Scheduling on dedicated machines with energy consumption limit

István Módos[1,2], Kiryl Kalodkin, Přemysl Šůcha[1] and Zdeněk Hanzálek[1]
[1]*Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic*
[2]*Department of Control Engineering, Czech Technical University in Prague, Czech Republic*
{*istvan.modos, premysl.sucha, zdenek.hanzalek*}*@cvut.cz*

Keywords: scheduling, dedicated machines, energy consumption limits.

Abstract: This work studies a problem of scheduling non-preemptive independent jobs on dedicated machines while considering an energy consumption limit. The problem is motivated by energy-demanding production processes, such as glass tempering and steel hardening, in which a material is heated to high temperature in furnaces. The production companies have contracts with electric utilities that specify a maximum energy consumption limit. If the heating in the furnaces is not planned carefully, the energy spikes overshoot the energy consumption limit, and the companies must pay large penalty fees. In this paper, we propose two exact methods that find schedules with the minimum makespan such that the energy limit is satisfied. The first proposed method is a Constraint Programming model and the second one finds the optimal solution by iteratively re-solving a Mixed Integer Linear Programming model with a decreasing scheduling horizon. The iterative algorithm exploits the fact that the start times do not need to be modeled explicitly, which leads to an efficient method for solving instances with a higher number of shorter jobs. The experimental results show that our methods outperform an adapted approach from the literature for a related problem.

1 INTRODUCTION

The motivation for this work comes from production processes in which batches of material are heated in furnaces to high temperature, e.g., glass tempering and steel hardening. Heating of the material is energy-demanding and causes spikes in the energy consumption profile if multiple furnaces are heating the material at the same time. These consumption spikes pose a problem since the production companies have contracts with an electric utility, which specify a *maximum energy consumption limit* in every 15 minutes *metering interval*. Failing to comply with the energy consumption limit leads to substantial penalty fees which are directly proportional to the consumed energy over the contracted limit.

Usually, the production companies are scheduling their production either manually or using specialized software. However, only classical scheduling criteria and constraints such as due dates, makespan, changeover times, etc., are considered. Consequently, the resulting production schedules may violate the contracted energy consumption limit. This work proposes two exact scheduling methods for dedicated machines that take into account these energy limits alongside traditional scheduling aspects.

1.1 Related Work

A similar problem of satisfying a *maximum power demand* was studied by (Bruzzone et al., 2012; Fang et al., 2013). Their models try to comply with the maximum power demand at every time instant. This problem can be seen as a special case of our problem in which the length of the metering interval is 1.

Another related problem to the energy consumption limit is *electrical load tracking* (Nolde and Morari, 2010; Haït and Artigues, 2011; Haït and Artigues, 2011; Hadera et al., 2015; Gajic et al., 2017), where the objective is to minimize the absolute difference between the actual and contracted energy consumption over all metering intervals. In the load tracking problem, both over-consumption and under-consumption of the energy are penalized and the authors usually consider minimization of the energy consumption deviation. Thus, the violation of the contracted energy consumption is a soft constraint in the electrical load tracking problem.

The energy consumption limits were considered in the domain of *lot sizing* (Masmoudi et al., 2017; Rapine et al., 2018). In contrast to scheduling, the goal of the lot sizing problem is to determine the quantity of the produced products by the machines in each period

so that the demand is satisfied.

The most similar problem to ours is the robust scheduling problem presented in (Módos et al., 2017). The authors proposed algorithms for designing robust schedules guaranteeing that the energy consumption limits are not violated even if some jobs are delayed due to unexpected circumstances. However, the authors consider only a single machine, thus their problem is a special case of the parallel machine scheduling problem. Although there is no machine allocation in the parallel scheduling problem with energy consumption limits, the schedules on the machines influence each other through the energy limit, thus the problem cannot be solved independently on each machine.

1.2 Contribution

In this paper, we propose two exact methods for the problem of scheduling non-preemptive independent jobs on dedicated machines so that the makespan is minimized. The resulting schedules must also satisfy the energy consumption limit in every metering interval. The first method is a Constraint Programming (CP) model and the second one is a Mixed Integer Linear Programming (MILP) based iterative algorithm which exploits the fact, that the start times of the jobs do not need to be modeled explicitly. As shown in the experiments, both of the methods outperform an adapted MILP model from (Haït and Artigues, 2011; Haït and Artigues, 2011) w.r.t. the total number of solved instances to optimality. Moreover, we believe that our novel ideas for modeling the overlap of the jobs with the metering intervals can be used to increase the performance of the models for scheduling with *real-time energy prices* (Merkert et al., 2015; Zhao et al., 2018), where the cost of energy may change every hour.

2 PROBLEM STATEMENT

Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of *jobs* and let $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ be a set of *machines*. For each job $J_j \in \mathcal{J}$ we define its *processing time* $p_j \in \mathbb{N}_{>0}$ and *dedicated machine* $\mu_j \in \mathcal{M}$, on which the job has to be processed without preemption. The set of jobs that have to be processed on machine M_k is denoted as $\mathcal{J}_k$. Moreover, for each job J_j we also define its *power consumption* $P_j \in \mathbb{R}_{>0}$ that represents the power consumption of machine μ_j when processing job J_j, i.e., it is the constant rate at which the energy is consumed at every time instant. Therefore, the total consumed energy by each job J_j is $p_j \cdot P_j$.

Let $s_j \in \mathbb{N}_{\geq 0}$ denotes a *start time* of job $J_j \in \mathcal{J}$. The jobs have to be scheduled within *scheduling horizon* $H \in \mathbb{N}_{>0}$, i.e., the jobs must complete at most at time H. The scheduling horizon is divided into a set of *metering intervals* $I = \{I_1, I_2, \ldots, I_{\frac{H}{D}}\}$ with equal *length* of $D \in \mathbb{N}_{>0}$ (it is assumed that H is a multiple of D). For all metering intervals, we define the same *energy limit* $E^{\max}$ representing the upper bound on the total energy consumption of the jobs in each metering interval[1]. We say that vector of start times $\mathbf{s} = (s_j)_{J_j \in \mathcal{J}}$ does not violate the energy limit in metering interval I_i if and only if

$$\sum_{J_j \in \mathcal{J}} \text{Overlap}(j, s_j, i) \cdot P_j \leq E^{\max}, \qquad (1)$$

where $\text{Overlap}(j, s_j, i)$ represents the overlap length of job J_j with metering interval I_i if starting at time s_j. Start times $\mathbf{s}$ are *feasible* if the jobs on each machine are not overlapping and the energy limit is not violated in any metering interval. An example of a feasible schedule of 30 jobs on four machines can be seen in Fig. 1.

The goal of our scheduling problem is to find feasible start times such that *makespan* $C_{\max}$, i.e., the maximum completion time of all jobs, is minimized. We denote this problem in Graham's notation as $m|E^{\max}|C_{\max}$.

The problem $m|E^{\max}|C_{\max}$ is $\mathcal{NP}$-hard since we can reduce 3-partition problem to the decision variant $1|p_j = 1, E^{\max}|-$ of our scheduling problem. In the 3-partition problem, we are given bound B and multiset $A = \{a_1, a_2, \ldots, a_{3 \cdot q}\}$ of integers such that $\frac{B}{4} < a_j < \frac{B}{2}$ and $\sum_{a_j \in A} a_j = q \cdot B$. In this problem we ask the question whether there exist sets $A_1, \ldots, A_q$ such that (i) $\cup_{i=1}^{q} A_i = A$, (ii) $A_{i_1} \cap A_{i_2} = \emptyset$ for all $i_1 \neq i_2$ and (iii) $\sum_{a_j \in A_i} a_j = B$ for all i. The reduction maps each integer a_j to a unit length job with power consumption $P_j = a_j$. The energy limit is set to B, length of each metering intervals to 3 and the number of metering intervals is q.

3 SOLUTION APPROACHES

In this section, we propose two exact approaches to the problem stated in Section 2: (i) a CP model (see Section 3.1) and (ii) a MILP-based iterative algorithm

[1] Usually, the energy consumption limit is contracted for a longer time period such as a month. Therefore, the energy limit is constant within the scheduling horizon. However, our solution methods can be easily generalized to the case where the energy limit varies, but due to the above mentioned reason and simplicity of the methods we consider only one energy limit.

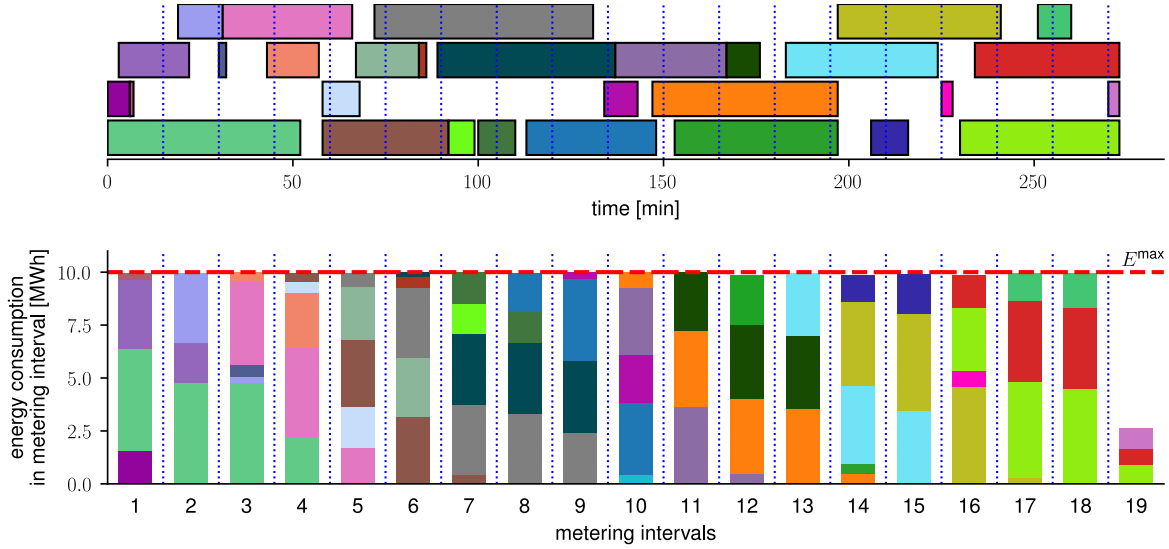

Figure 1: An example of a feasible schedule of 30 jobs on four machines. The figure below the schedule illustrates the energy consumption of the jobs in each 15 minutes metering interval, i.e., the height of each colored box equals $\text{Overlap}(j, s_j, i) \cdot P_j$. The same color in both figures represents the same job. The energy consumption limit is denoted by the dashed horizontal red line.

(see Section 3.2), which exploits the fact that the start times of the jobs do not need to be modeled explicitly (the start times from the model are obtained using an algorithm described in Section 3.2.2).

3.1 CP Model

The first proposed solution approach is a CP model which will be described using IBM CP Optimizer formalism (Laborie et al., 2018). CP uses a notion of *interval variables* representing a time interval, whose start time, completion time and length has to be decided by a solver. Our model has one interval variable x_j for each job $J_j \in \mathcal{J}$ that denotes the time interval in which the job is scheduled. The idea behind the model is to use CP function Overlap, which computes the length of the overlap between a job's interval variable and a metering interval given by its start and end. The overlaps in a metering interval are multiplied by the corresponding power consumption, added together and constrained to be at most $E^{\max}$.

Note that the *cumulative functions* cannot be used for modeling the energy consumption since the *height* of an elementary cumulative function is a constant whereas our problem would require the height to be a function of the overlap length with a metering interval.

The complete model follows

$$\min \max_{J_j \in \mathcal{J}} \text{EndOf}(x_j) \tag{2}$$

$$\text{EndOf}(x_j) \leq H, \quad \forall J_j \in \mathcal{J} \tag{3}$$

$$\text{LengthOf}(x_j) = p_j, \quad \forall J_j \in \mathcal{J} \tag{4}$$

$$\text{NoOverlap}(\{x_j : J_j \in \mathcal{J}_k\}), \quad \forall M_k \in \mathcal{M} \tag{5}$$

$$\sum_{J_j \in \mathcal{J}} \text{Overlap}(x_j, (i-1) \cdot D, i \cdot D) \cdot P_j \leq E^{\max}, \quad \forall I_i \in I \tag{6}$$

The objective (2) minimizes the maximum completion time of all the jobs. Constraint (3) guarantees that the completion time of every job is bounded by the scheduling horizon H. The following constraint (4) sets the length of an interval variable x_j corresponding to some job J_j to be p_j. Global constraint NoOverlap in (5) ensures that the jobs on each dedicated machine are not overlapping. Finally, the energy limit in every metering interval is enforced by constraint (6).

3.2 MILP-based Iterative Algorithm

In this section, we present a MILP formulation that does not explicitly model the start times of the jobs; we call it an *implicit model*. However, the implicit model can optimize the makespan only in the last metering interval of the scheduling horizon, thus it may

found suboptimal solutions to the scheduling problem presented in Section 2. This issue is mitigated by wrapping the implicit model in a loop that iteratively tightens the scheduling horizon until an optimal solution to the original problem is found.

First, we introduce the implicit MILP model, see Section 3.2.1. The following Section 3.2.2 shows how the start times could be reconstructed from a feasible solution to the implicit MILP model. The last Section 3.2.3 wraps the implicit MILP model into an iterative algorithm.

3.2.1 Implicit MILP Model

As noted above, the implicit model is solved iteratively with a decreasing horizon. Let us denote by $I_{i^{\max}}$ the last metering interval, which is considered in the model. To avoid notational clutter, the set of metering intervals in this subsection is redefined to $I = \{I_1, I_2, \ldots, I_{i^{\max}}\}$.

Similarly as the CP model, an overlap of the jobs with metering intervals needs to be formulated. However, MILP does not contain any constructs such as Overlap, thus we must model the overlap using linear expressions ourselves. We observed that for "shorter" jobs the overlap constraints can be modeled much simpler than for "longer" jobs, thus we split the set of jobs into *non-spannable* and *spannable* jobs, i.e., the set of spannable jobs is

$$\mathcal{J}^{\text{span}} = \{J_j \in \mathcal{J} : p_j \geq D+1\}, \tag{7}$$

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

For modeling purposes, we also need to determine the maximum number of consecutive metering intervals in which job J_j can have non-zero overlap for any feasible start times $\mathbf{s}$; this number will be denoted as Ω_j.

Observation 1. *The maximum number of consecutive metering intervals that can have non-zero overlap with job J_j for any feasible start times* $\mathbf{s}$ *is*

$$\Omega_j = \left\lceil \frac{p_j}{D} \right\rceil + 1\,. \tag{8}$$

Proof. See Appendix 5.1. □

The implicit MILP model uses the following variables: (i) $d_{j,i} \in \mathbb{N}_{\geq 0}$ denoting the length of the overlap between job J_j and metering interval I_i, (ii) $x^s_{j,i} \in \{0,1\}$ indicating whether spannable job J_j starts in metering interval I_i, (iii) $x^+_{j,i} \in \{0,1\}$ expressing whether job J_j has non-zero overlap in metering interval I_i, and (iv) $Z \in \mathbb{N}_{\geq 0}$ representing the objective value. Fig. 2 shows an example illustrating the variables used in the implicit MILP model.

The complete model follows

$$\min \quad Z \tag{9}$$

$$Z \geq \sum_{J_j \in \mathcal{J}_k} d_{j,i^{\max}}, \quad \forall M_k \in \mathcal{M} \tag{10}$$

$$\sum_{J_j \in \mathcal{J}_k} d_{j,i} \leq D, \quad \forall I_i \in I, \forall M_k \in \mathcal{M} \tag{11}$$

$$\sum_{J_j \in \mathcal{J}} d_{j,i} \cdot P_j \leq E^{\max}, \quad \forall I_i \in I \tag{12}$$

$$\sum_{I_i \in I} d_{j,i} = p_j, \quad \forall J_j \in \mathcal{J} \tag{13}$$

$$\text{SOS}_2(d_{j,1}, \ldots, d_{j,i^{\max}}), \quad \forall J_j \in \mathcal{J} \setminus \mathcal{J}^{\text{span}} \tag{14}$$

$$\sum_{I_i \in I} x^s_{j,i} = 1, \quad \forall J_j \in \mathcal{J}^{\text{span}} \tag{15}$$

$$d_{j,i} \leq D \cdot \sum_{i'=\max(1,i-\Omega_j+1)}^{i} x^s_{j,i'}, \quad \forall J_j \in \mathcal{J}^{\text{span}}, \forall I_i \in I \tag{16}$$

$$\begin{aligned} &D \cdot (x^+_{j,i-1} + x^+_{j,i+1} - 1) \leq d_{j,i}, \\ &\quad \forall J_j \in \mathcal{J}^{\text{span}}, \forall I_i \in I \setminus \{I_1, I_{i^{\max}}\} \end{aligned} \tag{17}$$

$$d_{j,i} \leq \min(D, p_j) \cdot x^+_{j,i}, \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \tag{18}$$

$$\begin{aligned} &x^+_{j_1,i} + x^+_{j_1,i+1} + x^+_{j_2,i} + x^+_{j_2,i+1} \leq 3, \\ &\quad \forall M_k \in \mathcal{M}, \forall I_i \in I \setminus \{I_{i^{\max}}\}, \\ &\quad \forall J_{j_1}, J_{j_2} \in \mathcal{J}_k : J_{j_1} \neq J_{j_2}, p_{j_1} \geq 2, p_{j_2} \geq 2, \\ &\quad p_{j_1} + p_{j_2} \leq 2 \cdot D \end{aligned} \tag{19}$$

In the following paragraphs, the model will be explained in detail.

The objective of the model is to minimize the maximum total overlap in $I_{i^{\max}}$ over all machines, see Eq. (10). A simple observation is that no idle time between two consecutive jobs on the same machine is necessary in $I_{i^{\max}}$, since the subsequent job is fully contained in $I_{i^{\max}}$ and shifting it to the completion time of the preceding job has no effect on the consumed energy in $I_{i^{\max}}$. Therefore, minimization of the maximum total overlap in $I_{i^{\max}}$ also minimizes the makespan in $I_{i^{\max}}$.

The following constraint (11) ensures, that the total overlap in every metering interval is at most the length of the metering intervals. Constraint (12) models the energy limit and constraint (13) restricts the sum of the overlaps of one job to be its processing time.

Next, the continuity of the jobs over metering intervals is modeled, i.e., only adjacent metering intervals can have non-zero overlap with a job. The non-preemption of the jobs is then ensured by the start times reconstruction procedure, see Section 3.2.2. As noted before, for non-spannable jobs this constraint can be modeled easier than for the spannable jobs,

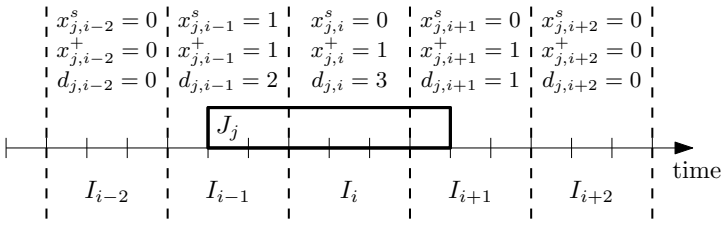

Figure 2: An example illustrating the values of variables for job J_j that are used in the implicit MILP model. The length of the metering intervals is 3.

thus the explanation is split into two parts. We start by handling the non-spannable jobs first.

Due to Observation 1, the non-spannable jobs can overlap at most two consecutive metering intervals. This can be exploited by modeling the continuity of the jobs using *special-ordered set* (Gurobi, 2018) constraint of type 2 (SOS_2). SOS_2 receives a list of ordered variables (can be continuous, binary or integer) and the constraint ensures that at most two consecutive variables in this list can have non-zero value. The continuity of every non-spannable job is then achieved by passing the ordered list of overlap variables to SOS_2 constraint, see constraint (14).

The spannable jobs may overlap more than two consecutive intervals, thus SOS_2 constraint cannot be used. The continuity is modeled using four constraints (15)-(18). The first constraint (15) requires that the job starts in some metering interval. The second constraint (16) ensures that at most Ω_j consecutive intervals have non-zero overlap with job J_j. The third constraint (17) enforces that if job J_j has non-zero overlap with metering intervals I_{i-1}, I_{i+1}, then the job spans the whole metering interval I_i. The last constraint (18) pushes variable $x^+_{j,i}$ to be 1 if job J_j has non-zero overlap with metering interval I_i.

Finally, the last constraint (19) of the model ensures that two jobs J_{j_1}, J_{j_2} on the same machine are not overlapping on a boundary of two consecutive metering intervals. Notice that due to the following lemma this constraint has to be included only for a pair of jobs with $p_{j_1} + p_{j_2} \leq 2 \cdot D$.

Lemma 1. *Let $J_{j_1}, J_{j_2} \in \mathcal{J}$ be two jobs such that $j_1 \neq j_2, \mu_{j_1} = \mu_{j_2}, p_{j_1} \geq 2, p_{j_2} \geq 2$ and $p_{j_1} + p_{j_2} \geq 2 \cdot D + 1$. Let $\mathbf{d} = (d_{j,i})_{J_j \in \mathcal{J}, I_i \in I}$ be values for some feasible solution to the MILP model from Section 3.2. Then Eq.* (19) *holds for J_{j_1}, J_{j_2} in any metering interval $I_i \in I \setminus \{I_{i^{max}}\}$.*

Proof. See Appendix 5.1. □

3.2.2 Reconstruction of the Start Times

As was noted above, the model does not include variables that represent the start times explicitly. However, a simple algorithm exists that reconstructs start times $\mathbf{s}$ from the values of feasible overlap variables $\mathbf{d}$. The full pseudocode of the reconstruction algorithm is listed in Algorithm 1.

The algorithm first assigns the start times to the jobs that lies on a boundary of at least two metering intervals. For these jobs, the start time is unambiguous since it equals $i^s \cdot D - d_{j,i^s}$, where I_{i^s} is the first metering interval in which the job has non-zero overlap.

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

Function `ReconstructStartTimes`($\mathbf{d}$)***:***
  foreach $I_i \in I$ ***do***
    earliestStartTimes$[i] \leftarrow (i-1) \cdot D$
  end
  foreach $\{J_j \in \mathcal{J} : |\{I_i \in I : d_{j,i} > 0\}| \geq 2\}$ ***do***
    $i^s \leftarrow \arg\min_{i=1,\ldots,i^{max}} d_{j,i} > 0$
    $i^c \leftarrow \arg\max_{i=1,\ldots,i^{max}} d_{j,i} > 0$
    $s_j \leftarrow i^s \cdot D - d_{j,i^s}$
    earliestStartTimes$[i^c] \leftarrow$
      earliestStartTimes$[i^c] + d_{j,i^c}$
  end
  foreach $\{J_j \in \mathcal{J} : |\{I_i \in I : d_{j,i} > 0\}| = 1\}$ ***do***
    $I_{i^s} \leftarrow I_i \in I : d_{j,i} > 0$
    $s_j \leftarrow$ earliestStartTimes$[i^s]$
    earliestStartTimes$[i^s] \leftarrow$
      earliestStartTimes$[i^s] + p_j$
  end
  return $\mathbf{s}$

Algorithm 1: Reconstruction of start times $\mathbf{s}$ from overlap variables $\mathbf{d}$.

3.2.3 Iterative Algorithm

Since the implicit MILP model only optimizes the makespan in $I_{i^{\max}}$, the optimal solution found by the implicit model might be suboptimal for the original problem stated in Section 2. The idea of the iterative algorithm is to progressively decrease $i^{\max}$ and re-solve the implicit model until an optimal solution to the original problem is found. The full pseudocode of the iterative algorithm is listed in Algorithm 2.

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

To determine the optimality of a feasible solution, the algorithm tests its objective value Z. A non-zero objective value means that the found solution is optimal to the original problem since makespan is greater than $(i^{\max}-1)\cdot D$. On the other hand, if the objective is zero, a shorter solution ending in the previous metering interval might exist to the original problem. Therefore, $i^{\max}$ is decreased by one, the implicit model is re-solved with the smaller horizon, and the tests on the objective value are repeated.

Notice that in every iteration the previously found start times are passed to `SolveImplicitMILP`, which are used as an initial solution (by transformation to the overlap variables) to reduce the running time of the implicit model.

```
Function IterativeImplicitMILP():
    i^max ← H/D
    s, Z ← SolveImplicitMILP(i^max, ∅)
    if s = ∅ then
        /* The original problem is
           infeasible.                */
        return ∅
    end
    while Z = 0 do
        i^max ← i^max − 1
        s, Z ← SolveImplicitMILP(i^max, s)
    end
    /* Found optimal solution to the
       original problem.              */
    return s
```

Algorithm 2: MILP-based iterative algorithm.

4 EXPERIMENTS

In the experiments, we compare the presented approaches from Section 3 w.r.t. finding the optimal solutions to a given set of generated instances and proving their optimality. Although in practice it is often necessary to find high-quality solutions in short time, proving that a solution is also optimal is valuable information. The aim of the experiments is to determine, how two different parameters affect the solving time: (i) number of spannable jobs in an instance and (ii) the energy limit tightness w.r.t. the power consumption of the jobs. In addition to the proposed experiments we also compare the solution approaches w.r.t. the quality of the found solutions (which can be sub-optimal).

For the comparison with the existing literature, we include in the experiments a MILP model that was adapted from a similar problem (Haït and Artigues, 2011). The model is described in Appendix 5.2, and we will refer to this model as Disjunctive MILP.

All experiments were executed on 2x Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz with 188GB of RAM (16 cores in total). For solving the MILP and CP models, we used Gurobi 8 and IBM CP Optimizer 12.8, respectively. The multi-threading settings for each solver was set to default, i.e., the solvers can use all the available cores for computation.

The source codes of the methods and the generated instances are publicly available at `https://github.com/CTU-IIG/EnergyLimitsScheduling` and `https://github.com/CTU-IIG/EnergyLimitsSchedulingDatasets`, respectively.

4.1 Instances

The instances for the experiments were generated as follows. The energy limit and the length of the metering intervals were fixed to 1000 and 15, respectively. To explore how the solving time is affected by the number of spannable jobs in an instance, we use parameter α

$$\alpha = \frac{|\mathcal{J}^{\text{span}}|}{|\mathcal{J}|}. \tag{20}$$

The processing time of each spannable job is sampled from discrete uniform distribution $\mathcal{U}\{D+1,60\}$ while the processing time of the non-spannable jobs is sampled from discrete uniform distribution $\mathcal{U}\{1,D\}$.

Another evaluated parameter in the experiments is β that controls the tightness of the energy limit. Given fixed β, the power consumption of each job is then sampled from continuous uniform distribution

$$\mathcal{U}\left(\beta\cdot\frac{E^{\max}}{m\cdot D}, 2\cdot\frac{E^{\max}}{m\cdot D}\right). \tag{21}$$

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

To distribute the jobs between the machines, each job is assigned to a random machine with uniform distribution. Finally, the length of the scheduling horizon is set to the sum of all jobs' processing times in an instance. The analysis of the results shows that this horizon is large enough to guarantee the existence of a feasible solution.

For each $n \in \{10,20,30\}$, $m \in \{2,4\}$, $\alpha \in \{0.1,0.25,0.5,0.75,0.9\}$ and $\beta \in \{0.8,1.0,1.2,1.4,1.6\}$ we randomly generated 10 instances using the scheme described above. Therefore, the generated dataset has 1500 instances in total.

4.2 Results

The results section is divided into two parts. The first part analyzes the ability of the proposed approaches to prove the solutions' optimality based on the examined parameters n, m, α, β. The second part compares the solution approaches w.r.t. the quality of the found solutions. In the subsequent text, the following abbreviations of the solution approaches are used: (i) CP: CP model (see Section 3.1), (ii) MILP-IMP: MILP-based iterative algorithm (see Section 3.2) and (iii) MILP-DIS: Disjunctive MILP model (see Appendix 5.2).

4.2.1 Finding Optimal Solutions and Proving Their Optimality

The results are shown in Tables 1 to 3; each of the tables provides a different view of the same results. Each row of the table shows the number of optimally solved instances by each approach. For example, in Table 2 the row denoted as $\alpha = 0.25$ shows the results for instances with $n \in \{10,20,30\}$, $m \in \{2,4\}$, $\alpha = 0.25$ and $\beta \in \{0.8,1.0,1.2,1.4,1.6\}$. To highlight the best method in each row, the best values are shown in bold. The time-limit given to each method for each instance was 300 s.

Table 1 shows, how the methods scale with the increasing size of the instances. Except of the smallest instances with $n = 10, m = 2$, MILP-IMP model either outperforms or is at least as good as the other approaches. For example, 53.2% of the instances with $n = 30, m = 4$ were solved optimally by MILP-IMP, whereas the CP approach solved 34.4% and MILP-DIS only 14.4% of the instances.

The next Table 2 compares the approaches according to the increasing spannable ratio α. We see that the performance of the implicit MILP model decreases with increasing α. The implicit MILP model outperforms the CP approach when the number of shorter jobs is higher or the same; on the other hand, CP is better for the higher spannable ratio. With exception of $\alpha = 0.9$, MILP-DIS is outperformed by MILP-IMP. The efficiency of MILP-IMP for the smaller spannable ratio can be explained by breaking of symmetries, i.e, the solver does not have to sequence the non-spannable jobs that are wholly processed within one metering interval.

Table 1: Number of proved optimal solutions out of 250 with varying n, m.

n	m	CP	MILP-IMP	MILP-DIS
10	2	**247**	207	234
10	4	**250**	**250**	**250**
20	2	14	**46**	16
20	4	127	**177**	144
30	2	3	**11**	0
30	4	86	**133**	36

Table 2: Number of proved optimal solutions out of 300 with varying α.

α	CP	MILP-IMP	MILP-DIS
0.1	154	**239**	187
0.25	165	**207**	157
0.5	151	**163**	134
0.75	**134**	128	113
0.9	**123**	87	89

The following Table 3 illustrates, how the tightness of the energy limit β affects the performance. We see that MILP-IMP outperforms other approaches. From the results we may infer that the tighter limit leads to harder instances, which corresponds with the intuition since our scheduling problem without the energy limit is trivially solvable (any left-shifted schedule is optimal).

4.2.2 Comparison of the Quality of the Found Solutions

In addition to the previous results, we also examined the total makespan achieved by each method within the time limit for varying n, m, see Table 4. Instances, for which at least one method did not find any solution, are not included (there are 5 such instances, in all cases a solution was not found by MILP-DIS method). The table reveals that CP outperforms all the other approaches, which is not surprising since IBM CP Optimizer is primarily designed to find good solutions. With the exception of the instances with

Table 3: Number of proved optimal solutions out of 300 with varying β.

β	CP	MILP-IMP	MILP-DIS
0.8	185	**198**	150
1.0	159	**181**	148
1.2	144	**162**	136
1.4	125	**152**	127
1.6	114	**131**	119

Table 4: Total makespan with varying n, m.

n	m	CP	MILP-IMP	MILP-DIS
10	2	**46419**	46451	46425
10	4	**26028**	**26028**	**26028**
20	2	**94673**	95736	97456
20	4	**49206**	49284	49394
30	2	**133389**	136744	140814
30	4	**69938**	70250	71033

$n = 10, m = 2$, MILP-IMP has smaller or the same total makespan as MILP-DIS.

5 CONCLUSION

Managing the production is becoming increasingly complex without computer-aided systems. On the other hand, traditional scheduling constraints and objectives are not enough to design realistic production schedules. In this work, we explored a scheduling problem with a practical production constraint of satisfying the contracted energy consumption limits. This problem emerges in energy-demanding production processes, such as glass tempering or steel hardening, in which a material is heated in furnaces to high temperature. The heating in the furnaces has to be planned carefully; otherwise, the resulting energy spikes lead to a violation of the energy limits.

We propose two exact algorithms for the studied scheduling problem with dedicated machines: (i) a Constraint Programming model and (ii) a Mixed Integer Linear Programming based iterative algorithm. The iterative algorithm exploits the fact that the start times do not need to be modeled explicitly, which allows us to efficiently solve instances having a higher number of shorter jobs. To compare the algorithms with the approaches from literature, we adapted an existing Mixed Integer Linear Programming model for a similar problem (Haït and Artigues, 2011; Haït and Artigues, 2011). To summarize the experimental results on a dataset consisting of instances with 10 to 30 jobs, our best method proves the optimality for 54.9% instances, whereas the Mixed Integer Linear Programming model inspired by the existing literature proves the optimality for 45.3% instances.

In the future, our aim is to extend our iterative algorithm to the job shop scheduling problem with energy consumption limit, where the research challenge is handling of the precedences between the operations of the same job.

ACKNOWLEDGEMENTS

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

REFERENCES

Bruzzone, A., Anghinolfi, D., Paolucci, M., and Tonelli, F. (2012). Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops. *CIRP Annals - Manufacturing Technology*, 61(1):459–462.

Fang, K., Uhan, N. A., Zhao, F., and Sutherland, J. W. (2013). Flow shop scheduling with peak power consumption constraints. *Annals of Operations Research*, 206(1):115–145.

Gajic, D., Hadera, H., Onofri, L., Harjunkoski, I., and Gennaro, S. D. (2017). Implementation of an integrated production and electricity optimization system in melt shop. *Journal of Cleaner Production*, 155:39 – 46. Sustainable Development of Energy, Water and Environmental Systems.

Gurobi (2018). Constraints. `http://www.gurobi.com/documentation/8.0/refman/constraints.html`. Accessed September 18, 2018.

Hadera, H., Harjunkoski, I., Sand, G., Grossmann, I. E., and Engell, S. (2015). Optimization of steel production scheduling with complex time-sensitive electricity cost. *Computers & Chemical Engineering*, 76:117–136.

Haït, A. and Artigues, C. (2011). A hybrid CP/MILP method for scheduling with energy costs. *European Journal of Industrial Engineering*, 5(4):471–489.

Haït, A. and Artigues, C. (2011). On electrical load tracking scheduling for a steel plant. *Computers & Chemical Engineering*, 35(12):3044–3047.

Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2018). IBM ILOG CP optimizer for scheduling. *Constraints*, 23(2):210–250.

Masmoudi, O., Yalaoui, A., Ouazene, Y., and Chehade, H. (2017). Lot-sizing in a multi-stage flow line production system with energy consideration. *International Journal of Production Research*, 55(6):1640–1663.

Merkert, L., Harjunkoski, I., Isaksson, A., Säynevirta, S., Saarela, A., and Sand, G. (2015). Scheduling and energy – industrial challenges and opportunities. *Computers & Chemical Engineering*, 72(0):183 – 198.

Módos, I., Šůcha, P., and Hanzálek, Z. (2017). Algorithms for robust production scheduling with energy consumption limits. *Computers & Industrial Engineering*, 112:391 – 408.

Nolde, K. and Morari, M. (2010). Electrical load tracking scheduling of a steel plant. *Computers & Chemical Engineering*, 34(11):1899–1903.

Rapine, C., Goisque, G., and Akbalik, A. (2018). Energy-aware lot sizing problem: Complexity analysis and exact algorithms. *International Journal of Production Economics*, 203:254 – 263.

Zhao, S., Grossmann, I. E., and Tang, L. (2018). Integrated scheduling of rolling sector in steel production with consideration of energy consumption under time-of-use electricity prices. *Computers & Chemical Engineering*, 111:55 – 65.

APPENDIX

5.1 Proofs

Observation 1. *The maximum number of consecutive metering intervals that can have non-zero overlap with job J_j for any feasible start times* $\mathbf{s}$ *is*

$$\Omega_j = \left\lceil \frac{p_j}{D} \right\rceil + 1 \,. \tag{8}$$

Proof. Let I_{i^s} be the metering interval, where the job starts and let I_{i^c} be the metering interval, where the job completes; thus $d_{j,i^s} \in \mathbb{N}_{>0}, d_{j,i^c} \in \mathbb{N}_{>0}$. Consider the following two cases

1. $I_{i^s} = I_{i^c}$: This implies that the job has non-zero overlap with only one metering interval in $\mathbf{s}$. Since $\Omega_j \geq 1$, the lemma holds.
2. $I_{i^s} < I_{i^c}$: Since the job is processed without preemption, the job completely fills up each metering interval between I_{i^s}, I_{i^c}, i.e., the overlap length of the job with each metering interval between I_{i^s}, I_{i^c} is D. The number of such metering intervals is $q = i^c - i^s$. The processing time can be then written as
$$p_j = q \cdot D + d_{j,i^s} + d_{j,i^c} \,. \tag{22}$$
To finish our proof, we need to show that
$$\Omega_j \geq q + 2 \,, \tag{23}$$
i.e., the number of metering intervals with non-zero overlap in $\mathbf{s}$ is at most Ω_j.
Notice that from the definition of the ceiling function it holds that
$$\Omega_j = \left\lceil \frac{p_j}{D} \right\rceil + 1 = \underbrace{\frac{p_j}{D} + \varepsilon}_{\in \mathbb{N}_{>0}} + 1 \tag{24}$$
where $\varepsilon \in [0,1)$. Therefore,
$$\begin{aligned} \Omega_j &= \frac{p_j}{D} + \varepsilon + 1 \\ &= q + \underbrace{\frac{d_{j,i^s}}{D} + \frac{d_{j,i^c}}{D} + \varepsilon}_{\in \mathbb{N}_{>0}} + 1 \\ &\geq q + 2 \,. \end{aligned}$$
The last inequality follows from $q \in \mathbb{N}_{\geq 0}, \Omega_j \in \mathbb{N}_{>0}, d_{j,i^s} \in \mathbb{N}_{>0}, d_{j,i^c} \in \mathbb{N}_{>0}$.

□

Lemma 1. *Let $J_{j_1}, J_{j_2} \in \mathcal{J}$ be two jobs such that $j_1 \neq j_2, \mu_{j_1} = \mu_{j_2}, p_{j_1} \geq 2, p_{j_2} \geq 2$ and $p_{j_1} + p_{j_2} \geq 2 \cdot D + 1$. Let $\mathbf{d} = (d_{j,i})_{J_j \in \mathcal{J}, I_i \in I}$ be values for some feasible solution to the MILP model from Section 3.2. Then Eq.* (19) *holds for J_{j_1}, J_{j_2} in any metering interval $I_i \in I \setminus \{I_{i^{max}}\}$.*

Proof. By contradiction. Assume that $I_i \in I \setminus \{I_{i^{\max}}\}$ is a metering interval such that Eq. (19) is violated, i.e., $d_{j_1,i} \geq 1, d_{j_1,i+1} \geq 1, d_{j_2,i} \geq 1, d_{j_2,i+1} \geq 1$. Since Eq. (11) holds, we know that

$$d_{j_1,i} + d_{j_2,i} \leq D \tag{25}$$
$$d_{j_1,i+1} + d_{j_2,i+1} \leq D, \tag{26}$$

thus

$$d_{j_1,i} + d_{j_1,i+1} + d_{j_2,i} + d_{j_2,i+1} \leq 2 \cdot D. \tag{27}$$

For $J_j \in \{J_{j_1}, J_{j_2}\}$, one of the following two cases are possible

1. $d_{j,i} + d_{j,i+1} = p_j$
2. $d_{j,i} + d_{j,i+1} < p_j$. This means that the job has non-zero overlap with either I_{i-1} or I_{i+2}. Due to Eq. (17), either $d_{j,i}$ or $d_{j,i+1}$ is pushed to be at least D, i.e., either $d_{j,i} \geq D, d_{j,i+1} \geq 1$ or $d_{j,i} \geq 1, d_{j,i+1} \geq D$.

The proof is now split according to combinations of those cases

- Case 1 holds for J_{j_1} and Case 1 holds for J_{j_2}: By substitution we get
$$p_{j_1} + p_{j_2} \leq 2 \cdot D, \tag{28}$$
which is a contradiction with our initial assumption $p_{j_1} + p_{j_2} \geq 2 \cdot D + 1$.
- Case 2 holds for J_{j_1} and Case 2 holds for J_{j_2}: By substitution we get
$$D + 1 + D + 1 \leq 2 \cdot D, \tag{29}$$
which is an obvious contradiction.
- Case 1 holds for J_{j_1} and Case 2 holds for J_{j_2}: W.l.o.g., assume that $d_{j_1,i} \geq D$. Since $d_{j_2,i} \geq 1$, we get $d_{j_1,i} + d_{j_2,i} \geq D + 1$, which is a contradiction with $d_{j_1,i} + d_{j_2,i} \leq D$.
- Case 2 holds for J_{j_1} and Case 1 holds for J_{j_2}: Proof is analogous to the previous one.

□

5.2 Disjunctive MILP Model

The disjunctive MILP model uses the formulation for computing the jobs' overlaps with the metering intervals that was presented in (Haït and Artigues, 2011; Haït and Artigues, 2011). We adapted the model proposed in (Haït and Artigues, 2011; Haït and Artigues, 2011) to the parallel machine scheduling with energy consumption limits and makespan as an objective.

The disjunctive MILP model uses the following variables: (i) $C_{\max} \in \mathbb{N}$ is the makespan of the schedule, (ii) $s_j \in \mathbb{N}_{\geq 0}$ denotes the start time of job J_j, (iii) $x_{j,j'} \in \{0,1\}$ represents whether job J_j is scheduled before job $J_{j'}$, (iv) $d_{j,i} \in \mathbb{N}_{\geq 0}$ expresses the length of the overlap between job J_j and metering interval I_i, (v) $z^{\mathrm{s}}_{j,i} \in \{0,1\}$ denotes whether $s_j \in [0, i \cdot D - 1]$, and (vi) $z^{\mathrm{c}}_{j,i} \in \{0,1\}$ denotes whether $s_j + p_j \in [0, (i-1) \cdot D]$. The complete model follows

$$\min \quad C_{\max} \tag{30}$$
$$C_{\max} \geq s_j + p_j, \quad \forall J_j \in \mathcal{J} \tag{31}$$
$$s_j + p_j \leq s_{j'} + M \cdot (1 - x_{j,j'}), \quad \forall M_k \in \mathcal{M}, \forall J_j, J_{j'} \in \mathcal{J}_k : j < j' \tag{32}$$
$$s_{j'} + p_{j'} \leq s_j + M \cdot x_{j,j'}, \quad \forall M_k \in \mathcal{M}, \forall J_j, J_{j'} \in \mathcal{J}_k : j < j' \tag{33}$$
$$s_j \geq i \cdot D \cdot (1 - z^{\mathrm{s}}_{j,i}), \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \setminus \{I_{\frac{H}{D}}\} \tag{34}$$
$$s_j \leq i \cdot D - 1 + M \cdot (1 - z^{\mathrm{s}}_{j,i}), \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \setminus \{I_{\frac{H}{D}}\} \tag{35}$$
$$z^{\mathrm{s}}_{j,i+1} \geq z^{\mathrm{s}}_{j,i}, \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \setminus \{I_{\frac{H}{D}}\} \tag{36}$$
$$z^{\mathrm{s}}_{j,\frac{H}{D}} = 1, \quad \forall J_j \in \mathcal{J} \tag{37}$$
$$s_j + p_j \geq (i-1) \cdot D \cdot (1 - z^{\mathrm{c}}_{j,i}) + 1, \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \tag{38}$$
$$s_j + p_j \leq (i-1) \cdot D + M \cdot (1 - z^{\mathrm{c}}_{j,i}), \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \tag{39}$$
$$z^{\mathrm{c}}_{j,i+1} \geq z^{\mathrm{c}}_{j,i}, \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \setminus \{I_{\frac{H}{D}}\} \tag{40}$$
$$d_{j,i} \leq D \cdot (z^{\mathrm{s}}_{j,i} - z^{\mathrm{c}}_{j,i}), \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \tag{41}$$
$$\sum_{I_i \in I} d_{j,i} = p_j, \quad \forall J_j \in \mathcal{J} \tag{42}$$
$$d_{j,i} \geq D \cdot (z^{\mathrm{s}}_{j,i-1} - z^{\mathrm{c}}_{j,i+1}), \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \setminus \{I_1, I_{\frac{H}{D}}\} \tag{43}$$
$$d_{j,i} \geq i \cdot D \cdot (1 - z^{\mathrm{s}}_{j,i-1}) - s_j - D \cdot z^{\mathrm{c}}_{j,i+1}, \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \setminus \{I_1, I_{\frac{H}{D}}\} \tag{44}$$
$$d_{j,i} \geq s_j + p_j - i \cdot D + D \cdot z^{\mathrm{s}}_{j,i-1} - M \cdot (1 - z^{\mathrm{c}}_{j,i+1}), \quad \forall J_j \in \mathcal{J}, \forall I_i \in I \setminus \{I_1, I_{\frac{H}{D}}\} \tag{45}$$
$$d_{j,1} \geq D \cdot z^{\mathrm{s}}_{j,1} - s_j - D \cdot z^{\mathrm{c}}_{j,2}, \quad \forall J_j \in \mathcal{J} \tag{46}$$
$$d_{j,\frac{H}{D}} \geq s_j + p_j - H + D \cdot z^{\mathrm{s}}_{j,\frac{H}{D}-1}, \quad \forall J_j \in \mathcal{J} \tag{47}$$
$$\sum_{J_j \in \mathcal{J}} d_{j,i} \cdot P_j \leq E^{\max}, \quad \forall I_i \in I \tag{48}$$

The objective (30) minimizes the makespan, which is the maximum of the completion times of the jobs, see constraint (31). The non-overlapping of the jobs is ensured by constraints (32) and (33). Constraints (34)-(47) models the overlaps of the jobs with the metering intervals; see (Haït and Artigues, 2011)

for details. The last constraint (48) enforces the energy limit.