Otestujte a porovnejte navržené algoritmy na bechmarkových sadách vycházející z PSPlib za využití reálných průběhů cen energií OTE z hlediska kvality řešení a škálovatelnosti.

Seznam doporučené literatury:

- [1] M. Gaggero, M. Paolucci, R. Ronco, Exact and heuristic solution approaches for energy-efficient identical parallel machine
scheduling with time-of-use costs, EJOR 311 (3) (2023) 845–866.
- [2] O. Benedikt, I. Modos, Z. Hanzalek, Power of pre-processing: production scheduling with variable energy pricing and power-saving states, Constraints 25 (3-4) (2020) 300–318.
- [3] H. Minh Hung, F. Hnaien, F. Dugardin, Electricity cost minimisation for optimal makespan solution in flow shop scheduling
under time-of-use tariffs, International Journal of Production Research (2020) 1–27.
- [4] Catanzaro, Daniele, Raffaele Pesenti, and Roberto Ronco. "Job scheduling under Time-of-Use energy tariffs for sustainable manufacturing: a survey." European Journal of Operational Research 308.3 (2023): 1091-1109.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Antonín Novák, Ph.D. katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.08.2024** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **15.02.2026**

Ing. Antonín Novák, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Acknowledgements

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Antonínu Novákovi, Ph.D. za vstřícný a trpělivý přístup, rychlou komunikaci, cenné rady a náměty a za nadstandardní množství času, které mi při tvorbě této práce věnoval. Dále bych chtěl poděkovat prof. Dr. Ing. Zdeňku Hanzálkovi za přínosné rady, poznámky a náměty. V neposlední řadě děkuji své rodině za podporu během celého studia.

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 7. ledna 2025

Abstract

The presented master thesis focuses on extending the Resource-Constrained Project Scheduling Problem (RCPSP) by the energy-awareness element. We assume a single stateful resource to be energy-consumption intense. We compute the total energy costs of this resource considering both the TOU pricing and power-saving states of the resource. We introduce a new parameter α, which allows us to balance the project makespan and the total energy cost in the bi-criterial objective function. Furthermore, we study the problem of rescheduling in reaction to the changes in the energy price predictions, allowing for various constraints to modify the schedule while also considering the working shifts of the resources. The work proposes exact methods for solving the problems using both Constraint Programming and Integer Linear Programming. The methods are evaluated in terms of scalability and quality of the solution on newly created instances based on datasets from PSPLIB.

Keywords: Scheduling, Resource-Constrained Project Scheduling Problem, Total Energy Consumption, Constraint Programming, Integer Linear Programming, Rescheduling, Power-saving states, Time-of-use pricing

Supervisor: Ing. Antonín Novák, Ph.D.
Praha,
Jugoslávských partyzánů 1580,
místnost: A-514

Abstrakt

Předložená diplomová práce se věnuje tématu rozšíření problému rozvrhování projektů s omezenými zdroji (RCPSP) o aspekt spotřeby energie. Uvažujeme právě jeden energeticky náročný zdroj s možností existence stavů pro úsporu energie. Celková cena za spotřebovanou energii tímto zdrojem je počítána s ohledem na proměnlivé ceny energie i na úsporné stavy zdroje. Zavádíme parametr α, který umožňuje vyvažovat celkovou cenu energie a délku trvání projektu v bi-kriteriální účelové funkci. Dále se práce věnuje problému přerozvrhování v reakci na změny v predikovaných cenách energie, přičemž jsou uvažována jak různá omezení pro změny v rozvrhu, tak i pracovní směny jednotlivých zdrojů. V práci jsou navrženy exaktní metody řešení zmíněných problémů, a to jak využitím programování s omezujícími podmínkami tak celočíselného lineárního programování. Metody jsou otestovány z hlediska škálovatelnosti a kvality dosaženého řešení na nově vytvořených instancích odvozených z datasetů PSPLIB.

Klíčová slova: Rozvrhování, Resource-Constrained Project Scheduling Problem, Celková spotřeba energie, Programování s omezujícími podmínkami, Celočíselné lineární programování, Přerozvrhování, Stavy pro úsporu energie, Ceny dle času spotřeby

Překlad názvu: Rozvrhování projektů se stavy strojů a variabilní cenou energií

Contents

Figures

Tables

Chapter 1

Introduction

The industrial sector accounts for about one-half of the world's total energy consumption [17]. Its energy consumption is further projected to grow by up to 62% by 2050 [51]. With rising energy prices, increasingly serious environmental problems, and growing pressure on companies to make their production more sustainable in recent years, energy-aware project scheduling has been attracting a considerable amount of attention [15].

In industrialized countries, many energy suppliers have begun to implement a so-called time of use (TOU) energy pricing. TOU pricing helps to balance daily energy consumption and to avoid concentration of consumption during certain hours [27]. This is achieved by changing the energy prices as a response to the demand, resulting in variable energy prices, which may differ every hour, creating a huge opportunity for savings by shifting the main energy consumption to off-peak intervals [21]. An example of a TOU price profile from OTE [41] is shown in Figure 1.1, with the traded energy volume shown in blue and the energy cost shown as the red line. It can be seen that there is a first spike in the energy cost in the morning hours and a second, even larger spike in the late evening. The prices of energy are visibly increasing in response to the increased demand.

Another promising approach for optimizing the energy demand, presented by Gahm et al. [20], is introducing the power-saving states of the energy-consumption intense resources, such as steel or glass hardening furnaces, robots in assembly lines or even computers in data centers. The introduction of the power-saving states allows us to not only optimize the start times of task processing with regard to the TOU energy pricing but also to select the

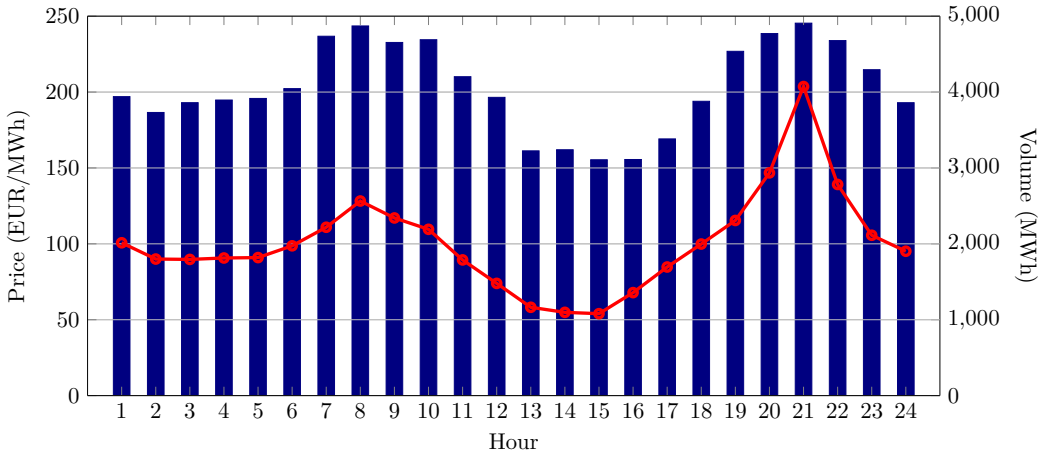


Figure 1.1: TOU price profile example from OTE [41]. The traded energy volume in each interval is shown in blue; the corresponding energy cost is shown by the red line.

optimal power-saving states in periods when the resource is underutilized. However, the switching between states of the resource needs to be planned carefully, considering the non-negligible time and cost of the transitions, yet when done right, a significant energy cost reduction can be attained [39].

In contrast to most literature, the presented thesis focuses not only on the optimization of the total energy costs on a single machine but rather on the optimization of the whole resource-constrained production process, with a single stateful energy-consumption intense resource being a part of it. The aim is to optimize both the total energy cost and the overall project makespan. An example of such a production line can be the gear shaft production line, with the electric vacuum steel hardening furnace representing the energy-consumption intense resource. Such a problem was addressed in the works of Benedikt et al. [5, 7], yet the works focused solely on the energy-consumption intense part of the production line.

Another challenge connected to the fluctuations in energy prices is their forecasting. Numerous mathematical models aim to predict energy prices based on various influencing factors [36]. This topic will be addressed in the presented thesis in the form of rescheduling in reaction to the changes in the energy price predictions while also allowing further savings and employee utilization by concentrating the work on the resources into work shifts. The rescheduling presents an interesting task, as adjustments to the schedule of energy-intensive resources — prompted by changes in energy prices — can lead to further modifications in the overall schedule to ensure feasibility. This can also lead to tradeoffs between the total energy consumption and the overall project makespan. Yet the amount of possible changes is subject to various other constraints.

1.1 Contributions and thesis outline

The presented master thesis focuses on the extension of the Resource-Constrained Project Scheduling Problem (RCPSP) by the energy-awareness element, building mainly on the results of Benedikt et al. [6]. We assume a single stateful resource to be energy-consumption intense. We compute the total energy costs of this resource with consideration of both the TOU pricing and power-saving states of the resource. We introduce a new parameter α, which allows us to balance both the project makespan (C_{max}) and the total energy cost (TEC) objective. Furthermore, we study the problem of rescheduling in reaction to the changes in the energy price predictions, allowing for various constraints for the modification of the schedule while also taking into account the working shifts of the resources. The main contributions of the thesis are the following:

- Formulation of the energy-aware RCPSP with a single energy-intense resource as a bi-criterial optimization task.
- Creation of a method for instance generation, based on the datasets from the PSPLIB [30] with the use of real data from OTE for the energy costs.
- Proposing exact methods for solving the problem using both Constraint Programming and Integer Linear Programming.
- Proposing a model for the task of rescheduling in reaction to the changes in the energy prices predictions.
- Evaluation of the presented methods in terms of both scalability and the quality of the solution.

The text is further structured in the following way: The rest of the Chapter 1 is dedicated to the formal problem statement. Chapter 2 provides an overview of the related work. Chapter 3 is dedicated to the preliminaries; firstly, it formalizes the optimal switching problem and describes a method for its solving; later, the method for instance generation is described. Chapter 4 presents the proposed methods for both the scheduling and the rescheduling problem. Chapter 5 shows the performed experiments, evaluating the proposed methods in terms of scalability and the quality of the solutions. Lastly, Chapter 6 is dedicated to the conclusion.

1.2 Problem statement

In this section, we start by describing the well-known problem of Resource-Constrained Project Scheduling and single-machine scheduling with the goal of minimizing the total energy cost, considering both time-of-use pricing and power-saving states of the machine. After the formal definition of these problems, we proceed with the introduction of the newly formulated bi-criteria problem statement for energy-aware Resource-Constrained project scheduling problem with both time-of-use pricing and power-saving states of the machine.

1.2.1 Resource-Constrained project scheduling problem

The Resource-Constrained Project Scheduling Problem (RCPSP) is a well-known classic optimization challenge frequently encountered in project management [24]. It was proven that the problem belongs to the class of NP-hard problems [9]. Informally, the goal is to create a feasible schedule for completing a set of tasks that minimizes the project makespan. All of the tasks are available from the start and are non-preemptive, meaning that once started, the task cannot be interrupted. Each task is defined by its duration, resource requirements, and a set of precedence constraints, expressing that the task can only start after other specified tasks are finished. Furthermore, each of the given resources has its given capacity, which is renewable, meaning that the sum of requirements of this resource cannot exceed this capacity at any given moment in the schedule.

Let us introduce the formal definition of the problem. Let $R = \{R_1, R_2, ..., R_m\}$ be the set of available renewable resources, where $R_i \in \mathbb{N}$ is the i-th resource capacity. Let $T = \{T_1, T_2, ..., T_n\}$ be a set of tasks that need to be scheduled, the tasks are not preemptive. Each task T_i is associated with its processing time duration $p_i \in \mathbb{N}$ and a vector $\mathbf{r} = (r_1, r_2, ..., r_m)$ $r_i \in \mathbb{N}_0 \; \forall i \in \{1, ..., m\}$ stating the task's requirements of the corresponding resource. We assign each task its start time s_i and completion time $C_i = s_i + p_i$. It must hold that the schedule contains all of the given tasks T. At each given time instant, the sum of j-th resource requirements of all ongoing tasks (either starting at this time or starting at some time $i - a$ where $a < p_k$ of the corresponding task T_k) must be lower or equal than the corresponding resource capacity R_j for each resource. Additionally, let $G = (V, E)$ be a directed acyclic graph. With V being a set of n vertices, each vertex v_i corresponds to the task T_i. The set of edges E of the graph represents the precedence relations between the tasks. For each $(i, j) \in E$ it must stand

that $C_i \leq s_j$. The objective is to minimize the maximum completion time $C_{max} = \max_{i \in \{1,\ldots,n\}} C_i$, which means minimizing the overall project makespan.

To illustrate the problem, consider the following example project. The project contains 8 tasks and 2 resources of capacities $R_1 = 5$ and $R_2 = 3$. The graph in Figure 1.2 shows the precedence relations between the tasks. Each task is also described with its associated processing time p and resource requirements given as (r_1, r_2).

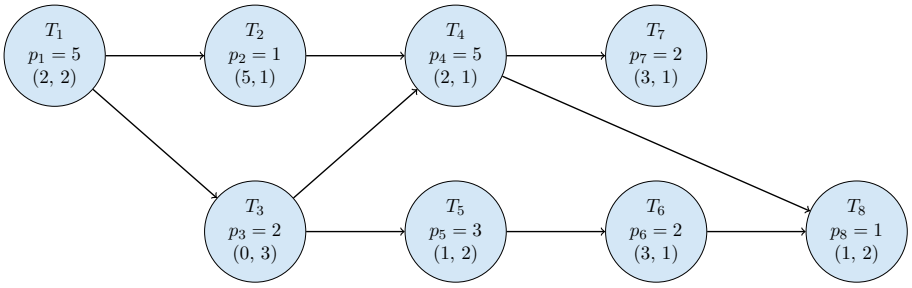


Figure 1.2: RCPSP example precedence relations graph, resource requirements are shown as (r_1, r_2).

The Gantt chart in Figure 1.3 shows the resulting schedule for the given problem, with C_{max} objective value of 15. Note that the resource capacities, together with precedence constraints between tasks, can prevent the resources from being operational at all times.

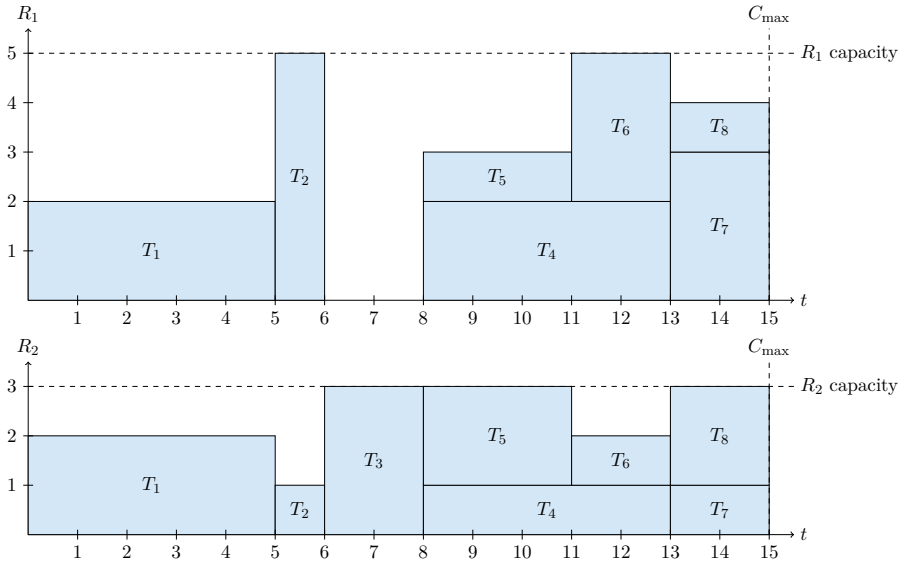


Figure 1.3: Example project schedule for resources R_1 and R_2.

1.2.2 Total energy cost minimization for single machine scheduling problem

The single-machine scheduling problem aiming to minimize the total energy cost (TEC) with time-of-use energy pricing (TOU) and power-saving states of the machine can be expressed in Graham's notation as 1, TOU | states | TEC [23]. Informally, we have a given scheduling horizon partitioned into equal intervals of length 1. Each of the intervals is associated with a given energy price. These intervals can, in general, correspond to any time unit (such as 1 hour or 15 minutes) based on the required granularity of the scheduling horizon as long as the energy cost remains constant for the whole duration of the interval. There is a given set of tasks to be scheduled within the scheduling horizon. Each task has a given duration. The tasks are non-preemptive. All of the tasks are available from the beginning of the scheduling horizon. There is a single machine on which the tasks must be processed. The machine can process at most one task at a time. Furthermore, the machine has multiple states between which it can be switched. There is an off state, meaning the machine is turned off and does not consume energy; exactly one processing state, which the machine is required to be in for the processing of the tasks; and finally, one or more power-saving states, which allow the machine to save energy without the need to be fully shut down. The machine is in all intervals, either transitioning between two states or staying in the same state. Each of the transitions is associated with the duration (number of intervals) and energy consumption. All of the possible transitions are expressed in a transition diagram. For a solution to be valid, its transitions must be possible with respect to this given diagram. To obtain the final energy cost of the transition, we take a sum of the energy consumptions multiplied by the cost of energy in the corresponding interval over the whole scheduling horizon.

The formal definition of the problem adopts the problem statement and notation presented in the work of Benedikt et al. [6] and is expressed as follows: Let $I = \{I_1, I_2, ..., I_h\}$ be a set of intervals, all of equal size 1, that partition the scheduling horizon of length h. Each interval is associated with an energy cost, given by the vector $\mathbf{c} = (c_1, c_2, ..., c_h)$ where $c_i \in \mathbb{Z}_0$. Let $T = \{T_1, T_2, ..., T_n\}$ be a set of tasks that need to be scheduled in the scheduling horizon; the tasks are not preemptive. Each task T_i is associated with its processing time duration $p_i \in \mathbb{N}$. The schedule assigns each task its start time s_i and completion time $C_i = s_i + p_i$. We consider the energy consumption per interval to be similar for all tasks. It stands that $n > 0$, meaning that the set of tasks is not empty. The machine has a non-empty set of available states Σ. In each interval, the machine is either staying in the state $\sigma \in \Sigma$ or is transitioning between two states. For simplicity, we say that when a machine stays in the same state during an interval, it is transitioning

from the state to itself.

The parameters of the transitions are expressed by two functions, the transition time function and the transition power function. Firstly, the transition time function is defined as $TF : \Sigma \times \Sigma \to \mathbb{N}_0 \cup \{\infty\}$ the value of $TF(\sigma, \sigma')$ represents the number of intervals needed for a direct transition from state σ to σ'. If the direct transition between the states σ and σ' does not exist the value of $TF(\sigma, \sigma')$ is set to ∞. This, however, doesn't mean the transition is not possible at all; there can still be a possible sequence of direct transitions from σ to σ'. As mentioned above, for the case of the machine staying in the same state (transitioning from the state to itself), we set the value of $TF(\sigma, \sigma)$ to 1. Secondly, the transition power function, defined as $PF : \Sigma \times \Sigma \to \mathbb{N}_0 \cup \{\infty\}$, expresses the power consumption during each interval of the transition from state σ to σ'. Equally, as for the transition time function, if there is not a direct transition between the states σ and σ', then $PF(\sigma, \sigma') = \infty$. The value $PF(\sigma, \sigma)$ expresses the machine's power consumption when staying in the same state. Additionally, if for two states $\sigma, \sigma' \in \Sigma$ the value of $TF(\sigma, \sigma') = 0$ the value of $PF(\sigma, \sigma')$ is also equal to 0, meaning that if the given direct transition does not take any time it can not be associated with a non-zero power consumption. This rule, however, does not apply the other way around. There can be a non-zero time transition with zero power consumption, as an example we can take the transition (off, off) where the machine is staying turned off. The transition time function and transition power function can be also expressed in the form of a transition diagram, with nodes representing the states of the machine and arrows representing possible transitions between them, described with the transition time and power consumption. An example of such a diagram can be seen in Figure 1.4.

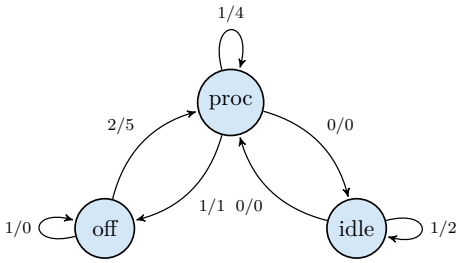


Figure 1.4: Transition diagram example, transitions are described as $TF(\sigma, \sigma')/PF(\sigma, \sigma')$ (taken from Benedikt et al. [6]).

There is exactly one processing state $proc \in \Sigma$, in which the machine must be during the processing of the tasks. There is also exactly one off state $off \in \Sigma$. We require the machine to be in this state during both the first and the last interval of the scheduling horizon. Therefore, before the first job can be processed, some initial transition or sequence of transitions between the

off state and processing state is needed, so we can denote the first interval in which a job can be processed as I_{early}. Similarly, there needs to be a transition or sequence of transitions from the last processing state to the off state during the last interval, hence we can also denote the latest interval in which a job can be processed as I_{late}. Furthermore, there can be one or more power-saving states such as *idle*. See the example of a transition diagram of a resource, shown in Figure 1.4.

A solution to this problem is a pair $(\mathbf{s}, \Omega)$, where $\mathbf{s}$ is a vector of length n, $(s_1, s_2, \ldots, s_n) \in \mathbb{N}_0$ and Ω is a vector of length h, $(\Omega_1, \Omega_2, \ldots, \Omega_h) \in (\Sigma \times \Sigma)^h$. The vector $\mathbf{s}$ contains the start times of the given tasks. The vector Ω contains the transitions the machine is undergoing in each interval of the scheduling horizon. The pair $(\mathbf{s}, \Omega)$ represents a feasible solution if all of the following conditions are satisfied:

1. The machine is not processing more than one task during any interval.
2. When a task is being processed, the machine is undergoing a $(proc, proc)$ transition, meaning that $\forall T_j \in T, \forall i \in \{s_j, \ldots, s_j + p_j - 1\} : \Omega_i = (proc, proc)$.
3. During the first and last interval of the scheduling horizon, the machine is undergoing the (off, off) transition, i.e., $\Omega_1 = \Omega_h = (off, off)$.
4. All transitions are correctly following the transition time function, i.e., $\forall i \in \{1, 2, \ldots, h-1\}$ such that $\Omega_i = (\sigma_1, \sigma_1')$, $\Omega_{i+1} = (\sigma_2, \sigma_2')$, it holds that:
   a. Ω_i and Ω_{i+1} are both existing direct transitions with non-zero time duration: $0 < TF(\Omega_i) < \infty$ and $0 < TF(\Omega_{i+1}) < \infty$. We do not allow zero time transitions here since Ω_i and Ω_{i+1} are showing the transitions the machine is undergoing throughout the interval; therefore, a transition that lasts 0 intervals cannot take this place.
   b. Only existing zero-time transition or sequence of existing zero-time transitions is allowed between Ω_i and Ω_{i+1}.
   c. All non-zero-time transitions do correctly follow the time transition function. If $\Omega_i \neq \Omega_{i+1}$ and $0 < T(\Omega_{i+1}) < \infty$ then $\forall j \in \{1, \ldots, TF(\sigma_2, \sigma_2') - 1\} : \Omega_{i+1+j} = \Omega_{i+1}$.

The TEC objective of the problem is then expressed as

$$\min \sum_{I_i \in I} c_i \cdot PF(\Omega_i)$$

For a better understanding of the problem, a small example instance of the problem is presented in Figure 1.5. In this instance, we consider the transition diagram given in Figure 1.4. The set of tasks is given as $T = \{2, 2, 1\}$. The vector of costs $\mathbf{c}$ is shown in the Figure 1.5. The line labeled as Machine state corresponds to the optimal transitions of the machine Ω, with $\nearrow \ = (off, proc)$ and $\searrow \ = (proc, off)$. The total energy cost line shows the energy cost in each interval. The optimal task start times are $\mathbf{s} = (9, 12, 3)$ and the TEC objective equals 133. It can be seen that, as required, the machine is undergoing the (off, off) transition in both the first interval I_1 and the last interval I_{16}. To avoid the spike in energy costs in intervals I_6 and I_7, the machine is turned off and turned back on again later to process the task T_1 when the prices get lower again. To avoid the processing of a task in the interval I_{12}, where we can again see a spike in the energy cost, the machine makes use of the *idle* state, waiting for one interval. The intervals I_{11} and I_{12} can be also used to illustrate conditions 4a and 4b; the machine is undergoing valid non-zero time transitions in both intervals, namely $(proc, proc)$ and $(idle, idle)$, in between them the machine undergoes an existing zero-time transition $(proc, idle)$, however, we do not see it in the vector of transitions since it is a zero-time transition and does not take up any interval. The example instance was adopted from [6].

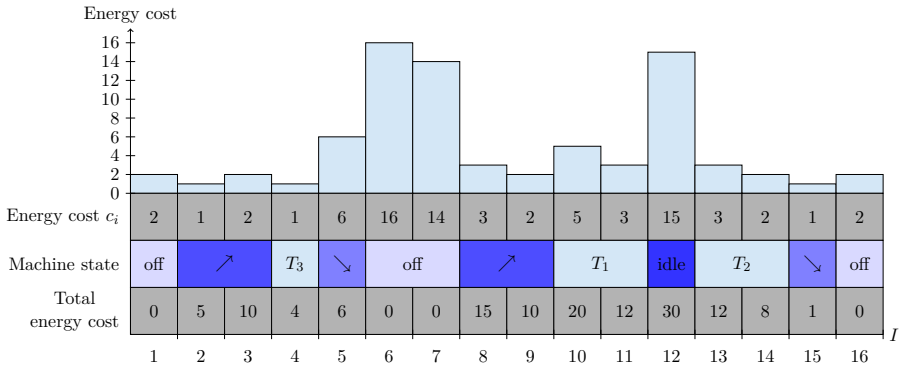


Figure 1.5: Example instance solution for single machine scheduling to minimize TEC.

1.2.3 Energy-Aware RCPSP with TOU pricing and power-saving states

In this subsection, we will present the problem statement for Energy-aware RCPSP with TOU pricing and power-saving states of the machine. It is posed as a bi-criteria optimization problem combining the project makespan and TEC. The problem statement is based on merging the above-described problem statements of RCPSP and TEC minimization for single-machine scheduling problem. We consider one of the resources to be significant for

energy consumption. The goal is to minimize both the TEC of the energy-consumption significant resource and the overall project makespan. The formal definition of the problem goes as follows.

Let $I = \{I_1, I_2, ..., I_h\}$ be a set of intervals, all of equal size 1 that partition the scheduling horizon. Each interval is associated with an energy cost, given by the vector $\mathbf{c} = (c_1, c_2, ..., c_h)$ where $c_i \in \mathbb{Z}_0$. Let $R = \{R_0\} \cup \{R_1, R_2, ..., R_m\}$ be the set of available renewable resources, where $R_i \in \mathbb{N}$ is the i-th resource capacity at each interval. Without loss of generality, we state that the added resource R_0 is the energy-consumption significant resource and its capacity $R_0 = 1$. This way, we ensure that the resource can be processing at most one task at a time.

Let $T = \{T_1, T_2, ..., T_n\}$ be a set of not preemptive tasks that need to be scheduled in the scheduling horizon h. Each task T_i is associated with its processing time duration $p_i \in \mathbb{N}$ and a vector $\mathbf{r}_i = (r_0^i, r_1^i, ..., r_m^i)$ $r_j^i \in \mathbb{N}_0$ $\forall R_j \in R$ stating the task's requirements of the corresponding resource, we will assign each task its start time s_i and completion time $C_i = s_i + p_i$, $s_i, C_i \in \mathbb{N}_0$. We state that if a task T_i requires the resource R_0, it is the only resource required by that task. Let $J = \{J_1, J_2, ..., J_k\} \subseteq T$ $J = \{T_i | r_0^i > 0\}$, meaning it is a subset of tasks that require the resource R_0 (need to be scheduled on the energy-consumption significant resource). We assume that J is a non-empty set. The tasks in J will be referred to as the energy consumption-intensive tasks.

Same as for the RCPSP problem, we define a directed acyclic graph $G = (V, E)$ with V being a set of n vertices. Each vertex $v_i \in V$ corresponds to the task T_i. The set of edges E of the graph represents the precedence relations between the tasks.

The energy-consumption significant resource R_0 has a non-empty set of available states Σ. There is exactly one processing state $proc \in \Sigma$, in which the R_0 must be during the processing of the tasks. There is also a single off state $off \in \Sigma$; we require R_0 to be in this state during both the first and the last interval of the scheduling horizon. Similarly to Section 1.2.2, we can therefore define the first interval in which a job can be processed as I_{early} and the latest interval in which a job can be processed as I_{late}.

In each interval I_i, the machine is transitioning between two states. When R_0 stays in the same state during an interval, we define it as transitioning from the state to itself. The parameters of the transitions are expressed by the transition time function TF and transition power function PF, both of them defined and described in Section 1.2.2.

To address the presence of work shifts of given resources, let us define a vector $\mathbf{w} = (w_0, w_1, ..., w_m)$ $w_i \in \{1, \ldots, 24\}$ $\forall R_i \in R$. This vector represents the length of the work shift of each resource. For simplicity, we consider the shifts to be consecutive, without interruptions, and periodically repeating with a period of 24, representing one day, starting from the first interval I_1. Since we are trying to minimize the TEC and utilize the capacity of the resource R_0, it would not make sense to limit its possibilities with working shifts; hence we set the w_1 to 24, meaning the resource is operational in all intervals.

A solution to this problem is a pair $(\mathbf{s}, \Omega)$, where $\mathbf{s}$ is a vector of length n, $(s_1, s_2, \ldots, s_n) \in \mathbb{N}_0^n$ and Ω is a vector of length h, $(\Omega_1, \Omega_2, \ldots, \Omega_h) \in (\Sigma \times \Sigma)^h$. The vector $\mathbf{s}$ contains the start times of the given tasks. The vector Ω contains the transitions the resource R_0 is undergoing in each interval of the scheduling horizon.

The pair $(\mathbf{s}, \Omega)$ represents a feasible solution if all of the following conditions are satisfied:

1. It holds that $C_i \leq h$ $\forall T_i \in T$, meaning that all the tasks are scheduled inside the given scheduling horizon.
2. At each interval I_i, the sum of j-th resource requirements of all tasks scheduled (either starting in this interval or starting in some interval I_{i-a} where $a < p$ of corresponding task) in this interval must be lower or equal to the corresponding resource capacity R_j for each resource.
3. The start times respect the precedence relations stated in the graph G: $\forall (i, j) \in E : s_i + p_i \leq s_j$.
4. The start times respect the work shifts of the resources: for each resource $R_i \in R$ all the tasks $T_j \in T$ requiring the resource R_i (therefore having $r_i^j > 0$) must be scheduled with their whole processing time duration inside some the working shifts of this resource, formally $\forall i \in \{0, 1, \ldots, m\}$, $\forall j \in \{1, \ldots, n\}$ such that $r_i^j > 0$, $\exists q : q = k \cdot 24, k \in \mathbb{N}_0, q \leq h$ such that $s_j \in \{q, \ldots, q + w_i - p_j\}$.
5. When a task is being processed on the resource R_0, it is undergoing a $(proc, proc)$ transition, meaning that $\forall T_j \in J, \forall i \in \{s_j, \ldots, s_j + p_j - 1\}$: $\Omega_i = (proc, proc)$.
6. During the first and last interval of the scheduling horizon R_0 is undergoing the (off, off) transition, i.e., $\Omega_1 = \Omega_h = (off, off)$.
7. All transitions are correctly following the transition time function, i.e., $\forall i \in \{1, 2, \ldots, h-1\}$ such that $\Omega_i = (\sigma_1, \sigma_1')$, $\Omega_{i+1} = (\sigma_2, \sigma_2')$, it holds that:

a. Ω_i and Ω_{i+1} are both existing direct transitions with non-zero time duration: $0 < TF(\Omega_i) < \infty$ and $0 < TF(\Omega_{i+1}) < \infty$.

b. Only existing zero-time transition or sequence of existing zero-time transitions is allowed between Ω_i and Ω_{i+1}.

c. All non-zero-time transitions are correctly following the time transition function. If $\Omega_i \neq \Omega_{i+1}$ and $0 < T(\Omega_{i+1}) < \infty$ then $\forall j \in \{1, \ldots, T(\sigma_2, \sigma_2') - 1\} : \Omega_{i+1+j} = \Omega_{i+1}$.

The overall project makespan objective is to minimize the maximum completion time C_{max}. It holds that $C_{max} \geq s_i + p_i \; \forall i \in \{1, \ldots, n\}$.

The total energy cost objective of the problem is then expressed as

$$TEC = \sum_{I_i \in I} c_i \cdot PF(\Omega_i).$$

To connect both aspects of the problem, we introduce a new parameter α which will be used as a balancing parameter; this leads to a bi-objective formulation of the whole problem defined as

$$\min\{\alpha \cdot \sum_{I_i \in I} c_i \cdot PF(\Omega_i) + (1 - \alpha) \cdot C_{max}\},$$

where $\alpha \in [0, 1]$. With α being set to 0, we do not consider the energy cost, meaning we are solving the RCPSP problem. Even though it might seem intuitive, with $\alpha = 1$, the problem is not necessarily equal to the single machine scheduling problem as defined in Section 1.2.2 since we must still consider the capacity of other resources and precedence constraints between the tasks.

For an illustration of the problem and the proposed notation, consider the following: We have a project of 8 tasks, processed on 3 resources of capacities $R_0 = 1, R_1 = 5$ and $R_2 = 3$. The precedence constraints, as well as information about the task processing times and resource requirements, are displayed in Figure 1.6. The tasks that need to be scheduled on the energy-significant resource R_0 are signified in the graph with a dark blue color. In this example, we consider all the resources to be operational at all intervals, hence setting $w_0 = w_1 = w_2 = 24$.

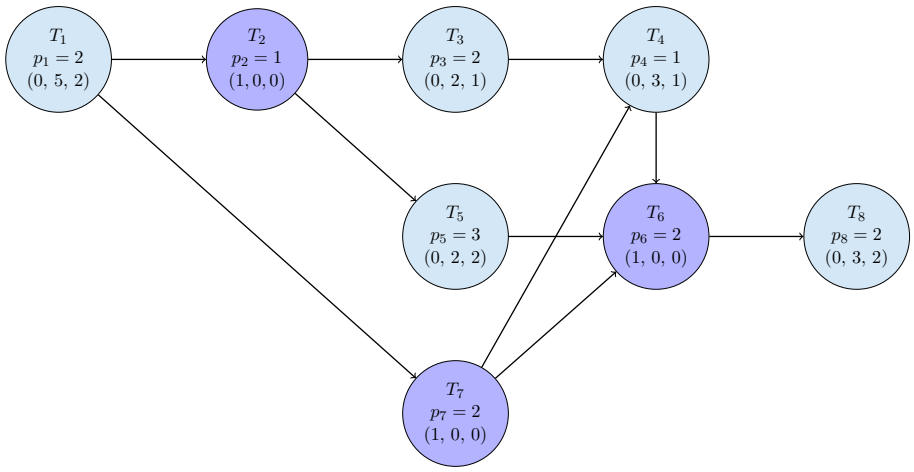


Figure 1.6: Energy-aware RCPSP example precedence relations graph, resource requirements are shown as (r_0, r_1, r_2).

Figure 1.7 shows the optimal schedule for the problem with α set to 0.75. It shows the optimal schedules for resources R_1 and R_2, as well as the optimal transitions between states for the resource R_0. The total project makespan is equal to 16, with a total energy cost of 133. The illustration shows that setting the α parameter toward the TEC objective results in prolonging the schedule and lowering the utilization of the resources in order to keep the energy costs lower.

On the other hand, Figure 1.8 shows the optimal schedule for the problem with α set to 0.25. Shifting the balancing parameter towards the C_{max} objective results in a shorter project makespan of 11 intervals and better utilization of the resources; however, this comes at a price of the increased total energy cost of 160.

1.2.4 Rescheduling

In this section, we will describe the problem of rescheduling as a reaction to changes in energy prices for the above defined Energy-aware RCPSP with TOU pricing and power-saving states of the machine. The problem of rescheduling lies in the modification of a given original feasible solution so that it has a better objective value under a new, modified vector of prices $\mathbf{c}$. However, the scale of possible changes in the schedule is subject to various constraints to limit deviations from the original solution.

Informally, we consider that there is a given feasible solution that was

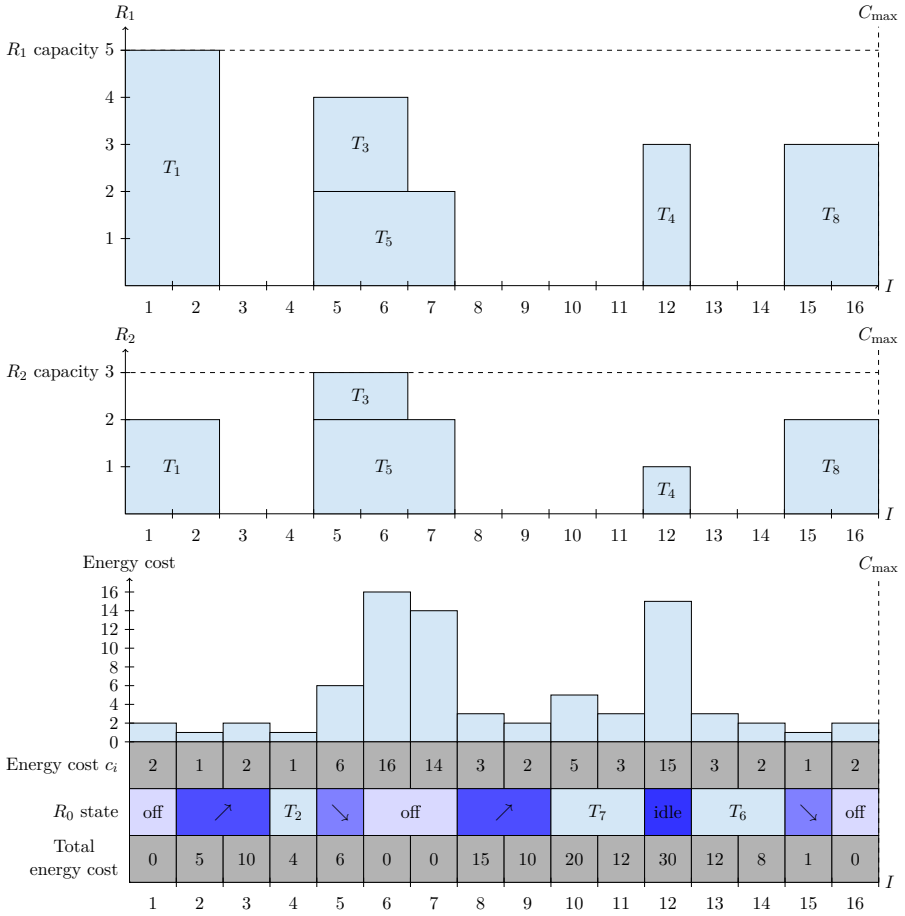


Figure 1.7: Example project schedule for resources R_1 and R_2 and the energy costs together with the optimal switching for R_0. Parameter α is set to 0.75.

created for a predicted vector of energy prices, and the production process is planned according to that schedule. However, after some time, we receive an updated vector of prices. The prices of the new predictions can be similar in the early intervals, giving us some time to react to the change. Yet it can also happen that the execution of some tasks in the production process cannot be changed anymore, meaning that there is a given number of intervals from the beginning of the scheduling horizon where the schedule cannot be changed. The goal is to create a new feasible solution, considering the update in prices. Additionally, the changes in start times of tasks might be limited either individually for each task or overall for all tasks combined.

Formally, we assume that for rescheduling, all of the definitions and rules from the problem statement remain the same; additionally, we define the following: Let $\mathbf{c}'$ be the vector of new energy prices, $|\mathbf{c}'| = |\mathbf{c}|$, meaning that the length of the scheduling horizon remains the same. We denote $I_{update} \in \{1, \ldots, h\}$ as the first interval for which the new vector of prices

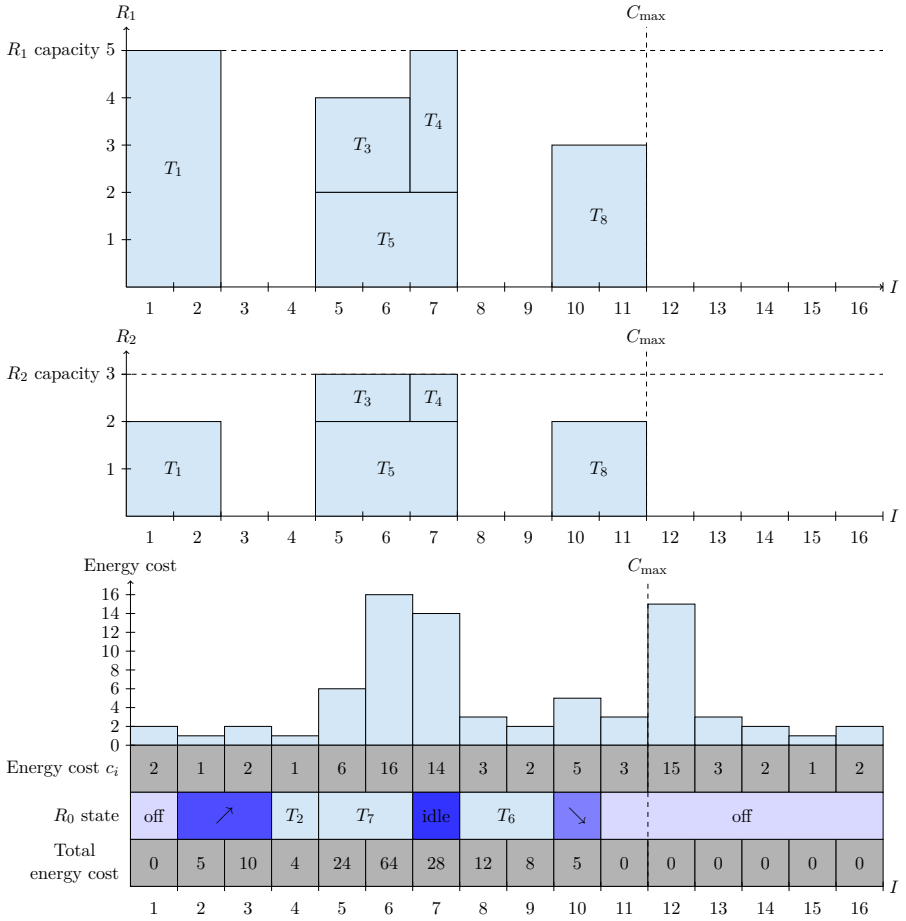


Figure 1.8: Example project schedule for resources R_1 and R_2 and the energy costs together with optimal switching for R_0. Parameter α is set to 0.25.

deviates from the original vector. Let $\mathbf{s}, |\mathbf{s}| = n$ be the vector of start times of tasks from the original feasible solution. Finally, let $I_{fix} \in \{1, \ldots, h\}$ be the given interval until which the schedule must remain the same as in the original solution. The number of intervals $I_{update} - I_{fix}$ is called the look-ahead window. The look-ahead window represents the time given to adapt to the upcoming price change.

To limit the number of differences in the solution, compared to the previous schedule, we define a difference function $DF : \mathbb{N}_0^n \times \mathbb{N}_0^n \to \mathbb{Z}_0$. The value $DF(\mathbf{s}, \mathbf{s}')$ expresses the difference in schedules between the original feasible solution start times $\mathbf{s}$ and the newly created solution start times, given by $\mathbf{s}'$. Further, let $\mu \in \mathbb{N}$ be a parameter expressing the maximal difference limit; we require that $DF(\mathbf{s}, \mathbf{s}') \leq \mu$.

Since the changes in the schedule of the resource R_0 are a desired reaction to the change in the energy prices, we do not limit the changes in the vector

of transitions Ω. However, since positions of the tasks scheduled up to I_{fix} must remain the same, it can happen that there are tasks with fixed positions that require R_0. Since the schedule must remain feasible, that naturally forces transitions of the resource in such a way that it is correctly undergoing the $(proc, proc)$ transition while processing tasks. Yet, if the different energy costs are making different transitions more suitable, we are allowed to make changes to the transitions anywhere in the schedule as long as they remain feasible.

Formally, for a rescheduling problem, solution pair $(\mathbf{s}', \Omega)$ is a feasible solution if, in addition to the constraints (1)-(7) specified in section 1.2.3, the following constraints are satisfied:

8. All of the tasks T_i, whose start time $\mathbf{s}_i \leq I_{fix}$, will remain scheduled at the same time: $\forall i \in \{1, \ldots, n \mid \mathbf{s}_i \leq I_{fix}\} : \mathbf{s}'_i = \mathbf{s}_i$.
9. All of the tasks T_i, whose start time $\mathbf{s}_i > I_{fix}$, will be scheduled after the interval I_{fix}: $\forall i \in \{1, \ldots, n \mid \mathbf{s}_i > I_{fix}\} : \mathbf{s}'_i > I_{fix}$.
10. The difference between the start times $\mathbf{s}$ and $\mathbf{s}'$, given by the function DF must be lower or equal to the parameter μ: $DF(\mathbf{s}, \mathbf{s}') \leq \mu$.

For an illustration of the problem, consider the same example as stated in Section 1.2.3, with 3 resources and 8 tasks. The precedence constraints, as well as the processing times and resource requirements of the tasks, are displayed in Figure 1.6. We consider all the resources to be operational at all intervals, setting $w_0 = w_1 = w_1 = 24$. This time the schedule is created for a new vector of prices $\mathbf{c}'$.

Figure 1.9 shows the optimal schedule for the new vector of prices for resources R_1 and R_2 as well as the optimal transitions between states for the resource R_0, with α set to 0.75. The new schedule for the resource R_0 is shown in red. For comparison, the original schedule for resource R_0, computed for the original vector of prices $\mathbf{c}$, shown in Figure 1.7, is shown in blue. For the original schedule, I_{fix} is set to 16, meaning that it is fixed for its whole duration. Note that while the tasks in the original schedule must, by definition, stay at the same interval, the transitions of resource R_0 are free to change as long as the solution remains feasible. Hence a different sequence of transitions is picked between intervals 5 and 9, leading to a TEC of 335. For comparison, the original switching behavior leads to a TEC of 338. On the other hand, in intervals 2 and 3, the resource needs to undergo the $(off, proc)$ transition despite high energy prices in order for a fixed task to be processed in interval 4. The TEC of the new schedule, created by the

rescheduling algorithm, results in a significantly lower TEC of 184, showing the potential of rescheduling in reaction to the changes in energy prices.

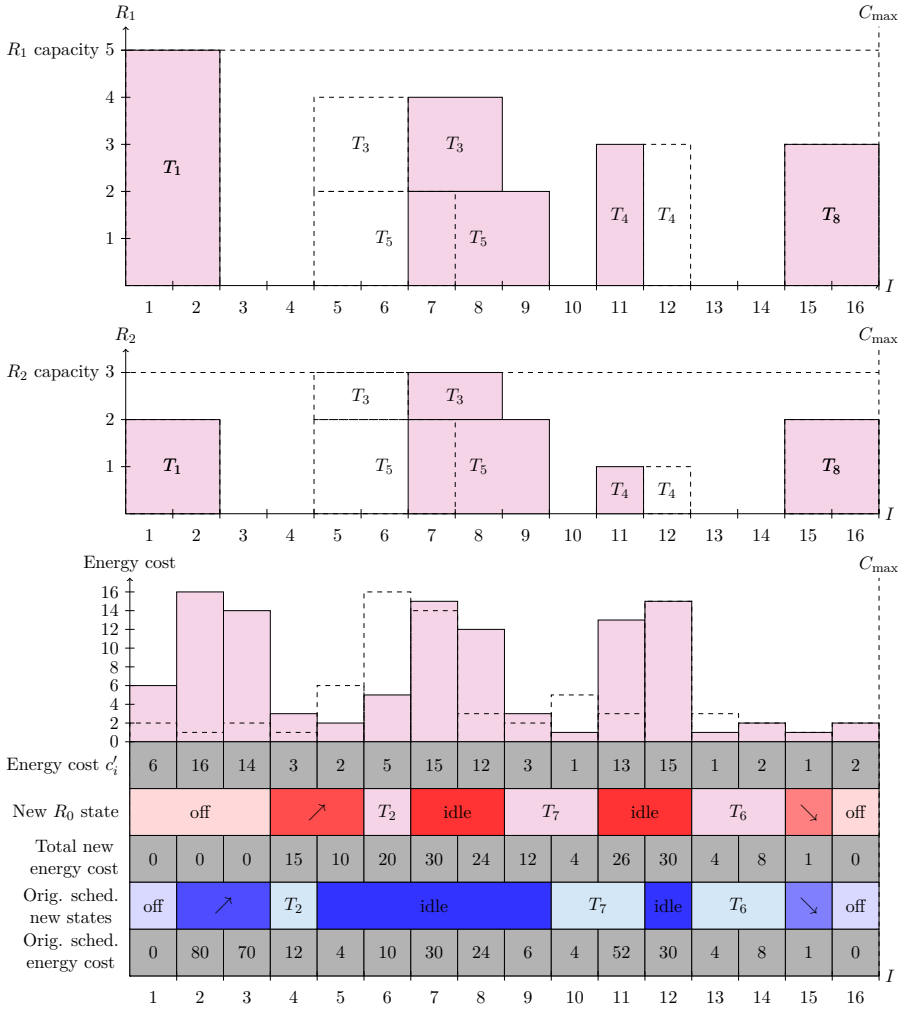


Figure 1.9: Example project schedule for new energy prices for resources R_1 and R_2 and the energy costs together with the optimal switching for R_0, shown in red. In blue, the original schedule for the tasks, when used with the new vector of prices, is shown. The parameter α is set to 0.75. Dashed lines show the original schedule and vector of prices.

Chapter 2

Related work

The project scheduling problem represents a well-known optimization challenge, and it is widely recognized as a topic of high importance for both theoretical research and practical applications. From a practical standpoint, good and effective project scheduling plays a key role in project management. Through its optimization, organizations can minimize delays, resource wastage, and other associated costs, which can directly impact the likelihood of project success. It comes as no surprise that project scheduling has captured a large amount of attention from researchers and has been extensively studied throughout the years in multiple variants and with diverse solution approaches [24].

In the early research, the scheduled tasks were often defined solely by their duration, which might be considered an oversimplification. To make the problem more applicable in real-world scenarios, additional constraints concerning precedence relations and resource limitations were added. This led to the rise of the Resource-Constrained Project Scheduling Problem, which is now widely regarded as a standard and foundational problem in project scheduling [24].

2.1 RCPSP

The RCPSP was first introduced by Wiest in 1963 [53, 45]. It was later proven by Blazewicz et al. that the problem belongs to the class of strongly

NP-hard problems [9]. Consequently, a large amount of literature is devoted to solving this problem using both exact and heuristic methods. The exact methods are often being used to solve small-sized problems to optimality [10, 28, 8, 56]; however, when attempting to solve large-scale problems, their time complexity is far from efficient [48]. That leads to the research of heuristic methods, capable of finding good-enough solutions in reasonable computational time [40, 48, 14, 22, 35, 44]. With project scheduling being a key application area for Constraint Programming, tools like the CP Optimizer have also proven effective in addressing this challenging problem, setting new lower bounds for some of the instances of RCPSP and, in some cases, even outperforming specialized algorithms [34].

To further address practical, real-world challenges, researchers have developed numerous variants of the RCPSP. These variants primarily differ in modifications to the fundamental assumptions regarding resources, task structures, and objective functions. For the resource constraints, it is worth mentioning the addition of non-renewable resources [32], meaning that the resource consumption is limited over the whole scheduling horizon and double-constrained resources [29], where both usage at every point of time and cumulative consumption over the scheduling horizon are limited. Another way of adapting the problem to real-world use cases lies in the addition of multi-mode tasks, where there exist multiple alternative ways to execute the task, each having its own specific processing time and resource requirements [25]. Other modifications can be found in changes to the objective function, focusing on minimization of the project lateness or (weighted) tardiness [4] instead of the makespan. The work of Habibi et al. provides an extensive overview of the mentioned and many other modifications and corresponding studies [24].

The work of Kolisch et al. [30] presents the project scheduling problem library PSPLIB, which is a set of benchmark instances for both single- and multi-mode RCPSP. It includes a detailed characterization of the instances, together with publicly available best-known solutions and lower bounds. The benchmark sets are still widely used in the evaluation and comparison of the algorithms [22, 35, 44].

2.2 Energy-efficient scheduling under TOU pricing

Similarly to RCPSP, the problem of energy-aware scheduling has, in recent years, become increasingly popular. A significant part of the literature is focused on the minimization of the total energy consumption under the time-

of-use (TOU) energy pricing. The TOU pricing system aims to flatten the demand for energy by introducing higher prices for the on-peak periods in contrast to the lower prices in off-peak periods. In this way, the pricing encourages users to schedule their consumption periods, presenting great opportunities for savings [21]. The given scheduling horizon is partitioned in discrete intervals, with assigned corresponding energy prices directly linked with the cost of processing a job in the given interval. The aim is to minimize the total energy cost while scheduling all of the tasks on a single machine. This formulation of the task was initially proposed in the work of Wan and Qi [52].

As the work already mentioned, the complexity of the problem is largely affected by the structure of the time slot costs. The problem can be solved in polynomial time in the case the prices in the intervals are non-decreasing [52]. Similarly, the work of Fang et al. presents a polynomial-time algorithm for the case when the TOU prices form a specific pyramidal structure [16]. However, those are particular cases that do not reflect real-world scenarios.

Over time, the research regarding scheduling under TOU pricing further developed in order to match the more complex real-world scenarios and machine environments. Such examples can be the parallel identical machines [19], unrelated parallel machines [12, 11], job shop [42, 33, 49] or flow shops [38, 3]. Some studies also explore integrating the total energy cost objective with other scheduling aspects, particularly duration. This approach gives rise to new bi-criteria objective functions that combine TEC with metrics such as total makespan [42, 19, 43] or tardiness [33, 13, 47].

2.3 Energy-aware RCPSP

When we focus on resource-constrained project scheduling concerning energy consumption, the literature becomes more limited and case-specific [54]. The observed trends in research suggest the main approach is based on the multi-mode RCPSP, where the energy consumption is directly linked to the execution mode of the resource, with higher energy-consumption modes being associated with faster processing time [54]. That makes space for optimization of switching between the modes considering the variable energy prices. The work of Hosseinian et al. presents the Energy-efficient multi-mode RCPSP formulation as a bi-objective mixed-integer linear programming model minimizing TEC and makespan. A multi-objective fruit fly optimization algorithm (MOFOA) is proposed and compared to other meta-heuristics on test samples from PSPLIB in terms of several performance measures [26].

The work of Zheng et al. focused on a similar kind of bi-criteria problem, yet instead of TEC, it is focused on the quantity of carbon emissions that are given for units of consumed energy. The study proposes a Pareto-based estimation of the distribution algorithm to solve the problem [55]. The carbon emissions are also a matter of concern in the work of Pouramin et al. [46]. Their study proposes a multi-objective mathematical model to address a Multi-Mode RCPSP with material ordering, taking into account energy-related factors such as renewable and non-renewable energy sources, TOU electricity tariffs, and carbon emissions. Two meta-heuristic algorithms, NSGA-II and MOPSO, were adopted to solve it. Other research approaches to the problem can be based on preemptive RCPSP under TOU tariffs [37] or even a combination of the mentioned approaches. Such an example can be the work of Peng et al. integrating the multi-mode RCPSP and the preemptive RCPSP for projects using energy-intensive equipment, using the Gurobi optimizer to solve small-scale projects, and developing meta-heuristic algorithms in the framework of the two non-dominated sorting genetic algorithm versions for large-scale projects [54].

2.4 Rescheduling

There is very little literature concerning the problem of rescheduling under TOU pricing. An example of such work is the study of Kong et al., considering on-peak and off-peak periods for energy costs. The scheduling is done on parallel machines; there is an optimal schedule of n_0 original jobs with n_N new jobs arriving at the beginning of the scheduling horizon, creating the need for rescheduling. A variable neighborhood search algorithm (VNS) is applied to obtain near-optimal solutions, and the work further improves the algorithm by proposing three novel swapping neighborhood structures [31]. Another approach to rescheduling was shown in the work of Forghani et al. with a two-step framework, first solving a MILP model to obtain an initial schedule and second suggesting a set of reactive rescheduling policies to address unexpected events [18].

2.5 Power-saving states

It is important to mention that most of the above-stated approaches to scheduling under TOU pricing do not consider the states of the resources, meaning that the machines can have, in addition to the processing state,

other power-saving states [1]. Optimal switching between the states presents another opportunity for significant energy savings but simultaneously adds another layer of complexity to the problem. The underlying single-machine scheduling problem minimizing the TEC with power-saving states of the machine can be defined in Graham's notation as 1, TOU | states | TEC. It was introduced by Shrouf et al. [50], initially considering the jobs as components of the global production schedule for a factory and, therefore, giving them a fixed order. The work also presents a mathematical model for solving the problem, treating it as an NP-hard problem. However, it was later shown in the work of Aghelinejad et al [2]. that with a fixed order of the jobs, the problem can be solved in polynomial time. The work also shows that the general version of the problem, without the fixed sequence of the jobs, is strongly NP-hard. Aghelinejad et al. proposed a generalized ILP model for solving the problem without a fixed sequence of the jobs [1]. However, only small instances of the model were solved optimally. A great improvement was later shown in the work of Benedikt et al., where a pre-processing technique called SPACES, used for computing the optimal switching of the machine states with respect to the energy costs, was proposed. The optimal switchings are associated with the shortest paths in an interval-state graph that describes all possible transitions between the machine states in time. Thanks to this technique, efficient models for both integer linear programming and constraint programming are presented. The best-performing model, called ILP-SPACES, was able to find optimal solutions even for large instances with up to 190 jobs and 1277 intervals within an hour of computation, outperforming the existing state-of-the-art method [6].

2.6 Conclusion

Even though, as shown in this chapter, there is a large amount of literature concerning project scheduling and Time-of-Use pricing, to the best of the author's knowledge, there was very little research done concerning the Resource-Constrained project scheduling problem with both time-of-use pricing and power-saving states of the machine as it was defined in the problem statement in Section 1.2. Since the problem statement is quite general, it can be used to model different problems simply through reductions. Similarly, very little literature addresses the problem of uncertainties in energy prices; rescheduling offers a promising approach to tackle this problem. Thus, we consider both of these problems an interesting field for research. With the presented master thesis, we aim to address this gap in the literature, explore its possibilities, and lay a foundation for future research on this topic.

Chapter 3

Preliminaries

The purpose of this chapter is to provide an introduction to the background knowledge of the methods and concepts that were utilized in this thesis. The understanding of these concepts is, therefore, key for further understanding of the proposed methods presented in Section 4. The chapter is split into two sections. The first section is dedicated to the description of the method for optimal switching precomputation. The second section focuses on the algorithm for generating instances for energy-aware RCPSP stated in Section 1.2, derived from the instances from PSPLIB [30].

3.1 Optimal switching problem

The work of Benedikt et al. [6] introduced a pre-processing technique called SPACES, which is used to compute the optimal switching between states in the single-machine scheduling problem, minimizing the TEC objective. This technique can efficiently pre-compute the optimal state transitions in polynomial time. Thanks to this precomputation, the transitions of the machine do not need to be explicitly formulated in the optimization models, which leads to a great simplification, lowering the number of both variables and constraints and consequently resulting in improved performance.

Given that SPACES can also be applied to determine the optimal state switching for the energy-consumption significant resource R_0 in the energy-aware RCPSP, stated in Section 1.2, we have utilized this method in the

presented thesis. Understanding this technique is crucial for grasping the presented methods; therefore, this section will provide a description of SPACES. For further detail, we refer the reader to the original work [6].

Informally, the optimal switching between two interval-state pairs is such a sequence of feasible transitions that the total energy cost is minimal. The optimal switching can be obtained by finding the shortest path in a weighted interval-state graph. In such a graph, each vertex represents a state and the beginning of a given interval with the addition of vertices representing the end of the last interval. The edges connecting the vertices represent feasible transitions with respect to the transition time function TF in both the duration and associated states. The weight of an edge is given as the total energy cost of the transition over all intervals of its duration with respect to the transition power function PF. If there is a feasible sequence of transitions between the states in given intervals, it can be found as the shortest path between the corresponding vertices; the total energy cost of the transition is equal to the sum of the weights of all edges in the shortest path.

Formally, let us consider a transition from interval-state pair (I_i, σ_1) to interval-state pair $(I_{i'}, \sigma_2)$, $i < i'$, with R_0 being in the state σ_1 at the beginning of interval I_i and in the state σ_2 at the beginning of the interval $I_{i'}$. The switching is a feasible sequence of transitions $(\Omega_i, \Omega_{i+1}, \ldots, \Omega_{i'-1})$, with respect to the function TF, $\Omega_j \in \Sigma \times \Sigma$ being a transition, that R_0 is undergoing in the interval I_j, with Σ being the set of available states of R_0. It must hold that $\Omega_i = (\sigma_1, \sigma)$ and i is the first interval of this transition. Similarly it must hold that $\Omega_{i'-1} = (\sigma', \sigma_2)$ and $i' - 1$ is the last interval of this transition, $\sigma_1, \sigma_2, \sigma, \sigma' \in \Sigma$. The optimal switching is such a switching that minimizes the following expression:

$$\sum_{j=i}^{i'-1} c_j \cdot PF(\Omega_j)$$

The interval-state graph is given by the triplet $G_{interval\_state} = (V, E, d)$, it represents all feasible transitions of R_0 between states in time, with V being the set of vertices representing states and the beginning of intervals, with the addition of the ending of the last interval. V is defined as:

$$V = \{v_{1,off}\} \cup \{v_{i,\sigma} : I_i \in I \backslash \{I_1\}, \sigma \in \Sigma\} \cup \{v_{h+1,off}\}$$

E is defined as the set of edges, each edge $(v_{i,\sigma}, v_{i',\sigma'}) \in E$ represents a feasible direct transition (σ, σ') with $TF(\sigma, \sigma') = i' - i$. We consider only transitions that start at the beginning of I_2 or later and are completed at the latest at the beginning of interval I_h, as in the intervals I_1 and I_h R_0 must by definition be undergoing the transition (off, off). E is defined as:

$$\begin{aligned} E = &\{(v_{1,off}, v_{2,off})\} \\ &\cup \{(v_{i,\sigma}, v_{i+TF(\sigma,\sigma'),\sigma'}) : \sigma, \sigma' \in \Sigma,\ I_i \in I \setminus \{I_1\}, \\ &\qquad TF(\sigma, \sigma') \neq \infty,\ (i-1) + TF(\sigma, \sigma') \leq h - 1\} \\ &\cup \{(v_{h,off}, v_{h+1,off})\} \end{aligned}$$

Lastly, $d : E \to \mathbb{N}_0$ is defined as a weight associated with each edge $(v_{i,\sigma}, v_{i',\sigma'}) \in E$, given by the total energy cost of the given transition computed as:

$$w(v_{i,\sigma}, v_{i',\sigma'}) = \sum_{j=i}^{i'-1} c_j \cdot PF(\sigma, \sigma')$$

The optimal switching between (σ_1, I_i) and $(\sigma_2, I_{i'})$ correspond to the shortest path between v_{i,σ_1} and v_{i',σ_2} and can be computed in polynomial time using shortest path algorithms, since the graph $G_{interval\_state}$ does not contain negative cycles. However, for the scheduling problem, we do not need to consider all of the transitions. Since all of the jobs need to be processed within intervals with the $(proc, proc)$ transition, we are interested only in the optimal switching around such intervals, namely between two consecutive intervals with the $(proc, proc)$ transition, between the very first interval with transition defined as (off, off) and the first processing interval with $(proc, proc)$ transition and finally the last interval with $(proc, proc)$ transition and the very last interval, where the transition is again required to be (off, off).

Let the cost of the optimal switching be denoted by the function $l : V \times V \to \mathbb{Z}_0$. The costs of optimal switching between states $\sigma, \sigma' \in \{off, proc\}^2$ are then given by function $c^* : (I \cup \{I_{h+1}\})^2 \to \mathbb{Z}_0$ as follows:

$$c^*(i,i') = \begin{cases} l(v_{i,proc}, v_{i',proc}) & \text{if } i > 2 \text{ and } i' < h \\ l(v_{i,off}, v_{i',proc}) & \text{if } i = 2 \text{ and } 2 < i' < h \\ l(v_{i,proc}, v_{i',off}) & \text{if } 2 < i < h \text{ and } i' = h \\ l(v_{i,off}, v_{i',off}) & \text{if } i, i' \in \{1,2\} \text{ or } i, i' \in \{h, h+1\} \\ \infty & \text{otherwise} \end{cases}$$

Note that when calling the function $c^*(i,i')$, the index i corresponds to the interval I_i in which the switching starts, the index i' then corresponds to the interval $I_{i'}$, with the switching ending at the beginning of interval $I_{i'}$, hence the need to define the function also for the value of $h+1$, which corresponds to the end of the last interval.

The SPACES algorithm computes the values of c^* efficiently in polynomial time, using the Dijkstra algorithm to obtain the shortest paths [6]. The algorithm performance can be further improved using parallel computing or caching. However, since this algorithm is not the main focus of the presented thesis, we will be using only a non-parallel version of the algorithm, and when comparing the performance of the later presented models, we will not consider the SPACES algorithm runtime since it is negligible in comparison to the computational times required by the ILP or CP solvers.

For illustration, consider the transition diagram given in Figure 1.4 and the example given in Figure 1.5. Figure 1.7 shows a part of the interval-state graph $G_{interval\_state}$. In the intervals I_4 and I_{10} the resource R_0 is undergoing the $(proc, proc)$ transitions marked in blue. The green path signifies the optimal switching sequence between these intervals corresponding to turning the resource off and then on again, with the TEC of the switching equal to 31. Another feasible switching sequence is marked in red, corresponding to keeping the machine idle. However, its TEC is equal to 82 and, therefore, will not be selected as an optimal switching behavior.

3.2 Instance generation

This section introduces a method for generating instances of the energy-aware RCPSP problem with TOU prices and power-saving states of the resource R_0. We have opted to present this topic in a separate section, as the task of instance generation for the stated problem is non-trivial. However,

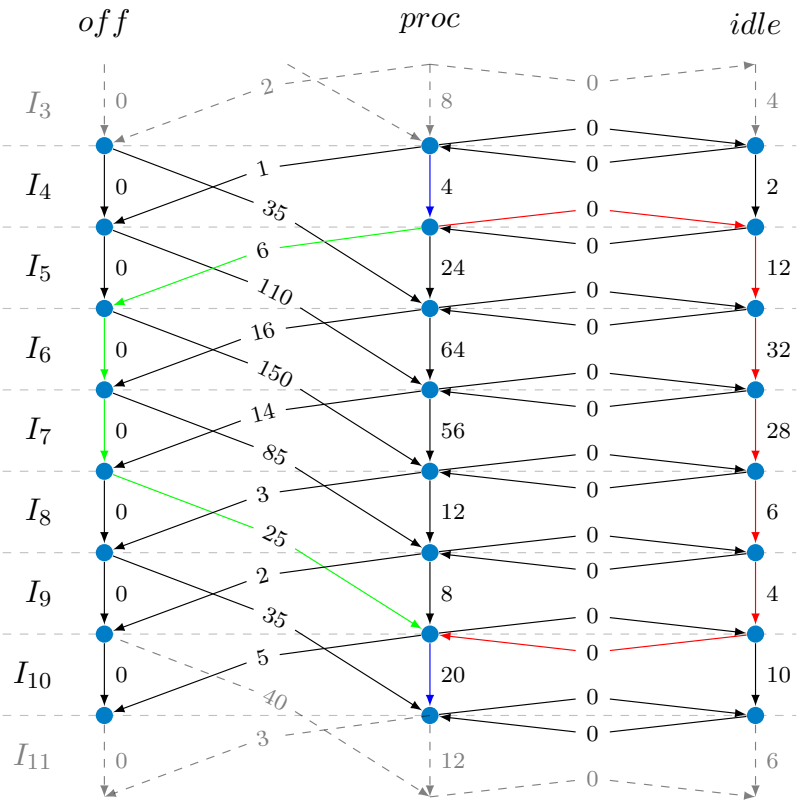


Figure 3.1: A part of the interval-state graph, corresponding to the example from Section 1.2.3, with optimal switching marked in green and sub-optimal feasible switching marked in red. Example adapted from Benedikt et al. [6].

understanding the process is not crucial for later proposed methods; therefore, if uninterested, we refer the reader to Section 4. The instance generating method utilizes the publicly available RCPSP instances from PSPLIB[1] and real data for TOU energy prices in Czechia from OTE[2]. Firstly, we will focus on the process of generating of scheduling instances; in the second part, we will describe the extension of this method for generating rescheduling instances, which also considers the addition of work shifts.

The idea behind the instance generating lies in taking a PSPLIB instance for RCPSP as input and extending it by adding the energy-consumption significant resource R_0 along with the vector of TOU prices. We then select a subset of tasks that will be changed to energy consumption-intensive tasks that require the resource R_0. PSPLIB provides instances for RCPSP with 30, 60, 90, and 120 tasks. With the growing size of the instances, the optimal project makespan also grows. However, in the case of the stated problem of energy-aware RCPSP, the whole project schedule must fit inside the considered scheduling horizon. In order to increase the number of tasks

[1] https://www.om-db.wi.tum.de/psplib/
[2] https://www.ote-cr.cz/en

without significantly increasing the length of the scheduling horizon, instead of selecting larger RCPSP instances, we will take multiple smaller RCPSP instances as input and merge them into one instance as parallel projects. These instances will share resources with one another and, therefore, not act as completely independent projects.

3.2.1 Scheduling instance generation

In order to observe differences in the behavior of the further presented methods, we allow to control from which parts of the initial schedule and in what percentage the tasks to be marked as energy consumption intensive will be selected.

The input of the instance generating method is the following: a set S of the RCPSP instances, the desired length of the scheduling horizon h, the vector of prices $\mathbf{c}_{init}$ from which the vector $\mathbf{c}$ will be constructed, part P of the schedule where tasks will be added to the set J of energy consumption intensive tasks and percentage p of the tasks belonging to P which will be added to J. The process of the instance generating goes as follows:

1. Solve each of the instances in S as an instance of RCPSP using a constraint programming model. In this thesis, we used IBM CP Optimizer to solve the constraint programming models. Given its specialization in scheduling problems, IBM CP Optimizer is mostly capable of solving the RCPSP instances quickly, and a 60-second timeout is enough for most instances [34]. However, the PSPLIB dataset still contains instances that were not yet solved to optimality within this timeout; since we do not want to include these instances in the dataset, it was decided that if the time limit is reached, the whole instance generating process is stopped as unsuccessful.
2. For each solved instance, we split its schedule based on the value of C_{max} of the instance into 3 equal parts called the beginning, middle, and end parts.
3. For each instance, we add a new resource with the capacity of 1. The new resource is added as R_0, and the other resource capacities remain unchanged. The same goes for resource requirements, r_0^i is set to 0. In total, we now have $m + 1$ resources.
4. For each instance, we select p percent of the tasks that are in their whole duration scheduled inside of the selected part of the schedule P. For

these tasks, we set the resource requirement r_0 to 1 and all other resource requirements to 0.

5. We merge the instances from S, modified in steps 1-4, into one; for each resource R_i, its capacity is set as a maximum of this resource's capacities over all of the RCPSP instances. If there is an RCPSP instance that has fewer resources than the number of resources in the merged instance, we add 0 as this resource requirement for each of its tasks. In the process of merging, we offset task indexes of the currently merged instance by the sum of the number of tasks in the previously merged instances so that there are no overlaps. Tasks with a processing time equal to 0 are not allowed; if there is such a task, we set the processing time to 1.

6. We add the vector of costs $\mathbf{c}$. We compose it by taking first h elements of $\mathbf{c}_{init}$. If h is higher than the number of elements in $\mathbf{c}_{init}$, we start periodically repeating its elements.

7. We save the instance into a file. The file follows the structure of PSPLIB instances, with the addition of vector $\mathbf{c}$ appended at the last line. Since this way of instance generation does not consider the work shifts, we consider the length of all of them to be equal to 24. The file structure is following (variables according to the problem statement):

```
n (m + 1)
R_0 R_1 ... R_m
p_1 r_0 r_1 ... r_m number_of_successsors successor_1 ...
p_2 r_0 r_1 ... r_m number_of_successsors successor_1 ...
.
.
.
p_n r_0 r_1 ... r_m number_of_successsors successor_1 ...
c_1 c_2 ... c_h
```

3.2.2 Rescheduling instance generating

The process of instance generating for rescheduling introduces a few changes. Firstly, the scheduling horizon length is fixed to 168. This represents 1 week of scheduling, with each interval taking 1 hour. Secondly, for each instance, we now create two files, each with two different vectors of prices $\mathbf{c}_1$ and $\mathbf{c}_2$. This represents the original instance and the instance with changes in electricity prices. The two vectors can be any arbitrary sequences of numbers; however, for the purpose of this thesis, we are again working with the values given by the day-ahead and inter-day markets from OTE. The reasoning behind

the selection of data will be addressed further in the work when describing datasets for each of the presented experiments. The last change lies in the addition of work shifts. The durations of jobs need to be normalized in such a way that the longest jobs are at most 8 intervals long so that they can be processed within an 8-hour working shift. We will further introduce an algorithm that helps us determine the lengths of the working shifts for each resource.

The scenario we aim for with the rescheduling instances is that there is a given project that fits into one factory workweek. As a reaction to the electricity prices, the schedule can be made in such a way that it prolongs the work to days of the weekend. That can lead to a lowering of the TEC. The addition of the work shifts must be done carefully so that it does not restrict the possibilities of the model too much and leaves enough space for task movements to take advantage of the variabilities in energy prices. This led to a series of preliminary experiments. Based on their results, it was settled that to achieve a good granularity of the results with changes of the balancing parameter α, the work shifts must be set in such a way that the whole schedule fits into the scheduling horizon of length 96 (corresponding to 4 days), the remaining time serves as a space for changes in the schedule to minimize TEC. Figure 3.2 shows two images with the C_{max} results on the x-axis and TEC results on the y-axis. Each point represents a solution for different α. It can be seen that when we only require the schedule with work shifts to fit into a scheduling horizon of length 120 (leaving 24 intervals less for manipulation according to the changes in energy prices compared to 96), we are, in this case, able to get only two different solutions, oppose to four different solutions when we require the schedule to fit into a scheduling horizon of length 96.

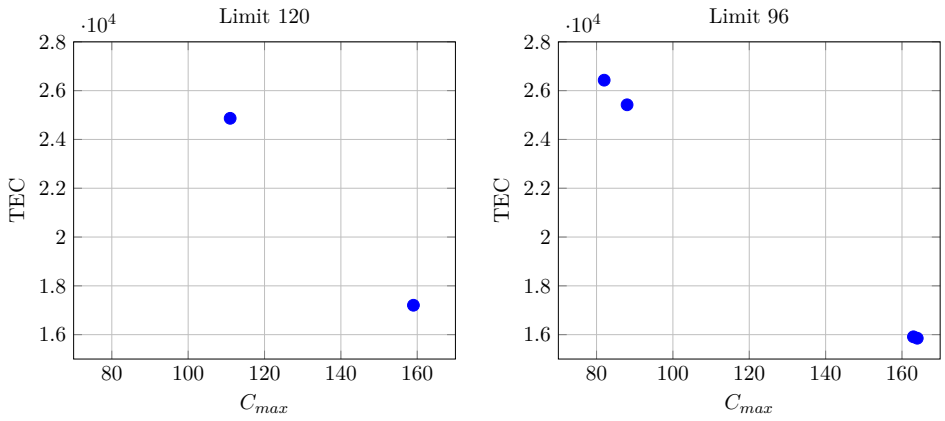


Figure 3.2: Numbers of different solutions with changes to α, based on the requirement to fit the whole schedule into 120 intervals (left) and 96 intervals (right).

The input of the instance generating method is the following: a set S of the RCPSP instances, the vectors of prices $\mathbf{c}_{init\_original}$ and $\mathbf{c}_{init\_changed}$ from which the vectors $\mathbf{c}_1$ and $\mathbf{c}_2$ will be constructed, the vectors are of length 168, part P of the schedule where tasks will be added to the set J of energy consumption intensive tasks and percentage p of the tasks belonging to P which will be added to J. The process of rescheduling instance generating goes as follows:

1. Solve each of the instances in S as an instance of RCPSP using the constraint programming model.
2. For each solved instance, we split its schedule based on the value of C_{max} of the instance into 3 equal parts called the beginning, middle, and end parts.
3. For each instance, we add a new resource with the capacity of 1. The new resource is added as R_0, and the other resource capacities remain unchanged. The same goes for resource requirements, r_0^i is set to 0. In total, we now have $m + 1$ resources.
4. For each instance, we select p percent of the tasks that are fully scheduled inside of the selected part of the schedule P. For these tasks, we set the resource requirement r_0 to 1 and all other resource requirements to 0.
5. For each instance, we normalize the processing times of its tasks. Let p_{max} be the maximal processing time of a task in the instance. All processing times of tasks that do not require the resource R_0 are then set to $\lfloor p_i/p_{max} \rfloor \cdot 8$; if there are tasks that result in a processing time of 0, we set their processing time to 1. The tasks that require resource R_0 are left unaffected.
6. We merge the instances from S, modified in steps 1-4 into one; for each resource R_i, its capacity is set as a maximum of this resource's capacities over all of the RCPSP instances. If there is an RCPSP instance that has fewer resources than the number of resources in the merged instance, we add 0 as this resource requirement for each of its tasks. In the process of merging, we offset task indexes of the currently merged instance by the sum of the number of tasks in the previously merged instances so that there are no overlaps. Tasks with a processing time equal to 0 are not allowed; if there is such a task, we set the processing time to 1.
7. In this step, we add the work shifts for the instance. As it was already mentioned, we require the whole schedule with work shifts to fit into a scheduling horizon of length 96. To achieve that, we designed an algorithm that iteratively determines the lowest lengths of work shifts needed to satisfy this condition. In the first step, we check that the schedule is shorter than 96 with all work shifts set to the length of 24;

that is done using the constraint programming model presented in step 1 in Section 3.2.1. If not, the whole instance generating process is stopped here as unsuccessful. We continue by setting the length of all of the shifts to 8 hours, except the resource R_0, which always has 24-hour long shifts. Then starts the iterative part of the algorithm. In each iteration, we check the length of the schedule with the given shifts; if the schedule is longer than 96, we check which tasks are at the end of the schedule and pick such tasks or tasks that end the latest, and their required resources do not have 24-hour shifts. We then add 2 hours to the length of the shifts of the resources required by those tasks, up to 24-hour long shifts, and continue with the next iteration. This repeats until the schedule with work shifts does not fit into the 96 intervals, resulting in the final lengths of work shifts for all resources.

To check the length of the schedule with shifts, we use a constraint programming model for RCPSP, which additionally forbids the tasks to extend outside of the shifts given in the iteration for all of the resources required by the task.

8. We save the instance into two files. The first file contains the vector of costs $\mathbf{c}_1$ composed from the values $\mathbf{c}_{init\_original}$ the second file contains the vector of costs $\mathbf{c}_2$ composed from the values $\mathbf{c}_{init\_changed}$ the vectors are in this case all of length 168 and therefore do not need any further modification. The file structure for both files is as follows (variables according to the problem statement):

```
n (m + 1)
R_0 R_1 ... R_m
w_0 w_1 ... w_m
p_1 r_0 r_1 ... r_m number_of_successsors successor_1 ...
p_2 r_0 r_1 ... r_m number_of_successsors successor_1 ...
.
.
.
p_n r_0 r_1 ... r_m number_of_successsors successor_1 ...
c_1 c_2 ... c_h
```

Chapter 4

Methods

In this chapter, we will introduce the newly designed models for the energy-aware RCPSP with TOU energy prices for both Constraint Programming (CP) and Integer Linear Programming (ILP) formalisms. Afterward, we will introduce the ILP model, which is capable of rescheduling as a reaction to changes in energy prices. Lastly, we will describe an objective balancing method, which allows us to more accurately control the balance between the TEC and C_{max} objectives using the α parameter.

4.1 Constraint Programming model

Before we start with the description of the created models, let us first briefly present the basic concepts used in Constraint Programming. We will adopt the notation from IBM CP Optimizer [34]. The main modeling expression used in CP is the *interval variable*, representing an activity that needs to be put in the schedule. Let x be an interval variable; its duration is denoted as LengthOf(x), its start time as StartOf(x), and its end time as EndOf(x). Another way of modeling activities in the schedule is the use of *optional interval variables*. The variable can be either present or absent in the schedule. Moreover, they can be constrained in a similar way to interval variables. Only if the variable is present in the schedule does it need to satisfy all the constraints; if absent, the variable is completely excluded from the schedule and is not considered when checking if the constraints are satisfied or a default value is used. The presence can be checked using the method PresenceOf, which returns 1 if the variable is active and 0 otherwise.

We will further briefly introduce a few methods that can be used to constrain variables in relation to one another. The NoOverlap constraint ensures that for the given set of interval variables, there is no pair of variables in the set that would overlap in the resulting schedule. The constraint EndBeforeStart takes as input an ordered pair of internal variables. It ensures that the activity given by the first interval variable is finished before the start of the activity given by the second interval variable. Further, the Element constraint takes as parameters a vector and a variable, indexing to this vector with the assigned value of the variable and returns the value of the indexed element. Compared to traditional indexing, the method allows us to index using the parameters of the interval variables, such as start or end time, which are dynamically changing during the optimization process. Lastly, we will introduce the method Pulse. The method takes as an input an interval variable x and an integer k. When called, it generates a stepwise function of value k between StartOf(x) and EndOf(x) and zero otherwise. We can further sum the values of such functions and constrain the maximal value; this technique can be used for modeling resource consumption.

In the rest of this section, we will present two models for CP. Both of the models use the precomputed optimal switching, obtained using the SPACES method, described in Section 3.1. The models differ in the way of modeling the optimal switching in spaces between the processing of jobs on resource R_0. The first approach models the spaces as optional variables with fixed start and end times. The second approach models the spaces without the start and end time fixation, allowing them to move freely around the scheduling horizon. At the same time, the number of variables is lowered compared to the first approach. This way of modeling is known to be the best-performing CP model for single-machine scheduling under TOU prices, from the work of Benedikt et al. [6]. If this precedence also holds for the energy-aware RCPSP studied in this thesis, will be a subject to further experiments in Chapter 5.

4.1.1 CP model with fixed spaces

Firstly, let us define a vector $\mathbf{c}_j^{job}$ for all tasks $T_j \in J$. The i-th element of vector $\mathbf{c}_j^{job}$ expresses the total energy cost for the processing of task T_j if started at interval I_i and can be computed as follows:

$$\mathbf{c}_{j,i}^{job} = \sum_{k=i}^{i+p_j-1} \mathbf{c}_k \cdot PF(proc, proc)$$

for each task $T_j \in J$ and $i \in \{1, \ldots, h - p_j + 1\}$.

Variables: to represent each task $T_j \in T$ we create a new interval variable x_j of a fixed length p_j, this variable corresponds to the processing of T_j, starting at the time StartOf(x_j) and ending at the time EndOf(x_j). To explain the indexing, consider each interval $I_i \in I$ to span from the time $i-1$ to the time i, meaning that the first interval I_1 spans from time 0 to 1 and the last interval I_h from time $h-1$ to h. By definition, each of the interval variables must be scheduled. To represent the optimal switching in spaces around the processing of tasks, we create an optional interval variable $z_{i,j}$ with fixed start and end time for each possible switching in the following way:

$$\begin{gathered} z_{i,j} : \text{StartOf}(z_{i,j}) = i, \text{EndOf}(z_{i,j}) = j \\ \forall i \in \{0, \ldots, h-1\} \quad \forall j \in \{i+1, \ldots, h\} \end{gathered} \tag{4.1.1}$$

Objective: the model minimizes the following objective

$$\begin{aligned} \alpha \cdot \Bigg(& \sum_{i=0}^{h-1} \sum_{j=i+1}^{h} \text{PresenceOf}(z_{i,j}) \cdot \mathbf{c}^*(i+1, j+1) \\ & + \sum_{T_j \in J} \text{Element}(\mathbf{c}_j^{job}, \text{StartOf}(x_j) + 1) \Bigg) \\ & + (1-\alpha) \cdot \max_{T_i \in T} \{\text{EndOf}(x_i)\} \end{aligned} \tag{4.1.2}$$

with the first part representing the TEC part of the objective and the second part representing the C_{max} part of the objective. Note that when computing the TEC of the tasks, using the Element function, we consider only tasks that are in the set of energy consumption-intensive tasks J, while for the C_{max} objective, we consider all of the tasks and hence iterate over the set of all tasks T.

The solution must be feasible with respect to the following constraints:

$$\sum_{T_i \in T} \text{Pulse}(x_i, r_j^i) \leq R_j \quad \forall R_j \in R \tag{4.1.3}$$

$$\text{EndBeforeStart}(x_i, x_j) \quad \forall (i,j) \in E \tag{4.1.4}$$

$$\text{StartOf}(x_j) \geq 1 + TF(off, proc) \quad \forall T_j \in J \tag{4.1.5}$$

$$\text{EndOf}(x_j) \leq h - TF(proc, off) - 1 \quad \forall T_j \in J \tag{4.1.6}$$

$$\text{EndOf}(x_j) \leq h \quad \forall T_j \in T \tag{4.1.7}$$

$$\begin{gathered}\text{NoOverlap}(\{x_j : T_j \in J\} \cup \\ \{z_{i,j} : \forall i \in \{0, \ldots, h-1\} \quad \forall j \in \{i+1, \ldots, h\}\})\end{gathered} \tag{4.1.8}$$

$$\sum_{T_j \in J} \text{LengthOf}(x_j) + \sum_{i=0}^{h-1} \sum_{j=i}^{h} \text{LengthOf}(z_{i,j}) = h \tag{4.1.9}$$

$$\begin{gathered}\text{ForbidExtent}(x_j, \bigcup_{n=0}^{\lfloor h/24 \rfloor + 1} [n \cdot 24 + w_i, (n+1) \cdot 24]) \\ n \in \mathbb{N} \quad \forall T_j \in T \quad \forall R_i \in R : r_i^j > 0\end{gathered} \tag{4.1.10}$$

Constraint (4.1.3) ensures that the resource capacity is not exceeded at any interval for all resources.

With E being the set of edges of the graph G that express the precedence relations, the constraint (4.1.4) ensures that all of the precedence relations are followed correctly.

The constraints (4.1.5) and (4.1.6) together ensure that all of the energy consumption-intensive tasks are scheduled only from the interval I_{early} to interval I_{late}, thus respecting the turn-on and turn-off time of the resource R_0 and the rule that R_0 must be undergoing the (off, off) transition in the first and last interval.

The constraint (4.1.7) ensures that all of the tasks are scheduled inside the scheduling horizon. Compared to the constraints (4.1.5) and (4.1.6), we consider all of the tasks in this case.

Using the NoOverlap method, the constraint (4.1.8) ensures that the resource R_0 is never processing a task and undergoing an optimal switching at the same time.

The constraint (4.1.9), together with the NoOverlap constraint (4.1.8), ensures that for the resource R_0, the exact length of the scheduling horizon is filled either with the processing of tasks or with optimal switching. Note that thanks to the optional variables, only the used optimal switchings contribute to the sum. Consequently, since the whole scheduling horizon needs to be filled without overlaps, and since the tasks can be only scheduled between the intervals I_{early} and I_{late}, this also forces the resource to take the optimal switching for the turn-on and turn-off of the resource, in the early and late intervals.

The constraint (4.1.10) can be used to address the work shifts, given in vector **w**. The method ForbidExtend ensures that the task can only be scheduled inside the working shifts of all the resources the task requires. However, as it will be later mentioned in Chapter 5, the experiments that included the CP models were only done on instances that considered all of the shifts to be 24 intervals long, and hence, this constraint was not directly used. This is due to the fact that the ILP model was proven to be more suitable for further scenarios that also concerned the work shifts.

Additionally, we also consider the constraints (4.1.1), mentioned during the description of the variables. Note that thanks to the SPACES method for optimal switching pre-computation, we do not need to include any additional constraints to model the states and transitions of the resource R_0.

4.1.2 CP model with free spaces

Compared to the previous model, this model differed in that the start and end times of the variables representing the optimal switching are not fixed. Instead, we will represent the switching with variables of a fixed length that will be allowed to move around the schedule as necessary. This also allows us to lower the number of variables needed to represent the optimal switching since we can exactly compute the maximal number of optimal switchings of each length that can be used in the schedule. This method of modeling the optimal switching was proposed in the work of Benedikt et al. [6].

Variables: similarly to the previous model, a new interval variable x_j of a fixed length p_j is created to represent each task $T_j \in T$. This variable corresponds to the processing of T_j, starting at the time StartOf(x_j) and ending at the time EndOf(x_j). By definition, each of the interval variables must be scheduled. Optional interval variables are used to represent the optimal switching in the spaces around the task processing on the resource R_0. Let $l_{max} = h - \sum_{T_j \in J} p_j$ be the upper bound on the optimal switching length. The function $K(l) = \left\lfloor \frac{h - \sum_{T_j \in J} p_j}{l} \right\rfloor$ then gives the upper bound on the number of optimal switching of the length l that can appear in the schedule. The optional variables to represent the optimal switching are created in the following way:

$$\text{LengthOf}(z_{l,k}) = l \quad \forall l \in \{1, \ldots, l_{max}\} \quad \forall k \in \{1, \ldots, K(l)\} \tag{4.1.11}$$

Objective: Similarly to the model with fixed variables for optimal switching, we use the vector $\mathbf{c}_j^{job}$ that expresses the total energy cost for the processing of task j based on the interval I_i it starts in. Additionally, since the variables representing the optimal switching are not fixed in their position, they are not directly associated with the total energy cost of the switching. Hence we define the vector $\mathbf{c}_l^{space}$, the i-th element of the vector corresponds to the total energy cost of the optimal switching of length l, if started in the interval I_i. The values of $\mathbf{c}_l^{space}$ are given as follows:

$$\mathbf{c}_{l,i}^{space} = \mathbf{c}^*(i, i+l) \quad \forall i \in \{1, \ldots, h-l+1\}$$

The model minimizes the following objective:

$$\begin{aligned} \alpha \cdot \Bigg(& \sum_{l=1}^{l_{max}} \sum_{k=1}^{K(l)} \text{Element}(\mathbf{c}_l^{space}, \text{StartOf}(z_{l,k}) + 1) \\ & + \sum_{T_j \in J} \text{Element}(\mathbf{c}_j^{job}, \text{StartOf}(x_j) + 1) \Bigg) \\ & + (1-\alpha) \cdot \max_{T_i \in T} \{\text{EndOf}(x_i)\} \end{aligned} \tag{4.1.12}$$

with the first part representing the TEC part of the objective and the second part representing the C_{max} part of the objective. Note that when computing the TEC, we consider only energy consumption-intensive tasks in the set J, while for the C_{max} objective, we consider all of the tasks. The addition of 1 in the Element method is done to associate the start time with the corresponding interval when indexing.

The solution must be feasible with respect to the following constraints:

$$\text{StartOf}(z_{l,k}) \geq 0 \quad \forall l \in \{1, \ldots, l_{max}\} \quad \forall k \in \{1, \ldots, K(l)\} \tag{4.1.13}$$

$$\text{EndOf}(z_{l,k}) \leq h \quad \forall l \in \{1, \ldots, l_{max}\} \quad \forall k \in \{1, \ldots, K(l)\} \tag{4.1.14}$$

$$\begin{gathered} \text{PresenceOf}(z_{l,i}) \geq \text{PresenceOf}(z_{l,i+1}) \\ l \in \{1, \ldots, l_{max}\} \quad \forall i \in \{1, \ldots, K(l)-1\} \end{gathered} \tag{4.1.15}$$

$$\begin{gathered} \text{EndBeforeStart}(z_{l,i}, z_{l,i+1}) \\ \forall l \in \{1, \ldots, l_{max}\} \quad \forall i \in \{1, \ldots, K(l)-1\} \end{gathered} \tag{4.1.16}$$

$$\text{StartOf}(x_j) \geq 1 + TF(off, proc) \quad \forall T_j \in J \tag{4.1.17}$$

$$\text{EndOf}(x_j) \leq h - TF(proc, off) - 1 \quad \forall T_j \in J \tag{4.1.18}$$

$$\text{EndOf}(x_j) \leq h \quad \forall T_j \in T \tag{4.1.19}$$

$$\begin{gathered}\text{NoOverlap}(\{x_j : T_j \in J\} \cup \\ \{z_{l,k} : \forall l \in \{1, \ldots, l_{max}\} \quad \forall k \in \{1, \ldots, K(l)\}\})\end{gathered} \tag{4.1.20}$$

$$\sum_{l=1}^{l_{max}} \sum_{k=1}^{K(l)} \text{LengthOf}(x_{l,k}) = l_{max} \tag{4.1.21}$$

$$\sum_{T_j \in T} \text{pulse}(x_j, r_i^j) \leq R_i \quad \forall R_i \in R \tag{4.1.22}$$

$$\text{EndBeforeStart}(x_i, x_j) \quad \forall (i,j) \in E \tag{4.1.23}$$

$$\begin{gathered}\text{ForbidExtent}(x_j, \bigcup_{n=0}^{\lfloor h/24 \rfloor + 1} [n \cdot 24 + w_i, (n+1) \cdot 24]) \\ n \in \mathbb{N} \quad \forall T_j \in T \quad \forall R_i \in R : r_i^j > 0\end{gathered} \tag{4.1.24}$$

The constraints (4.1.13) and (4.1.14) ensure that all of the optimal switching variables, if present, are scheduled inside of the scheduling horizon.

The constraints (4.1.15) and (4.1.16) give an ordering to the variables representing optimal switchings and ensure that the variables are used in an order given by their indexes for switchings of the same length. This removes redundant solutions that are different just in the order of use of the variables.

The constraints (4.1.17) and (4.1.18) ensure that all of the energy consumption-intensive tasks are scheduled only from the interval I_{early} to interval I_{late}, thus respecting the turn-on and turn-off time of the resource R_0 and the rule that R_0 must be undergoing the (off, off) transition in the first and last interval.

Constraint (4.1.19) ensures that all of the tasks are scheduled inside the scheduling horizon. Compared to the constraints (4.1.17) and (4.1.18), we consider all of the tasks in this case.

Using the NoOverlap method, constraint (4.1.20) ensures that the resource R_0 is never processing a task and undergoing an optimal switching at the same time.

The constraint (4.1.21) ensures that the length of all used optimal switchings in the solution is equal exactly to the value of l_{max}; the optional variables that are not present in the solution do not count towards this sum. By the definition of l_{max}, this constraint, together with the constraint (4.1.20), also ensures that for the resource R_0, the whole scheduling horizon is filled either by the processing of tasks or by optimal switchings in the spaces around.

The constraint (4.1.22) ensures that the resource capacity is not exceeded at any interval for all resources.

With E being the set of edges of the graph G that express the precedence relations, the constraint (4.1.23) ensures that all of the precedence relations are followed correctly.

The constraint (4.1.24) can be used to address the work shifts, given in vector $\mathbf{w}$. As was already mentioned in Section 4.1.1, the experiments that included the CP models were only done on instances that considered all of the shifts to be 24 intervals long due to the fact that the ILP model was proven to be more suitable for further scenarios that also concerned the work shifts. Thus this constraint was not directly used.

Additionally, consider also the constraint (4.1.11) mentioned during the description of the variables.

4.2 Integer Linear Programming model

In this section, we will present an Integer Linear Programming (ILP) model for the energy-aware RCPSP stated in Section 1.2. The model uses the SPACES method, described in Section 3.1, for precomputation of the optimal switching of the resource R_0. Thanks to this method, the number of required constraints can be greatly reduced since the model does not need to account for the optimality of the transition between states and only selects from the optimal switchings based on the intervals in which the tasks are processed.

The ILP model uses the time-indexed formulation. Before we describe the model, we will add a new task T_{n+1} to set T. The task has a processing time p_{n+1} of 0 and does not require any resources, $r_j^{n+1} = 0 \quad \forall R_j \in R$. Based on the problem statement, it is the only task with a processing time of 0 in T. Furthermore, we will add a new edge $(i, n+1) \quad \forall i \in \{1, \ldots, n\}$ to the set of edges E of the precedence graph G. This means that the task T_{n+1} must be scheduled after all other tasks. This task will be used to determine the schedule's makespan.

Similar to the CP models, we will start by defining a vector $\mathbf{c}_j^{job}$ for all tasks $T_j \in J$. The i-th element of vector $\mathbf{c}_j^{job}$ expresses the total energy cost

for the processing of task T_j if started at interval I_i and can be computed as follows:

$$\mathbf{c}_{j,i}^{job} = \sum_{k=i}^{i+p_j-1} \mathbf{c}_k \cdot PF(proc, proc)$$

for each task $T_j \in J$ and $i \in \{1, \ldots, h - p_j + 1\}$. The model formulation is the following:

Variables: we create binary variable $x_{j,i} \in \{0, 1\}$ for all $T_j \in T, I_i \in I$. The value 1 represents that the j-th task is starting at the beginning of the i-th interval. We will define a binary variable $z_{i,j} \in \{0, 1\}$ for all $i \in \{1, \ldots, h\}, j \in \{1, \ldots, h + 1\}$. Variable $z_{i,j}$ equals 1 if the resource R_0 is undergoing the optimal switching from the beginning of the interval I_i to the beginning of the interval I_j, respectively to the end of the interval I_h if $j = h + 1$.

However, an edge case arises with this formulation that needs to be addressed. Since all of the tasks need to be scheduled within the scheduling horizon, there can be a task that is still being processed within the last interval I_h; this is a feasible solution since the task T_{n+1} that must come after has a processing time of 0. However, since we are using the time-indexed formulation, there is no variable $x_{n+1,h+1}$, which would represent the start time of the task T_{n+1}. That would cause the solver not to produce a feasible solution. To address this problem, we denote, without loss of generality, that the task T_{n+1}, as the only task, can be scheduled in the last interval of processing of its preceding task. The final length of the schedule is then determined as the start time of the task T_{n+1} plus one for the remaining interval of the preceding task. This can be done since the set of tasks T is a non-empty set, and other tasks with a processing time of 0 are not allowed.

Objective: the model minimizes the following objective:

$$\begin{aligned} \alpha \cdot \left(\sum_{T_j \in J} \sum_{i=1}^{h} x_{j,i} \cdot \mathbf{c}_{j,i}^{job} + \sum_{i=1}^{h} \sum_{j=i}^{h+1} z_{i,j} \cdot \mathbf{c}_{i,j}^{*} \right) \\ +(1 - \alpha) \cdot \left(\sum_{k=1}^{h} k \cdot x_{n+1,k} + 1 \right) \end{aligned} \tag{4.2.1}$$

with the first part representing the TEC for R_0 state switching and processing of energy consumption-intensive tasks in J. The second part represents

the C_{max} objective.

The model must be feasible with respect to the following constraints:

$$\sum_{I_i \in I} x_{j,i} = 1 \quad \forall T_j \in T \tag{4.2.2}$$

$$\sum_{I_k \in I} k \cdot x_{j,k} - \sum_{I_k \in I} k \cdot x_{i,k} \geq p_i \quad \forall (i,j) \in E : j \neq n+1 \tag{4.2.3}$$

$$\sum_{I_k \in I} k \cdot x_{j,k} - \sum_{I_k \in I} k \cdot x_{i,k} \geq p_i - 1 \quad \forall (i,j) \in E : j = n+1 \tag{4.2.4}$$

$$\sum_{T_j \in T} \sum_{i'=max\{1,i-p_j+1\}}^{i} r_k^j \cdot x_{j,i'} \leq R_k \quad \forall R_k \in R \quad \forall I_i \in I \tag{4.2.5}$$

$$\begin{gathered} x_{j,i} = 0 \\ \forall T_j \in J \quad \forall i \in \{1, \ldots, 1 + TF(off, proc)\} \cup \\ \{h - TF(proc, off) - p_j + 1, \ldots, h\} \end{gathered} \tag{4.2.6}$$

$$\sum_{T_j \in J} \sum_{i'=max\{1,i-p_j+1\}}^{i} x_{j,i'} + \sum_{j=1}^{i} \sum_{k=i+1}^{h+1} z_{j,k} = 1 \quad \forall I_i \in I \tag{4.2.7}$$

$$x_{j,i} = 0 \quad \forall i \in I_i : (i \bmod 24) > w_m - p_j + 1, \forall m : r_m^j > 0, \forall T_j \in T. \tag{4.2.8}$$

The constraint (4.2.2) expresses that each task must be scheduled; we consider the task T_{n+1} to be part of T, and therefore, this constraint also holds for this task.

With E being the set of edges of the graph G that express the precedence relations, constraints (4.2.3) and (4.2.4) ensure that all of the precedence relations are followed correctly. It also considered the above-described edge case for the task T_{n+1}. Note that with the time-indexed representation and the added task T_{n+1}, the constraints for precedence and the constraint for all tasks to be scheduled combined also enforce that all of the tasks fit their whole processing time into the scheduling horizon h.

The constraint (4.2.5) ensures that the resource capacity is not exceeded for any resource in any interval.

The constraint (4.2.6) ensures that all of the energy consumption-intensive tasks are scheduled only from the interval I_{early} to interval I_{late}, thus respecting the turn-on and turn-off time of the resource R_0 and the rule that R_0 must be undergoing the (off, off) transition in the first and last interval.

The constraint (4.2.7) ensures that the resource R_0 is in all intervals either processing a task or undergoing an optimal switching between states that starts at the latest at the beginning of the interval and ends earliest at the beginning of the next interval. The constraint, at the same time, forbids overlaps between the job processing and the switching between states.

To address the work shifts, given in vector $\mathbf{w}$, we add the constraint (4.2.8), which ensures that each task fits its whole processing time into the work shifts of all resources the task requires.

4.3 Rescheduling model

As it will be detailed further in Chapter 5, the experiments revealed that the ILP model is more suitable for rescheduling. This decision is based on its superior ability to solve instances to optimality within a reasonable time limit, given the targeted instance sizes for rescheduling.

The solution space of a rescheduling problem is a subset of the solution space for a scheduling problem limited by additional constraints for the task fixations, given by I_{fix}, which represent the already planned or even already executed part of the schedule that cannot be changed, and for the allowed difference from the previous schedule, given as $\mathbf{s}$. Therefore, we can, without loss of generality, work with the ILP model for scheduling, described in Section 4.2, and extend it by additional constraints. As for the objective, consider the function $\mathbf{c}^*$ and the vector $\mathbf{c}_i^{job} \quad \forall T_i \in T$ to be computed using the new vector of prices $\mathbf{c}'$.

Firstly, we will address the fixation of the schedule up until the interval I_{fix} with the following constraints:

$$x_{j,s_j} = 1 \quad \forall j \in \{1, \ldots, n | s_j \leq I_{fix}\} \tag{4.3.1}$$

$$x_{j,i} = 0 \quad \forall j \in \{1, \ldots, n | s_j > I_{fix}\} \quad \forall I_i \leq I_{fix} \tag{4.3.2}$$

These constraints ensure that all of the tasks scheduled up to the interval I_{fix} will remain scheduled in the same intervals. Similarly, no task that was in the original schedule after I_{fix} can move into the fixed intervals.

The difference from the original schedule $\mathbf{s}$ is given by the difference function DF and limited by the parameter μ. We define two possible functions DF. We will call them task limitation and overall limitation.

In the task limitation difference function, we allow the absolute value of the difference between the start time given by $\mathbf{s}$ and the start time from the newly created solution, given by $\mathbf{s}'$, to be at most equal to μ for each task. For the overall limitation difference function, the sum of absolute values of the difference between the start time given by $\mathbf{s}$ and the start time of the new solution $\mathbf{s}'$ over all tasks must be at most equal to μ multiplied by the number of tasks n. Naturally, with the same value of μ, the set of task limitation solutions is a subset of a set of overall limitation solutions for a given problem since every valid solution created by task limitation is also a valid overall limitation solution.

- If we consider the task limitation, then the absolute value of the difference in start times will be lower or equal to μ for all tasks: $\forall i \in \{1, \ldots, n\} : |s_i - s'_i| \leq \mu$. Formally, the constraint is then defined as $DF_{tl}(\mathbf{s}, \mathbf{s}') = \max_{T_i \in T} |s_i - s'_i| \leq \mu$.

  We add this constraint to the model in the following way:

$$\sum_{I_k \in I} k \cdot x_{j,k} - s_j \leq \mu \quad \forall j \in \{1, \ldots, n\} \tag{4.3.3}$$

$$s_j - \sum_{I_k \in I} k \cdot x_{j,k} \leq \mu \quad \forall j \in \{1, \ldots, n\} \tag{4.3.4}$$

- If we consider the overall limitation, then the sum of absolute values of differences in start times over all tasks must be lower than μ, multiplied by the number of tasks n, formally, the constraint is then defined as $DF_{ol}(\mathbf{s}, \mathbf{s}') = \sum_{i=1}^{n} |s_i - s'_i| \leq \mu \cdot n$.

  We add this constraint to the model in the following way: First, we define a new integer variable $a_j \in \mathbb{N}_0$ for each task $T_j, j \in \{1, \ldots, n\}$, this variable will represent the absolute value of the difference in start times for each task. With $|s_i - s'_i| \leq \mu$ being a convex set, the absolute value can be represented as an intersection of two half-planes. We add the following constraints:

$$\sum_{I_k \in I} k \cdot x_{j,k} - s_j \leq a_j \quad \forall j \in \{1, \ldots, n\} \tag{4.3.5}$$

$$s_j - \sum_{I_k \in I} k \cdot x_{j,k} \leq a_j \quad \forall j \in \{1, \ldots, n\} \tag{4.3.6}$$

$$\sum_{j=1}^{n} a_j \leq \mu \cdot n \tag{4.3.7}$$

4.4 Objective balancing method

Since the values of TEC are multiplying the energy costs with the power consumption in each interval, its values are usually significantly higher than the length of the schedule, given by C_{max}. It comes as no surprise that without any balancing method, we can not use the parameter α to balance the objectives accurately. To address this problem, we came up with a method that aims to normalize the values of both objectives, using an estimation of their lower bounds. This way, in the solution, the objectives are scaled relatively to their lower bounds, allowing for more accurate balancing, using the parameter α. The lower bound estimations are obtained by solving a simplified problem for each part of the objective. The rest of this section will describe the methods used for finding the lower bounds.

To obtain the RCPSP objective lower bound lb_{RCPSP}, we solve the standard RCPSP problem without considering the energy consumption of R_0. To do so, we solve a constraint programming model using the IBM CP Optimizer; the solver is, thanks to its main focus on scheduling problems, capable of solving most of the small-sized RCPSP instances in a reasonable time, however, for this use-case, even a suboptimal solution is sufficient [34]. The model goes as follows:

Variables: to represent each task $T_i \in T$ we add a new interval variable x_i with fixed length of p_i.

Objective:

$$\min \max_{i \in \{1,\dots,n\}} \text{EndOf}(x_i) \tag{4.4.1}$$

the solution must be feasible with respect to the following constraints:

$$\sum_{i=1}^{n} \text{Pulse}(x_i, r_j^i) \leq R_j \quad \forall R_j \in R \tag{4.4.2}$$

$$\text{EndBeforeStart}(x_i, x_j) \quad \forall (i,j) \in E \tag{4.4.3}$$

Where E is the set of edges of the precedence graph G.

To obtain the TEC objective lower bound lb_{TEC}, we solve a simplified version of single-machine scheduling to minimize the TEC problem, considering only the energy consumption-intensive tasks in the set J, without considering

the precedence relations. We will still consider the TOU energy pricing; however, for simplification, we will only consider the energy consumption of the resource when processing a task. We can further simplify the problem by ignoring the switching of the states of the machine, only taking into account that tasks cannot be scheduled outside the interval between I_{early} and I_{late}.

To obtain the total energy cost of task processing, we will use the vector $\mathbf{c}_j^{job}$ as it was defined in Section 4.1. The value $\mathbf{c}_{j,i}^{job}$ represents the total energy cost of task T_j, with its processing starting in the interval I_i. For the formulation of this problem a time-indexed ILP model was used, as it has shown better performance compared to the CP formulation. The model formulation goes as follows:

Variables: we create a binary variable $x_{j,i} \in \{0, 1\}$ for all $T_j \in J, I_i \in I$. The value 1 represents that the i-th task starts at the beginning of the j-th interval.

Objective: the model minimizes the following objective:

$$\sum_{T_j \in J} \sum_{i=1}^{h} x_{j,i} \cdot \mathbf{c}_{j,i}^{job} \tag{4.4.4}$$

with respect to the following constraints:

$$\sum_{I_i \in I} x_{j,i} = 1 \quad \forall T_j \in J \tag{4.4.5}$$

$$\begin{gathered} x_{j,i} = 0 \\ \forall T_j \in J \quad \forall i \in \{1, \ldots, 1 + TF(off, proc)\} \cup \\ \{h - TF(proc, off) - p_j + 1, \ldots, h\} \end{gathered} \tag{4.4.6}$$

$$\sum_{T_j \in J} \sum_{i'=max\{1, i-p_j+1\}}^{i} x_{j,i} \leq 1. \tag{4.4.7}$$

The constraint (4.4.5) ensures that all of the tasks are scheduled. The constraint (4.4.6) ensures that all of the tasks are scheduled only from the interval I_{early} to interval I_{late}, thus respecting the turn-on and turn-off time of the resource R_0 and the rule that R_0 must be undergoing the (off, off) transition in the first and last interval. The constraint (4.4.7) ensures that there is no overlap between tasks, meaning that, at most, one task can be scheduled at each interval.

To put the lower bound balancing into use, the objective of the whole problem is modified to the following:

$$\min \left\{ \alpha \cdot \frac{1}{lb_{TEC}} \cdot \sum_{I_i \in I} c_i \cdot PF(\Omega_i) + (1-\alpha) \cdot \frac{1}{lb_{RCPSP}} \cdot C_{max} \right\}.$$

With lb_{RCPSP} and lb_{TEC} as input parameters, this change in objective can also be simply made in all of the above-described CP and ILP models. In practice and in all the experiments described in Chapter 5, this version of the objective will be used in the models instead of the original formulations.

Chapter 5

Experiments

In this chapter, we will describe the experiments conducted to observe and provide a comparison in terms of both the performance and the quality of the solutions obtained using the CP and ILP models described in Chapter 4. First, we will focus on the scheduling problem, evaluating the performance of different formulations of the CP models; in the following section, we will study the CP and ILP models in terms of both quality of solution and scalability and compare them. Lastly, we will focus on the rescheduling problem and observe the models' behavior and the solutions' features based on the changes in input parameters. All of the experiments contain a description of the input data, results, and conclusions made based on the results. The experiments are based on the publicly available instances from PSPLIB and real data for the TOU energy prices in Czechia from OTE (independent market operator) unless specifically stated otherwise.

For all of the instances, we used the transition diagram of the energy consumption-intensive resource, shown in Figure 1.4. Note that since we are using the SPACES method for the optimal switching computation, the structure of the transition diagram is not relevant as long as it follows the rules from the problem statement. This is due to the fact that the models operate only with the precomputed costs for optimal switchings in the vector $\mathbf{c}^*$. As was already mentioned, since the SPACES algorithm is not the main focus of this thesis, and its runtime is negligible in comparison to the runtime of the exact methods proposed in this thesis, we do not count the runtime of SPACES into the measured runtime of the solvers. Additionally, consider that for the purpose of the following experiments, we are working with a hypothetical transition diagram in terms of real energy consumption, and therefore, even though the data for TOU prices are real and measured in cost

in EUR per unit, we do not refer to any specific unit for the total energy cost.

The experiments were run using Python 3.11 on a server with Intel(R) Xeon(R) Silver 4110 CPU and 160GB of available RAM. For solving the CP models, the IBM CP Optimizer was used in version 22.1.0. For solving the ILP models, the Gurobi solver was used in version 11.0.3. In all experiments, the objective balancing method, presented in Section 4.4, was employed.

5.1 CP models comparison

The goal of this experiment was to compare the performance of the two presented models for constraint programming. Namely, it is the model that uses optional variables with fixed start and end times to represent the optimal switchings, presented in Section 4.1.1, which we will refer to as *CP fixed* and the model presented in Section 4.1.2, that allows the variables that represent the optimal switching to move freely around the schedule, which will be referred to as *CP free*.

5.1.1 Preliminary experiment

For the initial experiment, 9 instances were generated using the instance generator, described in Section 3.2. The instances are based on three PSPLIB instances from the j30 dataset; from each of the PSPLIB instances, we generated 3 new instances; in each of them, the energy consumption-intensive tasks were placed in a different part of the schedule, namely either in the beginning, middle or at the end of the schedule. The percentage p of tasks that should be marked as energy consumption-intensive in the given part of the schedule was set to 50%. To estimate the length of the scheduling horizon, the instances were solved using a generic CP model for RCPSP, the resulting length of the schedule was rounded up to the closest multiply of 24, and the final length of the schedule was set as double of this value. As for the vector of prices $\mathbf{c}$, we used a TOU price profile in the length of one day, corresponding to 24 intervals, and periodically repeated it over the whole length of the scheduling horizon. The resulting instances were named based on the number of PSPLIB instance it comes from and the part of the schedule which contains the energy consumption-intensive tasks.

The parameters of the instances are shown in Table 5.1, showing the number

Instance	$\|R\|$	$\|T\|$	J tasks position	$\|J\|$	h
1_beg	5	32	beginning	4	96
1_mid	5	32	middle	2	96
1_end	5	32	end	5	96
2_beg	5	32	beginning	6	96
2_mid	5	32	middle	3	96
2_end	5	32	end	3	96
3_beg	5	32	beginning	4	144
3_mid	5	32	middle	6	144
3_end	5	32	end	3	144

Table 5.1: Parameters of instances from the initial experiment.

of resources in the set R, the number of tasks in the set T, the position of the energy consumption-intensive tasks from set J in the original schedule, their count, and the final length of the scheduling horizon h.

The results of the initial experiment are shown in Table 5.2. Note that the value of α is the weight for the TEC objective, and the weight for the C_{max} objective is $1-\alpha$. The time limit for the solver was set to 600 seconds. In the table, the better objective values are highlighted. The gap value shows the relative difference between the found objective value and the estimated lower bound given by the solver; it does not show the gap to the actual, unknown optimum. Consider that the objective was computed with the objective balancing method in use, which explains the range of the reported objective values. It can be seen that within the given time limit, the *CP fixed* model was able to solve some of the instances to optimality, compared to the *CP free* model that reached the time limit (TLR) on all instances. Interestingly, if we look at the objective values, the *CP free* model was, in all cases, capable of finding a lower or equal objective value compared to the *CP fixed* model, yet it fails to prove its optimality within the given time limit. With the exception of the instances of the *CP fixed* model solved to optimality, the gaps do not differ greatly and, in some cases, are even better for the *CP free* model.

Note how, for both of the solvers, the gaps are, in all instances, clearly lower when the value of α gives a bigger priority to the C_{max} objective. This corresponds to the fact that the CP solver is, in general, empirically more effective in solving the RCPSP part of the problem and does a better job reducing the solution space when a priority is put on this objective. In comparison, when priority is put on the TEC objective, the CP solver was outperformed by other exact approaches such as the ILP [6].

The data in Table 5.2 also suggest that with the growing size of the

Inst.	α	CP free			CP fixed		
		Time (s)	Gap	Objective	Time (s)	Gap	Objective
1_beg	0.25	TLR	1.82%	**1.1266**	53.52	0.01%	**1.1266**
	0.5	TLR	6.14%	**1.1931**	93.58	0.01%	**1.1931**
	0.75	TLR	11.29%	**1.2597**	TLR	11.29%	**1.2597**
1_mid	0.25	TLR	3.95%	**1.1683**	TLR	3.83%	**1.1683**
	0.5	TLR	11.68%	**1.3367**	TLR	11.68%	**1.3367**
	0.75	TLR	16.19%	**1.4766**	TLR	16.19%	**1.4766**
1_end	0.25	TLR	14.98%	**1.4155**	TLR	14.98%	**1.4155**
	0.5	TLR	25.19%	**1.6596**	TLR	25.19%	**1.6596**
	0.75	TLR	31.30%	**1.7919**	TLR	31.30%	**1.7919**
2_beg	0.25	TLR	3.00%	**1.1347**	163.77	0.01%	**1.1347**
	0.5	TLR	9.65%	**1.2081**	358.29	0.01%	**1.2081**
	0.75	TLR	15.23%	**1.2816**	TLR	15.23%	**1.2816**
2_mid	0.25	TLR	9.70%	**1.2514**	TLR	9.70%	**1.2514**
	0.5	TLR	16.55%	**1.4855**	TLR	17.51%	1.5027
	0.75	TLR	21.65%	**1.7114**	TLR	21.65%	**1.7114**
2_end	0.25	TLR	8.15%	**1.2156**	TLR	8.15%	**1.2156**
	0.5	TLR	19.37%	**1.4181**	TLR	19.37%	**1.4181**
	0.75	TLR	22.71%	**1.5324**	TLR	22.84%	1.535
3_beg	0.25	TLR	5.33%	**1.0907**	TLR	7.56%	**1.0907**
	0.5	TLR	9.84%	**1.1814**	TLR	11.65%	**1.1814**
	0.75	TLR	19.07%	**1.2721**	TLR	19.07%	**1.2721**
3_mid	0.25	TLR	7.40%	**1.1923**	TLR	9.02%	1.2135
	0.5	TLR	19.12%	**1.3734**	TLR	21.22%	1.41
	0.75	TLR	24.84%	**1.5124**	TLR	27.41%	1.5659
3_end	0.25	TLR	11.69%	**1.2832**	TLR	11.12%	**1.2832**
	0.5	TLR	20.07%	**1.5663**	TLR	20.07%	**1.5663**
	0.75	TLR	27.12%	**1.8454**	TLR	31.50%	1.9632

Table 5.2: Results of the initial comparison between *CP free* and *CP fixed* models with 600 seconds time limit.

instances, the *CP free* model starts to outperform the *CP fixed* model. To observe if the trends revealed in this experiment also hold for bigger instances, we designed a follow-up experiment.

5.1.2 Scalability with respect to number of tasks

In the follow-up experiment, we generated new instances that are larger and more varied in size than the instances from the preliminary experiment, presented in Section 5.1.1. The new instances were generated from four different sets of PSPLIB instances. Set A contained a single instance from

the j30 dataset, set B contained two instances from the j30 dataset, set C contained a single instance from the j60 dataset, and set D contained two instances, one from the j30 and one from the j60 dataset. The instances were picked from the datasets randomly. Similarly to the initial experiment, from each set, three new instances were created based on the part of the schedule in which the energy consumption-intensive tasks were placed. To add more tasks to the set J this time, 75% of the tasks in the given part of the schedule were marked as energy consumption-intensive. With the growing size of instances, the number of tasks in J grows accordingly. To better observe the scalability of the models with respect to the number of tasks, the length of the scheduling horizon h was fixed to 168 intervals (i.e., one week-long scheduling horizon) for all instances. The vector $\mathbf{c}$ was obtained using one consecutive week of prices from the OTE day-ahead market, which corresponds exactly to 168 intervals.

Table 5.3 summarizes the parameters of the instances, showing the number of resources in set R, the number of tasks in set T, the position of the energy consumption-intensive tasks from the set J in the original schedule, their count, and the length of the scheduling horizon h.

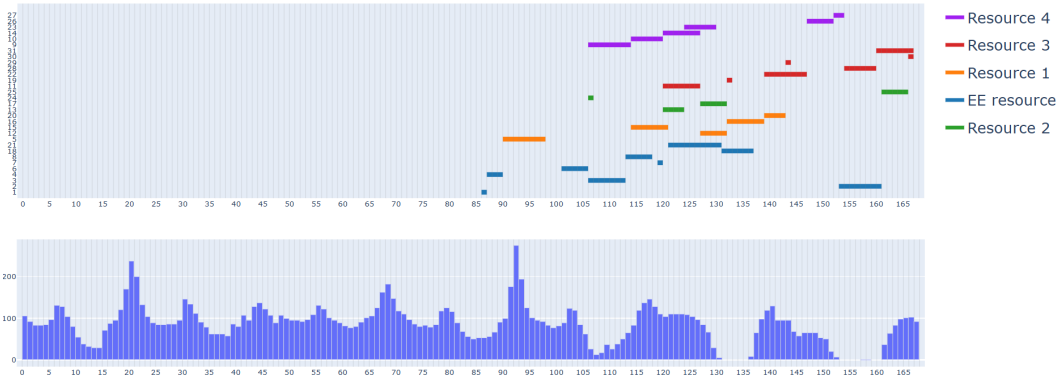
Name	$\|R\|$	$\|T\|$	J tasks position	$\|J\|$	h
1_beg	5	32	beginning	9	168
1_mid	5	32	middle	9	168
1_end	5	32	end	3	168
2_beg	5	64	beginning	15	168
2_mid	5	64	middle	15	168
2_end	5	64	end	6	168
3_beg	5	62	beginning	24	168
3_mid	5	62	middle	10	168
3_end	5	62	end	4	168
4_beg	5	94	beginning	22	168
4_mid	5	94	middle	19	168
4_end	5	94	end	9	168

Table 5.3: Parameters of instances from the follow-up experiment.

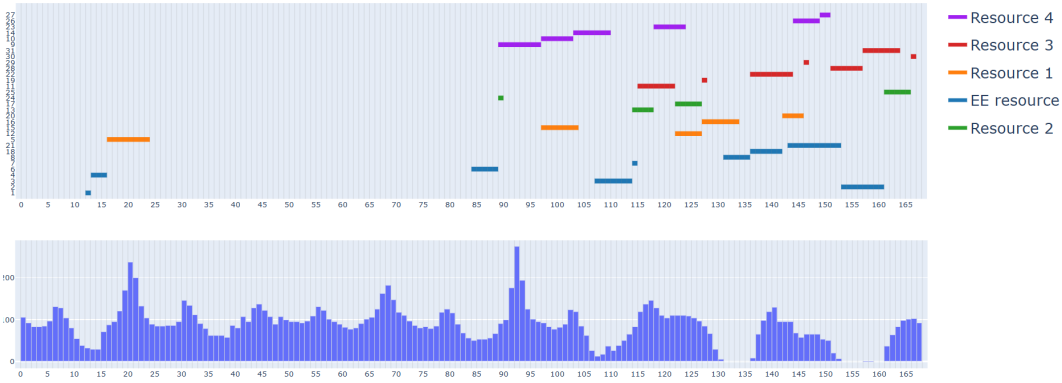
The results are shown in Table 5.4. The time limit was set to 600 seconds, and it was reached on all instances by both models. The objective balancing method was put into use. To better display the differences between the found solutions, the objective value is split into the TEC and C_{max} components. The highlighted values correspond to a better objective value. The gap value again shows the difference between the found objective and the best proven lower bound given by the solver.

In line with the results from the initial experiment, for the larger instances, the *CP free* model is capable of finding better or equal objectives, compared to the ones found by the *CP fixed* model with a single exception of the instance 4_end with α set to 0.75, where a better solution was found using the *CP fixed* model. Yet again, the *CP free* model is, in most cases, struggling to prove the optimality of the solutions, resulting in higher optimality gap values.

Even for the larger instances, it still holds for both solvers that the achieved gap values are considerably lower with lower α. That confirms that with both of the models, the solver performs better when the focus is shifted more towards the C_{max} objective.



(a) : Result for *CP fixed*, TEC = 11785, $C_{max} = 168$.



(b) : Result for *CP free*, TEC = 9432, $C_{max} = 168$.

Figure 5.1: Visualization of the results of *CP free* and *CP fixed* models on the instance 1_beg with $\alpha = 0.5$. Resource R_0 is shown in blue, R_1 in orange, R_2 in green, R_3 in red and R_4 in purple.

As an example of what the difference in solutions between solvers might look like in the final schedule, see Figure 5.1. The figure shows the resulting schedules for the instance 1_beg with α set to 0.5 obtained using the two compared models. The colors in the schedule denote each resource; below the schedules, there are the TOU price profiles. The resource R_0 is shown in blue color. Note that since the energy consumption-intensive tasks in the set J were selected from the beginning part of the initial RCPSP schedule, most of the tasks need to be shifted after the processing of the energy consumption-intensive tasks to satisfy the precedence constraints. It can be seen that in the case of the *CP free* model, the tasks scheduled on R_0 are placed exactly to the places with lower energy costs, while the solution obtained with the *CP fixed* model makes use of the lower energy prices yet fails to select them optimally in the given time limit.

Inst.	α	CP free			CP fixed		
		Gap	C_{max}	TEC	Gap	C_{max}	TEC
1_beg	0.25	30.00%	**82**	**18760**	18.75%	82	18778
	0.5	53.65%	**168**	**9432**	41.08%	168	11785
	0.75	83.75%	**168**	**10703**	58.55%	168	10975
1_mid	0.25	21.22%	**86**	**21114**	14.09%	84	22803
	0.5	56.78%	**168**	**12226**	43.14%	168	13143
	0.75	81.89%	**168**	**12226**	60.15%	168	13186
1_end	0.25	162.09%	**168**	**1713**	54.61%	168	2290
	0.5	231.92%	**168**	**1713**	63.18%	168	1746
	0.75	272.69%	**168**	**1713**	71.95%	**168**	**1713**
2_beg	0.25	20.37%	**93**	**32269**	15.34%	93	32269
	0.5	37.91%	**99**	**31178**	29.66%	93	32269
	0.75	60.05%	**168**	**23260**	48.88%	168	23944
2_mid	0.25	15.16%	**99**	**33640**	13.12%	101	35234
	0.5	44.79%	**99**	**33640**	39.11%	167	23773
	0.75	73.05%	**167**	**23487**	61.26%	167	23773
2_end	0.25	48.68%	**163**	**6363**	33.00%	166	6372
	0.5	86.86%	**164**	**6295**	54.70%	166	6372
	0.75	116.28%	**162**	**5759**	73.14%	166	6372
3_beg	0.25	22.42%	**146**	**51736**	20.36%	**146**	**51736**
	0.5	46.20%	**165**	**46236**	42.46%	165	46477
	0.75	68.18%	**167**	**46089**	62.58%	168	46357
3_mid	0.25	40.19%	**90**	**15596**	27.98%	90	16784
	0.5	76.26%	**167**	**8103**	50.65%	163	8890
	0.75	99.80%	**167**	**8057**	61.14%	163	8310
3_end	0.25	377.43%	**161**	**942**	73.25%	161	1026
	0.5	425.00%	**161**	**942**	82.13%	161	1041
	0.75	443.77%	**161**	**942**	85.25%	161	1026
4_beg	0.25	20.63%	**117**	**42209**	17.32%	**117**	**42209**
	0.5	44.61%	**120**	**41224**	38.73%	**120**	**41224**
	0.75	70.81%	**168**	**34027**	62.54%	168	34451
4_mid	0.25	19.27%	**97**	**37128**	15.04%	**97**	**37128**
	0.5	47.00%	**97**	**37128**	42.32%	165	26059
	0.75	76.89%	**166**	**25916**	66.31%	167	26182
4_end	0.25	32.32%	**60**	**14220**	32.71%	161	7499
	0.5	81.10%	**162**	**5496**	52.11%	166	6227
	0.75	114.32%	164	5904	68.84%	**162**	**5551**

Table 5.4: Results of the follow-up comparison between *CP free* and *CP fixed* models with 600 seconds time limit.

5.2 ILP and CP models comparison

In this section, we will present the experiments that were performed to compare the quality of solutions and scalability of the ILP model, presented in Section 4.2, with the CP quality of solutions and scalability of the CP models. Based on the comparison of CP models, described in Section 5.1, the *CP free* model was capable of finding a better solution in the given time limit. Thus, we will be comparing the ILP model to the results of the *CP free* model in this section.

5.2.1 Initial CP and ILP models comparison

In the initial CP and ILP models comparison, we compared the models on the same dataset that was used in the experiment, comparing the scalability of the CP models with respect to the number of tasks described in Section 5.1.2. This dataset allows us to compare the models on instances of various sizes and with the energy consumption-intensive tasks placed in three different parts of the schedule for each instance. The way the dataset was generated is described in detail in Section 5.1.2. Table 5.3 provides a summarization of the parameters of the instances, showing the number of resources in set R, the number of tasks in set T, the position of the energy consumption-intensive tasks from the set J in the schedule, their number, and the length of the scheduling horizon h.

The results of the experiments are shown in Table 5.5. The time limit was set to 600 seconds for both models. Both of the models used the objective balancing methods. The value of α represents the weight of the TEC objective. The objective was split into the TEC and C_{max} components; the values that correspond to the better objective are highlighted in the table. The value of the gap is given by the solvers as the relative difference between a lower bound confirmed by the solver and the found objective value.

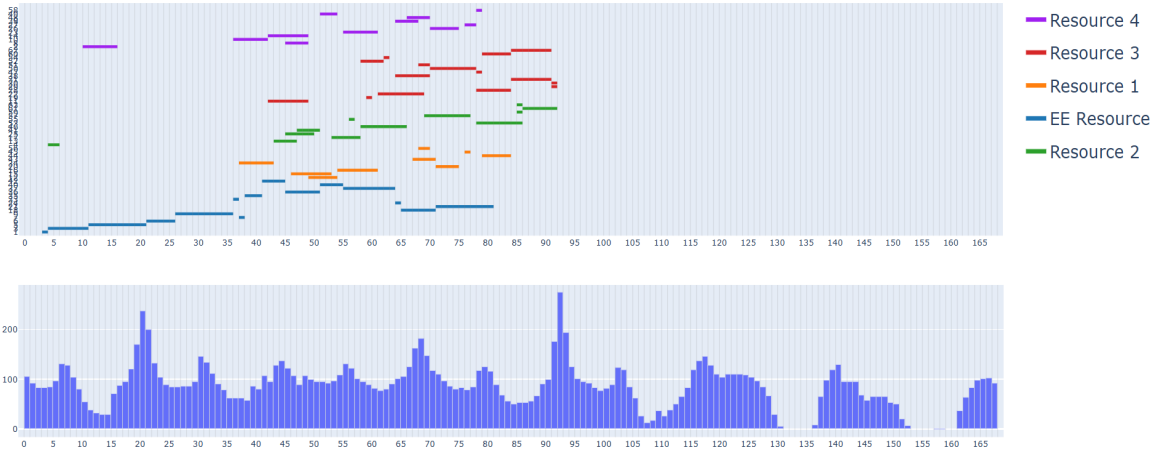
As it can be seen in Table 5.5, the ILP model clearly outperforms the *CP free* model on the smaller instances 1 to 3, being mostly capable of solving the instances in the given time limit and even for the ones where the solver timed out, it was capable of finding a better or equal solution in comparison with the *CP free* model. In some cases, the *CP free* model was able to find a similar solution but still ended up with a higher gap value, failing to prove its optimality. That, however, changes when it comes to the largest instances 4_beg, mid and end. In these instances, the *CP free* model is, despite the

higher gap values, in some cases, capable of finding a better solution than the ILP model. This suggests that the *CP free* model might be able to scale better with the growing size of the instances; however, with the size of this initial experiment, the results are inconclusive. Therefore, we decided to further compare the scalability of both models in a larger experiment, described in Section 5.3. Interestingly, the results further suggest that when the ILP model times out, the achieved gap is lower when the value of α is higher, and therefore gives more priority to the TEC objective, while it is the exact opposite for the CP solver.

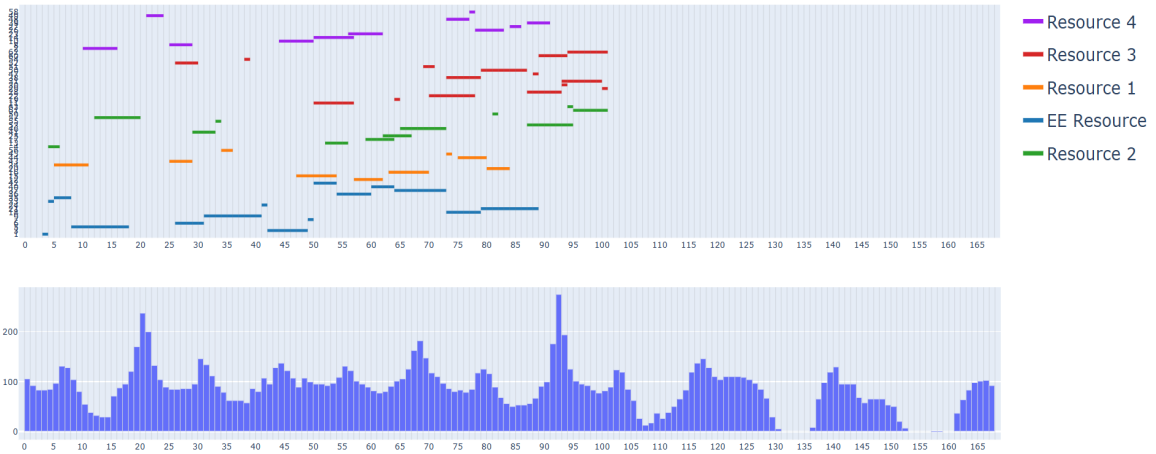
To illustrate how the differences in the value of α influence the resulting schedule, see Figure 5.2. The figure shows the results for the instance 2_beg, with α set to $0.25, 0.5$, and 0.75. Note how, with the increasing priority for the TEC objective, the makespan is prolonged, and the energy consumption-intensive tasks are positioned in such a way that it takes advantage of the lowered energy prices, and the peaks in the prices are avoided by either turning the machine off or by using the idle state.

Inst.	α	CP free				ILP			
		Time (s)	Gap	C_{max}	TEC	Time (s)	Gap	C_{max}	TEC
1_beg	0.25	TLR	30.00%	82	18760	31.54	0.00%	**82**	**18718**
	0.5	TLR	53.65%	**168**	**9432**	21.14	0.00%	**168**	**9432**
	0.75	TLR	83.75%	168	10703	7.33	0.00%	**168**	**9432**
1_mid	0.25	TLR	21.22%	**86**	**21114**	26.22	0.00%	**86**	**21114**
	0.5	TLR	56.78%	**168**	**12226**	31.14	0.00%	**168**	**12226**
	0.75	TLR	81.89%	**168**	**12226**	10.73	0.00%	**168**	**12226**
1_end	0.25	TLR	162.09%	**168**	**1713**	7.79	0.00%	**168**	**1713**
	0.5	TLR	231.92%	**168**	**1713**	7.88	0.00%	**168**	**1713**
	0.75	TLR	272.69%	**168**	**1713**	8.70	0.00%	**168**	**1713**
2_beg	0.25	TLR	20.37%	**93**	**32269**	384.01	0.00%	**93**	**32269**
	0.5	TLR	37.91%	99	31178	TLR	3.30%	**102**	**30659**
	0.75	TLR	60.05%	168	23260	76.90	0.00%	**168**	**22469**
2_mid	0.25	TLR	15.16%	**99**	**33640**	132.04	0.00%	**99**	**33640**
	0.5	TLR	44.79%	**99**	**33640**	169.03	0.00%	**99**	**33640**
	0.75	TLR	73.05%	167	23487	52.04	0.00%	**164**	**22899**
2_end	0.25	TLR	48.68%	163	6363	57.90	0.00%	**161**	**5776**
	0.5	TLR	86.86%	164	6295	20.54	0.00%	**161**	**5776**
	0.75	TLR	116.28%	**162**	**5759**	14.72	0.00%	**162**	**5759**
3_beg	0.25	TLR	22.42%	**146**	**51736**	TLR	11.09%	**146**	**51736**
	0.5	TLR	46.20%	165	46236	356.08	0.00%	**165**	**45923**
	0.75	TLR	68.37%	168	46357	510.40	0.00%	**167**	**45625**
3_mid	0.25	TLR	40.19%	90	15596	52.93	0.00%	**90**	**15449**
	0.5	TLR	76.26%	167	8103	23.33	0.00%	**166**	**7322**
	0.75	TLR	99.80%	167	8057	9.86	0.00%	**167**	**7300**
3_end	0.25	TLR	377.43%	**161**	**942**	9.33	0.00%	**161**	**942**
	0.5	TLR	425.00%	**161**	**942**	9.69	0.00%	**161**	**942**
	0.75	TLR	443.77%	**161**	**942**	9.27	0.00%	**161**	**942**
4_beg	0.25	TLR	20.63%	**117**	**42209**	TLR	19.59%	121	41024
	0.5	TLR	44.61%	**120**	**41224**	TLR	15.10%	122	40929
	0.75	TLR	70.81%	168	34027	TLR	3.86%	**168**	**33018**
4_mid	0.25	TLR	19.27%	**97**	**37128**	TLR	17.64%	**97**	**37128**
	0.5	TLR	47.00%	**97**	**37128**	TLR	16.74%	114	33773
	0.75	TLR	76.89%	166	25916	TLR	7.39%	**164**	**25248**
4_end	0.25	TLR	32.32%	**60**	**14220**	78.42	0.00%	**60**	**14220**
	0.5	TLR	81.10%	162	5496	22.78	0.00%	**162**	**5377**
	0.75	TLR	114.32%	164	5904	18.52	0.00%	**162**	**5377**

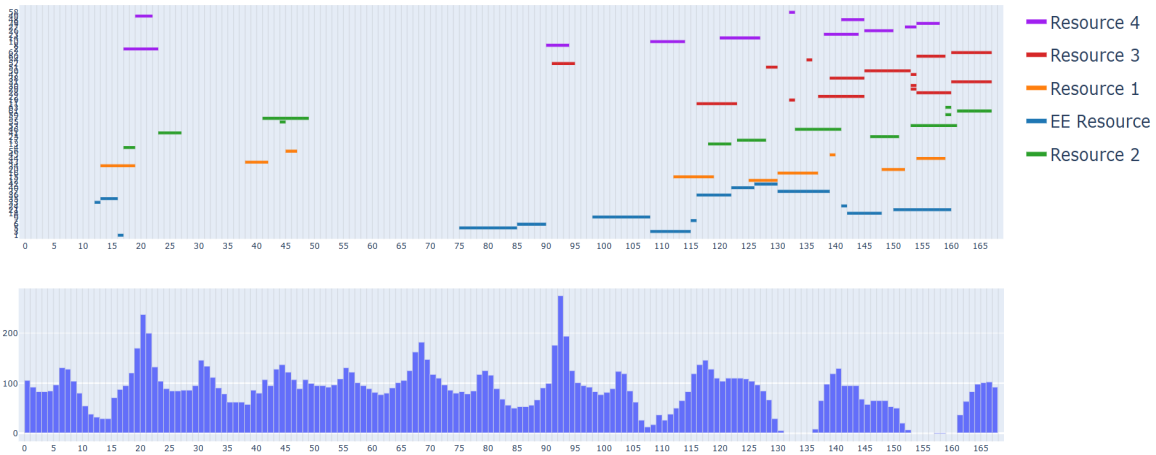
Table 5.5: Results of the initial comparison between CP free and ILP. The time limit was 600 seconds.



(a) : Result for $\alpha = 0.25$, TEC $= 32269$ and $C_{max} = 93$.



(b) : Result for $\alpha = 0.5$, TEC $= 30659$ and $C_{max} = 102$.



(c) : Result for $\alpha = 0.75$, TEC $= 22469$ and $C_{max} = 168$.

Figure 5.2: Visualization of the results of the ILP model with different values of α for the instance 2_beg. Resource R_0 is shown in blue, R_1 in orange, R_2 in green, R_3 in red and R_4 in purple.

5.3 Scalability comparison

The goal of this experiment was to put the ILP and *CP free* model into comparison on a large dataset containing instances of various sizes, allowing us to observe the scalability of the models. The created dataset is split into 6 sets of 10 instances of increasing size. Each of the sets, numbered one to six, contains instances created by merging the corresponding number of instances from the PSPLIB j30 dataset. The energy consumption-intensive tasks were picked by selecting 75% of the tasks in the middle section of the schedule. In this case, we omitted the instances created by picking the R_0 tasks from the beginning and end part of the schedule. This was done since, in the case of tasks picked from the middle, the solver still has to deal with both the tasks that come before and after the tasks scheduled on R_0. Additionally, the preliminary results suggested to be much more significantly affected by the number of tasks in set J than the position of the tasks itself. To set the scheduling horizon length h, the merged instance was solved using a CP model for RCPSP, setting the scheduling horizon length h as double of its minimum value. This ensures enough space for the tasks to move around while not making the scheduling horizon too long. The vector $\mathbf{c}$ was created for each instance by randomly picking one out of 40 available price profiles, each being created from 1 consecutive week (168 intervals) of TOU prices from the OTE day-ahead market. In cases where the horizon length h was longer, the price profile was periodically repeated.

The parameters of the dataset are summarized in Table 5.6. The table shows the number of resources in the set R, the number of tasks in the set T, the average number of tasks in the set J of energy consumption-intensive tasks, and the average length of the scheduling horizon h for each of the sets of instances.

Instance set	$\|R\|$	$\|T\|$	Average $\|J\|$	Average h
1	5.0	32.0	5.8	124.6
2	5.0	64.0	10.7	162.8
3	5.0	96.0	17.3	213.8
4	5.0	128.0	20.8	239.6
5	5.0	160.0	28.5	311.2
6	5.0	192.0	34.0	378.2

Table 5.6: Parameters of the instance sets for the scalability experiment.

The time limit for the solvers was set to 3600 seconds. The objective balancing method was put in use for both the ILP and the *CP free* model. Each instance was solved with α set to 0.25, 0.5 and 0.75. The Table 5.7 provides a summarization of the results. For each model, we observed its

ability to find some feasible solution for the given instance, the ability to find an optimal solution and prove its optimality, and finally, we compared the objectives of the found solutions. A solution is counted as the best-found solution if its objective value is better or equal to the objective value of the other solver; we do not take the gap into account in this case. Each number is given as the number of instances from the given set for which the solver satisfied the given condition. It can be seen that in accordance with the initial experiment, the ILP model clearly outperforms the *CP free* model when it comes to finding the optimal solution and confirming its optimality. However, the number of instances solved to optimality rapidly decreases with the growing size of the instances. Looking at the best-found solution column, it can be seen that the CP solver was able to find some of the optimal solutions yet failed to prove the optimality - for example, for $\alpha = 0.75$ and the instance set 1, 8 out of 10 optimal solutions were found using the *CP free* model, however, none of them was proven to be optimal by the CP solver in the given time limit, the ILP model was, in this case, able to solve and prove optimality for all 10 instances in the set.

On the other hand, it can be seen that starting from set 4, the ILP model, in some cases, fails to find any feasible solution for the instance in the given time limit. Specifically, in the case of set 6, it is the case for more than half of the instances for all values of α. In contrast, using the *CP free* model, a feasible solution is obtained for all instances in the dataset. However, the solution in a non-negligible number of cases is still worse than the solution obtained using the ILP model. As an example, we can take the results for set 6 with $\alpha = 0.75$. Using the ILP model, the solver was able to find a feasible solution for only 4 instances, whilst using the *CP free* model, a feasible solution was found for all 10 instances. However, 3 out of the 4 ILP solutions were better than the solutions obtained using the *CP free* model.

Figure 5.3 visualizes the average runtime of the ILP and CP solver, for instances in each set, for each value of α. It can be seen that the CP solver reached the time limit in all cases. As for the ILP model, the solver runtime grows in accordance with the growth in the size of the instances. The average runtime does not seem to be greatly affected by the value of α.

Instance set	α	Feasible solution		Best-found solution		Proved optimal solution	
		ILP	CP	ILP	CP	ILP	CP
1	0.25	10	10	10	6	10	0
	0.5	10	10	10	7	10	0
	0.75	10	10	10	8	10	0
2	0.25	10	10	9	10	9	0
	0.5	10	10	9	8	8	0
	0.75	10	10	10	7	9	0
3	0.25	10	10	8	10	5	0
	0.5	10	10	7	10	3	0
	0.75	10	10	6	8	3	0
4	0.25	9	10	6	10	2	0
	0.5	10	10	5	9	0	0
	0.75	10	10	6	5	0	0
5	0.25	6	10	2	10	1	0
	0.5	8	10	1	10	0	0
	0.75	9	10	4	7	0	0
6	0.25	4	10	0	10	0	0
	0.5	3	10	0	10	0	0
	0.75	4	10	3	7	0	0

Table 5.7: Results of the scalability tests for the instance sets. Each entry corresponds to the number of instances out of 10 in the set for which the given condition is satisfied.

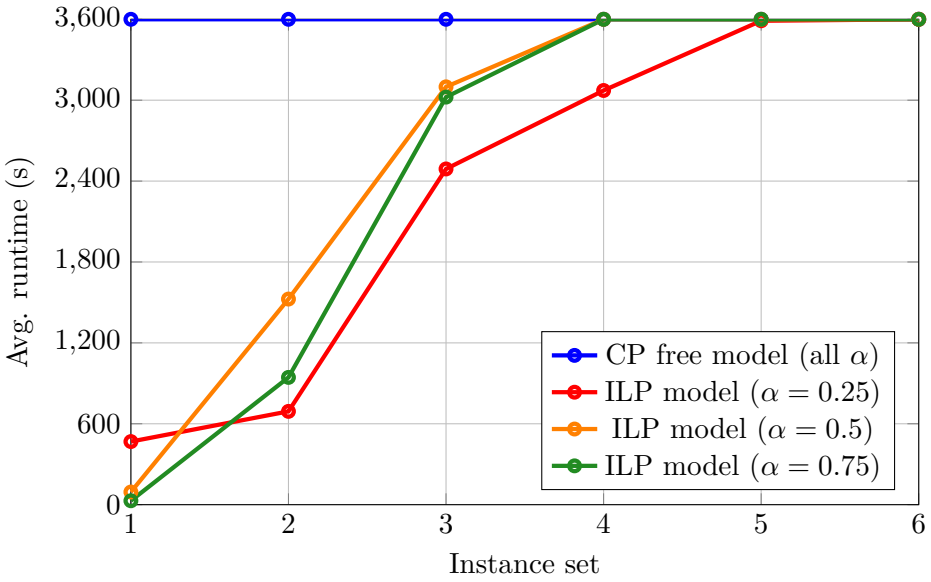


Figure 5.3: Runtimes of the CP and ILP models for the scalability instance sets.

As we already observed in the initial experiment, for the CP solver, the gap is lower for lower values of α, meaning that a priority is given towards the C_{max} objective. This is visualized in Figure 5.4, which shows the average gap of the CP solver for each set of instances, for each value of α. It can be seen that the average value of α differs greatly between the instances and does not seem to be clearly affected by the size of the instances. Yet, in all cases, the gap value is lower with a lower value of α. This holds not only for the average values but also for all of the instances in the dataset. The average values of the gap of the ILP solver for each set of instances for each value of α are shown in Figure 5.5. Only the instances where the solver found a feasible solution are taken into account. The value seems to grow in line with the growth in the size of the instances. Interestingly, as the instances get larger, we can observe that the average gap is lower for higher values of α, where the priority is given to the TEC objective. This is the exact opposite of what we observe for the *CP free* model.

In conclusion, the ILP model seems like a superior choice for smaller instances with under 100 tasks, given its ability to solve them optimally or with a low gap from the optimal solution. This seems to be a struggle with the *CP free* model, which has trouble lowering the gap, even for smaller instances. Furthermore, the ILP solver might be worth a try, even for larger instances, given that we want to prioritize the TEC objective. On the other hand, the *CP free* model is a good choice even for large instances, in case we want to find any feasible solution. Additionally, it shows good performance when more emphasis is given on the value of C_{max}.

Based on the results, we have decided that for the rescheduling problem, we will further continue to work solely with the ILP model since we will be aiming for optimal or near-optimal solutions in order to observe the potential savings with changes in energy prices.

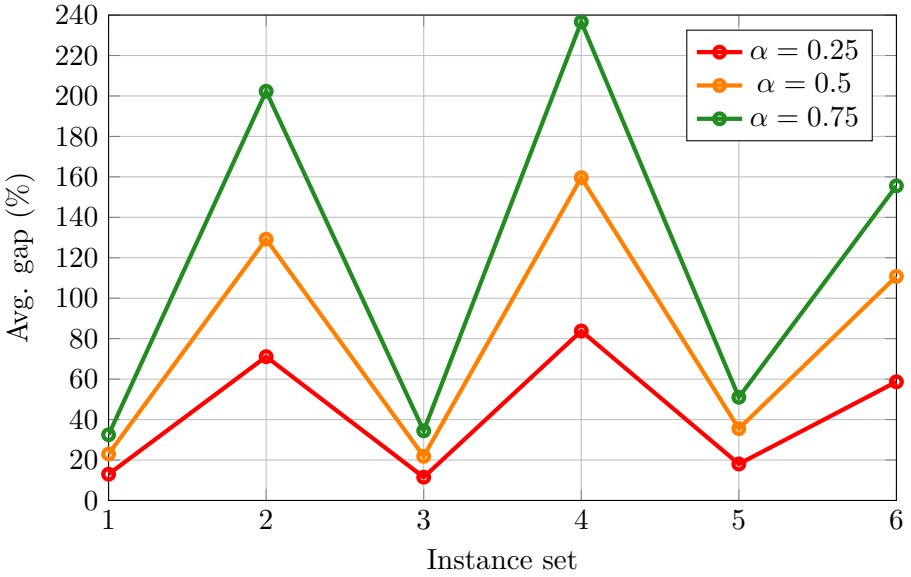


Figure 5.4: Average gap of the *CP free* model for the scalability instance sets.

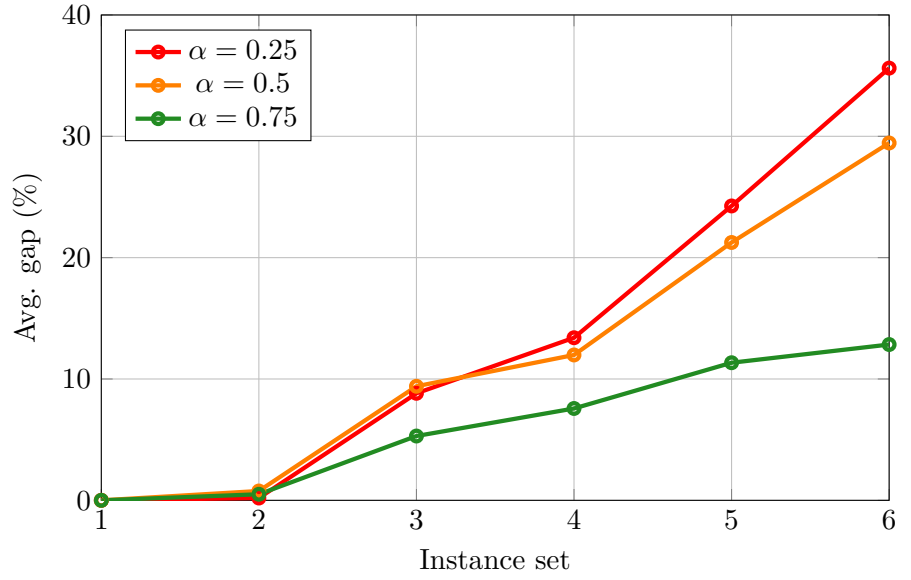


Figure 5.5: Average gap of the ILP model for the scalability instance sets.

5.4 Rescheduling experiments

In this section, we will describe the experiments performed on the ILP model for the rescheduling problem, introduced in Section 4.3. The scenario in the rescheduling experiments is the following: We will create a schedule for a horizon length h of 168 intervals, which corresponds to one week of 1-hour intervals. First, we create an initial schedule based on a prediction of prices for the given week. We will simulate the price prediction using the prices from the day ahead marked from OTE. Based on the sizes of the instances and the results from the scalability experiment, described in Section 5.3, we will use the ILP scheduling model for the creation of the initial schedule. Second, we focus on the rescheduling task, aiming to modify the initial schedule as a reaction to changes in energy prices. As for the new energy prices, we will use the interday market prices from OTE for the corresponding week, which report the actual average prices for which was the electricity traded in each given hour. The prices of the day-ahead market are published in advance and allow the buyers to buy the electricity for a clearly stated price. The interday prices correspond to the prices for which the electricity is traded on a real-time market based on the current supply and demand. Hence, we assume that this data poses a realistic simulation of the environment and is sufficient for the scope of the performed experiments. The characteristics of the market also allow for the creation of the look-ahead window, stating the number of intervals before the change in energy prices happens, as it is possible to secure a given volume of energy for a given time in advance using the day ahead market, before switching to the dynamic interday market.

In the following experiments, we will observe the features of the solutions obtained by using the ILP model based on the changes in the input parameters. Firstly, we will observe the potential for savings based on the varying accuracy of the electricity price predictions in different seasons. Secondly, we will observe how the potential savings are influenced based on the limitation of the changes in the schedule, given by the schedule difference function DF. Lastly, we will focus on the effects of the amount of fixed intervals from the previous schedule and the length of the look-ahead window.

5.4.1 Effects of the seasons of the year

In this experiment, we focused on how the potential savings created by modifying the initial schedule based on the new electricity prices change based on the season of the year, as the difference between the price prediction

and the real resulting price seems to vary. For this experiment, we created a new dataset. The dataset is based on 30 newly generated instances, each of them made by merging two randomly selected PSPLIB instances from the j30 dataset, using the described dataset generator, selecting 75% of the tasks from the middle part of the initial schedule as the energy consumption-intensive tasks. For each season - summer, winter, autumn, and spring we obtained 10 TOU price profiles of the length of one week from both the day-ahead and the corresponding interday markets. We randomly assigned 3 instances to each of the 10 profiles in each season, resulting in 120 final instances. Each instance contains 64 tasks; on average, 11 are in the set J of energy consumption-intensive tasks. Table 5.8 shows the average difference between the price prediction and the actual prices for each season in the created dataset. It can be seen that the summer season seems to be the least predictable in comparison with the other seasons. This may be caused by the unpredictable changes in the amount of solar energy.

Season	Avg. absolute difference (EUR/MWh)
summer	32.42
autumn	15.46
winter	17.84
spring	14.38

Table 5.8: Average absolute difference in the TOU prices between day-ahead and interday market in different seasons in the years 2022 and 2023.

Before performing the experiments, we adjusted the price profile in such a way that the prices were changed to the new price profile after 24 intervals while not fixing the schedule in any way; this gives us a look-ahead window of 24 intervals. We used the overall limitation for the possible changes in the schedule, with the value of μ set to 84, giving the solver quite a large flexibility in the number of changes to the schedule. The solver was run with a time limit of 1200 seconds for both the initial schedule and the rescheduling. To express the amount of possible savings, we first compute the blended objective and its separation back to the C_{max} and TEC objectives for the initial schedule when used with the new vector of prices. We express the savings as a percentage of these objectives that were saved when using the rescheduling model. If the value is negative, it means that the new objective value was higher than for the initial schedule. Note that this is quite common as the changes in reaction to the new energy prices are often made as a tradeoff between the objectives. We further ran the experiment both with and without taking the resource shifts into account in order to compare how the shifts can affect the potential savings.

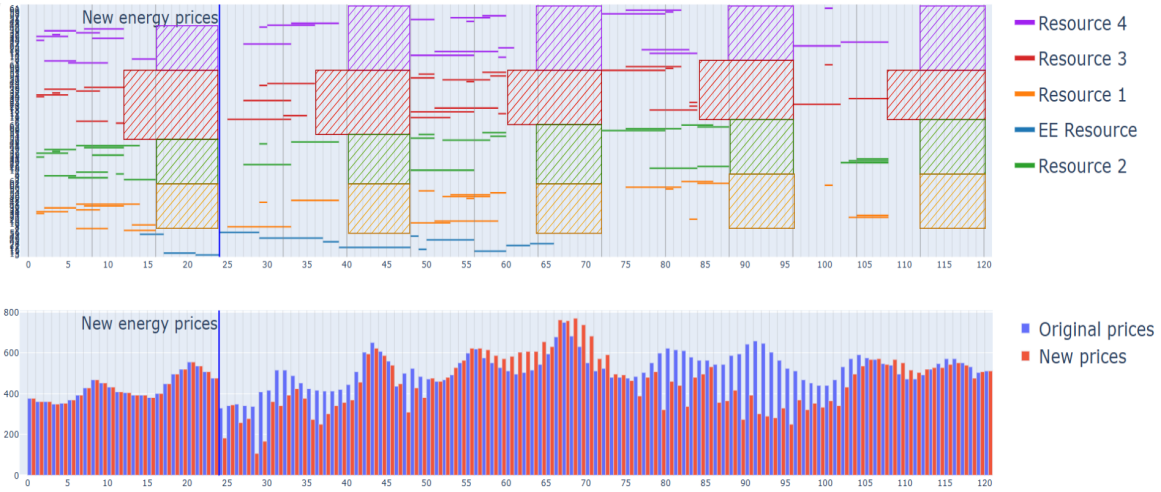
The results are shown in Table 5.9, showing the average savings for a given value of α and a given objective. It can be seen that the savings mostly appear as a tradeoff between the objectives, mostly achieving savings in terms of energy while prolonging the overall project makespan. The average savings on the overall objective are up to 2%, being naturally higher with larger values of α. This comes as no surprise, as the change in energy prices does not make a large difference when we focus mostly on the project makespan. Yet even in this case, some savings can be attained. In the case of the summer and spring datasets and $\alpha = 0.25$, it can be seen that the average overall objective even worsened; this is, however, caused by the model not being able to find the optimal solution within the given time limit on a few instances, resulting in a suboptimal solution for both the initial schedule and the rescheduling. Interestingly, there does not seem to be a great difference in the possible savings with regard to the work shifts. When we focus solely on the savings in energy costs, the highest average savings were attained on the summer dataset; however, in contrast to the initial expectations, the savings are quite comparable between the seasons, showing that the effect of the season is negligible.

α	Season	Savings with working shifts			Savings without working shifts		
		Objective	C_{max}	TEC	Objective	C_{max}	TEC
0.25	summer	-0.15%	-0.14%	-0.06%	0.09%	0.05%	0.13%
	spring	-0.14%	-1.36%	1.37%	0.27%	-2.39%	2.64%
	winter	0.45%	0.52%	0.30%	0.01%	0.00%	0.01%
	autumn	0.51%	0.44%	0.49%	0.09%	-0.04%	0.29%
0.5	summer	2.16%	-11.78%	8.22%	1.06%	-13.16%	6.48%
	spring	0.76%	3.12%	-2.39%	0.68%	-1.62%	0.84%
	winter	0.29%	0.47%	-0.05%	0.41%	-4.16%	2.33%
	autumn	1.39%	3.11%	-6.96%	1.62%	1.05%	-8.36%
0.75	summer	2.15%	-2.57%	3.72%	1.87%	-1.58%	3.17%
	spring	1.34%	-6.53%	3.06%	1.75%	-10.87%	3.47%
	winter	1.16%	-3.96%	1.42%	1.49%	-10.08%	4.29%
	autumn	1.75%	-7.33%	4.16%	2.01%	-9.55%	4.71%

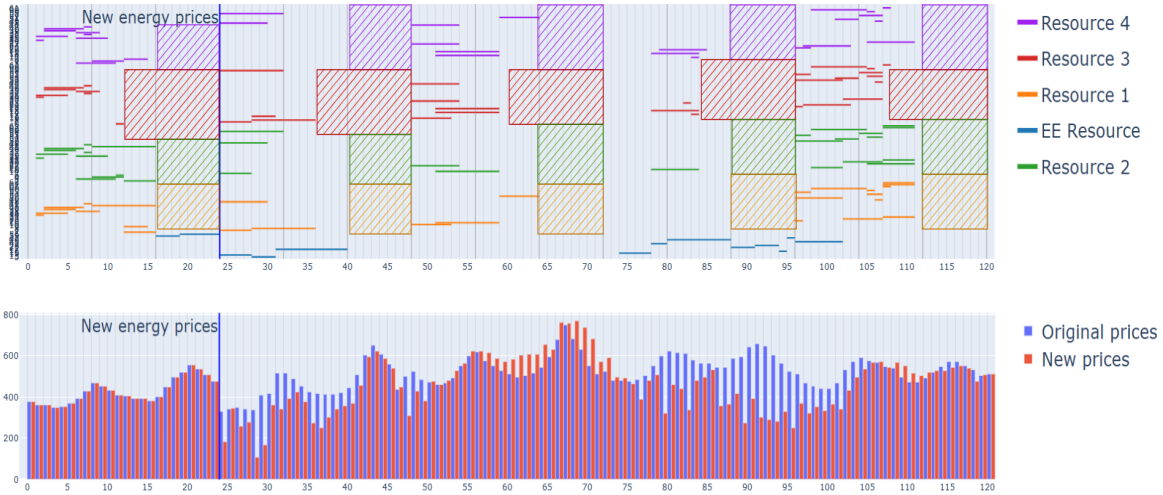
Table 5.9: Savings achieved by rescheduling based on the season, compared to the results obtained by the initial schedule when used with the new vector of prices. A negative value means that the new objective was higher than for the initial schedule.

For an example of a change in the schedule based on the new energy prices, see Figure 5.6. The tasks scheduled on the resource R_0 are shown in blue. The blue price profiles show the original energy prices $\mathbf{c}$ and the new energy prices $\mathbf{c}'$ are shown in red. You can notice how the rescheduling algorithm moves the tasks on resource R_0 away from the place with an increase in the

prices over the two places where the energy prices are considerably lower. This, however, comes at the price of a longer overall project makespan. Note also that the work shifts of the resources are being followed, $\mathbf{w} = (24, 16, 16, 12, 16)$.



(a) : Initial schedule for $\alpha = 0.75$, resulting in TEC $= 98222$ and $C_{max} = 109$.



(b) : Schedule after the rescheduling $\alpha = 0.75$, resulting in TEC $= 87721$ and $C_{max} = 112$.

Figure 5.6: Initial schedule and the schedule after the rescheduling. Resource R_0 is shown in blue, R_1 in orange, R_2 in green, R_3 in red and R_4 in purple. Original prices $\mathbf{c}$ are shown in blue, and new prices $\mathbf{c}'$ are shown in red. Change in prices is signified after 24 intervals, the cross out areas signify the intervals where the resource is not operating.

5.4.2 Effects of the limitations of the difference in the schedule

In Section 4.3, we introduced two possible definitions for the difference function DF. Namely, it is the task limitation, which states that the absolute value of the difference in start times between the original and the new schedule must be at most equal to μ for each task, and the overall limitation, which states that the sum of absolute values of the differences in start times between the original and the new schedule must be at most μ, multiplied by the number of tasks n. In this experiment, we will observe how the possible savings attained by rescheduling, as a reaction to the changes in energy prices, change based on the difference function and the value of the parameter μ.

For the purpose of this experiment, we created a new dataset of 20 instances. Each instance was created by merging two PSPLIB instances from the j30 dataset using the method, for instance generation, described in Section 3.2. The set J of energy consumption-intensive tasks was selected by taking 75% of tasks from the middle part of the original schedule. The length of the scheduling horizon h was set to 168 intervals. A randomly selected TOU price profile from OTE in the length of 168 intervals was assigned to each instance. The day-ahead market was used for the creation of the initial schedule, and the inter-day market was used as the new changed price prediction for rescheduling. The instances each contain 64 tasks and 5 resources, and on average, 11.4 tasks belong to the set J of energy consumption-intensive tasks.

Before running the experiments, we changed the price profile in such a way that the prices were changed to the new price profile after 24 intervals while not fixing the schedule in any way; this gives us a look-ahead window of 24 intervals for each instance. Each instance was run with both the overall and the task limitation, setting the value of μ to 8, 24, 48, and 168. The parameter α was set for each run to the values of 0.25, 0.5, and 0.75. The time limit was set to 1200 seconds for both the creation of the initial schedule and for the rescheduling. Similarly to the previous experiment, we first compute the objective of the original schedule when used with the new vector of prices and express the attained savings as a percentage of this objective that was saved by using the rescheduling model. Each instance was further solved using the model with shifts and without shifts (shifts on all resources are considered to be 24 hours long) to observe how the shifts affect the possible savings. This plays an important role, as when considering the work shifts, the lower values of μ of 8 or 24 intervals limit the possibility of moving tasks in between different shifts.

Table 5.10 shows the average savings for the given difference functions and values of μ and α. Similarly to the previous experiment, it can be seen that the savings are considerably higher for higher values of α, as the changes in the prices of energy do not play that significant role when the focus is shifted mainly towards the overall project makespan. This is even more significant when we do not consider the working shifts, as the schedule is, in general, shorter and allows for fewer movements of the tasks while keeping a similar makespan, thus resulting in 0% average savings in terms of the overall objective. Yet with higher values of α, the average savings are a bit higher when not considering the working shifts of the resources. With the same function DF and increasing value of μ, the optimal solution can only improve, yet we can see that for α set to 0.25 and 0.5 with the overall limitation, the average savings in some cases worsen with the increase of μ. This is caused by the fact that with an increasing value of μ, the solution space also grows in size, causing the model to time out, finding only suboptimal solutions for some instances. With the value of α set to 0.75, it can be clearly seen how the solutions improve with the growth in the value of μ. Since every feasible solution of the task limitation is also a feasible solution for the overall limitation but not vice versa, it is clear that the solution space of the task limitation is only a subset of the solution space of the overall limitation. This can also be seen in the results with higher values of α, as the overall limitation results in better savings for the same value of μ; for the task limitation, the value of μ needs to grow significantly in order to match the results of the model with overall limitation. The savings with both limitations match up with μ set to 168 since, in this case, all tasks can move around the whole schedule with both variants of DF, as the horizon length h is equal to 168 intervals.

To observe how the different functions DF affect the runtime of the solvers, consider Figure 5.7 showing the average runtime of the solver using the overall limitation as DF with respect to different values of μ and α and Figure 5.8, showing the average runtime of the solver using the task limitation as DF with respect to different values of μ and α. It can be seen that for the task limitation, the average solver runtime grows accordingly with the increase in the value of μ. This comes as no surprise as the increase in the value of μ corresponds clearly to the growing size of the solution space. Interestingly, this is not the case for the overall limitation, where the average runtime does not seem to vary greatly with respect to the value of μ, resulting in roughly the same average runtime for all values of μ as the task limitation does for the value of μ set to 168.

DF	μ	α	Savings with shifts			Savings without shifts		
			Objective	C_{max}	TEC	Objective	C_{max}	TEC
Overall lim.	8	0.25	0.34%	0.64%	-0.24%	0.00%	0.00%	0.01%
		0.5	0.43%	-0.46%	1.20%	0.43%	0.06%	0.99%
		0.75	1.62%	-1.49%	2.78%	1.92%	-1.17%	3.18%
Overall lim.	24	0.25	0.03%	-0.17%	0.45%	0.00%	0.00%	0.01%
		0.5	0.44%	0.47%	0.58%	0.43%	0.06%	0.99%
		0.75	1.98%	-4.14%	2.95%	2.23%	-4.96%	3.94%
Overall lim.	48	0.25	0.14%	0.12%	0.20%	0.00%	0.00%	0.01%
		0.5	0.41%	0.34%	0.62%	0.43%	0.06%	0.99%
		0.75	2.23%	-5.48%	3.57%	2.66%	-6.73%	4.75%
Overall lim.	168	0.25	0.14%	0.12%	0.19%	0.00%	0.00%	0.01%
		0.5	0.41%	0.34%	0.62%	0.43%	0.06%	0.99%
		0.75	2.31%	-4.56%	3.43%	2.67%	-6.67%	4.75%
Task lim.	8	0.25	0.01%	0.00%	0.02%	0.00%	0.00%	0.01%
		0.5	0.02%	0.00%	0.05%	0.02%	0.06%	-0.04%
		0.75	0.22%	0.00%	0.31%	0.17%	0.06%	0.23%
Task lim.	24	0.25	0.42%	0.76%	-0.24%	0.00%	0.00%	0.01%
		0.5	0.29%	-0.06%	0.51%	0.39%	0.15%	0.77%
		0.75	0.99%	-0.94%	1.63%	1.57%	-0.21%	2.30%
Task lim.	48	0.25	0.08%	0.17%	-0.07%	0.00%	0.00%	0.01%
		0.5	0.41%	0.34%	0.62%	0.43%	0.06%	0.99%
		0.75	1.52%	-2.16%	2.71%	2.14%	0.70%	2.78%
Task lim.	168	0.25	0.44%	0.70%	-0.07%	0.00%	0.00%	0.01%
		0.5	0.42%	-0.57%	1.25%	0.43%	0.06%	0.99%
		0.75	2.31%	-4.56%	3.43%	2.67%	-6.67%	4.75%

Table 5.10: Average savings achieved by rescheduling, based on the function DF and the parameter μ, given in number of intervals.

Effects of the limitation of the difference in the schedule with a simplified prediction

As it can be seen in the price profiles from Figure 5.6, the day-ahead market TOU prices from OTE serve as quite a good prediction of the interday market TOU prices. Even when the prices do not align perfectly, the prediction is still fairly good in terms of estimating the highs and the lows of the prices on most days. In this experiment, we want to observe how the average savings change when using a greatly simplified price prediction for the creation of the original schedule. We will use the same set of 20 instances, as described in Section 5.4.2; however, the initial price prediction will be switched from the day-ahead prices from OTE to a simple prediction, composed of periodically repeating 4 intervals of lower prices, followed by 8 intervals of higher prices. This simplification very roughly corresponds to the development of TOU

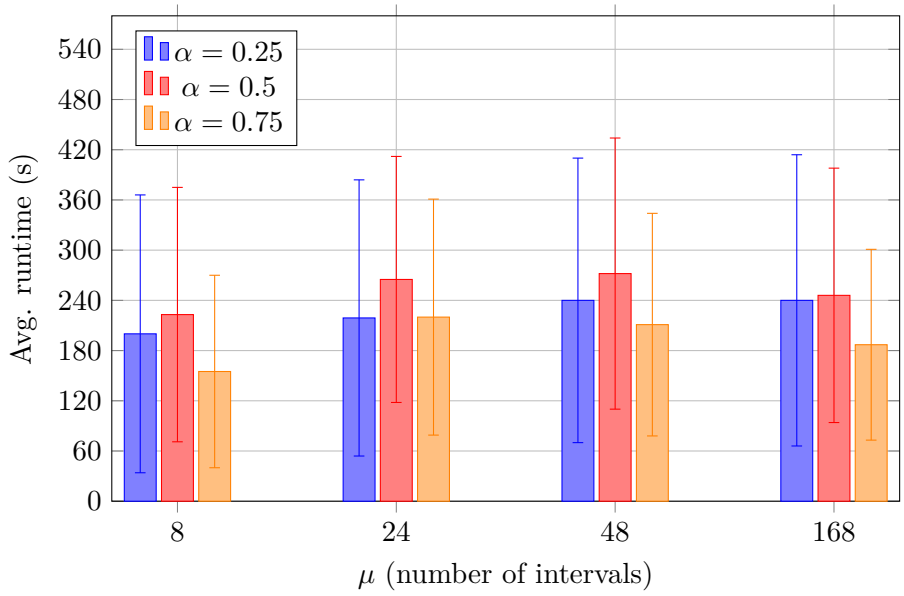


Figure 5.7: Average solver runtime, when using the overall limitation as DF.

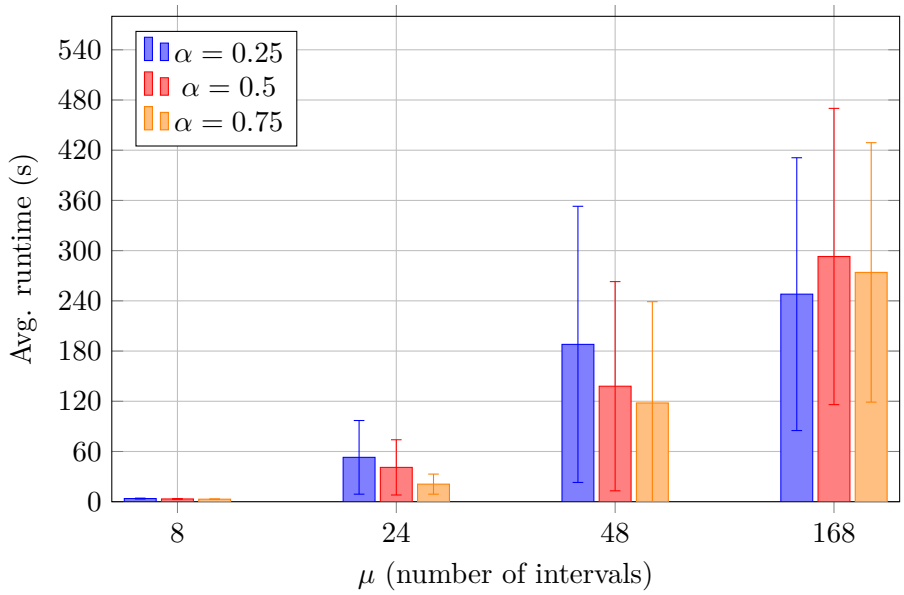


Figure 5.8: Average solver runtime, when using the task limitation as DF.

prices each day. Since this prediction is artificial and does not correspond to real data, we cannot give the solver any lookahead period, meaning that the new prices will start already from the first interval. A cutout from the TOU price profiles comparison can be seen in Figure 5.9 showing the simplified price prediction in blue and the new vector of prices in red. Note that since the new prices are used only for the creation of the initial schedule and the objective balancing method is put in use, it does not matter what values are used to represent the high and low energy prices periods. For better

visualization along the new vector of prices, in our case, the values were set to 50 for the lows and 250 for the high parts.

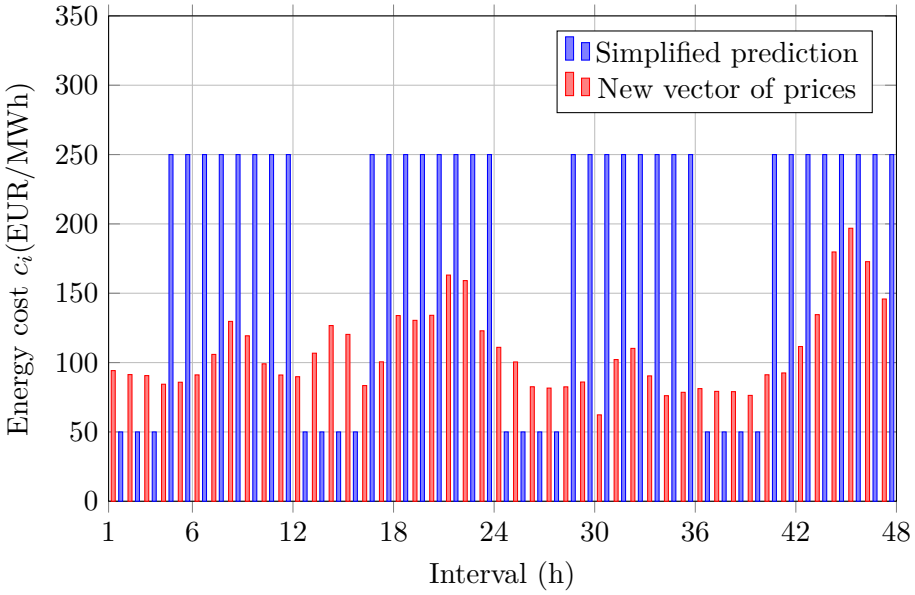


Figure 5.9: Comparison between the simplified price profile prediction, shown in blue, and the new vector of TOU prices from the OTE interday market, shown in red.

The solver was run with both the task limitation and the overall limitation, with values of μ set to 8, 24, 48, and 168 and with values of α set to 0.25, 0.5, and 0.75. The timeout was set to 1200 seconds. The work shifts of the resources are taken into account. The average savings are shown in Table 5.11. It can be again seen that the savings are considerably greater with higher values of α; in this case, the difference is even larger, as the simplified price prediction is periodical, causing the initial schedule to be concentrated in the early intervals, as there is no energy cost benefit in the later intervals, compared to the structure of the real data, with lower energy prices by the end of the schedule. The savings again mostly appear as savings on the TEC objective at the cost of prolonging the overall project makespan. It can be seen that for the overall limitation, the values do not differ greatly between the values of μ set to 48 and 168 intervals, meaning that the solver is able to effectively pick the right tasks to move whilst keeping others close to the initial position. In contrast, for the task limitation, there is a clear relation between the value of μ and the attained savings. The possible average savings of up to 22.55% on the TEC objective with α set to 0.75 show the power of rescheduling in case of a poor TOU price prediction; however, to attain these savings, it must be allowed to move the tasks around the schedule without a large limitation. This also highlights the importance of good and reliable price predictions in the case the new schedule must be close to the initially

given schedule, as otherwise, a possibility for great savings can be missed.

DF	μ	α	Savings		
			Objective	C_{max}	TEC
Overall limitation	8	0.25	1.28%	-0.11%	4.38%
		0.5	2.52%	-0.60%	4.84%
		0.75	7.26%	-17.47%	12.06%
Overall limitation	24	0.25	1.32%	-0.04%	4.38%
		0.5	3.20%	-9.00%	9.17%
		0.75	11.26%	-31.64%	19.68%
Overall limitation	48	0.25	1.25%	-0.17%	4.39%
		0.5	3.65%	-8.07%	9.18%
		0.75	12.63%	-38.42%	22.78%
Overall limitation	168	0.25	1.28%	-0.04%	4.25%
		0.5	3.60%	-9.13%	9.79%
		0.75	12.64%	-37.66%	22.55%
Task limitation	8	0.25	0.59%	0.00%	1.87%
		0.5	0.92%	0.00%	1.62%
		0.75	2.22%	0.33%	2.72%
Task limitation	24	0.25	1.20%	0.26%	3.35%
		0.5	2.05%	-0.41%	3.91%
		0.75	5.88%	-9.66%	9.41%
Task limitation	48	0.25	1.29%	0.00%	4.19%
		0.5	2.54%	0.33%	4.25%
		0.75	7.56%	-14.60%	12.26%
Task limitation	168	0.25	1.20%	-0.17%	4.25%
		0.5	3.57%	-9.13%	9.75%
		0.75	12.64%	-37.66%	22.55%

Table 5.11: Average savings achieved by rescheduling, based on the function DF and the parameter μ, given as a number of intervals, with a simplified prediction.

5.4.3 Effects of the look-ahead window

In the problem statement in Section 1.2, we defined the term look-ahead window as the number of intervals between the interval I_{fix} until which the schedule, created in rescheduling must remain similar to the initial schedule and the interval I_{update}, which is the first interval in which the new vector of prices is different from the original one. In short, the look-ahead window represents the time given to the solver to adapt to the upcoming change in energy prices. In this experiment, we will focus on how its size affects the possible savings that can be attained by the rescheduling algorithm.

To do so, we will work with the same dataset of 20 instances that was presented in the experiment in Section 5.4.2. Each instance is modified so that the new vector of prices is similar to the original vector for the first 24 intervals, thus setting the I_{update} to 24. We use the ILP model for rescheduling, presented in Section 4.3, using the overall limitation as the difference function DF and setting the value of μ to 168, meaning that the tasks can be moved freely around the schedule since the horizon length h is also equal to 168. We will observe the changes when setting the value of I_{fix} to 0, 12, and 24, creating look-ahead windows of 24, 12, and 0 intervals.

The results are presented in Table 5.12, showing the average savings for both when we do and when we do not consider the working shifts of the resources. The time limit was set to 1200 seconds. The savings are expressed as a percent of the objectives of the original schedule when put onto the new vector of energy prices that are saved using the rescheduling algorithm. It can be seen that with one single exception caused by the timeout of the solver while finding a suboptimal solution, the savings are getting lower with the shrinking length of the look-ahead window. This holds for both when we do and do not consider the working shifts of the resources. Interestingly, the savings for α of 0.25 when considering working shifts seem to lower considerably between the 24 and 12-hour look-ahead windows, whilst, for the other values of α, the largest change happens between the 12-hour and no-look-ahead windows. This is caused by the differences in the structure of the schedule based on the value of α. With a lower value of α, the schedule is concentrated in the early intervals, as the prioritized objective is the overall project makespan. Therefore, there is a larger difference between fixing none and 12 intervals. With higher values of α, this changes, as the schedule is more spread out, the biggest difference appears between fixation of 12 and all 24 intervals. When not considering the working shifts, the changes appear only for the α set to 0.75, meaning that for low values of α, there is no difference in savings caused by the look-ahead window length. In conclusion, the value of the lookahead window should be given carefully, based on the objective balancing parameter α.

Figure 5.10 shows the average runtime for the solvers, given the various lengths of the look-ahead window and values of α. It can be clearly seen that the runtime increases for all given values of α, with the growth in the size of the lookahead window. This is caused by the fixation of the part of the schedule, simplifying the task and reducing the search space for the solver.

Lookahead	α	Savings with shifts			Savings without shifts		
		Objective	C_{max}	TEC	Objective	C_{max}	TEC
24 hour	0.25	0.44%	0.70%	-0.07%	0.00%	0.00%	0.01%
	0.5	0.42%	-0.52%	1.21%	0.43%	0.06%	0.99%
	0.75	2.32%	-4.51%	3.43%	2.66%	-6.55%	4.71%
12 hour	0.25	0.14%	0.12%	0.20%	0.00%	0.00%	0.01%
	0.5	0.43%	-0.46%	1.19%	0.43%	0.06%	0.99%
	0.75	2.19%	-4.51%	3.22%	2.49%	-7.72%	4.94%
None	0.25	0.11%	0.12%	0.12%	0.00%	0.00%	0.01%
	0.5	0.36%	-0.15%	0.80%	0.43%	0.06%	0.99%
	0.75	1.90%	-4.20%	2.76%	1.98%	-6.89%	4.13%

Table 5.12: Average savings achieved by rescheduling, based on the size of the look-ahead window.

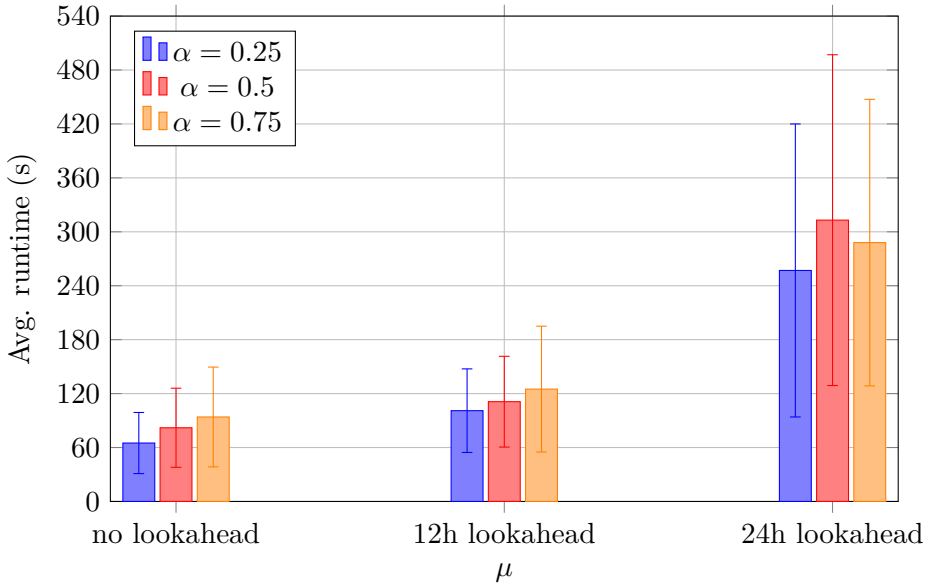


Figure 5.10: Average runtime of the solver with various look-ahead window sizes and various values of α.

Effects of the look-ahead window with simplified prediction

Similar to the experiment described in Section 5.4.2, we ran the experiment to observe the effects of the look-ahead window with a simplified TOU prices prediction used for the creation of the initial schedule. For the experiment, we used the same set of 20 instances, with the initial schedule being created for a vector of prices, consisting of periodically repeated 4 intervals of lower prices and 8 intervals of higher prices, instead of the real prices from OTE. See

Figure 5.9 for an example. Since the simplified prediction does not correspond to the real values of the energy costs, we will use the new energy prices for the whole scheduling horizon, including the first 24 intervals. With the I_{fix} being set to 0, 12, and 24, this creates negative look-ahead windows, where the schedule is fixed even after the change in energy prices as the value of I_{update} is, in this case, equal to zero. The solver was again run with the time limit of 1200 seconds.

The results are shown in Table 5.13. It can be seen that the attained savings are considerably higher, allowing for saving up to 22.62% on the total energy cost. Interestingly, even for the negative look-ahead windows, the trend still holds that for the higher values of α, the greatest difference can be observed between the 12 and 24 fixed intervals, while for the value of α set to 0.25, the most significant change appears between 0 and 12 fixed intervals. This shows that even with a simplified prediction, the look-ahead window must be considered with respect to the value of α, as too strict a look-ahead window can lead to a significant decrease in the attained savings.

Lookahead	α	Savings		
		Objective	C_{max}	TEC
None	0.25	1.79%	0.55%	4.48%
	0.5	3.50%	-9.32%	9.75%
	0.75	12.64%	-37.66%	22.55%
-12 hour	0.25	1.24%	-0.09%	4.30%
	0.5	3.38%	-9.30%	9.56%
	0.75	12.52%	-38.37%	22.62%
-24 hour	0.25	0.94%	-0.04%	3.10%
	0.5	2.55%	-8.94%	7.82%
	0.75	9.36%	-29.22%	16.82%

Table 5.13: Average savings achieved by rescheduling, based on the size of the look-ahead window with simplified prediction.

Chapter 6

Conclusion

The presented master thesis focused on the problem of energy-aware RCPSP with a single energy-significant stateful resource, operating under TOU pricing, minimizing both the TEC and the overall project makespan. The Chapter 1 provided an introduction to the problematics, providing a formalized problem statement and outlining the goals and contributions of the work. Chapter 2 provided an overview of the related work, identifying a gap in the literature concerning the studied topic. Chapter 3 was dedicated to the preliminaries, describing the optimal switching problem, the SPACES precomputation method for its solving, taken from literature [6], and the newly created method for generating instances for both the scheduling and rescheduling problem, based on the datasets from PSPLIB [30]. Chapter 4 presented the newly designed models for both CP and ILP, the ILP model for rescheduling, and the objective balancing method used to better control the balance of the objectives with the parameter α. Finally, Chapter 5 presents various experiments to compare the different CP models, the CP and the ILP model in both terms of scalability and the quality of solution, and the experiments to observe the possible savings that can be attained using rescheduling as a reaction to the changes in the energy prices, with regard to various factors. Thus, the thesis fulfills all the requirements stated in the assignment.

The results have shown that for larger instances, the *CP free* model outperforms the *CP fixed* model, yet the *CP fixed* model has shown to be more successful in proving the optimality of solutions for smaller instances. In the comparison of the ILP and CP models, it was shown that whilst the ILP model is capable of finding optimal solutions for smaller instances of up to 64 tasks, clearly outperforming the CP model, with the growth in the size of the instances, the CP model proves to be more capable of finding a feasible

solution within an hour of computation, even for larger instances of up to 192 tasks. Yet, the CP model fails to lower the gap and still, in some cases, provides a worse solution compared to the ILP model, which starts to struggle with finding any feasible solution with instances of around 128-160 tasks within an hour of computation. With the desired sizes of the instances for rescheduling and the goal of mostly solving them to optimality, it was decided to further work only with the ILP model. The experiments have shown that with the emphasis put on the TEC objective, rescheduling in reaction to the changes in energy prices can lower the objective by up to around 2% and save up to around 4% of the energy cost compared to the initial schedule on real data from OTE in various seasons of the year. The experiments have further shown how the savings are influenced by the various functions DF, limiting the differences between the new and the original schedule or how the savings are affected by varying size of the look-ahead window, providing an insight into how the accuracy of the energy price predictions and the flexibility of the schedule can limit the possible savings. Furthermore, experiments were performed to show the power of the rescheduling when considering a simplified, less accurate price prediction; in this case, in our instances, the rescheduling algorithm was able to attain savings of up to 12% on the objective, with up to 22.5% on the TEC, compared to the initial schedule, when the focus was shifted towards the TEC objective.

As for future work, the thesis can be built upon in various directions. Firstly, the thesis provided the formal problem statement together with multiple exact methods for solving both the scheduling and rescheduling problem; however, as it is common for the exact approaches, they are very limited in the size of instances that can be solved. This issue can be addressed by developing other heuristic approaches for solving the problem, which can be used to solve even larger instances. The solutions can be compared using instances based on the PSPLIB datasets generated using the proposed instance generation methods. Secondly, in the presented thesis, real data from the interday and day-ahead markets from OTE were used to simulate the energy price predictions and the changes in the vector of prices. The solutions could be further improved by incorporating the use of an actual energy prices prediction model. This would allow for better observations of the possible attainable savings by re-running the solver with each update in the price predictions, fixing the already contracted energy consumption, and allowing for changes in the schedule based on the employee's shifts and other constraints given by the movement of the material, etc. Lastly, future work can also lay in an extension of the problem statement. The problem can be extended to allow us to take the energy consumption of multiple resources into account, even with various state diagrams and TOU price profiles, which can be used to represent various energy sources. This would not be a complicated extension as we can use the SPACES method to precompute the optimal switching separately for each resource and encode the transitions into the models in a similar way. It would, however, likely increase the time complexity of the solution, as it

would lead to an increase in both the number of variables and the number of constraints. Another approach to the problem extension could be to allow for the maximum and minimum number of intervals a resource can stay in a state to be limited. This would allow the need for maintenance breaks or the need to shut a resource down to prevent overheating to be taken into account. Using a price profile of human labor in various shifts throughout the day would also allow us to take the cost of the machine operators into account, allowing us to use a transition diagram to model the need for breaks and maximum length of shits.

Appendix A

Bibliography

[1] MohammadMohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Production scheduling optimisation with machine state and time-dependent energy costs. *International Journal of Production Research*, 56:1–18, 12 2017.

[2] MohammadMohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Complexity analysis of energy-efficient single machine scheduling problems. *Operations Research Perspectives*, 6:100105, 2019.

[3] MohammadMohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Energy-cost-aware flow-shop scheduling systems with state-dependent energy consumptions. *IOP Conference Series: Earth and Environmental Science*, 463:012163, 04 2020.

[4] J. Bagherinejad and Z. R. Majd. Solving the mrcpsp/max with the objective of minimizing tardiness/earliness cost of activities with double genetic algorithms. *International Journal of Advanced Manufacturing Technology*, 2014.

[5] Ondřej Benedikt, Baran Alikoç, Přemysl Šůcha, Sergej Čelikovský, and Zdeněk Hanzálek. A polynomial-time scheduling approach to minimise idle energy consumption: An application to an industrial furnace. *Computers & Operations Research*, 128:105167, 2021.

[6] Ondřej Benedikt, Istvan Modos, and Zdeněk Hanzálek. Power of preprocessing: Production scheduling with variable energy pricing and power-saving states. *Constraints*, 25, 12 2019.

[7] Ondřej Benedikt, Přemysl Šůcha, and Zdeněk Hanzálek. On idle energy consumption minimization in production: Industrial example and math-

ematical model. *9th International Conference on Operations Research and Enterprise Systems*, 05 2020.

[8] Timo Berthold, Stefan Heinz, Marco E. Lübbecke, Rolf H. Möhring, and Jens Schulz. A constraint integer programming approach for resource-constrained project scheduling. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 2010.

[9] Jacek Blazewicz, Jan Karel Lenstra, and AHG Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics*, 5(1), 1983.

[10] Ripon K Chakrabortty, R Sarker, and D Essam. Resource constrained project scheduling: A branch and cut approach. In *Proceedings of the 45th international conference on computers and industrial engineering Metz*, volume 132, 2015.

[11] Ada Che, Shibohua Zhang, and Xueqi Wu. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production*, 156:688–697, 2017.

[12] Jian-Ya Ding, Shiji Song, Rui Zhang, Raymond Chiong, and Cheng Wu. Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. *IEEE Transactions on Automation Science and Engineering*, 13(2):1138–1154, 2016.

[13] Junwen Ding, Sven Schulz, Liji Shen, Udo Buscher, and Zhipeng Lü. Energy aware scheduling in flexible flow shops with hybrid particle swarm optimization. *Computers & Operations Research*, 125:105088, 2021.

[14] Olfa Dridi, Saoussen Krichen, and Adel Guitouni. Solving resource-constrained project scheduling problem by a genetic local search approach. In *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, 2013.

[15] Baigang Du, Tian Tan, Jun Guo, Yibing Li, and Shunsheng Guo. Energy-cost-aware resource-constrained project scheduling for complex product system with activity splitting and recombining. *Expert Systems with Applications*, 173, 2021.

[16] K. Fang, N. A. Uhan, F. Zhao, and J. W. Sutherland. Scheduling on a single machine under time-of-use electricity tariffs. *Annals of Operations Research*, 238, 2016.

[17] Kan Fang, Nelson Uhan, Fu Zhao, and John W. Sutherland. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems*, 30(4):234–240, 2011. Selected Papers of 39th North American Manufacturing Research Conference.

[18] Asefe Forghani, M.M. Lotfi, Mohammad Ranjbar, and Ahmad Sadegheih. A two-step scheduling and rescheduling framework for integrated production and usage-based maintenance planning under tou electricity tariffs: A case study of the tile industry. *Journal of Cleaner Production*, 416:137844, 2023.

[19] Mauro Gaggero, Massimo Paolucci, and Roberto Ronco. Exact and heuristic solution approaches for energy-efficient identical parallel machine scheduling with time-of-use costs. *European Journal of Operational Research*, 311(3):845–866, 2023.

[20] Christian Gahm, Florian Denz, Martin Dirr, and Axel Tuma. Energy-efficient scheduling in manufacturing companies: A review and research framework. *European Journal of Operational Research*, 248(3):744–757, 2016.

[21] Kaifeng Geng, Chunming Ye, Zhen hua Dai, and Li Liu. Bi-objective re-entrant hybrid flow shop scheduling considering energy consumption cost under time-of-use electricity tariffs. *Complexity*, 2020:1–17, 2020.

[22] Evgenii N. Goncharov. An improved genetic algorithm for the resource-constrained project scheduling problem. In *Advances in Optimization and Applications*, 2022.

[23] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. Elsevier, 1979.

[24] Farhad Habibi, Farnaz Barzinpour, and Seyed Sadjadi. Resource-constrained project scheduling problem: review of past and recent developments. *Journal of Project Management*, 01 2018.

[25] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 2010.

[26] Amir Hossein Hosseinian and Vahid Baradaran. An energy-efficient mathematical model for the resource-constrained project scheduling problem: an evolutionary algorithm. *Iranian Journal of Management Studies*, 12(1):91–119, 2019.

[27] Ying-Chao Hung and George Michailidis. Modeling and optimization of time-of-use electricity pricing systems. *IEEE Transactions on Smart Grid*, 10(4):4116–4127, 2018.

[28] A. Jalilvand, S. Khanmohammadi, and F. Shabaninia. Scheduling of sequence-dependant jobs on parallel multiprocessor systems using a branch and bound-based petri net. *Proceedings of the IEEE Symposium on Emerging Technologies, 2005.*, 2005.

[29] Konstantin Kogan and Eugene Khmelnitsky. Tracking demands in optimal control of managerial systems with continuously-divisible, doubly constrained resources. *Journal of Global Optimization*, 1998.

[30] Rainer Kolisch and Arno Sprecher. Psplib - a project scheduling problem library: Or software - orsep operations research software exchange program. *European Journal of Operational Research*, 1997.

[31] Min Kong, Jin Xu, Tinglong Zhang, Shaojun Lu, Chang Fang, and Nenad Mladenovic. Energy-efficient rescheduling with time-of-use energy cost: Application of variable neighborhood search algorithm. *Computers & Industrial Engineering*, 156:107286, 2021.

[32] Georgios Kopanos and Michael Georgiadis. Milp formulations for single- and multi-mode resource-constrained project scheduling problems. *Computers and Chemical Engineering*, 01 2011.

[33] Budi Kurniawan, Wei Song, Wenjun Weng, et al. Distributed-elite local search based on a genetic algorithm for bi-objective job-shop scheduling under time-of-use tariffs. *Evolutionary Intelligence*, 14(4):1581–1595, 2021.

[34] Philippe Laborie, Jerome Rogerie, Paul Shaw, and Petr Vilím. Ibm ilog cp optimizer for scheduling: 20+ years of scheduling with constraints at ibm/ilog. *Constraints*, 23, 03 2018.

[35] Yongping Liu, Lizhen Huang, Xiufeng Liu, Guomin Ji, Xu Cheng, and Erling Onstein. A late-mover genetic algorithm for resource-constrained project-scheduling problems. *Information Sciences*, 2023.

[36] Hongfang Lu, Xin Ma, Minda Ma, and Senlin Zhu. Energy price prediction using data-driven models: A decade review. *Computer Science Review*, 39:100356, 2021.

[37] Hamed Maghsoudlou, Behzad Afshar-Nadjafi, and Seyed Taghi Akhavan Niaki. A framework for preemptive multi-skilled project scheduling problem with time-of-use energy tariffs. *Energy Systems*, 12:431–458, 2021.

[38] Ho Minh HUng, Faicel Hnaien, and Frederic Dugardinr. Exact method to optimize the total electricity cost in two-machine permutation flow shop scheduling problem under time-of-use tariff. *Computers & Operations Research*, 144:105788, 03 2022.

[39] Gilles Mouzon, Mehmet Yildirim, and Janet Twomey. Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research*, 45:4247–4271, 09 2007.

[40] Mudarmeen Munlin. Solving resource-constrained project scheduling problem using metaheuristic algorithm. In *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, 2018.

[41] OTE. Day ahead market - 8.8.2024. https://www.ote-cr.cz/en/short-term-markets/electricity/day-ahead-market?date=2024-08-08.

[42] Myoung-Ju Park and Andy Ham. Energy-aware flexible job shop scheduling under time-of-use pricing. *International Journal of Production Economics*, 248:108507, 2022.

[43] Zhi Pei, Wan Mingzhong, Zhong-Zhong Jiang, Ziteng Wang, and Xu Dai. An approximation algorithm for unrelated parallel machine scheduling under tou electricity tariffs. *IEEE Transactions on Automation Science and Engineering*, PP:1–14, 05 2020.

[44] Robert Pellerin, Nathalie Perrier, and François Berthaut. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 2020.

[45] Maryam Pouramin, Hadi Mosadegh Abolfazl Mirzazadeh, Hamed Davari-Ardakani, and Edris Alajegerdi. Multi-mode resource-constrained project scheduling problem along with material ordering under time-of-use electricity tariffs and carbon taxes. *Annals of Operations Research*, 2024.

[46] Mehdi Pouramin, Ali Mirzazadeh, Hossein Davari-Ardakani, et al. Multi-mode resource-constrained project scheduling problem along with material ordering under time-of-use electricity tariffs and carbon taxes. *Annals of Operations Research*, 2024.

[47] Julian Rocholl, Lars Mönch, and John Fowler. Bi-criteria parallel batch machine scheduling to minimize total weighted tardiness and electricity cost. *Journal of Business Economics*, 90:1345–1381, 2020.

[48] Karam M. Sallam, Ripon K. Chakrabortty, and Michael J. Ryan. A two-stage multi-operator differential evolution algorithm for solving resource constrained project scheduling problems. *Future Generation Computer Systems*, 2020.

[49] Maximilian Selmair, Thorsten Claus, Frank Herrmann, Andreas Bley, and Marco Trost. Job shop scheduling with flexible energy prices. *European Conference on Modelling and Simulation*, 06 2016.

[50] Fadi Shrouf, Joaquin Ordieres-Meré, Alvaro García-Sánchez, and Miguel Ortega-Mier. Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *Journal of Cleaner Production*, 67:197–207, 2014.

[51] U.S. Energy Information Administration. International energy outlook, 2023. Accessed: 2024-12-29.

[52] Guohua Wan and Xiangtong Qi. Scheduling with variable time slot costs. *Naval Research Logistics (NRL)*, 2010.

[53] Jerome D. Wiest. A heuristic model for scheduling large projects with limited resources. *Management Science*, 13, 1967.

[54] Dongmin Yu Wuliang Peng and Fang Xie. Multi-mode resource-constrained project scheduling problem with multiple shifts and dynamic energy prices. *International Journal of Production Research*, 0(0):1–24, 2024.

[55] Huan yu Zheng and Ling Wang. Reduction of carbon emissions and project makespan by a pareto-based estimation of distribution algorithm. *International Journal of Production Economics*, 164:421–432, 2015.

[56] Guidong Zhu, Jonathan F. Bard, and Gang Yu. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, 2006.

Appendix B

Attached media structure

The attached media contains the instance generator and a folder for each performed experiment, which contains instances, experiment results in both JSON and CSV formats, and the Python source codes used to run the experiment, including the presented mathematical models. The attached media structure outline is shown below. Further details are provided in README.