Constraints
https://doi.org/10.1007/s10601-020-09317-y

Power of pre-processing: production scheduling with variable energy pricing and power-saving states

Ondřej Benedikt[1,2] · István Módos[1,2] · Zdeněk Hanzálek[2]

Accepted: 20 October 2020Published online: 16 November 2020

Abstract

This paper addresses a single machine scheduling problem with non-preemptive jobs to minimize the total electricity cost. Two latest trends in the area of the energy-aware scheduling are considered, namely the variable energy pricing and the power-saving states of a machine. Scheduling of the jobs and the machine states are considered jointly to achieve the highest possible savings. Although this problem has been previously addressed in the literature, the reported results of the state-of-the-art method show that the optimal solutions can be found only for instances with up to 35 jobs and 209 intervals within 3 hours of computation. We propose an elegant pre-processing technique called SPACES for computing the optimal switching of the machine states with respect to the energy costs. The optimal switchings are associated with the shortest paths in an interval-state graph that describes all possible transitions between the machine states in time. This idea allows us to implement efficient integer linear programming and constraint programming models of the problem while preserving the optimality. The efficiency of the models lies in the simplification of the optimal switching representation. The results of the experiments show that our approach outperforms the existing state-of-the-art exact method. On a set of benchmark instances with varying sizes and different state transition graphs, the proposed approach finds the optimal solutions even for the large instances with up to 190 jobs and 1277 intervals within an hour of computation.

Keywords Single machine production scheduling · Machine states · Variable energy costs · Total energy cost minimization

This article belongs to the Topical Collection: *Special Issue on Constraint Programming, Artificial Intelligence, and Operations Research*
Guest Editors: Emmanuel Hebrard and Nysret Musliu

✉ Ondřej Benedikt
ondrej.benedikt@cvut.cz

István Módos
istvan.modos@cvut.cz

Zdeněk Hanzálek
zdenek.hanzalek@cvut.cz

[1] Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic

[2] Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Prague, Czech Republic

1 Introduction

Energy-efficient scheduling has been attracting a considerable amount of attention lately, as reported by both Gahm et al. [6] and Gao et al. [7]. The trend is most likely to continue in the future since the energy-efficient scheduling helps to achieve sustainability of the production by both decreasing the production cost and minimizing its environmental impact. Gahm et al. [6] identified promising approaches to the energy-aware scheduling, including, among others, (i) the optimization of the energy demand by considering the power-saving states of the machines, and (ii) the participation in demand response programs, which are used by the electric utilities to reward the energy consumers for shifting their energy consumption to *off-peak* intervals [14].

In this work, we study a single machine scheduling problem to minimize the total energy cost (TEC) of the production. We consider both the power-saving states of the machine and the time-of-use (TOU) pricing. The TOU pricing is one of the demand response programs, in which the electricity price may differ every hour. The scheduling problems with TOU pricing have been extensively addressed in the literature [5, 9, 10].

Considering the power-saving states of the machine, Mouzon et al. [15] identified that a significant energy cost reduction could be attained. However, the switchings between the machine states need to be planned carefully because of their non-negligible energy costs and transition times.

The integration of the power-saving states and the TOU pricing was initially proposed by Shrouf et al. [17], who designed an integer linear programming (ILP) model for the single machine problem with the fixed order of the jobs. However, it was proven by Aghelinejad et al. [2] that the problem with the fixed order of the jobs can be solved in polynomial time. Aghelinejad et al. [1] improved and generalized the existing ILP model by removing the restriction on the fixed order of the jobs, in which case the problem is $\mathcal{NP}$-hard [2]. However, in both works of Aghelinejad et al. [1] and Shrouf et al. [17], only small instances of the problem have been solved optimally.

In this paper, we study a single machine problem introduced by Shrouf et al. [17] and further studied by Aghelinejad et al. [1], and describe a novel pre-processing technique to solve it efficiently. Our pre-processing technique pre-computes the optimal switching behavior in time w.r.t. the energy costs. The pre-computed costs of the optimal switchings allow us to design exact ILP and constraint programming (CP) models. In contrast, the ILP model proposed by Aghelinejad et al. [1] explicitly formulates the transition behavior of the machine, which needs to be optimized jointly with the scheduling of the jobs. As shown by the experiments, our approach outperforms the existing ILP model [1], which is, to the best of our knowledge, the state-of-the-art among the exact methods for this problem. Our ILP model can solve all benchmark instances with up to 190 jobs and 1277 pricing intervals within the time limit. On the other hand, the state-of-the-art ILP model from the literature scales only up to instances with 60 jobs and 316 intervals. This shows that our approach can tackle production-size problems. For example, creating a schedule of one workweek (5 days) with start times granularity of 5 minutes requires 1440 intervals.

2 Problem statement

Let $\mathcal{I} = \{I_1, I_2, \ldots, I_h\}$ be a set of *intervals*, which partition the scheduling horizon. The *energy costs* for the intervals are given by the vector $\boldsymbol{c} = (c_1, c_2, \ldots, c_h)$, where $c_i \in \mathbb{Z}_{\geq 0}$ is the energy (electricity) cost associated with interval I_i. It is assumed that every interval is one time unit long, i.e., $I_1 = [0, 1)$, $I_2 = [1, 2), \ldots, I_h = [h - 1, h)$. Note that the physical representation of the time unit length can be different depending on the required granularity of the scheduling horizon.

Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of *jobs*, which must be scheduled on a single machine that is available throughout the whole scheduling horizon; we assume that $n \geq 1$. Each job J_j is characterized by its *processing time* $p_j \in \mathbb{Z}_{>0}$, given in the number of intervals. Scheduling of the jobs is non-preemptive, and the machine can process at most one job at a time. All the jobs are available at the beginning of the scheduling horizon.

During each interval, the machine is operating in one of its *states* $s \in \mathcal{S}$ or transits from one state to another. Let us denote the *transition time function* by $T : \mathcal{S} \times \mathcal{S} \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$, and the *transition power function* by $P : \mathcal{S} \times \mathcal{S} \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$. The direct transition from state s to state $s' \neq s$ lasts $T[s, s']$ intervals and has power consumption $P[s, s']$, which is the constant rate of the consumed energy at every time unit. The value ∞ means that the direct transition does not exist. For ease of notation, we set $T[s, s] = 1$ for each $s \in \mathcal{S}$, by which we represent that the machine is remaining in the state s for the duration of one interval. Correspondingly, we assume that $P[s, s]$ denotes the power consumption of the machine while staying in state s for the duration of one interval.

Note that the transition time/power functions are general enough to represent many kinds of machines used throughout the literature [1, 3, 15, 17]. Often, the machine states and transitions are represented by a *transition graph* instead of the transition time/power function. Then, the graph nodes correspond to the machine states, while the edges represent the allowed direct transitions between the states. The edges are labeled by the corresponding values of the transition time/power functions. An example of a transition graph is shown in Fig. 1.

During the first and the last interval, the machine is assumed to be in off state `off` $\in \mathcal{S}$. Besides, the machine has a single processing state, `proc` $\in \mathcal{S}$, which must be active during the processing of the jobs. Due to the transition from/to the initial/last `off` state, the machine cannot be in `proc` state during the early/late intervals. Hence, we denote the *earliest* and the *latest* interval during which the machine can be in `proc` state by I_{earl} and I_{late}, respectively.

A *solution* is a pair $(\boldsymbol{\sigma}, \boldsymbol{\Omega})$, where $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathbb{Z}^n_{\geq 0}$ is the vector denoting the start time of the jobs, and $\boldsymbol{\Omega} = (\Omega_1, \Omega_2, \ldots, \Omega_h) \in (\mathcal{S} \times \mathcal{S})^h$ represents the active state or transition in each interval.

The solution is feasible if the following four conditions are satisfied.

1. the machine processes at most one job at a time;

$s \backslash s'$	proc	idle	off
proc	4	0	1
idle	0	2	∞
off	5	∞	0

$P[s, s']$

$s \backslash s'$	proc	idle	off
proc	1	0	1
idle	0	1	∞
off	2	∞	1

$T[s, s']$

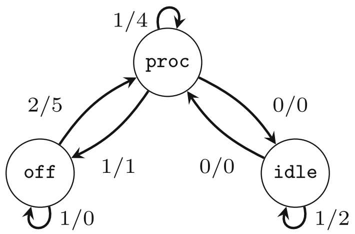


Fig. 1 Parameters of the transition power function $P[s, s']$ and the transition time function $T[s, s']$, and the corresponding transition graph, where every edge from s to s' is labeled by $T[s, s']/P[s, s']$

2. the jobs are processed when the machine is in $\texttt{proc}$ state, i.e., $\forall J_j \in \mathcal{J}\ \forall i \in \{\sigma_j + 1, \ldots, \sigma_j + p_j\} : \Omega_i = (\texttt{proc}, \texttt{proc})$, where $\{a, \ldots, b\}$ is $\{a, a+1, \ldots, b\}$;
3. the machine is in $\texttt{off}$ state during the first and the last interval, i.e., $\Omega_1 = (\texttt{off}, \texttt{off})$, and $\Omega_h = (\texttt{off}, \texttt{off})$;
4. all transitions are valid with respect to the transition time function, i.e., $\forall i \in \{1, 2, \ldots, h-1\}$ such that $\Omega_i = (s, s')$, $\Omega_{i+1} = (s'', s''')$, it holds that
   (a) Ω_i and Ω_{i+1} encode only feasible states/transitions: $0 < T[s, s'] < \infty$, $0 < T[s'', s'''] < \infty$,
   (b) only feasible zero-time transitions are allowed between Ω_i and Ω_{i+1}: either $s' = s''$ or there exists a sequence of states $(s', s_1, \ldots, s_k, s'')$ such that $T[s', s_1] = T[s_1, s_2] = \cdots = T[s_{k-1}, s_k] = T[s_k, s''] = 0$,
   (c) the non-zero-time transitions respect the transition time function: if $s'' \neq s'''$ and $\Omega_i \neq \Omega_{i+1}$ then there exists Ω_ℓ with ℓ being the smallest index larger than $(i+1)$ such that $\Omega_\ell \neq \Omega_{i+1}$, and it holds that $\ell - i - 1 = T[s'', s''']$.

The total energy cost (TEC) of solution $(\boldsymbol{\sigma}, \boldsymbol{\Omega})$ is

$$\sum_{I_i \in \mathcal{I}} c_i \cdot P[\Omega_i], \tag{1}$$

where $P[\Omega_i]$ represents $P[s, s']$ for $\Omega_i = (s, s')$. The goal of the scheduling problem is to find a feasible solution minimizing the total energy cost (1).

The above-defined problem was introduced by Shrouf et al. [17] and is denoted in standard Graham's notation as $1, \mathrm{TOU}|\,\mathrm{states}\,|\,\mathrm{TEC}$. The problem was shown to be $\mathcal{NP}$-hard [2].

Example Here, we present a small example to illustrate the proposed notation. Let us consider a scheduling horizon consisting of 16 intervals, $\mathcal{I} = \{I_1, \ldots, I_{16}\}$, and the associated energy costs $\boldsymbol{c} = (2, 1, 2, 1, 6, 16, 14, 3, 2, 5, 3, 15, 3, 2, 1, 2)$. Let us have three jobs, $\mathcal{J} = \{J_1, J_2, J_3\}$ with processing times $p_1 = 2$, $p_2 = 1$, and $p_3 = 2$. Considering the machine states, we assume $\mathcal{S} = \{\texttt{proc}, \texttt{off}, \texttt{idle}\}$. The values of the transition time function and the transition power function are given in Fig. 1. For the given transition time function, we have $I_{\mathrm{earl}} = I_4$ and $I_{\mathrm{late}} = I_{14}$. Note that the same machine states and transitions were originally proposed by Shrouf et al. [17].

The optimal solution to the given instance is depicted in Fig. 2, where

$$\begin{aligned}
\boldsymbol{\sigma} &= (9, 3, 12), \text{ and} \\
\boldsymbol{\Omega} &= ((\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{proc}), (\texttt{off}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{off}), (\texttt{off}, \texttt{off}), \\
&\quad (\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{proc}), (\texttt{off}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{idle}, \texttt{idle}), \\
&\quad (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{proc}), (\texttt{proc}, \texttt{off}), (\texttt{off}, \texttt{off})).
\end{aligned}$$

The TEC of this optimal solution is equal to 133.

3 Solution approach

In this section, we first describe how to pre-compute the optimal switching behavior of the machine and the corresponding costs. Afterward, we design efficient ILP

Interval I_i	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
Energy cost c_i	2	1	2	1	6	16	14	3	2	5	3	15	3	2	1	2
Processing				4						20	12		12	8		
Idle												30				
Off	0					0	0									0
Turn-off ($\searrow$)					6										1	
Turn-on ($\nearrow$)		5	10					15	10							
Schedule	off	$\nearrow$		J_2	$\searrow$	off		$\nearrow$		J_1		idle	J_3		$\searrow$	off

Fig. 2 The optimal schedule for the example instance. Each cell, corresponding to interval I_i and a state/transition, contains the value $c_i \cdot P[\Omega_i]$. The sum over all these values gives the TEC equal to 133

(called ILP-SPACES) and CP models that integrate the pre-computed optimal switching costs.

3.1 Instance pre-processing: computation of the optimal switching

Given two states s, s' in which the machine is during two intervals $I_i, I_{i'}$ such that $i < i'$, the pre-processing computes the optimal transitions from (s, I_i) to $(s', I_{i'})$ over all possible states w.r.t. the energy cost. Formally, the pre-processing solves the following optimization problem

$$\min_{\Omega_{i+1},\Omega_{i+2},\ldots,\Omega_{i'-1}} \sum_{i''=i+1}^{i'-1} c_{i''} \cdot P[\Omega_{i''}]. \tag{2}$$

such that $((s, s), \Omega_{i+1}, \Omega_{i+2}, \ldots, \Omega_{i'-1}, (s', s'))$ are valid transitions w.r.t. to the transition time function. We call this an *optimal switching problem*. As an illustration, the cost of the optimal switching in Fig. 2 from $(\texttt{proc}, I_4)$ to $(\texttt{proc}, I_{10})$ equals to 31. Interestingly, the optimal switching problem can be solved in polynomial time by finding the shortest path in an *interval-state* graph, which is explained in the rest of this section.

The interval-state graph is defined by a triplet (V, E, w), where V is the set of *vertices*, E is the set of *edges* and $w : E \to \mathbb{Z}_{\geq 0}$ are the *weights* of the edges. The set of the vertices and edges of this graph are defined as follows:

$$V = \{v_{1,\texttt{off}}\} \cup \{v_{i,s} : I_i \in \mathcal{I} \setminus \{I_1\}, s \in \mathcal{S}\} \cup \{v_{h+1,\texttt{off}}\}, \tag{3}$$

$$\begin{aligned} E = \{&(v_{1,\texttt{off}}, v_{2,\texttt{off}})\} \\ &\cup \{(v_{i,s}, v_{i+T[s,s'],s'}) : s, s' \in \mathcal{S}, I_i \in \mathcal{I} \setminus \{I_1\}, \\ &\qquad T[s,s'] \neq \infty, (i-1) + T[s,s'] \leq h-1\} \\ &\cup \{(v_{h,\texttt{off}}, v_{h+1,\texttt{off}})\}. \end{aligned} \tag{4}$$

Informally, each vertex $v_{i,s} \in V$ represents that at the beginning of interval I_i the machine is in state s. Each edge $(v_{i,s}, v_{i',s'}) \in E$ corresponds to the direct transition from state s to state s' that lasts $T[s, s'] = (i' - i)$ intervals. The condition $(i - 1) + T[s, s'] \leq h - 1$ ensures, that only transitions completing at most at the beginning of interval I_h are present in the interval-state graph.

The edges are weighted by the total energy cost of the corresponding transition w.r.t. the costs of energy in intervals, i.e., *weight* of edge $(v_{i,s}, v_{i',s'}) \in E$ is defined as

$$w[v_{i,s}, v_{i',s'}] = \sum_{i''=i}^{i'-1} c_{i''} \cdot P[s, s'] \,. \tag{5}$$

Note that by the definition, the interval-state graph encodes all the feasible transitions between the machine states in time.

Returning to the optimal switching problem (2), the optimal transitions from (s, I_i) to $(s', I_{i'})$ w.r.t. the energy cost can be obtained by finding the shortest path from $v_{i+1,s}$ to $v_{i',s'}$ in the interval-state graph. We denote the cost of the optimal switching by function $l : V \times V \to \mathbb{Z}_{\geq 0}$.

Example continued Continuing with the Example, Fig. 3 shows the whole interval-state graph for the given instance. The green dashed path shows the optimal transition of the machine assuming that the machine is in `proc` state during intervals I_4 and I_{10}; at first the machine is turned off (during I_5), then it remains off (during intervals I_6 and I_7), and is turned on afterward (intervals I_8, I_9). In this case, $l[v_{5,\texttt{proc}}, v_{10,\texttt{proc}}] = 31$.

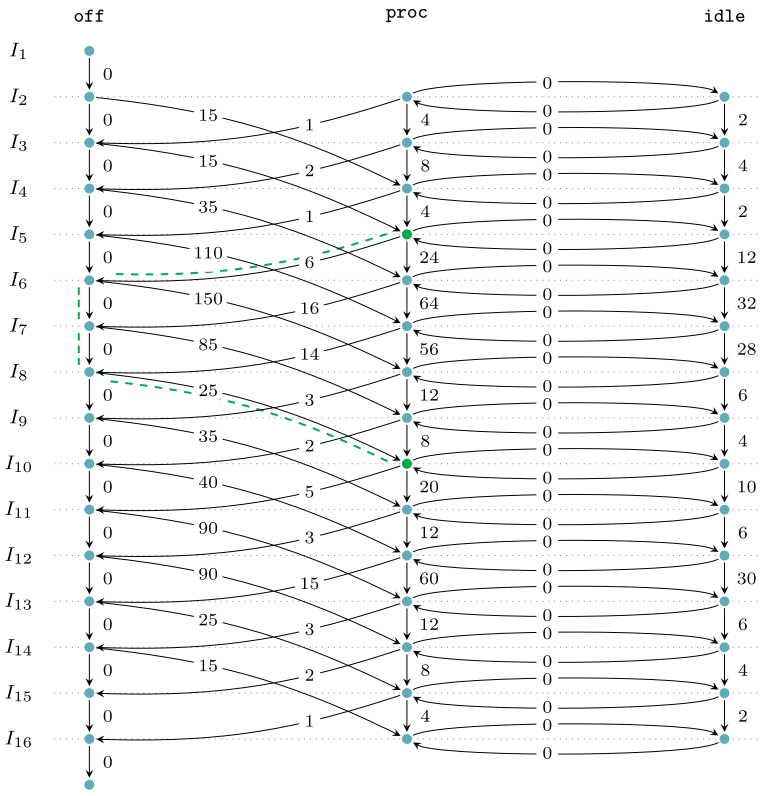


Fig. 3 Interval-state graph for the Example instance from Fig. 2 with highlighted optimal switching behavior from (`proc`, I_4) to (`proc`, I_{10})

The values of l can be computed using the Floyd-Warshall algorithm in $\mathcal{O}(h^3 \cdot |\mathcal{S}|^3)$ time. However, for the scheduling decisions, only some of the switchings are interesting. Since all the jobs need to be scheduled in the proc state, the optimal switchings have to be resolved only in the 'space', i.e., the sequence of intervals: (i) between the two consecutive intervals with proc; (ii) between the first off and the first proc; and (iii) the last proc and the last off. The cost of the switchings between $s, s' \in \{\texttt{off}, \texttt{proc}\}^2$ are recorded by function $c^* : \mathcal{I}^2 \rightarrow \mathbb{Z}_{\geq 0}$ defined as

$$c^*[i, i'] = \begin{cases} l[v_{i+1,\texttt{proc}}, v_{i',\texttt{proc}}] & i > 1, i' < h \qquad \text{case (i)} \\ l[v_{2,\texttt{off}}, v_{i',\texttt{proc}}] & i = 1, i' < h \qquad \text{case (ii)} \\ l[v_{i+1,\texttt{proc}}, v_{h,\texttt{off}}] & i > 1, i' = h \qquad \text{case (iii)} \end{cases} \tag{6}$$

for each $i < i'$. The vector of states corresponding to $c^*[i, i']$, i.e., the *optimal switching behavior* of the machine between i and i', is denoted by $\Omega^*(i, i')$. As an example, see the Fig. 2, where intervals $\{I_5, I_6, \ldots, I_9\}$ represent the space between two consecutive jobs J_2, J_1 with cost $c^*(4, 10) = l[v_{5,\texttt{proc}}, v_{10,\texttt{proc}}] = 31$. The optimal switching behavior is

$$\Omega^*[4, 10] = ((\texttt{proc}, \texttt{off}), (\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{off}), (\texttt{off}, \texttt{proc}), (\texttt{off}, \texttt{proc}))\,, \tag{7}$$

which is depicted by the green dashed path in Fig. 3.

Values of c^* can be computed efficiently using an algorithm that we call the *Shortest Path Algorithm for Cost Efficient Switchings* (SPACES). In every iteration $I_i \in \mathcal{I} \setminus \{I_h\}$, SPACES computes all values $c^*[i, i+1], c^*[i, i+2], \ldots, c^*[i, h]$ by finding the shortest paths from $v_{i+1,\texttt{proc}}$ (or $v_{2,\texttt{off}}$ if $i = 1$) to all other vertices in the interval-state graph. The shortest paths are obtained with Dijkstra algorithm that runs in $\mathcal{O}(|E| + |V| \cdot \log |V|)$ if implemented using the priority queues. Since the Dijkstra algorithm is started h times, the complexity of SPACES is

$$\mathcal{O}(h \cdot (|E| + |V| \cdot \log |V|)) = \mathcal{O}(h^2 \cdot |\mathcal{S}| \cdot (|\mathcal{S}| + \log h \cdot |\mathcal{S}|))\,. \tag{8}$$

To increase the performance further, iterations i can be computed in parallel since they are independent of each other. Moreover, the shortest paths between states could be cached, and might be re-used between the iterations of the algorithm. However, for the size of the benchmark instances considered in the experiments, the runtime of the SPACES without caching is negligible in comparison to the total solving time. Hence, we did not implement SPACES with the shortest paths' caching.

3.2 Integer linear programming model ILP-SPACES

In the ILP model proposed by Aghelinejad et al. [1], the state transition functions are explicitly encoded. In contrast, our ILP-SPACES model works only with the optimal switching costs pre-computed by the SPACES algorithm, thus encoding the transitions implicitly without sacrificing the optimality. The only task of the ILP solver is then to schedule the jobs, and select appropriate spaces in between, such that the TEC is minimized. Thus, the structure of our model is greatly simplified, with positive impact on its performance.

Formally, the variables used in the ILP-SPACES model are

- job start time $s_{j,i} \in \{0, 1\}$: equals 1 if job J_j starts at the beginning of interval I_i, otherwise 0;
- space activation $x_{i,i'} \in \{0, 1\}$: equals 1 if the machine undergoes the optimal switching defined by $\Omega^*(i, i')$, otherwise 0.

The complete model follows.

$$\min \sum_{\substack{I_i, I_{i'} \in \mathcal{I} \\ i < i'}} x_{i,i'} \cdot c^*[i, i'] + \sum_{\substack{J_j \in \mathcal{J} \\ I_i \in \mathcal{I}}} s_{j,i} \cdot c^{(\text{job})}_{j,i}, \tag{9}$$

$$\sum_{I_i \in \mathcal{I}} s_{j,i} = 1, \ \forall J_j \in \mathcal{J}, \tag{10}$$

$$s_{j,i} = 0, \ \forall J_j \in \mathcal{J}, \forall i \in \{1, \ldots, \text{earl} - 1\} \cup \{\text{late} - p_j + 2, \ldots, h\}, \tag{11}$$

$$\sum_{J_j \in \mathcal{J}} \sum_{i'=\max\{2, i-p_j+1\}}^{i} s_{j,i'} + \sum_{i'=1}^{i-1} \sum_{i''=i+1}^{h} x_{i',i''} = 1, \ \forall I_i \in \{I_2, I_3, \ldots, I_{h-1}\}. \tag{12}$$

The objective (9) minimizes the total energy cost, consisting of the optimal switching cost of the active spaces, and the cost of the jobs processing, where

$$c^{(\text{job})}_{j,i} = \sum_{i'=i}^{i+p_j-1} c_{i'} \cdot P[\texttt{proc}, \texttt{proc}] \tag{13}$$

for job $J_j \in \mathcal{J}$ and $i \in \{\text{earl}, \ldots, \text{late} - p_j + 1\}$.

Constraint (10) forces every job to be scheduled exactly once, and constraint (11) forbids the job to be scheduled before I_{earl} and after I_{late}. Finally, the last constraints (12) force the machine to be processing a job or to be undergoing some transition during every interval and forbid overlaps between them.

3.2.1 Search space reduction

Various methods can be employed to reduce the search space without sacrificing the optimality. One of such methods is pruning of the spaces variables that lead to infeasible solutions if activated.

The pruning works as follows. For each $I_i, I_{i'}$ such that $i < i'$, the available time for processing the jobs is computed for both left (before I_i) and right (after $I_{i'}$) part of the scheduling horizon, i.e., $i - \text{earl} + 1$ and $\text{late} - i' + 1$, respectively. Then, activating the switching behavior $\Omega^*[i, i']$ leads to an infeasible solution if one of the following *pruning conditions* holds

PC.1: The largest job can be fitted in neither part, i.e.,

$$\max_{J_j \in \mathcal{J}} p_j > i - \text{earl} + 1 \quad \wedge \quad \max_{J_j \in \mathcal{J}} p_j > \text{late} - i' + 1\,. \tag{14}$$

PC.2: The total available time for processing is less than the sum of all the processing times, i.e.,

$$(i - \text{earl} + 1) + (\text{late} - i' + 1) < \sum_{J_j \in \mathcal{J}} p_j\,. \tag{15}$$

If any of these conditions holds, the corresponding space variable $x_{i,i'}$ is not created in ILP-SPACES.

3.3 Constraint programming models

Thanks to the expressiveness of the CP, there are multiple possibilities on how to model our scheduling problem. Since the performance of each model is not easily predictable beforehand, we decided to try different combinations of the jobs' and spaces' modeling.

In the end, we selected the best model by performing a preliminary experiment, see Section 4.2. The best model is called Element-Free-SumLengths and is denoted as CP-SPACES.

In the following, we describe the CP-SPACES model formally. High-level description of all the tested CP models is in Appendix A. Also, all the source codes are publicly available at https://github.com/CTU-IIG/EnergyStatesAndCostsScheduling. In the text, we use the IBM CP formalism [12] for describing the models.

3.3.1 CP-SPACES model

The idea of the CP-SPACES model is similar to the ILP-SPACES, with the exception that the spaces are not fixed – they are allowed to 'float' within the scheduling horizon. In consequence, the spaces do not have fixed costs because the cost depends on the position of the space in the horizon and its length. In our CP-SPACES model, costs are formulated with an *Element* expression, which is integrated in the objective.

Variables Two types of interval variables are used in the CP-SPACES model.

To represent each job $J_j \in \mathcal{J}$ and the intervals in which the job is allocated, we use interval variable z_j of fixed length p_j. This interval variable represents the time interval in which the corresponding job is processed, i.e, the job starts at time $\text{StartOf}(z_j)$ and completes at time $\text{EndOf}(z_j)$.

The spaces in the schedule are modeled by using the optional interval variables, where $x_{\ell,k}$ represents the 'floating' spaces of fixed length ℓ. For each possible *length* $\ell \in \{1, 2, \ldots, h - 2 - \sum_{J_j \in \mathcal{J}} p_j\}$, we create $K(\ell) = \left\lfloor \frac{h-2-\sum_{J_j \in \mathcal{J}} p_j}{\ell} \right\rfloor$ variables that are indexed by $k \in \{1, 2 \ldots, K(\ell)\}$. Note that the number $K(\ell)$ gives the upper bound on the number of the spaces of length ℓ that may appear in a feasible schedule, while $\ell_{\max} = h - 2 - \sum_{J_j \in \mathcal{J}} p_j$ gives an upper bound on the space length.

Constraints Since the machine is assumed to be in `off` state during I_1 and I_h, the earliest and the latest interval during which a switching might occur is I_2 and I_{h-1}, respectively. Hence, starts (ends) of the spaces are restricted by

$$\left.\begin{aligned} \text{StartOf}(x_{\ell,k}) &\geq 1 \\ \text{EndOf}(x_{\ell,k}) &\leq h-1 \end{aligned}\right\} \forall \ell \in \{1, \ldots, \ell_{\max}\}, k \in \{1, \ldots, K(\ell)\}. \tag{16}$$

As mentioned previously, the spaces have fixed lengths, i.e.,

$$\text{LengthOf}(x_{\ell,k}) = \ell, \ \forall \ell \in \{1, \ldots, \ell_{\max}\}, k \in \{1, \ldots, K(\ell)\}. \tag{17}$$

To ensure that the jobs and the spaces are not overlapping, we use the *NoOverlap* constraint,

$$\text{NoOverlap}(\{x_{\ell,k} : \ell \in \{1, \ldots, \ell_{\max}\}, k \in \{1, \ldots, K(\ell)\}\} \cup \{z_j : J_j \in \mathcal{J}\}). \tag{18}$$

The lengths of the spaces are constrained by

$$\sum_{\ell=1}^{\ell_{\max}} \sum_{k=1}^{K(\ell)} \text{LengthOf}(x_{\ell,k}) = \ell_{\max}, \tag{19}$$

to ensure that the whole scheduling horizon is filled.

Objective The objective is to minimize the TEC, here expressed as

$$\sum_{\ell=1}^{\ell_{\max}} \sum_{k=1}^{K(\ell)} \text{Element}(\boldsymbol{c}_{\ell}^{(\text{space})}, \text{StartOf}(x_{\ell,k})) + \qquad + \sum_{J_j \in \mathcal{J}} \text{Element}(\boldsymbol{c}_{j}^{(\text{job})}, \text{StartOf}(z_j) + 1), \tag{20}$$

where the first part corresponds to the cost for optimal switchings between the job processings, and the second part corresponds to the cost for job processing. To compute the cost of the present spaces, vector

$$\boldsymbol{c}_{\ell}^{(\text{space})} = (c^*[1, 1 + \ell + 1], c^*[2, 2 + \ell + 1], \ldots, c^*[h - \ell - 1, h]) \tag{21}$$

is used to represent the optimal switching costs for the given length ℓ addressed by the start of space $x_{\ell,k}$ (indexed from 1). Similarly, to compute the cost of the jobs, vector

$$\boldsymbol{c}_{j}^{(\text{job})} = (c_{j,1}^{(\text{job})}, c_{j,2}^{(\text{job})}, \ldots, c_{j,h}^{(\text{job})}) \tag{22}$$

is used.

3.3.2 Interval-state graph as a global constraint

Note that a structure like the interval-state graph could be used even for the filtering of the variables domains, and could be embedded to a global constraint in CP. Imagine having variables $x_1, \ldots, x_h$ with domains $D_1, \ldots, D_h$, representing the states of the machine in each interval. Then whenever a value is filtered from a domain, we could remove the edges in the interval-state graph that are leading to the state corresponding to the removed value. Afterward, by forward and backward search in the graph, we could make the other domains consistent. This is similar to the filtering techniques used for the grammar constraints [11, 16] or knapsack constraints [18]. We believe that efficient global propagation based on the interval-state graph for this unrolled transition diagram can be designed. However, its formal derivation is beyond the scope of this paper.

3.4 Symmetry breaking constraints

Symmetries in connection with the CP and ILP are widely studied [4, 8, 13]. Some of the symmetries in combinatorial problems arise when multiple feasible solutions with the same objective value, and which differ only in the values of the variables, correspond to the same "canonical" feasible solution. For example, the jobs in our scheduling problem are identical, thus replacing one job in a feasible solution with another one having the same processing time will not influence the objective value.

The symmetries might negatively affect the performance of the models due to enlarged search space. To break the symmetries without losing the optimality, *symmetry breaking* *constraints* are employed. In the CP models, we use the following symmetry breaking constraints.

1. Fixed ordering on the jobs having the same processing time: Let $J_j, J_{j'} \in \mathcal{J}$, such that $j < j'$, be two jobs having the same processing time. The symmetry breaking constraint is $\mathrm{EndBeforeStart}(z_j, z_{j'})$.

   To reduce the size of the models, the constraint is not created for every pair of jobs; instead, the jobs having the same processing time are sorted by their indices and the constraint is created only for every pair of two consecutive jobs along this sorted sequence.
2. Fixed ordering on the spaces having the same length: Similarly as with breaking the symmetries on identical jobs, we can break symmetries on identical spaces for FREE space modeling. That is, let $\ell \in \{1, \ldots, h\}$ be a space length and let $k, k' \in \{1, 2, \ldots, K(\ell)\}$ such that $k < k'$. Then we add the following two constraints

$$\mathrm{PresenceOf}(x_{\ell,k'}) \leq \mathrm{PresenceOf}(x_{\ell k}) \,, \tag{23}$$

$$\mathrm{BeforeStart}(x_{\ell,k}, x_{\ell k'}) \,. \tag{24}$$

We also tried to fix the ordering of the jobs with the same processing time in the ILP-SPACES model using either constraint (25a) or (25b)

$$\sum_{I_i \in \mathcal{I}} s_{j,i} \cdot i + p_j \leq \sum_{I_i \in \mathcal{I}} s_{j',i} \cdot i, \quad \forall J_j, J_{j'} \in \mathcal{J}, p_j = p_{j'}, j < j' \,, \tag{25a}$$

$$s_{j',i} \leq \sum_{I_{i'} \in \mathcal{I} : i' < i} s_{j,i'}, \quad \forall J_j, J_{j'} \in \mathcal{J}, p_j = p_{j'}, j < j', \forall I_i \in \mathcal{I} \,. \tag{25b}$$

However, the performance of the resulting model was inferior to the original model without the constraints. Thus, the symmetry breaking constraints are omitted for the ILP-SPACES.

4 Experiments

This section evaluates how ILP-SPACES and the CP models perform in comparison to the ILP-REF model proposed by Aghelinejad et al. [1]. The comparison is made on a set of randomly generated instances; see Section 4.1 for the description of the generated dataset. Due to the space constraints, we only compare the best CP model which is selected according to the results of the preliminary experiment; see Section 4.2. The final results are presented in Section 4.3.

All the experiments were executed on 2x Intel(R) Xeon(R) Silver 4110 CPU 2.10 GHz with 188 GB of RAM (16 cores in total). For solving the ILP and CP models, we used Gurobi 8 and IBM CP Optimizer 12.9, respectively. Except for the time-limit and the search phases in CP-SPACES, which branched on the jobs first, all the solver parameters were set to the default values.

The source codes and the experimental data (instances and solutions) are publicly available at https://github.com/CTU-IIG/EnergyStatesAndCostsScheduling and https://github.com/CTU-IIG/EnergyStatesAndCostsSchedulingData, respectively.

4.1 Instances

The instances in the dataset can be divided according to

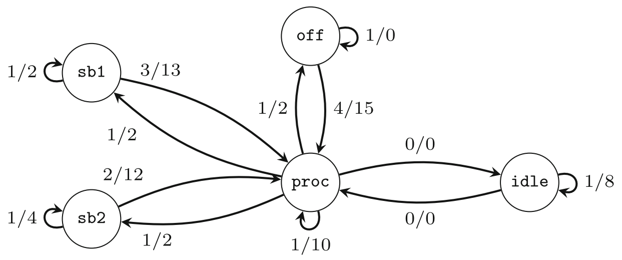


Fig. 4 Example of a transition graph with multiple standby states; every edge from s to s' is labeled by $T[s, s']/P[s, s']$

1. a number of jobs:
   (a) MEDIUM: medium instances with $n \in \{30, 60, 90\}$;
   (b) LARGE: large instances with $n \in \{150, 170, 190\}$;
2. a machine transition graph:
   (a) NOSBY: a simple graph with no standby state [1, 17], see Fig. 1 for its description;
   (b) TWOSBY: a graph with two standby states shown in Fig. 4.

For fixed n and a machine transition graph, 12 random instances are generated in the following way (48 instances in the whole dataset). The processing times of the jobs are randomly sampled from discrete uniform distribution $\mathcal{U}\{1, 5\}$. The number of intervals in each instance is obtained as a multiple of the total processing time plus the required number of intervals to turn the machine on and off, where this multiple H^{mul} is taken from set $\{1.3, 1.6, 1.9, 2.2\}$. The energy cost in each interval is randomly sampled from $\mathcal{U}\{1, 10\}$. For instances differing only in the number of intervals, the energy costs are sampled gradually, i.e., the energy costs of all the intervals in an instance with a shorter horizon are the same as for the corresponding intervals in an instance with a longer horizon.

The dataset for the preliminary CP experiments is generated using a similar scheme as described above with the following differences: $n \in \{30, 60\}$, $H^{\text{mul}} = 1.2$ and NOSBY transition graph is used. For each fixed n, we generate 6 random instances. Thus, the dataset for the preliminary experiments has 12 instances in total. We denote this dataset as PRELIM.

Note that the distributions for sampling the processing times and the energy costs of the intervals for all the datasets are the same as proposed by Aghelinejad et al. [1] and Shrouf et al. [17].

4.2 Preliminary CP experiments: results for PRELIM

The purpose of the preliminary experiments is to compare different CP modeling strategies and to select the best-performing model that will be compared against ILP-SPACES and ILP-REF.

The results aggregated over the instances are presented in Table 1, where *gap* stands for the average optimality gap, *time* is the average runtime in seconds and *#best* represents the number of times the model achieved the best upper bound among all the tested models including the ILP-REF. The optimality gap on each instance is defined as

$$\frac{ub - lb^{\text{best}}}{ub} \cdot 100\,[\%]\,, \tag{26}$$

Table 1 Comparison of different CP modeling techniques on PRELIM dataset

Model	gap [%]	time [s]	#best [-]
Element-Free-SumLengths	0.15	600.0	6
Optional-Free-Pulse	0.18	600.0	6
StepFunction-Free-SumLengths	0.22	600.0	6
Optional-Free-StartOfNext	0.25	600.0	4
Element-Free-StartOfNext	0.26	600.0	5
StepFunction-Free-Pulse	0.28	600.0	5
Element-Free-Pulse	0.33	600.0	6
Optional-No	0.33	600.0	5
Optional-Free-SumLengths	0.35	600.0	4
Logical-Free-SumLengths	0.37	600.0	6
Logical-Fixed-Pulse	0.42	600.0	4
Logical-Free-StartOfNext	0.43	600.0	5
StepFunction-Free-StartOfNext	0.43	600.0	4
Logical-Free-Pulse	0.47	600.0	4
Element-Fixed-Pulse	0.52	600.0	4
Overlap-Free-SumLengths	0.54	600.0	3
Element-Fixed-SumLengths	0.59	600.0	4
Overlap-Free-Pulse	0.61	600.0	4
Logical-Fixed-SumLengths	0.65	600.0	2
StepFunction-Fixed-Pulse	0.70	600.0	4
StepFunction-Fixed-SumLengths	0.73	600.0	3
Overlap-Free-StartOfNext	0.82	600.0	2
Optional-Fixed-SumLengths	0.82	600.0	3
Optional-Fixed-Pulse	0.83	600.0	3
StepFunction-Fixed-StartOfNext	0.98	600.0	4
Optional-Fixed-StartOfNext	0.98	600.0	4
StepFunction-No	1.02	600.0	2
Overlap-Fixed-Pulse	1.06	600.0	2
Overlap-Fixed-SumLengths	1.12	600.0	2
Overlap-Fixed-StartOfNext	1.17	600.0	3
Element-Fixed-StartOfNext	1.19	600.0	3
Logical-Fixed-StartOfNext	1.22	600.0	3
Overlap-No	1.66	600.0	2
Element-No	1.69	600.0	1
Logical-No	2.24	600.0	0
ILP-SPACES	0.00	4.6	12

where lb^{best} is the best lower bound obtained over all models (including the ILP-SPACES) on that instance. The time limit is set to 600 s per instance.

For each instance of PRELIM dataset, none of the CP models is able to prove the optimality of the found solution in the given time-limit. In Table 1, it can be observed how the quality of the best found solution varies across the tested models. Based on the lowest achieved gap, we choose the model Element-Free-SumLengths for the following experiments, which we denote as CP-SPACES.

Since we are looking for the optimal solutions, we have also experimented with the settings of `FailureDirectedSearchEmphasis` parameter for the CP-SPACES model. We run one additional experiment with CP-SPACES model on PRELIM dataset, but with `FailureDirectedSearchEmphasis` set to 16, i.e., 16 threads of the CP solver were dedicated to failure-directed search. However, we were not able to obtain better lower bounds or upper bounds within the given time-limit, compared with the default setting of the parameter. Hence, the default setting of `FailureDirectedSearchEmphasis` is used for the rest of the experiments.

4.3 Results for MEDIUM and LARGE instances with different machine transition graphs

Both the presented Tables 2 and 3 have the same structure: each row represents one instance characterized by the number of jobs n and the number of intervals h. The objective value *ub*

Table 2 Comparison of upper bound *ub*, lower bound *lb* and runtime t between the models on MEDIUM dataset

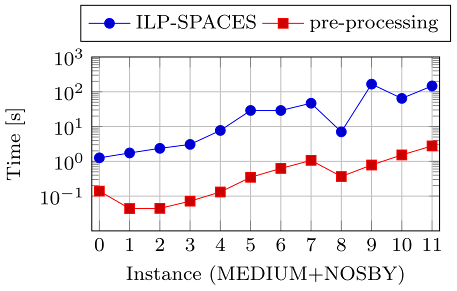
Instance		ILP-REF [1]			CP-SPACES			ILP-SPACES		
n	h	*ub* [-]	*lb* [-]	t [s]	*ub* [-]	*lb* [-]	t [s]	*ub* [-]	*lb* [-]	t [s]
MEDIUM + NOSBY										
30	104	**1426**	**1426**	3.7	1435	496	TLR	**1426**	**1426**	1.3
30	127	**1394**	**1394**	4.9	1396	488	TLR	**1394**	**1394**	1.7
30	150	**1394**	**1394**	5.7	1396	484	TLR	**1394**	**1394**	2.4
30	173	**1394**	**1394**	7.0	1408	484	TLR	**1394**	**1394**	3.1
60	258	**4290**	**4290**	88.5	4339	1724	TLR	**4290**	**4290**	7.7
60	316	**3994**	**3994**	344.7	4048	1584	TLR	**3994**	**3994**	29.0
60	374	**3836**	3826	TLR	3925	1424	TLR	**3836**	**3836**	29.0
60	432	3956	3800	TLR	3986	1380	TLR	**3833**	**3833**	46.9
90	363	6044	5839	TLR	5963	2328	TLR	**5920**	**5920**	7.0
90	445	5778	5567	TLR	5769	2232	TLR	**5686**	**5686**	166.0
90	528	5916	4695	TLR	5670	2168	TLR	**5431**	**5431**	64.5
90	610	5901	4514	TLR	5590	1832	TLR	**5373**	**5373**	147.1
Average time [s]:				337.9			> 600			42.1
Average optimality gap [%]:				1.99			1.73			0.00
MEDIUM+TWOSBY										
30	106	**3815**	**3815**	29.4	**3815**	1240	TLR	**3815**	**3815**	1.4
30	129	**3804**	**3804**	30.7	3815	1220	TLR	**3804**	**3804**	2.3
30	152	**3804**	**3804**	42.0	3815	1210	TLR	**3804**	**3804**	7.0
30	175	**3804**	**3804**	61.4	3815	1210	TLR	**3804**	**3804**	9.5
60	254	**10863**	**10863**	588.1	**10863**	4190	TLR	**10863**	**10863**	2.0
60	311	10289	10087	TLR	10401	3860	TLR	**10248**	**10248**	43.3
60	368	**9917**	9696	TLR	10104	3470	TLR	**9917**	**9917**	82.1
60	426	20346	9133	TLR	9954	3340	TLR	**9874**	**9874**	233.9
90	370	17179	14818	TLR	15401	5900	TLR	**15379**	**15379**	140.2
90	454	22808	12951	TLR	14973	5680	TLR	**14923**	**14923**	138.6
90	538	25992	11868	TLR	14729	5500	TLR	**14548**	**14548**	403.8
90	621	29558	11406	TLR	14900	4620	TLR	**14392**	**14392**	225.8
Average time [s]:				412.6			> 600			107.5
Average optimality gap [%]:				16.02			0.84			0.00

Time-limit is 600 s and TLR stands for time-limit reached of the best found feasible solution, lower bound *lb* and the running time *t* are reported for each tested model. If the objective value or the lower bound is in bold font, the corresponding value is known to be optimal. Therefore, if both objective and the lower bound are in bold, the solver was able to prove the solution optimality within the time-limit. If the solver

Table 3 Comparison of upper bound *ub*, lower bound *lb* and runtime *t* between the models on LARGE dataset

Instance		ILP-REF [1]			ILP-SPACES			P-P
n	*h*	*ub* [-]	*lb* [-]	*t* [s]	*ub* [-]	*lb* [-]	*t* [s]	*t* [s]
LARGE+NOSBY								
150	527	**8582**	8567	TLR	**8582**	**8582**	187	1.0
150	647	8726	8240	TLR	**8409**	**8409**	277	2.9
150	767	8557	7787	TLR	**8132**	**8132**	624	5.5
150	888	8976	6780	TLR	**8078**	**8078**	511	9.1
170	650	10596	9628	TLR	**10068**	**10068**	290	2.3
170	799	10794	8832	TLR	**9820**	**9820**	1087	4.6
170	948	10940	8343	TLR	**9637**	**9637**	806	9.3
170	1097	11189	8124	TLR	**9620**	**9620**	1345	13.4
190	757	12555	11206	TLR	**12008**	**12008**	246	3.9
190	930	12882	10521	TLR	**11758**	**11758**	942	6.9
190	1104	12791	9949	TLR	**11611**	**11611**	3147	13.3
190	1277	12757	0	TLR	**11465**	**11465**	1348	22.7
Average time [s]:				> 3600			901	7.9
Average optimality gap [%]:				7.58			0.00	
LARGE+TWOSBY								
150	529	**21910**	21562	TLR	**21910**	**21910**	130	1.1
150	649	29425	20685	TLR	**21821**	**21821**	702	3.1
150	769	37764	18140	TLR	**21353**	**21353**	949	5.2
150	890	43929	16799	TLR	**21266**	**21266**	701	8.5
170	651	28425	24983	TLR	**25807**	**25807**	809	2.6
170	799	39095	21981	TLR	**25518**	**25518**	1244	5.0
170	948	46083	20709	TLR	**25279**	**25279**	2922	8.5
170	1096	53177	20091	TLR	**25279**	**25279**	2162	14.1
190	756	38471	27984	TLR	**30563**	**30563**	797	4.2
190	929	46319	26166	TLR	**30224**	**30224**	1069	7.5
190	1102	53751	24630	TLR	**30224**	**30224**	2069	13.6
190	1275	61547	0	TLR	**30071**	**30071**	2572	23.7
Average time [s]:				>3600			1344	8.1
Average gap [%]:				34.39			0.00	

Time-limit is 3600s and TLR stands for time-limit reached



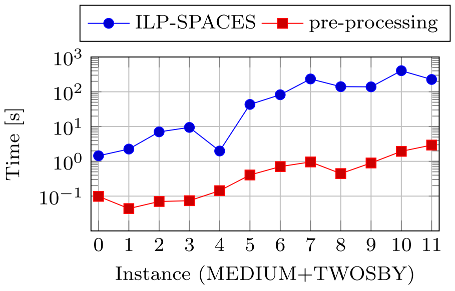


Fig. 5 Comparison between the running times of the pre-processing and ILP-SPACES model for MEDIUM dataset

reached its given time-limit on an instance without proving the optimality of a solution, the value in the corresponding cell in t column is TLR.

The last rows in each table show the average running time of each model and the average optimality gap, defined by (26). The average time is computed over all instances; if the solver timed-out on some instance, the specified time-limit is taken as the running time on that instance.

Additionally, we report the pre-processing time P-P for the large instances. For the medium-size instances, the comparison between the pre-processing time and the solving time of ILP-SPACES is shown in Fig. 5. The pre-processing takes in average only 2.8% of the the total solving time (pre-processing plus the solving of the ILP-SPACES model) for NOSBY and 1.8% for TWOSBY. Overall medium-size instances, the average and maximum pre-processing times were 0.69 s and 2.93 s, respectively.

4.3.1 Results for medium instances

The results of this experiment are shown in Table 2. In this table we can see that ILP-SPACES solves all the instances within the time-limit (600 s). On the other hand, the model ILP-REF proposed by Aghelinejad et al. [1] finds the optimal solution and proves the optimality only for 11 instances out of 24 within the time-limit. Moreover, some of the non-optimal solutions found by ILP-REF are far from the optimum, for example, the objective of the solution found for $n = 90$, $h = 621$ on TWOSBY is more than twice the objective of the optimal one found by ILP-SPACES.

Unfortunately, CP-SPACES is not able to prove the optimality of any instance within the time-limit. However, the average optimality gaps (1.73% for NOSBY and 0.84% for TWOSBY) reveals that it can find near-optimal solutions. The performance of both CP-SPACES and ILP-SPACES is slightly influenced by a more complex transition graph, whereas the performance of ILP-REF deteriorates significantly (average optimality gap 1.99% for NOSBY increased to 16.02% for TWOSBY).

4.3.2 Results for large instances

The results of this experiment are shown in Table 3. The results for CP-SPACES are not included, since we were unable to obtain solutions to all the instances from the IBM CP Optimizer. We observed that the solver used all the available RAM and started swapping, which negatively affected the runtime. Thus, we excluded CP-SPACES from the comparison on the LARGE dataset.

Looking at the results of ILP-SPACES, we can see that it solved all 24 instances within the time-limit (3600 s). On the other hand, ILP-REF was able to find the optimal solutions for only two smallest instances. Comparing the average optimality gaps, ILP-REF achieved 7.58% on NOSBY transition graph and 34.39% on TWOSBY, whereas ILP-SPACES achieved 0% optimality gap on both transition graphs.

5 Conclusions

Continuing on the recent research of the single-machine scheduling problem with the variable energy costs and power-saving machine states, we propose a pre-processing algorithm SPACES, which pre-computes the optimal switching behavior of the machine for all possible spaces in the schedule. The pre-processing runs in polynomial time and works well even for large instances of the problem, e.g., it takes 23 s to pre-process our largest benchmark instance with 190 jobs and 1277 intervals. The pre-computed switching costs are successfully integrated into novel ILP and CP models, which are compared to the state-of-the-art exact ILP model on a set of benchmark instances. Results show that our approach outperforms the existing methods considering all aspects – the runtime, the provided lower bounds, and the upper bounds. Using our models, we obtain the optimal solutions even for the large instances with up to 190 jobs and 1277 intervals, which have been previously tackled only heuristically [1].

Acknowledgements This work was funded by EU Structural funds and Ministry of Education, Youth and Sport of the Czech Republic within the project Cluster 4.0 number CZ.02.1.01/0.0/0.0/16_026/0008432. Moreover, we would like to thank the anonymous reviewers for their suggestions that improved the paper and the solution methods.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

Appendix A : Evaluated CP models

In this appendix, we describe the variations of the CP-SPACES model, which were implemented and evaluated in the preliminary experiment. The description is divided into three parts, namely the *modeling of the jobs*, the *modeling of the spaces*, and the *linking constraints*.

Modeling of the jobs:

All the models contain an interval variable z_j with a fixed length of p_j for each job $J_j \in \mathcal{J}$. This interval variable represents the time interval in which the corresponding job is processed, i.e, the job starts at time StartOf(z_j) and completes at time EndOf(z_j). The models for the jobs differ primarily in how the cost of scheduling the jobs is formulated in the objective.

– OPTIONAL: For each job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$, create an *optional* interval variable $z_{j,i}$ having fixed length p_j and fixed start meaning that the job J_j starts to be processed at the beginning of interval I_i. By enforcing Alternative(z_j, $\{z_{j,i} : \forall I \in \mathcal{I}\}$)

on every job $J_j \in \mathcal{J}$, we constraint that only one such variable will be present in the schedule. Then, every pair of job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$ adds term PresenceOf$(z_{j,i}) \cdot c^{(\text{job})}_{j,i}$ into the objective.

- LOGICAL: Every pair of job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$ adds term (StartOf$(z_j) = i - 1) \cdot c^{(job)}_{j,i}$ into the objective, i.e., if J_j starts at the beginning of I_i, the contribution of J_j into objective is $c^{(job)}_{j,i}$.
- ELEMENT: Each $J_j \in \mathcal{J}$ adds term $c^{(\text{job})}_{j,\text{StartOf}(z_j)+1}$ into the objective. The indexing by a variable can be done using Element expression.
- OVERLAP: Every pair of job $J_j \in \mathcal{J}$ and interval $I_i \in \mathcal{I}$ adds term Overlap$(z_j, I_i) \cdot c_i \cdot P[\texttt{proc}, \texttt{proc}]$ into the objective.
- STEPFN: For every unique processing time $p \in \{p_j : J_j \in \mathcal{J}\}$, create a *step function* F_p representing the cost of starting a job with processing time p_{NoJob} at the beginning of an interval. Each job $J_j \in \mathcal{J}$ adds term Start(EvalStartOf(z_j), F_{p_j}) into the objective.

Modeling of the spaces:

Similarly as with the jobs, the modeling techniques for the spaces differ in how they contribute into the objective.

- FIXED: For each pair of intervals $I_i, I_{i'} \in \mathcal{I}$ such that $i < i'$, create an *optional* interval variable $x_{i,i'}$ having fixed start to i and having fixed end to $i' - 1$. Each such pair of intervals adds term PresenceOf$(x_i, i') \cdot c^*[i, i']$ into the objective.
- FREE: For each possible space length $\ell \in \{1, \ldots, h\}$, we create $K(\ell) = \lfloor \frac{h}{\ell} \rfloor$ optional interval variables $x_{\ell k}$ that are indexed by $k \in \{1, 2, \ldots, K(\ell)\}$ and have fixed length ℓ. Each pair of length $\ell \in \{1 \ldots, h\}$ and $k \in \{1, 2, \ldots, K(\ell)\}$ adds term $c^*[\text{StartOf}(),_\ell\, k, \text{EndOf}()(x_\ell, k) + 1]$ into the objective.

  The difference between FIXED and FREE is that in FREE the start times of the space variables are not fixed.
- NOVARS: In this case, the spaces are not modeled by variables at all. Instead, we create a sequence variable π over all jobs variables z_j. Each position $\ell \in \{1, \ldots, n-1\}$ in π adds term $c^*(\text{EndOf}(\pi_\ell), \text{StartOf}(\pi_{\ell+1}) + 1)$ into the objective (for brevity of the description, the cost of the switching from the first `off` and to the last `off` is omitted).

Linking constraints: The formulations of the jobs and spaces must be linked together with a linking constraint ensuring that every time instant of the scheduling horizon is occupied by either a job or a space interval variable. Moreover, we use NoOverlap on the jobs and spaces variables so that they do not overlap each other. Note that since NOVARS does not use variables for modeling the spaces, the linking constraint is not necessary in this case.

- SUM: Here we simply constraint that the sum of the lengths of all present jobs and spaces variables equals to the length of the scheduling horizon.
- PULSE: For each job and space variable, we create a *pulse* function having *height* of 1. Then, all the pulse functions are summed together into a *cumul* expression, which is forced to be 1 in every time instant of the scheduling horizon.
- STARTOFNEXT: We create a sequence variable π over all jobs and spaces variables. Then we constraint that each variable on position π_ℓ ends at the start of variable $\pi_{\ell+1}$.

References

1. Aghelinejad, M., Ouazene, Y., Yalaoui, A. (2018). Production scheduling optimisation with machine state and time-dependent energy costs. *International Journal of Production Research*, *56*(16), 5558–5575. https://doi.org/10.1080/00207543.2017.1414969.
2. Aghelinejad, M., Ouazene, Y., Yalaoui, A. (2019). Complexity analysis of energy-efficient single machine scheduling problems. *Operations Research Perspectives*, *6*, 100105. https://doi.org/10.1016/j.orp.2019.100105.
3. Benedikt, O., Šůcha, P., Módos, I., Vlk, M., Hanzálek, Z. (2018). Energy-aware production scheduling with power-saving modes. In van Hoeve, W.J. (Ed.) *Integration of constraint programming, artificial intelligence, and operations research* (pp. 72–81). Cham: Springer International Publishing, https://doi.org/10.1007/978-3-319-93031-2_6.
4. Cohen, D., Jeavons, P., Jefferson, C., Petrie, K., Smith, B. (2006). Symmetry definitions for constraint satisfaction problems. *Constraints*, *11*, 115–137. https://doi.org/10.1007/s10601-006-8059-8.
5. Fang, K., Uhan, N.A., Zhao, F., Sutherland, J.W. (2016). Scheduling on a single machine under time-of-use electricity tariffs. *Annals of Operations Research*, *238*(1), 199–227. https://doi.org/10.1007/s10479-015-2003-5.
6. Gahm, C., Denz, F., Dirr, M., Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: a review and research framework. *European Journal of Operational Research*, *248*(3), 744–757. https://doi.org/10.1016/j.ejor.2015.07.017.
7. Gao, K., Huang, Y., Sadollah, A., Wang, L. (2019). A review of energy-efficient scheduling in intelligent production systems Complex & Intelligent Systems.
8. Gent, I.P., Petrie, K.E., Puget, J.F. (2006). Chapter 10 - symmetry in constraint programming. In Rossi, F., van Beek, P., Walsh, T. (Eds.) *Handbook of Constraint Programming, Foundations of Artificial Intelligence*, (Vol. 2 pp. 329–376): Elsevier, https://doi.org/10.1016/S1574-6526(06)80014-3.
9. Gong, X., Pessemier, T.D., Martens, L., Joseph, W. (2019). Energy- and labor-aware flexible job shop scheduling under dynamic electricity pricing: a many-objective optimization investigation. *Journal of Cleaner Production*, *209*, 1078–1094. https://doi.org/10.1016/j.jclepro.2018.10.289.
10. Hadera, H., Harjunkoski, I., Sand, G., Grossmann, I.E., Engell, S. (2015). Optimization of steel production scheduling with complex time-sensitive electricity cost. *Computers & Chemical Engineering*, *76*, 117–136. https://doi.org/10.1016/j.compchemeng.2015.02.004.
11. Kadioglu, S., & Sellmann, M. (2010). Grammar constraints. *Constraints*, *15*(1), 117–144. https://doi.org/10.1007/s10601-009-9073-4.
12. Laborie, P., Rogerie, J., Shaw, P., Vilím, P. (2018). IBM ILOG CP Optimizer for scheduling. *Constraints*, *23*(2), 210–250. https://doi.org/10.1007/s10601-018-9281-x.
13. Margot, F. (2010). Symmetry in integer linear programming. In Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (Eds.) *50 Years of integer programming 1958-2008: from the early years to the state-of-the-art* (pp. 647–686). Berlin: Springer, https://doi.org/10.1007/978-3-540-68279-0_17.
14. Merkert, L., Harjunkoski, I., Isaksson, A., Säynevirta, S., Saarela, A., Sand, G. (2015). Scheduling and energy – industrial challenges and opportunities. *Computers & Chemical Engineering*, *72*, 183–198. https://doi.org/10.1016/j.compchemeng.2014.05.024.
15. Mouzon, G., Yildirim, M.B., Twomey, J. (2007). Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research*, *45*(18–19), 4247–4271. https://doi.org/10.1080/00207540701450013.
16. Pesant, G. (2004). A regular language membership constraint for finite sequences of variables. In Wallace, M. (Ed.) *Principles and practice of constraint programming – CP 2004* (pp. 482–495). Berlin: Springer, https://doi.org/10.1007/978-3-540-30201-8_36.
17. Shrouf, F., Ordieres-Meré, J., García-Sánchez, A., Ortega-Mier, M. (2014). Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *Journal of Cleaner Production*, *67*, 197–207. https://doi.org/10.1016/j.jclepro.2013.12.024.
18. Trick, M. (2001). A dynamic programming approach for consistency and propagation for knapsack constraint. *Annals of Operations Research*, *118*, 73–84. https://doi.org/10.1023/A:1021801522545.