

Algorithms for robust production scheduling with energy consumption limits

István Módos, Přemysl Šůcha and Zdeněk Hanzálek

DOI: https://doi.org/10.1016/j.cie.2017.08.011 Cite as: I. Módos, P. Šůcha, and Z. Hanzálek. Algorithms for robust production scheduling with energy consumption limits. *Computers & Industrial Engineering*, 112:391 – 408, 2017 Source code: https://github.com/CTU-IIG/RSECLP

 \odot 2017. This manuscript version is made available under the CC-BY-NC-ND 4.0 license, see <code>http://creativecommons.org/licenses/by-nc-nd/4.0/</code>

Algorithms for robust production scheduling with energy consumption limits

István Módos^{a,b}, Přemysl Šůcha^{a,b}, Zdeněk Hanzálek^{a,b}

^aDepartment of Control Engineering, Faculty of Electrical Engineering, Czech Technical University, Czech Republic ^bCzech Institute of Informatics, Robotics and Cybernetics, Czech Technical University, Czech Republic

Abstract

In this work, we consider a scheduling problem faced by production companies with large electricity consumption. Due to the contract with the electric utility, the production companies are obligated to comply with the total energy consumption limits in the specified time intervals (usually 15-minutes long); otherwise, the companies pay substantial penalty fees. Although it is possible to design production schedules that consider these limits as hard constraints, uncertainties occurring during the execution of the schedules are usually not taken into account. This may lead to situations in which the unexpected delays of the operations cause the violations of the energy consumption limits. Our goal is to design robust production schedules pro-actively guaranteeing that the energy consumption limits are not violated for the given set of uncertainty scenarios. We consider scheduling on one machine with release times of the operations and total tardiness as the objective function.

To tackle this problem, we first propose a pseudo-polynomial algorithm for finding the optimal robust schedule for the given permutation of the operations. This algorithm is then utilised in three different algorithms for finding the optimal permutation: two exact (Branch-and-Bound and logic-based Benders decomposition) and one heuristic algorithm (tabu search). All the algorithms were experimentally evaluated on random instances with different sizes of the uncertainty scenarios set. Using the tabu search algorithm, we are able to solve large instances within one minute.

Keywords: robust production scheduling, energy consumption limits, uncertainty scenarios, maximum power demand

1. Introduction

In the domain of scheduling energy-demanding production, it is no longer sufficient to consider only traditional aspects such as due dates, machine capacities, tardiness, schedule length, etc. To produce efficient schedules, the energy consumption of the operations has to be also considered [19] since significant financial savings could be achieved if the utilisation of the energy is optimised. Although integration of the energy-awareness into production scheduling is getting more and more attention [1, 18, 25], there is still a gap between industrial needs and academic research [23].

One of the practical problems addressed in this work is uncertainty during production in relation to the energy consumption limits. Based on the contract with the electric utility, the companies are obligated to comply with the energy consumption limits in every 15 minutes intervals; otherwise, large penalty fees have to be paid. However, due to the unpredictability of the operation's preparation time, it often happens that some of the operations are delayed and thus causing the violation of the contracted energy limits. To guarantee compliance with the energy limits, reactive policies are usually employed on the shop floor.

Email addresses: modosist@fel.cvut.cz (István Módos), suchap@fel.cvut.cz (Přemysl Šůcha), hanzalek@fel.cvut.cz (Zdeněk Hanzálek)

However, using only reactive policies may lead to sub-optimal schedules or long downtimes if the schedules are not devised in a robust way (e.g. high and low energy-consuming operations are not alternating).

Therefore, we focus on constructing pro-active production schedules for one machine that guarantee compliance with the contracted energy consumption limits if the operations' start times are delayed within a predetermined range; we call this a Robust Scheduling with Energy Consumptions Limits problem (RSECLP).

1.1. Motivation for Robust Scheduling with Energy Consumption Limits

The motivation for our work comes from the manufacturing and production companies with significant electricity consumption. Specifically, we were motivated by a glass tempering process during which glass panels are heated to 620°C in a furnace. In the considered scheduling problem, the furnace is a resource, and the heating of the glass panels represent the operations to be scheduled. Due to technological requirements, heating of the glass panels cannot be interrupted (i.e. preemption is not allowed). Although the production process also contains pre-processing and post-processing stages, we consider only scheduling of the heating stage because it is the most energy-demanding one. However, the pre-processing and the post-processing production stages are not completely ignored since they appear as release times and due dates of the operations, respectively. To ensure the smoothness of the production, it is reasonable to minimise the total tardiness.

According to the negotiated contract with the electric utility, the companies are obligated to keep their power demand below a contracted maximum power demand. Otherwise, the companies pay substantial penalty fees; in the Czech Republic, the penalty is regulated, and it is approximately $10\,000 \in$ per consumed MW over the maximum power demand [6]. The measurement of the demand is taken in every 15 minutes metering interval of a day, and it is measured as an average power demand during the corresponding metering interval. Since the consumed energy can be computed as a product of the average power demand and the length of the metering interval, an equivalent formulation is that in every metering interval the total energy consumption cannot exceed the maximum energy consumption limit. By considering a proper order of the energy-demanding operations or inserting short idle times, it is possible to design production schedules that do not violate these energy limits, e.g. see Fig. 1.



Figure 1: A baseline schedule of four operations $\{1, 2, 3, 4\}$ on one machine which satisfies the energy consumption limits in every metering interval $\{1, 2, 3\}$. The energy consumption limits are denoted by the dashed horizontal red lines.



(a) An example of a realised schedule in which operation 2 is delayed by 3 minutes from its baseline start time (see Fig. 1) thus increasing the total energy consumption in metering interval 2 above the energy limit.



(b) An application of a reactive policy on schedule from Fig. 2a. Operation 4 is forcibly delayed so that the energy consumption limit is not violated in metering interval 2.

Figure 2: Handling of operations' delays by a reactive policy.

However, in reality, unexpected events can cause delays of the operations' start times. In the glass production example, the responsible worker has to carefully put the glass panels on the furnace conveyor, mark the panels and set the furnace parameters before the glass panels are heated. This preparation process may take minutes, and due to various reasons (inexperienced seasonal workers, delays in the preceding production stages, etc.), it might happen that the heating of a glass panel starts later than expected. We call the delayed start time a *realised start time*, whereas the initial non-delayed start time is referred to as a *baseline start time*. The issue is that delaying an operation may cause an increase in the energy consumption in some metering interval above the energy limit if the energy demanding operations are started consecutively in the baseline schedule, e.g. see Fig. 2a. In such a situation, the company pays the penalty fee even though the baseline schedule (see Fig. 1) does not violate the energy limits. Therefore, to design robust baseline schedules, these uncertainties have to be considered so that the energy consumption limits are not violated and the penalty fees are avoided.

One possible approach to tackling these uncertainties is to employ *reactive scheduling policies*, i.e. when the total energy consumption approaches the energy limit, the remaining unfinished operations are delayed until the start of the next metering interval. For example, in Fig. 2b the start time of operation 4 is forcibly delayed by a worker responsible for monitoring the production process. However, relying only on the reactive policies may cause long downtimes in the production if the order of the operations is not chosen reasonably in a baseline schedule. A more viable approach is to combine the reactive policies with a *pro-active scheduling*, i.e. the baseline schedule is designed in such a way that the hazardous situations are avoided if the deviations of the operations are reasonably small. For example, if the order of the operations from Fig. 1 would be (3, 2, 1, 4), as illustrated in Fig. 3a, then even if operation 2 is delayed by 3 minutes the energy consumption limits are not violated (see Fig. 3b). However, longer production delays (e.g. furnace breakdown) are still handled by reactive policies or by a complete rescheduling.



(a) A robust baseline schedule of four operations $\{1, 2, 3, 4\}$ on one machine which satisfies the energy consumption limits in every metering interval $\{1, 2, 3\}$.



(b) An example of a realised schedule in which operation 2 is delayed by 3 minutes from its baseline start time (see Fig. 3a). Even though the total energy consumption increased in metering interval 2, the energy limit is not violated.

Figure 3: Handling of operations' delays by a pro-active scheduling.

1.2. Related Work

The related work to the RSECLP can be categorised into two main groups: scheduling with energy constraints and robust scheduling.

The problem of maximum power demand was studied in [7, 2], although the models presented in these works do not consider 15-minutes intervals but rather complying with the maximum power demand at every time instant. Another related problem to the maximum energy consumption limits is the problem of *electrical load tracking* [22, 12, 11, 10], where the objective is to minimise the absolute difference between the actual and pre-agreed energy consumption over all metering intervals w.r.t. penalty-free deviation range. Contrary to the RSECLP, both over-consumption and under-consumption of the energy are penalised in the load tracking problem.

Robust scheduling is a well-studied problem in the domain of *resource constrained project scheduling* [13, 16]. The robustness is obtained either by a robust resource allocation or inserting time buffers between activities. In the domain of the resource constrained project scheduling, the closest problem to the RSECLP is presented in [24]. The goal of this work is to find a partial-ordering of the activities so that if the activities are arbitrarily delayed (w.r.t. to the ordering), the total demand of the resources in every time instant is below the respective capacities. The difference from our problem is that we limit the integral of the operations' demands w.r.t. the intersection length of the operations with the metering intervals.

A particular interest for us is the modelling using *uncertainty scenarios* [5, 3], which are used when the probability distribution of uncertain events is either not known or is uniform. An uncertainty scenario is one realisation of uncertain events, e.g. a time occurrence of a machine breakdown. In general, the objective of the scheduling with uncertainty scenarios is to mitigate the worst-case execution over all uncertainty scenarios.

To the best of our knowledge, only few works deal with both robust scheduling and energy limits. One of such works is [21], where the goal is to reduce the peak power consumption of flow shop schedules under uncertain processing times of the operations. The method proposed by the authors inserts idle times into the schedule to reduce the expected peak power demand. The time points for idle times are computed by evaluating all the possible schedules originating from the set of possible scenarios and, therefore, the running time of the algorithm can increase significantly with the size of this set.

1.3. Contribution and Outline

Our work addresses an issue of integrating energy-awareness with robust production scheduling; we argue that this is necessary so that scheduling algorithms can be used in practice. As seen from the related literature section, no work considers this problem in its entirety.

The main contributions of this paper are:

1. A pseudo-polynomial algorithm that finds, for a given permutation of the operations, a *robust* and *optimal schedule* w.r.t. the total tardiness objective function. The algorithm can be incorporated into a wide variety of methods for solving the RSECLP that are based on searching the space of the permutations of the operations. To demonstrate the universality of the algorithm we employ it in two exact and one heuristic algorithms for solving the RSECLP (see Contributions 2 a 3).

The complexity of the algorithm is pseudo-polynomial since it is linear in the maximum deviation of the operations (see Section 3.3).

- 2. Two exact algorithms (see Section 4) for solving the RSECLP: (i) a Branch-and-Bound algorithm and (ii) a logic-based Benders decomposition algorithm with no-good cuts based on the optimal robust schedules.
- 3. A tabu search heuristic (see Section 5) for solving the RSECLP. The experiments (see Section 6) show that the heuristic can solve instances with 100 operations within a minute.

We partly studied this problem in the conference paper [20], where we introduced the RSECLP and presented (i) a decomposition algorithm with simple cuts and (ii) a procedure for deciding whether the given schedule is robust or not. Except for the master problem used in the logic-bases Benders decomposition algorithm (see Section 4.2.1), all the proposed contributions of this paper, as specified above, are novel.

The paper is organised as follows. Section 2 states the problem in a formal way. Section 3 describes how the optimal robust baseline schedule is constructed for the given permutation of the operations. The next Section 4 is concerned with the exact algorithms, and Section 5 describes the heuristic algorithm. In Section 6 the proposed algorithms are experimentally evaluated. Finally, the last section concludes the paper.

2. Problem Statement

The production scheduling problem outlined above is formally defined in this section. First, the scheduling problem considering only the energy consumption limits without robustness is described in Section 2.1. This scheduling problem is then extended in Section 2.2 with the deviations of the start times.

In the rest of the text, we use the notation $[a..b] = [a,b] \cap \mathbb{Z}$ to denote the integer interval for given $a, b \in \mathbb{Z}$. Moreover, the length of intersection of two intervals will be denoted as $\operatorname{lenint}([a_1, b_1], [a_2, b_2]) = \max(0, \min(b_1, b_2) - \max(a_1, a_2)).$

2.1. Non-robust Scheduling with Energy Consumption Limits Problem

Let $\mathcal{J} = [1 .. n]$ be a set of operations that have to be scheduled on a single machine without preemption. For each operation $j \in \mathcal{J}$ we define release time $r_j \in \mathbb{Z}_{\geq 0}$, processing time $p_j \in \mathbb{Z}_{>0}$ and due date $d_j \in \mathbb{Z}_{\geq 0}$. Moreover, for each operation $j \in \mathcal{J}$ we also define $P_j \in \mathbb{R}_{\geq 0}$ representing the power consumption of the machine when processing operation j, i.e. it is the constant rate at which the energy is consumed in every time instant. Therefore, the total consumed energy by each operation j is $p_j \cdot P_j$.

The operations have to be scheduled on a single machine within scheduling horizon $H \in \mathbb{Z}_{\geq 0}$, i.e. the operations must complete at most at time H. The scheduling horizon is divided into a set of metering intervals $\Omega = \begin{bmatrix} 1 & \frac{H}{D} \end{bmatrix}$ with equal length of $D \in \mathbb{Z}_{>0}$ (it is assumed that H is a multiple of D). For each metering interval $\omega \in \Omega$, an energy consumption limit is denoted as E_{ω}^{\max} , which represents the upper bound on the total energy consumption of the operations in metering interval ω . Moreover, let us denote a start of interval ω as $\tau_{\omega} = (\omega - 1) \cdot D$ and its end as $\tau_{\omega} + D$.

Baseline schedule $bs \in \mathbb{Z}_{\geq 0}^n$ is a vector, where each element bs_j represents the baseline start time of operation $j \in \mathcal{J}$ and the operations are not overlapping in bs. If operation j starts at time bs_j , then a tardiness of operation j is defined as $T_j = \max(0, (bs_j + p_j) - d_j)$. Moreover, the *intersection length* between metering interval $\omega \in \Omega$ and operation $j \in \mathcal{J}$ starting at time t is denoted as $p(\omega, j, t) = \operatorname{lenint}([\tau_{\omega}, \tau_{\omega} + D], [t, t + p_j])$. The goal of this scheduling problem is to find baseline schedule **bs** such that

min $\sum T_i$ (1)

$$\int_{\mathcal{I}} \mathcal{I} = \int_{\mathcal{I}} \mathcal{I$$

$$s.t. \quad r_j \ge 0s_j, \ j \in J \tag{2}$$

$$bs_j + p_j \le H, \ j \in \mathcal{J}$$
(3)

$$\sum_{j \in \mathcal{J}} p(\omega, j, bs_j) \cdot P_j \le E_{\omega}^{\max}, \ \omega \in \Omega$$
(4)

where Constraint (4) enforces that the energy consumption limit is not violated in any metering interval. We describe this problem in Graham's notation [9] as $1|r_i, E_{\omega}^{\max}| \sum T_i$.

2.2. Robust Scheduling with Energy Consumption Limits Problem (RSECLP)

Since unexpected events may occur during the execution of a baseline schedule, the actual start times of the operations may be delayed from their baseline start times; we call the carried out schedule *a realised schedule*. The goal of the robust scheduling is to guarantee that the energy consumption limits are not violated in any possible realised schedule.

To formally define the realised schedules, the notion scenarios and the maximum deviation has to be introduced. Let $\delta^{\max} \in \mathbb{Z}_{\geq 0}$ be a maximum deviation of any operation. Then $\Delta = [0 \dots \delta^{\max}]^n$ is a set of all scenarios such that scenario $\delta \in \Delta$ is a vector where each element δ_j represents the deviation of operation $j \in \mathcal{J}$. The maximum deviation is a user parameter which can be set according to the required range of the covered realised schedules.

Let $\pi : [1..n] \to \mathcal{J}$ be a bijective function representing a *permutation of the operations*. Operation on k-th *position* in permutation π is denoted as $\pi(k)$. We will say that π is the *corresponding* permutation of **bs** if the order of operations in **bs** is the same as in π .

From baseline schedule bs, its corresponding permutation π and arbitrary scenario $\delta \in \Delta$, one can derive a realised schedule using recursive vector function RS as

$$RS(\boldsymbol{bs},\delta)_{\pi(k)} = \begin{cases} bs_{\pi(1)} + \delta_{\pi(1)} & k = 1\\ \max(bs_{\pi(k)}, RS(\boldsymbol{bs},\delta)_{\pi(k-1)} + p_{\pi(k-1)}) + \delta_{\pi(k)} & \text{otherwise.} \end{cases}$$
(5)

Please notice that the definition of the realised schedules implies $\delta_{\pi(k)} \leq RS(\mathbf{bs}, \delta)_{\pi(k)} - bs_{\pi(k)}$; to make the distinction clear, the value of $RS(\mathbf{bs}, \delta)_{\pi(k)} - bs_{\pi(k)}$ is called a *delay*. To summarise, deviation δ_j of operation j is independent of the deviations of the other operations, whereas the delay is not.

To ensure that no operation completes outside of horizon H even if all operations are delayed, a maximum start time is defined as $bs^{\max} = H - (n \cdot \delta^{\max} + \max_{j \in \mathcal{J}} p_j)$ where value $n \cdot \delta^{\max}$ represents the maximum possible delay of any operation in any realised schedule from its baseline start time. Although bs^{\max} is a pessimistic bound w.r.t. horizon H, it is simple to compute and can be incorporated into algorithms requiring a solution-independent upper bound on the baseline start times. Moreover, since bs^{\max} increases with the scheduling horizon (which is a user-provided parameter), setting the horizon large enough will loosen bs^{\max} .

The goal of the RSECLP is to find baseline schedule \boldsymbol{bs} such that

$$\min \quad \sum_{j \in \mathcal{J}} T_j \tag{6}$$

s.t.
$$r_j \le bs_j, \ j \in \mathcal{J}$$
 (7)

$$bs_j \le bs^{\max}, \ j \in \mathcal{J}$$
 (8)

$$\sum_{j \in \mathcal{J}} p(\omega, j, RS(\boldsymbol{bs}, \delta)_j) \cdot P_j \le E_{\omega}^{\max}, \ \omega \in \Omega, \delta \in \Delta$$
(9)

where Constraint (9) enforces that energy consumption limit is not violated in any metering interval in any realised schedule (or, equivalently, in no scenario). Notice that Constraint (9) reduces to Constraint (4) when $\delta = (0, 0, ..., 0)$. A baseline schedule which does not violate Constraint (9) is called a robust baseline schedule. We describe the RSECLP in Graham's notation as $1|r_j, E_{\omega}^{\max}, \delta_j^{\max} = \delta^{\max}|\sum T_j$.

The RSECLP is illustrated on the following simple example with 5 operations $\mathcal{J} = \{1, 2, 3, 4, 5\}$. Let $D = 15, \Omega = [1..5], \delta^{\max} = 3$ and $E_{\omega}^{\max} = 1200$, where $\omega \in \Omega$. The parameters of the operations are provided in Tab. 1. One particular baseline schedule **bs** is shown in Tab. 2 and realised schedule $\mathbf{rs} = RS(\mathbf{bs}, \delta)$ for scenario $\delta = (3, 0, 3, 2, 0)$ is provided in Tab. 3. The visualisation of the baseline and the realised start times is in Fig. 4. Notice that $rs_4 - bs_4 = 5 > \delta^{\max}$ due to the delay of operation 3. Moreover, $rs_5 - bs_5 > 0$ even though $\delta_5 = 0$; this is due to the delays of the preceding operations. The total tardiness in the baseline schedule equals to 4 and the total energy consumption in the realised schedule in metering intervals 1 and 2 are $2 \cdot 50 + 2 \cdot 70 + 3 \cdot 150 + 0 \cdot 120 + 0 \cdot 30 = 690$ and $0 \cdot 50 + 0 \cdot 70 + 4 \cdot 150 + 4 \cdot 120 + 3 \cdot 30 = 1170$, respectively. Therefore, the energy consumption limits are not violated in **rs** (the energy consumption in every other metering interval is 0).



Figure 4: A visualisation of the example.

The \mathcal{NP} -hardness of the RSECLP was shown in [20], which follows from the underlying problem $1 || \sum T_j$. Moreover, even checking the robustness of a given baseline schedule is not an easy problem at first sight since a naïve algorithm would check all $(\delta^{\max} + 1)^n$ scenarios.

3. Algorithm for Finding Optimal Robust Schedule for Fixed Permutation

As it has been shown at the end of the previous section, the RSECLP is a difficult combinatorial problem. To solve it, we first focus on a simpler, related problem: is it possible to find an optimal robust schedule (i.e. satisfying Constraints (7)-(9)) for a fixed permutation π of the operations quickly? If the answer is "yes", then the RSECLP can be solved by a natural decomposition into two parts: (i) search the space of the permutations and (ii) for the given permutation, find the optimal robust schedule.

In this section, we introduce an algorithm with pseudo-polynomial complexity of $\mathcal{O}(n^3 \cdot \delta^{\max} + n \cdot |\Omega|)$ that creates an optimal robust schedule from the given permutation π of the operations. This algorithm is the cornerstone of the exact and the heuristic approaches described in Section 4 and Section 5, respectively. To better explain the main concepts behind the algorithm, the description is split into several subsections.

3.1. Latest start time and right-shift schedules

Before we explain the algorithm, the notions of *latest start times* and *right-shift schedules* have to be defined. Both definitions assume some fixed baseline schedule bs and its corresponding permutation π .

Latest start time schedule ls is defined using vector function LS as

$$LS(\boldsymbol{bs})_{\pi(k)} = RS(\boldsymbol{bs}, (\delta^{\max}, \delta^{\max}, \dots, \delta^{\max}))_{\pi(k)}, \ k \in [1 \dots n]$$

$$(10)$$

It represents the maximum possible starting time over all realised schedules for fixed baseline schedule bs.

Let $\pi(k')$ be some operation and $t \in [bs_{\pi(k')} \dots LS(bs)_{\pi(k')}]$ its arbitrary realised start time. Then right-shift schedule **rss** is defined using recursive vector function RSS as

$$RSS(\boldsymbol{bs}, k', t)_{\pi(k)} = \begin{cases} t & k = k' \\ \min(LS(\boldsymbol{bs})_{\pi(k)}, RSS(\boldsymbol{bs}, k', t)_{\pi(k+1)} - p_{\pi(k)}) & k \in [1 \dots k' - 1] \end{cases}$$
(11)

Informally, a right-shift schedule is obtained from **bs** by fixing the start time of operation $\pi(k')$ to t and shifting all the operations on positions k < k' to the right as much as possible while respecting the latest start times and the no-overlap constraint. Notice that a right-shift schedule defines starting times only for the operations on positions [1 .. k'].

An important property of the right-shift schedules is that they are also realised schedules, i.e. for each right-shift schedule rss there exists scenario $\delta \in \Delta$ whose corresponding realised schedule rs is the same as rss (see Lemma 2 in Appendix).

3.2. Earliest robust baseline schedule

The algorithm for finding the optimal robust schedule is based on the iterative computation of the *earliest* robust baseline start time for each operation in the order given by π . Robust baseline start time of $\pi(\overline{k})$ is a baseline start time such that there is no realised schedule of operations $\pi(1), \pi(2), \ldots, \pi(\overline{k})$ in which some energy consumption limit is violated. More formally, $bs_{\pi(\overline{k})}$ is robust relative to baseline start times $bs_{\pi(1)}, bs_{\pi(2)}, \ldots, bs_{\pi(\overline{k}-1)}$ if

$$\sum_{k=1}^{\overline{k}} p(\omega, \pi(k), RS(\boldsymbol{bs}, \delta)_{\pi(k)}) \cdot P_{\pi(k)} \le E_{\omega}^{\max}, \quad \omega \in \Omega, \delta \in \Delta$$
(12)

The earliest robust baseline start time is simply a robust baseline start time that is the smallest possible relative to the baseline start times of the preceding operations. It can be proven (see Theorem 2 in Appendix) that baseline schedule **bs** is robust and optimal w.r.t. the total tardiness objective function if every operation starts at its earliest robust time in **bs**. Since the earliest robust start time of $\pi(\overline{k})$ depends only on the baseline start times of $\pi(1), \pi(2), \ldots, \pi(\overline{k}-1)$, the earliest robust start times can be computed one-by-one according to the ascending order of the positions in the given permutation π , see Algorithm 1.

1 Function ComputeOptimalRobustBaselineSchedule(π) $bs \leftarrow (\infty, \infty, \dots, \infty)$ $\mathbf{2}$ $ls \leftarrow (\infty, \infty, \dots, \infty)$ 3 foreach $k = 1, \ldots, n$ do $\mathbf{4}$ /* Computation of the earliest robust baseline start time of $\pi(\overline{k})$. */ $\mathbf{5}$ if ComputeEarliestRobustStartTime $(\pi, \overline{k}, bs, ls) = \mathsf{INFEASIBLE}_\mathsf{PERMUTATION then}$ 6 **return** INFEASIBLE_PERMUTATION, ∅ 7 return OK, bs8

Algorithm 1: Optimal Robust Baseline Schedule for Fixed Permutation

Algorithm 1 terminates either with computing the earliest robust baseline schedule (indicated by return value OK) or concluding that permutation π is infeasible (indicated by return value

INFEASIBLE_PERMUTATION), i.e. for the given permutation π , it is not possible to find a robust baseline start time for some operation.

The algorithm for computing the earliest robust baseline start time of $\pi(\overline{k})$ is presented in the following subsections.

3.2.1. Naïve algorithm for computing the earliest robust baseline start time of $\pi(\overline{k})$

From now on, assume that **bs** is a baseline schedule where operations $\pi(1), \pi(2), \ldots, \pi(\overline{k}-1)$ start at their earliest robust baseline start time, and we want to find the earliest robust baseline start time for $\pi(\overline{k})$ (all other operations are not yet assigned to any start time). A naïve algorithm (see Algorithm 2) directly applies the definition of the earliest robust baseline start time: iterate over every possible baseline start time $bs_{\pi(\overline{k})} \in \left[\max(r_{\pi(\overline{k})}, bs_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)}) \dots bs^{\max}\right]$ in increasing order and select the earliest baseline start time such that Eq. (12) is not violated. However, such algorithm is inefficient since the number of realised schedules of **bs** is exponential in n.



Algorithm 2: Earliest Robust Baseline Start Time of $\pi(\bar{k})$ for Fixed Permutation: naïve version

3.2.2. Increasing the efficiency of the naïve algorithm: energy consumption dominance of the right-shift schedules

The first key observation for obtaining an efficient algorithm is the "energy consumption dominance" of the right-shift schedules, which is illustrated in Fig. 5. Let rs be some realised schedule of operations $\pi(1), \pi(2), \ldots, \pi(\bar{k}-1)$, then it can be proven that the energy consumption in the metering intervals intersected by $\pi(\bar{k}-1)$ in $rss = RSS(bs, \bar{k}-1, rs_{\pi(\bar{k}-1)})$ is not less than in rs, i.e.

$$p(\omega, \pi(\overline{k}-1), rs_{\pi(\overline{k}-1)}) > 0 \implies \sum_{k=1}^{\overline{k}-1} p(\omega, \pi(k), rs_{\pi(k)}) \cdot P_k \le \sum_{k=1}^{\overline{k}-1} p(\omega, \pi(k), rss_{\pi(k)}) \cdot P_k, \quad \forall \omega \in \Omega$$
(13)



Figure 5: The right-shift schedule maximises the energy consumption, $\delta^{\max} = 2$.

since if some energy Therefore, suffices to consider only the right-shift schedules it limit is violated in rsthen it will also be violated in *rss* (see Theorem 1 in Appendix). Lines 8-12 in Algorithm 2 can be replaced with the following pseudo-code

1 if $\overline{k} = 1$ then for each $(\delta_{\pi(\overline{k})},\omega)\in [0\,..\,\delta^{\max}]\times\Omega$ do $\mathbf{2}$ $rs_{\pi(\overline{k})} \leftarrow bs_{\pi(\overline{k})} + \delta_{\pi(\overline{k})}$ 3 $\begin{array}{l} \text{if } p(\omega, \pi(\overline{k}), rs_{\pi(\overline{k})}) \cdot P_{\pi(\overline{k})} > E_{\omega}^{\max} \text{ then} \\ | \quad energyLimitViolated \leftarrow \textbf{true} \end{array}$ 4 5 break 6 7 else $\begin{array}{l} \mathbf{foreach} \ (t, \delta_{\pi(\overline{k})}, \omega) \in \left[bs_{\pi(\overline{k}-1)} \dots LS(\mathbf{bs})_{\pi(\overline{k}-1)} \right] \times [0 \dots \delta^{\max}] \times \Omega \ \mathbf{do} \\ \left| \begin{array}{c} \mathbf{rs} \leftarrow RSS(\mathbf{bs}, \pi(\overline{k}-1), t) \\ \mathbf{rs}_{\pi(\overline{k})} \leftarrow \max(bs_{\pi(\overline{k})}, \mathbf{rs}_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)}) + \delta_{\pi(\overline{k})} \\ \mathbf{if} \ \sum_{k=1}^{\overline{k}} p(\omega, \pi(k), \mathbf{rs}_{\pi(k)}) \cdot P_{\pi(k)} > E_{\omega}^{\max} \ \mathbf{then} \\ \left| \begin{array}{c} energyLimitViolated \leftarrow \mathbf{true} \\ \mathbf{breads} \end{array} \right| \\ \end{array} \right|$ 8 9 10 11 12break 13

Although such algorithm does not have exponential complexity anymore, it is still not very efficient since it asymptotically depends on the length of the horizon.

3.2.3. Increasing the efficiency of the naïve algorithm: maximum possible intersection of the operations with the metering intervals

As was noted at the end of the previous subsection, the complexity of the naïve algorithm depends on the length of the horizon since whenever a realised schedule violating any energy consumption limit is found, the baseline start time of $\pi(\bar{k})$ is increased by 1, see line 16 in Algorithm 2. The question is whether it is possible to identify a range of non-robust baseline start times of $\pi(\bar{k})$ and, therefore, "jump" by more than 1 time unit on line 16 in Algorithm 2. This is possible by considering a maximum possible intersection of operation $\pi(\bar{k})$ with metering intervals.

Assume that rs is some realised schedule of operations $\pi(1), \pi(2), \ldots, \pi(\overline{k})$ such that the energy consumption limit is violated in some metering interval $\omega \in \Omega$. One of the following two cases occurs:

Case 1 $p(\omega, \pi(\overline{k} - 1), rs_{\pi(\overline{k} - 1)}) > 0$: since the baseline start times of $\pi(1), \pi(2), \ldots, \pi(\overline{k} - 1)$ are robust, $p(\omega, \pi(\overline{k}), rs_{\pi(\overline{k})}) > 0$ must hold. We may ask what is the maximum possible intersection of $\pi(\overline{k})$ in ω relative to realised start times rs of $\pi(1), \pi(2), \ldots, \pi(\overline{k} - 1)$ without violating the energy limit

$$maxPossibleIntersection_{\omega} = \left[\frac{E_{\omega}^{\max} - \sum_{k=1}^{\overline{k}-1} p(\omega, \pi(k), rs_{\pi(k)}) \cdot P_{\pi(k)}}{P_{\pi(\overline{k})}}\right]$$
(14)

It can be proven (see Lemma 5 in Appendix) that all baseline start times $\begin{bmatrix} \max(r_{\pi(\overline{k})}, bs_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)}) \dots \tau_{\omega} + D - \maxPossibleIntersection_{\omega} - 1 \end{bmatrix} \text{ of } \pi(\overline{k}) \text{ are not robust, i.e.}$ $\tau_{\omega} + D - \maxPossibleIntersection_{\omega} \leq bs_{\pi(\overline{k})} \text{ must hold, otherwise } E_{\omega}^{\max} \text{ is violated.}$

Case 2 $p(\omega, \pi(\overline{k}-1), rs_{\pi(\overline{k}-1)}) = 0$: in this case, $\pi(\overline{k})$ is the only operation having a non-zero intersection with ω in rs. Therefore, it holds that $p(\omega, \pi(\overline{k}), rs_{\pi(\overline{k})}) \cdot P_{\pi(\overline{k})} > E_{\omega}^{\max}$. We can compute the maximum possible intersection of $\pi(\overline{k})$ with ω

$$maxPossibleIntersection_{\omega} = \left\lfloor \frac{E_{\omega}^{\max}}{P_{\pi(\bar{k})}} \right\rfloor$$
(15)

which represents the maximum intersection length between ω and $\pi(\overline{k})$ without violating the energy consumption limit. Assuming that all baseline start times $\left[\max(r_{\pi(\overline{k})}, bs_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)}) \dots bs_{\pi(\overline{k})} - 1\right]$ of $\pi(\overline{k})$ are not robust, it is easy to see that $\tau_{\omega} + D - maxPossibleIntersection_{\omega} \leq bs_{\pi(\overline{k})}$ must hold to assure that the intersection of $\pi(\overline{k})$ with ω in any realised schedule is not larger than $maxPossibleIntersection_{\omega}$.

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

- Step 1 the earliest robust baseline start time relative to $\pi(1), \pi(2), \ldots, \pi(\overline{k}-1)$: for each right-shift schedule $RSS(bs, \pi(\overline{k}-1), t)$, where $t \in \left[bs_{\pi(\overline{k}-1)} \dots LS(bs)_{\pi(\overline{k}-1)}\right]$, find the earliest baseline start time of $\pi(\overline{k})$ using the maximum possible intersection which does not violate E_{ω}^{\max} , where $\omega \in \Omega$ is the last metering interval having a non-zero intersection with $\pi(\overline{k}-1)$. Notice, that due to Lemma 5 it is efficient to check t in decreasing order since if some energy consumption limit is violated for some $rs_{\pi(\overline{k})}$, then all baseline start times $bs_{\pi(\overline{k})} \leq rs_{\pi(\overline{k})}$ cannot be robust and the algorithm can continue directly with the second step.
- Step 2 the earliest robust baseline start time relative to only $\pi(\overline{k})$: After the first step, it is easy to see that for every baseline start time $\left[bs_{\pi(\overline{k})} \dots bs^{\max}\right]$ of $\pi(\overline{k})$ there is no realised schedule of operation $\pi(1), \pi(2), \dots, \pi(\overline{k})$ in which both $\pi(\overline{k}-1), \pi(\overline{k})$ have a non-zero intersection with some metering interval $\omega \in \Omega$ and E_{ω}^{\max} would be violated. However, the energy limits can still be violated in metering intervals in which only $\pi(\overline{k})$ have a non-zero intersection.

Consider the example from Fig. 6. For each $t \in \left[bs_{\pi(\overline{k})} \dots ls_{\pi(\overline{k})}\right]$, there exists a realised schedule rs of operations $\pi(1), \pi(2), \dots, \pi(\overline{k})$ such that $rs_{\pi(\overline{k})} = t$. Therefore, the maximum intersection of $\pi(\overline{k})$ with every metering interval $\omega \geq \left\lfloor \frac{bs_{\pi(\overline{k})}}{D} \right\rfloor$ is

$$maxIntersection_{\omega} = \min(p_{\pi(\bar{k})}, \operatorname{lenint}([\tau_{\omega}, \tau_{\omega} + D], [bs_{\pi(\bar{k})}, ls_{\pi(\bar{k})} + p_{\pi(\bar{k})}]))$$
(16)

Then, $bs_{\pi(\overline{k})}$ is not robust if there exists metering interval ω such that the maximum intersection is larger than the maximum possible intersection in ω , i.e.

$$maxPossibleIntersection_{\omega} = \left\lfloor \frac{E_{\omega}^{\max}}{P_{\pi(\bar{k})}} \right\rfloor < maxIntersection \tag{17}$$

If this is the case, then the earliest baseline start time that can be robust is $\tau_{\omega} + D - maxPossibleIntersection_{\omega}$.



Figure 6: The time interval intersected by operation $\pi(\overline{k})$.

The complete algorithm for computing the robust baseline start time that combines all the discussed ideas is shown in Algorithm 3.

3.3. Algorithm complexity of the algorithm for computing the robust baseline schedule

The complexity of Step 1 of Algorithm 3 is $\mathcal{O}(n^2 \cdot \delta^{\max})$ since the number of unique right-shift schedules of $\pi(\overline{k}-1)$ is bounded above by $n \cdot \delta^{\max}$ and the computing a right-shift schedule can be done in $\mathcal{O}(n)$. The complexity of Step 2 is $\mathcal{O}(|\Omega|)$. Since Algorithm 3 is repeated for each position in the permutation (see Algorithm 1), the complexity of computing the optimal robust schedule for the given permutation is $\mathcal{O}(n^3 \cdot \delta^{\max} + n \cdot |\Omega|)$; the pseudo-polynomiality of the algorithm arises due to term δ^{\max} . Notice that if the operations cannot violate the energy consumptions limits by themselves, i.e. $\forall \omega \in \Omega, \forall j \in \mathcal{J} : \min(p_j, D) \cdot P_j \leq E_{\omega}^{\max}$, then Step 2 is not necessary and the complexity is $\mathcal{O}(n^3 \cdot \delta^{\max})$.

1 Function ComputeEarliestRobustStartTime $(\pi, \overline{k}, bs, ls)$ $\mathbf{2}$ $bs_{\pi(\overline{k})} \leftarrow r_{\pi(\overline{k})}$ /* Step 1: Earliest robust baseline start time relative to $\pi(1), \pi(2), \dots, \pi(\overline{k}-1)$. 3 */ if $\overline{k} > 1$ then $\mathbf{4}$ $bs_{\pi(\overline{k})} \leftarrow \max(r_{\pi(\overline{k})}, bs_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)})$ 5 $t \leftarrow ls_{\pi(\overline{k}-1)}$ 6 while $t \geq \min(ls_{\pi(\overline{k}-1)}, \max(bs_{\pi(\overline{k}-1)}, r_{\pi(\overline{k})} - p_{\pi(\overline{k}-1)}))$ do 7 $rss \leftarrow RSS(bs, \pi(\overline{k}-1), t)$ 8 $\omega \leftarrow \left\lfloor \frac{rss_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)} - 1}{D} \right\rfloor$ 9 $maxPossibleIntersection = \left| \frac{E_{\omega}^{\max} - \sum_{k=1}^{\overline{k}-1} p(\omega, \pi(k), rss_{\pi(k)}) \cdot P_{\pi(k)}}{P_{\pi(\overline{k})}} \right|$ 10 if $p_{\pi(\overline{k})} \leq maxPossibleIntersection$ then 11 /* Small optimisation: Entire operation can be fitted into $\boldsymbol{\omega}$ without 12 violating E_{ω}^{\max} , therefore continue with metering interval $\omega-1$. */ $t \leftarrow \tau_{\omega} - p_{\pi(\overline{k}-1)} - 1$ 13 else if maxPossibleIntersection $\geq (\tau_{\omega} + D) - (rss_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)})$ then 14 /* E_{ω}^{\max} is not violated. */ 15 $t \leftarrow t - 1$ 16 else 17/* E_{ω}^{\max} is violated. */ 18 $bs_{\pi(\overline{k})} \leftarrow \max(r_{\pi(\overline{k})}, (\tau_{\omega} + D) - maxPossibleIntersection)$ 19 break 20 /* Step 2: Earliest robust baseline start time relative to only $\pi(\overline{k})$. */ $\mathbf{21}$ $ls_{\pi(\overline{k})} \leftarrow LS(bs)_{\pi(\overline{k})}$ 22 $\omega \leftarrow \left| \frac{bs_{\pi(\overline{k})}}{D} \right|$ $\mathbf{23}$ while $\omega \in \Omega$ do 24 maxPossibleIntersection $\leftarrow \left\lfloor \frac{E_{\omega}^{\max}}{P_{\pi(\bar{k})}} \right\rfloor$ $\mathbf{25}$ maxIntersection $\leftarrow \min(p_{\pi(\overline{k})}, \operatorname{lenint}([\tau_{\omega}, \tau_{\omega} + D], [bs_{\pi(\overline{k})}, ls_{\pi(\overline{k})} + p_{\pi(\overline{k})}]))$ 26 if maxIntersection = 0 then $\mathbf{27}$ break 28 else if maxPossibleIntersection < maxIntersection then 29 $bs_{\pi(\overline{k})} \leftarrow \tau_{\omega} + D - maxPossibleIntersection$ 30 $ls_{\pi(\overline{k})} \leftarrow LS(bs)_{\pi(\overline{k})}$ 31 $\omega \leftarrow \omega + 1$ 32 if $bs_{\pi(\overline{k})} > bs^{\max}$ then 33 return INFEASIBLE_PERMUTATION $\mathbf{34}$ 35 else return OK 36

Algorithm 3: Earliest Robust Baseline Start Time of $\pi(\overline{k})$ for Fixed Permutation

4. Exact Algorithms for $1|r_j, E_{\omega}^{\max}, \delta_j^{\max} = \delta^{\max}|\sum T_j|$

We propose two exact algorithms for $1|r_j, E_{\omega}^{\max}, \delta_j^{\max} = \delta^{\max}|\sum T_j$: (i) Branch-and-Bound and (ii) a logic-based Benders decomposition algorithm. Both algorithms exploit the fact that the optimal robust schedule for the fixed permutation can be found by Algorithm 1.

4.1. Branch-and-Bound

Since robustness of a schedule with fixed operations' order can be solved independently by Algorithm 1, the proposed Branch-and-Bound algorithm (BB) is very simple. The algorithm searches through the space of partial permutations, i.e. in each node of the search tree, BB constructs the earliest robust schedule for partial permutation $\pi(1), \pi(2), \ldots, \pi(\overline{k} - 1)$ using Algorithm 1. For each remaining operation $\pi(\overline{k}) \in \mathcal{J} \setminus {\pi(1), \pi(2), \ldots, \pi(\overline{k} - 1)}$, a new branch is created by appending operation $\pi(\overline{k})$ to the end of the current partial permutation π , i.e. the set of new branches is

$$\{\pi(1), \pi(2), \dots, \pi(\bar{k}-1), \pi(\bar{k}) \mid \pi(\bar{k}) \in \mathcal{J} \setminus \{\pi(1), \pi(2), \dots, \pi(\bar{k}-1)\}\}$$
(18)

Our BB is implemented as Depth-First Search, and the branches are prioritised according to the due dates of $\mathcal{J} \setminus \{\pi(1), \pi(2), \ldots, \pi(\overline{k}-1)\}$.

The solution space is pruned using Chu's lower bound [4]. The lower bound is designed for problem $1|r_j| \sum T_j$ and can be computed as follows. W.l.o.g. assume that the due dates of operations are sorted, i.e. $d_1 \leq d_2 \leq \cdots \leq d_n$. Schedule the operations according to the Shortest Remaining Processing Time rule (i.e. the operations can be preempted); let $C_{[k]}$ be the completion time in the baseline schedule of k-th completed operation. Then Chu's lower bound on the total tardiness is $\sum_{k=1}^{n} \max(C_{[k]} - d_k, 0)$.

4.2. Logic-based Benders decomposition algorithm

Logic-based Benders decomposition (LBBD) [14] is a generalisation of the classical Benders decomposition that is used for solving large-scale optimisation problems. In the classical Benders decomposition, the subproblem is only a continuous linear or non-linear problem whereas in LBBD the subproblem may have an arbitrary form. We use a specific form of the LBBD in which the cuts remove infeasible solutions (*no-good* cuts).

The idea of LBBD is to decompose the original problem into two parts: (i) *master problem*, which is a relaxation of the original problem and (ii) *subproblem*. After the master problem is solved to optimality, its solution is checked by the subproblem whether it is feasible in the original problem or not. If yes, then the decomposition algorithm finishes since an optimal feasible solution for the original problem has been found. If not, a *cut* constraint is generated in which the infeasible solution is violated. The cut is added to the master problem, and the whole procedure is repeated.

In modern implementations of LBBD, the cuts are added gradually during solving the master problem [17]. This approach is more integrated into an Mixed Integer Linear Programming (MILP) solvers and therefore more efficient since the master problem does not need to be resolved from scratch every time a new cut is generated; the state-of-the-art solvers such as Gurobi or CPLEX support adding cuts dynamically using *lazy constraints generation* mechanism.

In our case, the master problem is essentially a MILP model of $1|r_j, E_{\omega}^{\max}| \sum T_j$. The solution of the master problem, i.e. baseline schedule bs', is checked in the subproblem whether it is robust or not. If bs' is not robust, a no-good cut is generated for the master problem.

4.2.1. Master Problem

The MILP formulation of the master problem corresponds to Constraints (20)-(24) and objective (19). It is modelled as a time-indexed formulation, which is suitable for generating the cuts. There are two types of variables in the program: (i) a binary baseline start time of operation j in time t defined as $bs_{j,t} = 1$ iff j starts at t in the baseline schedule and (ii) the energy consumed in time t denoted as E_t^{time} .

$$\min_{\substack{j \in \mathcal{J} \\ h_{c}^{\max}}} \sum_{t=r_{j}}^{bs^{\max}} bs_{j,t} \cdot \max(0, t+p_{j}-d_{j})$$
(19)

s.t.
$$\sum_{t=r_j}^{bs^{max}} bs_{j,t} = 1 , \qquad j \in \mathcal{J}$$
(20)

$$\sum_{\substack{\in \mathcal{J} \ t' = \max(r_j, t-p_j+1)}}^{\min(t, bs^{\max})} bs_{j,t'} \le 1 \quad , \qquad t \in \left[\min_{j \in \mathcal{J}} r_j \dots H - 1\right]$$
(21)

$$\sum_{e \in \mathcal{J}} \sum_{t'=\max(r_j,t-p_j+1)}^{\min(t,bs^{\max})} bs_{j,t'} \cdot P_j = E_t^{\text{time}} , \qquad t \in \left[\min_{j \in \mathcal{J}} r_j \dots H - 1 \right]$$
(22)

$$\sum_{t=\tau_{\omega}}^{\tau_{\omega}+D-1} E_t^{\text{time}} \le E_{\omega}^{\text{max}} , \qquad \omega \in \Omega$$
(23)

$$\sum_{m=\max(0,\tau_{\omega}-\delta)}^{\tau_{\omega}+D-1-\delta} E_t^{\text{time}} \le E_{\omega}^{\max}, \quad \forall \delta \in [1..\delta^{\max}], \forall \omega \in \Omega$$
(24)

The objective (19) of the master problem is the minimisation of $\sum_{j \in \mathcal{J}} T_j$. Constraint (20) ensures that each operation starts in some time that is at least its release time and at most the maximum start time bs^{\max} . Constraint (21) enforces that each time can be occupied by at most one operation. Computation of consumed energy in time t is in Constraint (22). Constraint (23) ensures that the energy consumption limit in each metering interval is not violated for the baseline schedule. Finally, Constraint (24) strengthens the master problem by taking into account a subset of scenarios in which only a single operation deviated. Although this constraint is valid for problem $1|r_j, E_{\omega}^{\max}, \delta_j^{\max} = \delta^{\max}|\sum T_j$, it is not exact, i.e. not every infeasible realised schedule, in which one operation deviated, is cut out. For more details about Constraint (24), see [20].

4.2.2. Subproblem: robustness check and cuts

When integer baseline schedule bs' is found by the MILP solver, it is checked if it is robust, i.e. whether all Constraints (9) are satisfied. If the schedule is not robust, a no-good cut is generated for the master problem such that bs' violates it. For checking the robustness of bs', a pseudo-polynomial algorithm introduced in [20] is employed.

Now assume that schedule bs' is not robust. In general, our cuts have a form of

$$\sum_{j \in \mathcal{J}} \sum_{t \in I'_j} bs_{j,t} \le n - 1 \tag{25}$$

where I'_j is a *cutting interval* of operation j for schedule bs', i.e. the cut enforces that at least one operation starts outside of its cutting interval.

Simple cutting intervals that forbid one particular schedule bs' can be defined as $\forall j \in \mathcal{J} : I'_j = \{bs'_j\}$. In the current work, we introduce cutting intervals that exploit the knowledge of the optimal robust schedule bs^* obtained by Algorithm 1 for permutation π' corresponding to schedule bs'. Informally, the cutting intervals are defined in such a way that the start times of the operations in the baseline schedules having the same order as in π' are "pushed" towards bs^* .

The type of the generated cuts depends on the return value of Algorithm 1 for permutation π'

1. INFEASIBLE_PERMUTATION, \emptyset : this means that any baseline schedule having the same order as π' cannot be robust. Therefore, a cut must be generated that "forbids" π' . Such cut can be formulated

using binary precedence variables $y_{j,j'}$ [15] such that $y_{j,j'} = 1$ if operation j precedes operation j', 0 otherwise. However, to model these variables, Big M constraints are usually employed which have poor relaxation. Alternatively, we can forbid only a subset of all schedules having the same order as π' with a simpler constraint described below.

Consider two operations $\pi'(k)$, $\pi'(k+1)$ that are executed consecutively and let $bs'_{\pi'(k)}$, $bs'_{\pi'(k+1)}$ be their baseline start times, respectively. Consider another schedule bs'' in which the start time of $\pi'(k+1)$ is at least $bs'_{\pi'(k+1)}$. Then to guarantee that $\pi'(k)$ starts before $\pi'(k+1)$ in bs'', the start time of $\pi'(k)$ must be at most $bs'_{\pi'(k+1)} + p_{\pi'(k+1)} - 1$. Therefore, in any integer schedule bs such that

$$bs_{\pi'(k)} \in \left[bs'_{\pi'(k)} \dots \min(bs'_{\pi'(k+1)} + p_{\pi'(k+1)} - 1, bs^{\max}) \right]$$
(26)

$$bs_{\pi'(k+1)} \in \left[bs'_{\pi'(k+1)} \dots bs^{\max} \right]$$

$$\tag{27}$$

operation $\pi'(k)$ is executed before $\pi'(k+1)$.

Such intervals can be derived for whole permutation π' . Therefore, the cutting intervals are

$$I'_{\pi'(k)} = \begin{cases} \left[bs'_{\pi'(k)} \dots \min(bs'_{\pi'(k+1)} + p_{\pi'(k+1)} - 1, bs^{\max}) \right] & k \in [1 \dots n - 1] \\ \left[bs'_{\pi'(k)} \dots bs^{\max} \right] & k = n \end{cases}$$
(28)

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

2. OK, bs^* : since $bs' \neq bs^*$ (otherwise bs' would be robust), there exists position \overline{k} in permutation π' such that

$$bs'_{\pi'(k)} = bs^*_{\pi'(k)}, \quad k \in \left[1 \dots \overline{k} - 1\right]$$

$$\tag{29}$$

$$bs'_{\pi'(\overline{k})} \neq bs^*_{\pi'(\overline{k})} \tag{30}$$

One of the following two cases occurs

(a) $bs'_{\pi'(\overline{k})} < bs^*_{\pi'(\overline{k})}$: Consider the example in Fig. 7 illustrating this case



The idea of the cut is that if in any baseline schedule **bs** the operations on positions $1, \ldots, \overline{k}$ are the same as in π' , i.e.

$$\pi(k) = \pi'(k), \quad \forall k \in \left[1 \dots \overline{k}\right], \tag{31}$$

then the start time of $\pi'(\overline{k})$ should be pushed towards $bs_{\pi'(\overline{k})}^*$ since all baseline start times before $bs_{\pi'(\overline{k})}^*$ are not robust for $\pi'(\overline{k})$. Since the earliest robust baseline start time of $\pi'(\overline{k})$ is by definition dependent only on positions $k \leq \overline{k}$ (see Eq. (12)), the order of the operations on positions $k > \overline{k}$ is not important, the cut only has to guarantee that they are not executed before $\pi'(\overline{k})$. Moreover, we have to take care of the case when some operation $\{\pi'(k) \mid k \in [\overline{k} + 1 \dots n]\}$ starts before $bs_{\pi'(\overline{k})}^*$ in bs'; to make sure that feasible schedule bs in which some operation $\{\pi'(k) \mid k \in [\overline{k} + 1 \dots n]\}$

starts before $\pi'(\overline{k})$ is not cut out, $bs'_{\pi'(\overline{k}+1)}$ bounds from above the cutting interval of $\pi(\overline{k})$ and from below the cutting intervals of $\{\pi'(k) \mid k \in [\overline{k}+1..n]\}$. Therefore, the cutting intervals are

$$I'_{\pi'(k)} = \begin{cases} \begin{bmatrix} bs'_{\pi'(k)} \dots \min(bs'_{\pi'(k+1)} + p_{\pi'(k+1)} - 1, bs^{\max}) \end{bmatrix} & k \in [1 \dots \overline{k} - 1] \\ \begin{bmatrix} bs'_{\pi'(k)} \dots bs^{*}_{\pi'(k)} - 1 \end{bmatrix} & k = \overline{k} \wedge \overline{k} = n \\ \begin{bmatrix} bs'_{\pi'(k)} \dots \min(bs^{*}_{\pi'(\overline{k})}, bs'_{\pi'(\overline{k}+1)} + p_{\pi'(\overline{k}+1)}) - 1 \end{bmatrix} & k = \overline{k} \wedge \overline{k} < n \\ \begin{bmatrix} \min(bs^{*}_{\pi'(\overline{k})}, bs'_{\pi'(\overline{k}+1)} + p_{\pi'(\overline{k}+1)} - p_{\pi'(k)}) \dots bs^{\max} \end{bmatrix} & k \in [\overline{k} + 1 \dots n] \end{cases}$$
(32)

(b) $bs'_{\pi'(\overline{k})} > bs^*_{\pi'(\overline{k})}$: this case is analogous to the previous one with the difference that operation $\pi'(\overline{k})$ is being pushed to the left to $bs^*_{\pi'(\overline{k})}$

$$I'_{\pi'(k)} = \begin{cases} \left[bs'_{\pi'(k)} \dots \min(bs'_{\pi'(k+1)} + p_{\pi'(k+1)} - 1, bs^{\max}) \right] & k \in [1 \dots \overline{k} - 1] \\ \left[bs^*_{\pi'(k)} + 1 \dots bs'_{\pi'(k)} \right] & k = \overline{k} \\ \left[bs'_{\pi'(\overline{k})} - p_{\pi'(k)} + 1 \dots bs^{\max} \right] & k \in [\overline{k} + 1 \dots n] \end{cases}$$
(33)

Notice, that even though the cutting intervals may cut out robust baseline schedules in which the operations are not starting at their earliest robust start times, they do not cut out the optimal schedule bs^* . Therefore, the lazy constraints approach is exact.

5. Heuristic Algorithm for $1|r_j, E_{\omega}^{\max}, \delta_j^{\max} = \delta^{\max}|\sum T_j|$

Since exact approaches are not able to provide solutions in a reasonable time for larger instances, the optimality is often sacrificed for efficiency in practical applications. In this section, we provide a heuristic that exploits the fact that for a given permutation, the optimal robust schedule can be found by Algorithm 1. Therefore, it is enough to search through the space of distinct permutations instead of a larger space of the baseline start times.

The heuristic has two-stages: construction of the initial permutation using a greedy algorithm and a tabu search algorithm that improves the initial permutation.

5.1. Greedy algorithm for initial permutation

The initial permutation is constructed in a greedy way using Algorithm 4. In each iteration \overline{k} of the algorithm, some operation from the set of not assigned operations *notAssignedOperations* is assigned to position \overline{k} in the permutation. The algorithm selects such operation $j \in notAssignedOperations$ which minimises the lower bound on the total tardiness of the operations in *notAssignedOperations*. To compute the lower bound, the baseline start time of operation j relative to the current partial permutation is found by Algorithm 3 and the remaining operations $j' \in notAssignedOperations \setminus \{j\}$ are allocated to the maximum of their release time and the completion time of operation j (notice that operations $notAssignedOperations \setminus \{j\}$ may overlap). If any two operations achieve the same value of the lower bound, the algorithm selects the operation which completes the earliest.



Algorithm 4: Initial permutation.

5.2. Tabu Search

To improve the initial solution found by Algorithm 4, we employ a simple tabu search [8]. The tabu search explores the space of the permutations of the operations and the tabu list contains the previously visited permutations. The neighbourhood of some permutation is generated randomly using two moves: swapping of two randomly selected operations and moving one randomly selected operation to another position. The best permutation in the neighbourhood, which is not tabu, is selected as a basis for the next iteration. The tabu search finishes after executing the predefined number of iterations.

The parameters for the tabu search were set according to the preliminary experiments as follows

Parameter	Value
Number of restarts	5 (including the initial run)
Number of iterations	200
Neighbourhood size	50
Tabu list length	5

Table 4: Tabu search parameters.

6. Experiments

The performance and efficiency of the proposed exact and heuristic algorithms were evaluated using the following two experiments: (i) comparison of the exact and heuristic algorithms on small instances relative to the obtained objective values and (ii) tabu search algorithm evaluation on large instances. All experiments were executed on an Intel(R) Core(TM) i5–4460 CPU @ 3.20GHz computer with 8GB of RAM running Fedora 23 operating system. Gurobi Optimizer 7.0 was used for solving the master problem (see Section 4.2.1) while the rest of the algorithms were programmed in C++ and compiled with GCC 5.3.1.

The source code of the algorithms and the generated instances are publicly available at https://github.com/CTU-IIG/RSECLP.

6.1. Fist experiment: comparison of the exact and heuristic algorithms on small instances

6.1.1. Instances

Three different sets of instances were generated randomly for various values of $n \in \{5, 10, 15\}$. The number of metering intervals and the length of the metering intervals was fixed to $3 \cdot n$ and 15, respectively. For simplicity reason, the maximum energy consumption in each metering interval is the same and was fixed to $E^{\max} = 100$. In each instance set, three values $\alpha_1, \alpha_2, \alpha_3$ were used to control how the parameters of the instances were generated: p_j was sampled from discrete uniform distribution $\mathcal{U}\{1, D\}$; exponential distribution with mean $\alpha_1 \cdot \frac{\sum_{j \in \mathcal{J}} p_j}{n}$ was used to sample the interarrival time of the operations, i.e. the difference between the release times of two consecutive operations; for generating d_j , value of $d_j - (r_j + p_j)$ was sampled from $\mathcal{U}\left\{0, \left[\alpha_2 \cdot \sum_{j \in \mathcal{J}} p_j\right]\right\}$; P_j was sampled from continuous uniform distribution $\mathcal{U}\left(\frac{\alpha_3 \cdot E^{\max}}{p_j}, \frac{E^{\max}}{p_j}\right)$. Notice that to avoid generating infeasible instances, the operations itself cannot violate the energy limit.

For each triple $(\alpha_1, \alpha_2, \alpha_3) \in \{0.6, 0.9\} \times \{0.1, 0.3\} \times \{0.1, 0.3, 0.5\}$, parameters p_j, P_j, r_j, d_j were randomly sampled 10-times according to the description above. For the sampled parameters p_j, P_j, r_j, d_j , three instances differing only in the maximum deviation $\delta^{\max} \in \{0, 3, 5\}$ were generated. Therefore, each instance set $n \in \{5, 10, 15\}$ consisted of $2 \cdot 2 \cdot 3 \cdot 10 \cdot 3 = 360$ instances.

6.1.2. Results

In the following text, the evaluated exact and heuristic algorithms are denoted as follows: *Greedy* for the algorithm finding the initial permutation (see Section 5.1), *Tabu* for the tabu search (see Section 5.2), *BB* for the Branch-and-Bound (see Section 4.1) and *LBBD* for the logic-based Benders decomposition (see Section 4.2). The time-limit given to each algorithm for solving each instance is 20 minutes. The schedules found by *Tabu* were used as an upper bound for the exact algorithms (the execution time of *Tabu* is not reflected in the time-limit of the exact solvers since its running time for such small instances is negligible).

The average objective values with a standard deviation obtained by the algorithms for n = 5, n = 10 and n = 15 are shown in Tables 5, 6 and 7, respectively. To make the tables more comprehensible, the instances were grouped by α_3 and δ^{\max} . As expected, the objective value increases with the increasing maximum deviation, since idle times and different permutations of the operations are necessary to make the schedules robust. The same observation applies for increasing $\frac{\alpha_3 \cdot E^{\max}}{p_j}$, i.e. the lower bound on the power consumption of the operations. Instances with n = 5 (see Tab. 5) are small enough that both *LBBD* and *BB* can found optimal solutions for every instance. All instances with n = 10 (see Tab. 6) can still be solved optimally with *BB*. On the other, instances with n = 15 (see Tab. 7) are hard for both *LBBD* and *BB* algorithms. *LBBD* approach can solve the majority of the instances optimally if the maximum deviation is zero, which is not true for the *BB*. In all cases, we see that *Tabu* is able to find very good solutions.

The average running times of the algorithms with a standard deviation for n = 5, n = 10 and n = 15 are shown in Tables 8, 9 and 10, respectively. The results for *LBBD* and *BB* correspond to the number of found optimal solutions.

The conclusion of the experiment is that with increasing the maximum deviation and the lower bound on the power consumption of the operations, the instances are harder to solve. Tab. 7 suggests that n = 15is a tipping point for any exact algorithm based on the MILP formulation of the master problem (see

		Objective value, average \pm std					ptimality [%]
α_3	δ^{\max}	Greedy	Tabu	LBBD	BB	LBBD	BB
0.1	0	32.0 ± 22.7	27.2 ± 19.0	27.2 ± 19.0	27.2 ± 19.0	100.0	100.0
0.1	3	36.5 ± 23.3	34.1 ± 22.6	34.1 ± 22.6	34.1 ± 22.6	100.0	100.0
0.1	5	45.1 ± 25.4	42.0 ± 24.4	42.0 ± 24.4	42.0 ± 24.4	100.0	100.0
0.3	0	37.4 ± 19.1	35.0 ± 19.0	35.0 ± 19.0	35.0 ± 19.0	100.0	100.0
0.3	3	46.8 ± 21.7	44.5 ± 19.8	44.5 ± 19.8	44.5 ± 19.8	100.0	100.0
0.3	5	55.9 ± 24.8	53.6 ± 23.0	53.6 ± 23.0	53.6 ± 23.0	100.0	100.0
0.5	0	55.8 ± 33.4	51.2 ± 29.8	51.2 ± 29.8	51.2 ± 29.8	100.0	100.0
0.5	3	70.5 ± 34.4	67.5 ± 32.6	67.5 ± 32.6	67.5 ± 32.6	100.0	100.0
0.5	5	78.8 ± 33.0	77.0 ± 32.4	77.0 ± 32.4	77.0 ± 32.4	100.0	100.0

Section 4.2.1) since it is not able to solve all the instances when the maximum deviation is zero (i.e. no cuts are necessary to be generated).

Table 5: Aggregated objective value, $n=5\,$

			Proven optimality [%]				
α_3	δ^{\max}	Greedy	Tabu	LBBD	BB	LBBD	BB
0.1	0	119.8 ± 69.9	101.4 ± 59.7	101.4 ± 59.7	101.4 ± 59.7	100.0	100.0
0.1	3	164.5 ± 82.7	142.7 ± 69.9	142.7 ± 69.9	142.7 ± 69.9	65.0	100.0
0.1	5	202.6 ± 89.3	173.8 ± 79.2	173.7 ± 79.1	173.7 ± 79.1	45.0	100.0
0.3	0	178.9 ± 91.9	153.2 ± 78.1	153.2 ± 78.1	153.2 ± 78.1	100.0	100.0
0.3	3	236.8 ± 95.7	212.4 ± 89.0	212.4 ± 89.0	212.3 ± 89.0	42.5	100.0
0.3	5	275.4 ± 96.7	248.8 ± 94.8	248.8 ± 94.8	248.8 ± 94.8	17.5	100.0
0.5	0	231.1 ± 106.9	190.4 ± 90.0	190.2 ± 89.6	190.2 ± 89.6	100.0	100.0
0.5	3	295.8 ± 103.8	267.7 ± 103.2	267.7 ± 103.2	267.7 ± 103.2	35.0	100.0
0.5	5	336.6 ± 105.3	310.6 ± 109.1	310.6 ± 109.1	310.6 ± 109.1	10.0	100.0

Table 6: Aggregated objective value, $n=10\,$

			Objective value, average \pm std				
α_3	δ^{\max}	Greedy	Tabu	LBBD	BB	LBBD	BB
0.1	0	284.1 ± 163.3	223.3 ± 141.8	221.9 ± 140.7	222.7 ± 141.3	97.5	32.5
0.1	3	401.5 ± 170.2	332.4 ± 160.1	332.4 ± 160.1	330.8 ± 160.7	12.5	17.5
0.1	5	490.4 ± 197.0	392.6 ± 182.6	392.6 ± 182.6	392.1 ± 182.2	10.0	12.5
0.3	0	437.4 ± 176.9	337.9 ± 146.4	337.6 ± 146.0	337.9 ± 146.4	82.5	10.0
0.3	3	550.6 ± 176.1	493.0 ± 169.1	493.0 ± 169.1	492.8 ± 169.0	2.5	5.0
0.3	5	652.2 ± 189.4	583.1 ± 185.6	583.1 ± 185.6	582.8 ± 185.6	2.5	2.5
0.5	0	541.0 ± 214.7	445.8 ± 181.7	445.1 ± 181.1	445.7 ± 181.6	77.5	12.5
0.5	3	715.3 ± 227.9	631.2 ± 196.5	631.2 ± 196.5	630.3 ± 196.5	0.0	0.0
0.5	5	804.5 ± 225.7	748.9 ± 230.4	748.9 ± 230.4	747.4 ± 232.3	0.0	0.0

Table 7: Aggregated objective value, n=15

		Time [s], average \pm std						
α_3	δ^{\max}	Greedy	Tabu	LBBD	BB			
0.1	0	0.0 ± 0.0	0.1 ± 0.0	0.1 ± 0.1	0.0 ± 0.0			
0.1	3	0.0 ± 0.0	0.1 ± 0.0	0.3 ± 0.3	0.0 ± 0.0			
0.1	5	0.0 ± 0.0	0.1 ± 0.0	0.7 ± 0.9	0.0 ± 0.0			
0.3	0	0.0 ± 0.0	0.1 ± 0.0	0.1 ± 0.1	0.0 ± 0.0			
0.3	3	0.0 ± 0.0	0.1 ± 0.0	0.5 ± 0.8	0.0 ± 0.0			
0.3	5	0.0 ± 0.0	0.1 ± 0.0	1.1 ± 1.5	0.0 ± 0.0			
0.5	0	0.0 ± 0.0	0.1 ± 0.0	0.2 ± 0.4	0.0 ± 0.0			
0.5	3	0.0 ± 0.0	0.1 ± 0.0	1.0 ± 1.3	0.0 ± 0.0			
0.5	5	0.0 ± 0.0	0.1 ± 0.0	2.0 ± 2.2	0.0 ± 0.0			

Table 8: Aggregated running time, n = 5

			Time [s], average \pm std						
α_3	δ^{\max}	Greedy	Tabu	LBBD	BB				
0.1	0	0.0 ± 0.0	0.1 ± 0.0	4.2 ± 7.2	1.0 ± 1.1				
0.1	3	0.0 ± 0.0	0.1 ± 0.0	483.1 ± 536.0	1.3 ± 1.2				
0.1	5	0.0 ± 0.0	0.2 ± 0.0	781.2 ± 502.6	1.6 ± 1.3				
0.3	0	0.0 ± 0.0	0.1 ± 0.0	28.0 ± 65.9	1.6 ± 1.2				
0.3	3	0.0 ± 0.0	0.1 ± 0.0	759.5 ± 525.0	2.0 ± 1.3				
0.3	5	0.0 ± 0.0	0.2 ± 0.0	1036.6 ± 370.4	2.4 ± 1.4				
0.5	0	0.0 ± 0.0	0.1 ± 0.0	18.8 ± 30.3	2.0 ± 1.7				
0.5	3	0.0 ± 0.0	0.1 ± 0.0	961.7 ± 395.3	2.6 ± 1.8				
0.5	5	0.0 ± 0.0	0.2 ± 0.0	1141.9 ± 208.5	3.0 ± 1.8				

Table 9: Aggregated running time, n=10

			Time [s], average \pm std							
α_3	δ^{\max}	Greedy	Tabu	LBBD	BB					
0.1	0	0.0 ± 0.0	0.1 ± 0.0	137.8 ± 230.6	896.8 ± 479.5					
0.1	3	0.0 ± 0.0	0.3 ± 0.1	1056.3 ± 379.3	1062.5 ± 361.3					
0.1	5	0.0 ± 0.0	0.4 ± 0.1	1080.8 ± 354.8	1081.4 ± 320.2					
0.3	0	0.0 ± 0.0	0.1 ± 0.0	423.0 ± 423.6	1103.6 ± 311.3					
0.3	3	0.0 ± 0.0	0.2 ± 0.0	1169.1 ± 186.9	1153.8 ± 210.3					
0.3	5	0.0 ± 0.0	0.3 ± 0.1	1174.4 ± 153.8	1170.6 ± 183.7					
0.5	0	0.0 ± 0.0	0.1 ± 0.0	460.9 ± 442.3	1136.0 ± 192.3					
0.5	3	0.0 ± 0.0	0.2 ± 0.0	1199.0 ± 0.0	1200.0 ± 0.0					
0.5	5	0.0 ± 0.0	0.3 ± 0.1	1199.0 ± 0.0	1200.0 ± 0.0					

Table 10: Aggregated running time, n=15

6.2. Second experiment: evaluation of the tabu search on large instances

6.2.1. Instances

The instances were generated in a similar manner as in Section 6.1 with the exception that the number of the operations was fixed to 100.

6.2.2. Results

We compared a simple Earliest Due Date First (EDF) ordering rule with Tabu. For this experiment, the stopping condition of Tabu was modified: instead of running for the pre-specified number of iterations, Tabu is stopped if the objective was not improved in the last 50 iterations; the rest of the parameters are the same as in Table 4.

The average objective values with a standard deviation obtained by the algorithms are shown in Table 11 and the average running times of the algorithms with a standard deviation are shown in Table 12. We can see that Tabu is able to find significantly better solutions than the EDF rule within one minute. On average, over all instances, solutions found by *Greedy* and *Tabu* algorithms are 26.5% and 40.2% better than EDF, respectively. *Tabu* improves the initial solution found by *Greedy* algorithm by 18.6%.

		Objective value, average \pm std						
α_3	δ^{\max}	EDF	Greedy	Tabu				
0.1	0	20101.8 ± 5958.5	12854.1 ± 4866.3	8898.8 ± 4639.8				
0.1	3	32171.8 ± 6047.1	19130.7 ± 4704.1	15966.2 ± 4792.4				
0.1	5	39465.8 ± 6345.3	25117.9 ± 5198.9	19330.0 ± 4874.8				
0.3	0	22810.2 ± 6448.9	17310.3 ± 5870.9	12195.6 ± 5746.0				
0.3	3	35228.0 ± 7264.9	23290.3 ± 5583.9	20129.2 ± 5949.5				
0.3	5	41343.4 ± 6993.2	29707.7 ± 5998.8	24152.6 ± 5894.9				
0.5	0	29926.3 ± 6233.8	25845.9 ± 6109.9	19967.6 ± 5976.9				
0.5	3	41215.0 ± 6057.4	33252.8 ± 5604.8	29037.8 ± 6223.5				
0.5	5	45246.8 ± 6183.4	39422.1 ± 6058.4	34274.7 ± 6862.9				

Table 11: Aggregated objective value, n = 100

		Time	Time [s], average \pm std					
α_3	δ^{\max}	EDF	Greedy	Tabu				
0.1	0	0.0 ± 0.0	0.0 ± 0.0	3.0 ± 0.4				
0.1	3	0.0 ± 0.0	0.0 ± 0.0	11.8 ± 3.9				
0.1	5	0.0 ± 0.0	0.0 ± 0.0	28.9 ± 12.3				
0.3	0	0.0 ± 0.0	0.0 ± 0.0	3.0 ± 0.4				
0.3	3	0.0 ± 0.0	0.0 ± 0.0	11.2 ± 3.4				
0.3	5	0.0 ± 0.0	0.0 ± 0.0	26.0 ± 8.1				
0.5	0	0.0 ± 0.0	0.0 ± 0.0	3.0 ± 0.4				
0.5	3	0.0 ± 0.0	0.0 ± 0.0	9.8 ± 2.5				
0.5	5	0.0 ± 0.0	0.0 ± 0.0	16.2 ± 5.8				

Table 12: Aggregated running time, n = 100

7. Conclusion

In this work, we tackled the scheduling problem of satisfying the energy consumption limits by the manufacturing or production companies with high energy demand under production uncertainties. The problem with uncertainty often occurs in reality since the carried-out schedule is usually different from the proposed baseline schedule.

Our main contribution is an algorithm that finds the optimal robust baseline schedule for the fixed permutation of the operations. This algorithm can be used either as: (i) a method for making the existing baseline schedule robust or (ii) operator incorporated in a scheduling algorithm that seeks the optimal permutation. We employed the algorithm in two exact (logic-based Benders decomposition and Branch-and-Bound) algorithms and one heuristic algorithm (tabu search). The exact algorithms were evaluated on small instances. The Branch-and-Bound algorithm is less sensitive to increase of the maximum deviation δ^{\max} , whereas the logic-based Benders decomposition approach is viable if the number of cutting constraints needed to generate robust schedules is low (which is typical if the energy limits are not very tight and the maximum deviation is small). In the experiments, we showed that with increasing the maximum deviation and the lower bound on the power consumption of the operations the instances are harder to solve.

To assess the scalability of our approach, the tabu search heuristic was evaluated on large instances with 100 operations. On average, the instances were solved within 1 minute w.r.t. to the stopping criteria (the search stops if the objective value is not improved in the last 50 iterations). On average, the tabu search improved the objective by 40.2% over the Earliest Due Date First ordering rule.

Acknowledgement

The work in this paper was supported by the Technology Agency of the Czech Republic under the Centre for Applied Cybernetics TE01020197.

References

- K. Biel, C. H. Glock, Systematic literature review of decision support models for energy-efficient production planning, Computers & Industrial Engineering 101 (2016) 243 – 259.
- [2] A. Bruzzone, D. Anghinolfi, M. Paolucci, F. Tonelli, Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops, CIRP Annals - Manufacturing Technology 61 (1) (2012) 459–462.
- [3] A. Cheref, C. Artigues, J.-C. Billaut, A new robust approach for a production scheduling and delivery routing problem, IFAC-PapersOnLine 49 (12) (2016) 886–891, 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, Troyes, France, 28—30 June 2016.
- [4] C. Chu, A branch-and-bound algorithm to minimize total tardiness with different release dates, Naval Research Logistics (NRL) 39 (2) (1992) 265–283.
- [5] E. Coban, A. Heching, J. N. Hooker, A. Scheller-Wolf, Robust scheduling with logic-based Benders decomposition, in: M. Lübbecke, A. Koster, P. Letmathe, R. Madlener, B. Peis, G. Walther (eds.), Operations Research Proceedings (OR 2014), Springer International Publishing, 2016, pp. 99–105.
- [6] Energetický regulační úřad, Energetický regulační věstník 10/2016, https://portal.gov.cz/portal/publikujici/ eeuaau7/vestniky/20387_doc.pdf (in Czech), accessed March 7, 2017.
- [7] K. Fang, N. A. Uhan, F. Zhao, J. W. Sutherland, Flow shop scheduling with peak power consumption constraints, Annals of Operations Research 206 (1) (2013) 115–145.
- [8] F. Glover, Tabu search-part i, ORSA Journal on computing 1 (3) (1989) 190-206.
- [9] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of discrete mathematics 5 (1979) 287–326.
- [10] H. Hadera, I. Harjunkoski, G. Sand, I. E. Grossmann, S. Engell, Optimization of steel production scheduling with complex time-sensitive electricity cost, Computers & Chemical Engineering 76 (2015) 117–136.
- [11] A. Haït, C. Artigues, A hybrid CP/MILP method for scheduling with energy costs, European Journal of Industrial Engineering 5 (4) (2011) 471–489.
- [12] A. Haït, C. Artigues, On electrical load tracking scheduling for a steel plant, Computers & Chemical Engineering 35 (12) (2011) 3044–3047.
- [13] W. Herroelen, Generating robust project baseline schedules, in: Tutorials in Operations Research: OR Tools and Applications : Glimpses of Future Technologies, TutORials in Operations Research, chap. 8, Institute for Operations Research and the Management Sciences (INFORMS), 2007, pp. 124–144.
- [14] J. N. Hooker, Planning and scheduling by logic-based Benders decomposition, Operations Research 55 (3) (2007) 588–602.
- [15] O. Koné, C. Artigues, P. Lopez, M. Mongeau, Event-based MILP models for resource-constrained project scheduling problems, Computers & Operations Research 38 (1) (2011) 3–13, project Management and Scheduling.
- [16] O. Lambrechts, E. Demeulemeester, W. Herroelen, Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities, Journal of Scheduling 11 (2) (2007) 121–136.
- [17] S. Lin, G. J. Lim, J. F. Bard, Benders decomposition and an IP-based heuristic for selecting IMRT treatment beam angles, European Journal of Operational Research 251 (3) (2016) 715–726.
- [18] S. A. Mansouri, E. Aktas, U. Besikci, Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption, European Journal of Operational Research 248 (3) (2016) 772–788.
- [19] L. Merkert, I. Harjunkoski, A. Isaksson, S. Säynevirta, A. Saarela, G. Sand, Scheduling and energy industrial challenges and opportunities, Computers & Chemical Engineering 72 (0) (2015) 183 – 198.
- [20] I. Módos, P. Šůcha, Z. Hanzálek, Robust scheduling for manufacturing with energy consumption limits, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016, pp. 1–8.

- [21] K. Nagasawa, Y. Ikeda, T. Irohara, Robust flow shop scheduling with random processing times for reduction of peak power consumption, Simulation Modelling Practice and Theory 59 (2015) 102 113.
- [22] K. Nolde, M. Morari, Electrical load tracking scheduling of a steel plant, Computers & Chemical Engineering 34 (11) (2010) 1899–1903.
- [23] S. Plitsos, P. P. Repoussis, I. Mourtos, C. D. Tarantilis, Energy-aware decision support for production scheduling, Decision Support Systems 93 (2017) 88–97.
- [24] N. Policella, A. Cesta, A. Oddi, S. F. Smith, From precedence constraint posting to partial order schedules: A csp approach to robust scheduling, AI Commun. 20 (3) (2007) 163–180.
- [25] D. Van Den Dooren, T. Sys, T. A. M. Toffolo, T. Wauters, G. Vanden Berghe, Multi-machine energy-aware scheduling, EURO Journal on Computational Optimization (2016) 1–23.

Appendix A. Proofs

Lemma 1. Let $bs^{(1)}$, $bs^{(2)}$ be two baseline schedules with the same permutation π such that for some position $\overline{k} \in [1 .. n]$ holds $\forall k \in [1 .. \overline{k}] : bs^{(1)}_{\pi(k)} \leq bs^{(2)}_{\pi(k)}$. Then $\forall k \in [1 .. \overline{k}] : LS(bs^{(1)})_{\pi(k)} \leq LS(bs^{(2)})_{\pi(k)}$.

Proof. Proof by induction on k

1. basis, k = 1: $LS(\boldsymbol{bs}^{(1)})_{\pi(1)} = bs_{\pi(1)}^{(1)} + \delta^{\max} \le bs_{\pi(1)}^{(2)} + \delta^{\max} = LS(\boldsymbol{bs}^{(2)})_{\pi(1)}$ 2. induction step, $1 < k \le \overline{k}$:

$$LS(\boldsymbol{bs}^{(1)})_{\pi(k)} = \max(bs_{\pi(k)}^{(1)}, LS(\boldsymbol{bs}^{(1)})_{\pi(k-1)} + p_{\pi(k-1)}) + \delta^{\max}$$
(A.1)

$$\leq \max(bs_{\pi(k)}^{(2)}, LS(bs^{(2)})_{\pi(k-1)} + p_{\pi(k-1)}) + \delta^{\max}$$
(A.2)

$$= LS(\boldsymbol{bs}^{(2)})_{\pi(k)} \tag{A.3}$$

Lemma 2. Let **bs** be a baseline schedule and π be the corresponding permutation. Let $RSS(\mathbf{bs}, \overline{k}, t)$ be a right-shift schedule for some $\overline{k} \in [1 .. n]$ and $t \in [bs_{\pi(\overline{k})} .. LS(\mathbf{bs})_{\pi(\overline{k})}]$. Then there exist scenario $\delta \in \Delta$ such that $\forall k \in [1 .. \overline{k}] : RS(\mathbf{bs}, \delta)_{\pi(k)} = RSS(\mathbf{bs}, \overline{k}, t)_{\pi(k)}$.

Proof. Let $\mathbf{rss} = RSS(\mathbf{bs}, \overline{k}, t)$ and $\mathbf{rs} = RS(\mathbf{bs}, \delta)$. First we prove that $\forall k \in [1 ... \overline{k}] : rss_{\pi(k)} \in [bs_{\pi(k)} ... LS(\mathbf{bs})_{\pi(k)}]$. The property holds trivially from the definition for $k = \overline{k}$. To prove the property for $k < \overline{k}$ assume by contradiction that $k < \overline{k}$ is the largest position such that $rss_{\pi(k)} \notin [bs_{\pi(k)} ... LS(\mathbf{bs})_{\pi(k)}]$. Since $rss_{\pi(k)} \leq LS(\mathbf{bs})_{\pi(k)}$ holds from the definition of the right-shift start time, it must $rss_{\pi(k)} < bs_{\pi(k)} \leq LS(\mathbf{bs})_{\pi(k)}$ and therefore.

$$rss_{\pi(k)} = \min(LS(\boldsymbol{bs})_{\pi(k)}, rss_{\pi(k+1)} - p_{\pi(k)}) = rss_{\pi(k+1)} - p_{\pi(k)}$$
(A.4)

However, this leads to contradiction

$$rss_{\pi(k+1)} - p_{\pi(k)} = rss_{\pi(k)} < bs_{\pi(k)} \le bs_{\pi(k+1)} - p_{\pi(k)} \le rss_{\pi(k+1)} - p_{\pi(k)}$$
(A.5)

Now we prove that if δ is defined as

$$\delta_{\pi(k)} = \begin{cases} rss_{\pi(1)} - bs_{\pi(1)} & k = 1\\ rss_{\pi(k)} - \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)}) & k \in [2 \dots \overline{k}] \end{cases}$$
(A.6)

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

- 1. basis, k = 1:
 - $\delta_{\pi(1)} \in [0 \dots \delta^{\max}]$:

$$\delta_{\pi(1)} = rss_{\pi(1)} - bs_{\pi(1)} \ge 0 \tag{A.7}$$

$$\delta_{\pi(1)} = rss_{\pi(1)} - bs_{\pi(1)} \le LS(\mathbf{bs})_{\pi(1)} - bs_{\pi(1)} = bs_{\pi(1)} + \delta^{\max} - bs_{\pi(1)} = \delta^{\max}$$
(A.8)

•
$$rs_{\pi(1)} = rss_{\pi(1)}$$
: $rs_{\pi(1)} = bs_{\pi(1)} + \delta_{\pi(1)} = bs_{\pi(1)} + rss_{\pi(1)} - bs_{\pi(1)} = rss_{\pi(1)}$

- 2. induction step, $1 < k \leq \overline{k}$:
 - $\delta_{\pi(k)} \ge 0$: consider cases

(a)
$$bs_{\pi(k)} \ge rss_{\pi(k-1)} + p_{\pi(k-1)}$$
:

$$\delta_{\pi(k)} = rss_{\pi(k)} - \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)}) = rss_{\pi(k)} - bs_{\pi(k)} \ge 0 \tag{A.9}$$

(b) $bs_{\pi(k)} < rss_{\pi(k-1)} + p_{\pi(k-1)}$:

$$\delta_{\pi(k)} = rss_{\pi(k)} - \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)})$$
(A.10)

$$= (rss_{\pi(k)} - p_{\pi(k-1)}) - rss_{\pi(k-1)}$$
(A.11)

$$\geq \min(LS(bs)_{\pi(k-1)}, rss_{\pi(k)} - p_{\pi(k-1)}) - rss_{\pi(k-1)}$$
(A.12)

$$= rss_{\pi(k-1)} - rss_{\pi(k-1)} \tag{A.13}$$

- = 0(A.14)

• $\delta_{\pi(k)} \leq \delta^{\max}$: consider cases (a) $rss_{\pi(k-1)} = LS(\mathbf{bs})_{\pi(k-1)}$: since

$$LS(\mathbf{bs})_{\pi(k)} = \max(bs_{\pi(k)}, LS(\mathbf{bs})_{\pi(k-1)} + p_{\pi(k-1)}) + \delta^{\max}$$
(A.15)

$$= \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)}) + \delta^{\max}$$
(A.16)

therefore

$$\delta_{\pi(k)} = rss_{\pi(k)} - \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)}) = rss_{\pi(k)} - LS(\boldsymbol{bs})_{\pi(k)} + \delta^{\max} \leq \delta^{\max}$$
(A.17)
(b) $rss_{\pi(k-1)} = rss_{\pi(k)} - p_{\pi(k-1)}$:

$$\delta_{\pi(k)} = rss_{\pi(k)} - \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)}) = rss_{\pi(k)} - \max(bs_{\pi(k)}, rss_{\pi(k)}) \le 0 \le \delta^{\max}$$
(A.18)

•
$$rs_{\pi(k)} = rss_{\pi(k)}$$
:

$$rs_{\pi(k)} = \max(bs_{\pi(k)}, rs_{\pi(k-1)} + p_{\pi(k-1)}) + \delta_{\pi(k)}$$
(A.19)

$$= \max(b_{\pi(k)}, rs_{\pi(k-1)} + p_{\pi(k-1)}) + \delta_{\pi(k)}$$

$$= \max(b_{\pi(k)}, rs_{\pi(k-1)} + p_{\pi(k-1)}) + rs_{\pi(k)} + rs_{\pi(k-1)} + rs_{$$

$$= \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)}) + rss_{\pi(k)} - \max(bs_{\pi(k)}, rss_{\pi(k-1)} + p_{\pi(k-1)})$$
(A.21)
$$= rss_{\pi(k)}$$
(A.22)

Lemma 3. Let $bs^{(1)}, bs^{(2)}$ be two baseline schedules with the same permutation π such that for some position $\overline{k} \in [1 \dots n]$

$$bs_{\pi(\bar{k})}^{(1)} = bs_{\pi(\bar{k})}^{(2)} \tag{A.23}$$

$$bs_{\pi(\overline{k})}^{(1)} \le bs_{\pi(\overline{k})}^{(2)}, \quad k \in [1 .. \overline{k} - 1]$$
 (A.24)

Then for every $t \in \left[bs_{\pi(\overline{k})} \dots LS(\boldsymbol{bs}^{(1)})_{\pi(\overline{k})}\right]$ and every $\delta \in \Delta$ such that $t = RS(\boldsymbol{bs}^{(1)}, \delta)_{\pi(\overline{k})}$ it holds that

$$RS(\boldsymbol{bs}^{(1)}, \delta)_{\pi(\overline{k})} = RSS(\boldsymbol{bs}^{(2)}, \overline{k}, t)_{\pi(\overline{k})}$$
(A.25)

$$RS(\boldsymbol{bs}^{(1)}, \delta)_{\pi(k)} \le RSS(\boldsymbol{bs}^{(2)}, \overline{k}, t)_{\pi(k)}, \quad k \in \left[1 \dots \overline{k} - 1\right]$$
(A.26)

Proof. Proof by induction on k

- 1. basis, $k = \overline{k}$: holds from the assumptions.
- 2. induction step, $1 \le k < \overline{k}$: consider two cases
 - (a) $LS(\boldsymbol{bs}^{(2)})_{\pi(k)} > RSS(\boldsymbol{bs}^{(2)}, \overline{k}, t)_{\pi(k+1)} p_{\pi(k)}$: then $RS(\boldsymbol{bs}^{(1)}, \delta)_{\pi(k)} \le RS(\boldsymbol{bs}^{(1)}, \delta)_{\pi(k+1)} - p_{\pi(k)} \le RSS(\boldsymbol{bs}^{(2)}, \overline{k}, t)_{\pi(k+1)} - p_{\pi(k)} = RSS(\boldsymbol{bs}^{(2)}, \overline{k}, t)_{\pi(k)}$ (A.27)

(b)
$$LS(\boldsymbol{bs}^{(2)})_{\pi(k)} \leq RSS(\boldsymbol{bs}^{(2)}, \overline{k}, t)_{\pi(k+1)} - p_{\pi(k)}$$
: then from Lemma 1
 $RS(\boldsymbol{bs}^{(1)}, \delta)_{\pi(k)} \leq LS(\boldsymbol{bs}^{(1)})_{\pi(k)} \leq LS(\boldsymbol{bs}^{(2)})_{\pi(k)} = RSS(\boldsymbol{bs}^{(2)}, \overline{k}, t)_{\pi(k)}$ (A.28)

Lemma 4. Let $s^{(1)}$, $s^{(2)}$ be two schedules (not necessarily baseline, realised, etc.) with the same permutation π such that for some position $\overline{k} \in [1..n]$

$$s_{\pi(\bar{k})}^{(1)} = s_{\pi(\bar{k})}^{(2)} \tag{A.29}$$

$$s_{\pi(k)}^{(1)} \le s_{\pi(k)}^{(2)}, \quad k \in \left[1 \dots \overline{k} - 1\right]$$
 (A.30)

Let $\omega \in \Omega$ be arbitrary metering interval such that $p(\omega, \pi(\overline{k}), s_{\pi(\overline{k})}^{(1)}) > 0$. Then

$$\sum_{k=1}^{\overline{k}} p(\omega, \pi(k), s_{\pi(k)}^{(1)}) \cdot P_{\pi(k)} \le \sum_{k=1}^{\overline{k}} p(\omega, \pi(k), s_{\pi(k)}^{(2)}) \cdot P_{\pi(k)}$$
(A.31)

Proof. The Lemma obviously holds for metering intervals such that $\tau_{\omega} \geq s_{\pi(\overline{k})}^{(1)}$, therefore assume that $\tau_{\omega} < s_{\pi(\overline{k})}^{(1)} < \tau_{\omega} + D$. We prove the Lemma by showing

$$\forall k \in \left[1 \dots \overline{k}\right] : p(\omega, \pi(k), s_{\pi(k)}^{(1)}) \le p(\omega, \pi(k), s_{\pi(k)}^{(2)}) \tag{A.32}$$

The inequality obviously holds for $k = \overline{k}$, therefore assume $k < \overline{k}$. Since $s_{\pi(k)}^{(1)} + p_{\pi(k)} \leq s_{\pi(\overline{k})}^{(1)}$ and $s_{\pi(k)}^{(2)} + p_{\pi(k)} \leq s_{\pi(\overline{k})}^{(1)}$, it holds that

$$p(\omega, \pi(k), s_{\pi(k)}^{(1)}) = \max(0, s_{\pi(k)}^{(1)} + p_{\pi(k)} - \max(\tau_{\omega}, s_{\pi(k)}^{(1)}))$$
(A.33)

$$p(\omega, \pi(k), s_{\pi(k)}^{(2)}) = \max(0, s_{\pi(k)}^{(2)} + p_{\pi(k)} - \max(\tau_{\omega}, s_{\pi(k)}^{(2)}))$$
(A.34)

Now consider function $f(x) = x + p_{\pi(k)} - \max(\tau_{\omega}, x)$. It is easy to see that f(x) is non-decreasing in x and since from assumption we know that $s_{\pi(k)}^{(1)} \leq s_{\pi(k)}^{(2)}$, it holds that

$$p(\omega, \pi(k), s_{\pi(k)}^{(1)}) = \max(0, f(s_{\pi(k)}^{(1)})) \le \max(0, f(s_{\pi(k)}^{(2)})) = p(\omega, \pi(k), s_{\pi(k)}^{(2)})$$
(A.35)

Theorem 1. Let **bs** be some baseline schedule with corresponding permutation π . Let $\overline{k} \in [1..n]$, **rs** be some realised schedule of operations $\pi(1), \pi(2), \ldots, \pi(\overline{k})$ and $\mathbf{rss} = RSS(\mathbf{bs}, \overline{k}, rs_{\pi(\overline{k})})$. Then

$$\forall \omega \in \Omega : p(\omega, \pi(\overline{k}), rs_{\pi(\overline{k})}) > 0 \implies \sum_{k=1}^{\overline{k}} p(\omega, \pi(k), rs_{\pi(k)}) \cdot P_k \le \sum_{k=1}^{\overline{k}} p(\omega, \pi(k), rss_{\pi(k)}) \cdot P_k$$
(A.36)

Proof. From Lemma 3 it holds that

$$rs_{\pi(\overline{k})} = rss_{\pi(\overline{k})} \tag{A.37}$$

$$rs_{\pi(k)} \le rss_{\pi(k)}, \quad k \in \lfloor 1 \dots k - 1 \rfloor$$
(A.38)

By applying Lemma 4 on $s^{(1)} = rs$ and $s^{(2)} = rss$, the Theorem is proven.

Theorem 2. Let **bs** be a baseline schedule with corresponding permutation π . If every operation starts at its earliest robust time in **bs** then **bs** is robust and optimal for permutation π .

Proof.

- 1. *Robustness*: The robustness of **bs** is follows from the definition of the robust baseline start time, see Eq. 12.
- 2. Optimality: To illustrate the proof, we will use the following figure with different schedules



Figure A.8: Example illustration of the schedules.

Let bs^* be the optimal schedule for permutation π . We need to show that $\forall k \in [1 .. \overline{k}] : bs_{\pi(k)} \leq bs^*_{\pi(k)}$. Assume by contradiction that \overline{k} is the first position in π such that

$$bs_{\pi(\overline{k})} > bs_{\pi(\overline{k})}^* \tag{A.39}$$

$$bs_{\pi(k)} \le bs^*_{\pi(k)}, \quad k \in \left[1 \dots \overline{k} - 1\right]$$
(A.40)

Construct schedule bs' such that

$$bs'_{\pi(\overline{k})} = bs^*_{\pi(\overline{k})} \tag{A.41}$$

$$bs'_{\pi(k)} = bs_{\pi(k)}, \quad k \in \left[1 \dots \overline{k} - 1\right]$$
(A.42)

i.e. bs' is the same as bs with exception of operation $\pi(\overline{k})$ that starts at time $bs^*_{\pi(\overline{k})}$. Since $bs'_{\pi(\overline{k})}$ is not the earliest robust time in bs, there exist some realised schedule rs' of bs' such that

$$E_{\omega}^{\max} < \sum_{k=1}^{\overline{k}} p(\omega, \pi(k), rs'_{\pi(k)}) \cdot P_{\pi(k)}$$
(A.43)

in some metering interval $\omega \in \Omega$. Since operations $\pi(1), \ldots, \pi(\overline{k}-1)$ start at their earliest robust time in both schedules **bs** and **bs**', metering interval ω must have non-zero intersection with $\pi(\overline{k})$ in **rs**'. Construct right-shift schedule $\mathbf{rss}^* = RSS(\mathbf{bs}^*, \overline{k}, rs'_{\pi(\overline{k})})$; the construction is possible since from Lemma 1 it holds that $rs'_{\pi(\overline{k})} \in \left[bs^*_{\pi(\overline{k})} \dots LS(\mathbf{bs}^*)_{\pi(\overline{k})}\right]$. By applying Lemma 3 for $\mathbf{bs}^{(1)} = \mathbf{bs}', \mathbf{bs}^{(2)} = \mathbf{bs}^*$ and Lemma 4 for $\mathbf{s}^{(1)} = \mathbf{rs}', \mathbf{s}^{(2)} = \mathbf{rss}^*$ we conclude that

$$E_{\omega}^{\max} < \sum_{k=1}^{\bar{k}} p(\omega, \pi(k), rs'_{\pi(k)}) \cdot P_{\pi(k)} \le \sum_{k=1}^{\bar{k}} p(\omega, \pi(k), rss^*_{\pi(k)}) \cdot P_{\pi(k)}$$
(A.44)

which is a contradiction.

Lemma 5. Let **bs** be a baseline schedule with the corresponding permutation π . Assume that the baseline start times of operations $\pi(1), \pi(2), \ldots, \pi(\overline{k}-1)$ are robust for some $\overline{k} \in [1..n]$. Moreover, assume that there exists some realised schedule **rs** of **bs** such that for some metering interval $\omega \in \Omega$

$$E_{\omega}^{max} < \sum_{k=1}^{\overline{k}} p(\omega, \pi(k), rs_{\pi(k)}) \cdot P_{\pi(k)}$$
(A.45)

holds and $p(\omega, \pi(\overline{k}-1), rs_{\pi(\overline{k}-1)}) > 0$. Then all baseline start times $\left[bs_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)} \dots rs_{\pi(\overline{k})}\right]$ of $\pi(\overline{k})$ are not robust.

Proof. To illustrate the proof, we will use the following figure with different schedules



Figure A.9: Example illustration of the schedules.

Assume by contradiction that there is baseline schedule bs' such that

$$bs'_{\pi(\overline{k})} \in \left[bs_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)} \dots rs_{\pi(\overline{k})} \right]$$
(A.46)

$$bs'_{\pi(k)} = bs_{\pi(k)}, \quad k \in [1..k-1]$$
 (A.47)

and $bs'_{\pi(\overline{k})}$ is robust. Construct schedule $\boldsymbol{rs'}$ of $\boldsymbol{bs'}$ such that

$$rs'_{\pi(\bar{k})} = \max(bs'_{\pi(\bar{k})}, rs'_{\pi(\bar{k}-1)} + p_{\pi(\bar{k}-1)})$$
(A.48)

$$rs'_{\pi(k)} = rs_{\pi(k)}, \quad k \in [1..\overline{k} - 1]$$
 (A.49)

It is easy to see that rs' is a realised schedule in which the deviation of $\pi(\overline{k})$ is 0.

Since $rs_{\pi(\overline{k})} \geq rs'_{\pi(\overline{k})} \geq rs'_{\pi(\overline{k})} \geq rs'_{\pi(\overline{k}-1)} + p_{\pi(\overline{k}-1)}$, it holds that $p(\omega, \pi(\overline{k}), rs_{\pi(\overline{k})}) \cdot P_{\pi(\overline{k})} \leq p(\omega, \pi(\overline{k}), rs'_{\pi(\overline{k})}) \cdot P_{\pi(\overline{k})}$; the argument is analogous to the one shown in proof of Lemma 4. But this leads to contradiction

$$E_{\omega}^{\max} < \sum_{k=1}^{\bar{k}} p(\omega, \pi(k), rs_{\pi(k)}) \cdot P_{\pi(k)} \le \sum_{k=1}^{\bar{k}} p(\omega, \pi(k), rs'_{\pi(k)}) \cdot P_{\pi(k)}$$
(A.50)