# Scheduling on a single machine under time-of-use electricity tariffs

Kan Fang[*] Nelson A. Uhan[†] Fu Zhao[‡] John W. Sutherland[§]

Original version: April 2014
This version: June 2015

**Abstract**

We consider the problem of scheduling jobs on a single machine to minimize the total electricity cost of processing these jobs under time-of-use electricity tariffs. For the uniform-speed case, in which all jobs have arbitrary power demands and must be processed at a single uniform speed, we prove that the non-preemptive version of this problem is inapproximable within a constant factor unless P = NP. On the other hand, when all the jobs have the same workload and the electricity prices follow a so-called pyramidal structure, we show that this problem can be solved in polynomial time. For the speed-scalable case, in which jobs can be processed at an arbitrary speed with a trade-off between speed and power demand, we show that the non-preemptive version of the problem is strongly NP-hard. We also present different approximation algorithms for this case, and test the computational performance of these approximation algorithms on randomly generated instances. In addition, for both the uniform-speed and speed-scaling cases, we show how to compute optimal schedules for the preemptive version of the problem in polynomial time.

*Keywords:* scheduling; time-of-use tariff; electricity cost; dynamic speed scaling; approximation algorithm

## 1 Introduction

In the United States, about one-third of all the end-use energy consumption is associated with industrial activities. As a result, improving the energy efficiency of manufacturing technologies is crucial as our world faces rising energy costs and mounting energy security challenges. It is well-known that electricity is an efficient and safe way to move energy from one place to another, and most countries use it as the main energy source for manufacturing (Park et al. 2009). So, both electricity-intensive customers and providers have huge opportunities to save costs by improving their efficiency in consuming electricity. Recently, various approaches have been proposed to create

[*]College of Management and Economics, Tianjin University, Tianjin 300072, China, e-mail: `zjumath@gmail.com`

[†]Corresponding author. Mathematics Department, United States Naval Academy, Annapolis, MD 21403, USA, phone: +1 410-293-6713, fax: +1 410-293-4883, e-mail: `uhan@usna.edu`

[‡]Environmental and Ecological Engineering and School of Mechanical Engineering, Purdue University, West Lafayette, IN 47904, USA, e-mail: `fzhao@purdue.edu`

[§]Environmental and Ecological Engineering, Purdue University, West Lafayette, IN 47904, USA, e-mail: `jwsuther@purdue.edu`

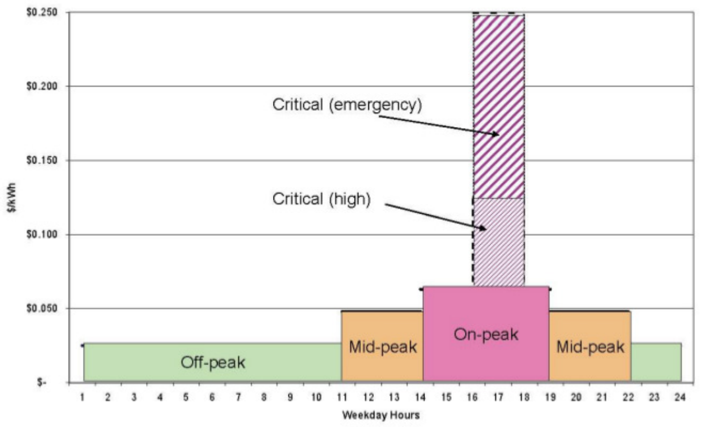

Figure 1.1: A variable TOU tariff scheme (Braithwait et al. 2007)

energy efficient systems, such as exploiting the *variable pricing of electricity* and implementing the technology of *dynamic speed scaling.*

Enabled by the growing infrastructure for advanced metering and monitoring, more and more electricity suppliers have started to implement variable pricing to manage the balance between electricity supply and demand and improve the reliability and efficiency of electrical power grids. For example, with *time-of-use (TOU) tariffs*, retail energy prices to customers vary hourly to reflect changes in wholesale energy prices, which are typically announced a day ahead or an hour ahead. Such price structures are used to shift electricity use from peak hours to off-peak hours. Figure 1.1 shows a typical variable TOU tariff scheme. In a typical TOU tariff scheme, as the demand increases, the cost goes up disproportionately as suppliers have to include older, more inefficient generation plants and more expensive and nonrenewable resources to meet the demand (Shapiro and Tomain 2005); the variation in electricity prices can be as high as a factor of 10 from one hour to the next. As a result, under TOU tariffs, optimizing energy consumption and optimizing energy costs can be quite different: shifting electricity usage from a high cost period to a low cost period can lead to cost savings, even with the same total energy consumption. The wide differences in prices between peak and off-peak hours encourage customers to reduce or reschedule their demand in response to the electricity prices and avoid consuming electricity during hours of peak prices. The cost savings can be significant, especially for large energy consumers such as data centers or manufacturing production lines.

The technology of dynamic speed scaling allows a job to be processed at an arbitrary speed, with a tradeoff between speed and power demand: generally, the higher the machine speed, the higher the power demand. Scheduling jobs on speed-scalable machines to optimize energy-related objectives has received much attention in the computer science community. In particular, a great deal of research has been done on scheduling to optimize energy consumption under the assumptions that jobs can be processed at arbitrary continuous speeds and that the power demand is an exponential function of the speed, especially in single machine environments. We give a few examples. Yao et al. (1995) showed that minimizing the total energy consumption of jobs with release dates and deadlines on a single machine can be done in polynomial time when preemptive processing is allowed. Antoniadis and Huang (2013) showed that the non-preemptive version of this problem is strongly NP-hard, and gave an approximation algorithm[1] with a constant performance guarantee. For the non-preemptive version of this problem in single and parallel machine environments, Bampis et al. (2015) designed approximation algorithms based on the idea of transforming preemptive schedules into non-preemptive schedules. Albers and Fujiwara (2007) studied minimizing the total flow time plus energy consumption on a single machine with non-preemptive processing in an online setting. Similarly, Bansal et al. (2007) proposed online algorithms to minimize total weighted flow time plus energy consumption on a single machine with preemptive processing. Bampis et al. (2012) studied the problem of minimizing the maximum lateness plus energy consumption on a single machine with non-preemptive processing, and proved that different variants of the problem are strongly NP-hard. For further pointers to work on speed scaling, we refer the reader to the articles by Irani and Pruhs (2005) and Albers (2010).

There has also been an increasing amount of work on the impact of energy-aware schedules outside of computer science, in particular, in manufacturing enterprises (e.g., Subai et al. 2006; Mouzon et al. 2007; Wang et al. 2009). For example, researchers have considered using the technology of dynamic speed scaling or multi-power states on scheduling problems in manufacturing (e.g., Sun et al. 2011; Fang et al. 2011, 2013). However, most of these efforts have been aimed at minimizing energy consumption, as in the computer science literature.

Research on using scheduling to reduce energy costs under time-dependent energy pricing in both computer science and manufacturing appears to be rather sparse. Castro et al. (2009) and Mitra et al.

[1]A *$\rho$-approximation algorithm* finds a solution whose objective value is within a factor $\rho$ of the optimal value, and runs in time polynomial in the input size. The factor $\rho$ is known as the *performance guarantee.*

(2012) both studied mixed integer programming formulations to minimize energy costs in industrial scheduling problems with time-sensitive prices. Inspired by scheduling problems in manufacturing systems, Moon et al. (2013) proposed a hybrid genetic algorithm to minimize the weighted sum of makespan and time-dependent electricity costs on unrelated parallel machines. Kulkarni and Munagala (2013) studied scheduling jobs online on a uniform-speed machine to minimize the total weighted flow time and electricity cost, assuming that electricity prices can take two values over time (low and high) and that all jobs have the same power demand. Very recently, Antoniadis et al. (2015) considered minimizing the total electricity cost of processing jobs preemptively on a speed-scalable machine with release dates, deadlines, speed limits, and variable electricity costs.

Researchers have also studied the *time slot scheduling problem* to minimize the energy consumption and other quality-of-service objectives in mobile telecommunication systems and wireless sensor networks. In these problems, the time horizon is generally divided into time slots of equal duration with a given maximum capacity, and each job may only be assigned to a given subset of the slots. For example, Kannan and Wei (2006) studied minimizing the energy consumption of jobs over equal duration time slots for duty-cycle and rate-constrained wireless sensor node transmission. Detti et al. (2009) proposed a strongly polynomial time algorithm for the problem of maximizing the number of scheduled jobs in time slots. Most related to our focus here on TOU tariffs is the work by Wan and Qi (2010), who studied single machine time slot scheduling problems in which each time slot has a corresponding cost, and the objective is to minimize a linear combination of the total time slot costs and some traditional scheduling objective.

In this work, we study problem of scheduling jobs on a single machine to minimize electricity costs under TOU tariffs. For simplicity, we call this problem the *single machine scheduling problem with electricity costs*, or the SMSEC problem for short. We consider the problem when (i) all jobs have arbitrary power demand requirements and must be processed at a single uniform speed, and (ii) each job can be processed at an arbitrary continuous speed with power demand as an exponential function of speed. We refer to these machine environments as *uniform-speed* and *speed-scalable* respectively. We consider both preemptive and non-preeemptive versions of this problem.

For the uniform-speed case, we show that when jobs must be processed non-preemptively, this problem is strongly NP-hard, and in fact inapproximable within a constant factor, unless P = NP. On the other hand, we show how to compute an optimal schedule in polynomial time when

(i) preemption is allowed, and (ii) preemption is forbidden, but all jobs have the same workload and the TOU tariffs satisfy a so-called pyramidal structure.

For the speed-scalable case, we first consider the preemptive version of the problem, for which we determine structural properties of an optimal schedule with the help of a convex programming formulation of the problem and the Karush-Kuhn-Tucker (KKT) optimality conditions. Using these properties, we show how to compute an optimal schedule in polynomial time when jobs can be processed preemptively, and that the problem is strongly NP-hard when jobs must be processed non-preemptively. In addition, by transforming optimal preemptive schedules, we design approximation algorithms for the non-preemptive version of the problem. Finally, we empirically test the performance of these approximation algorithms on a set of randomly generated instances.

To the best of our knowledge, we are the first to study the SMSEC problem as defined here. Very recently, Antoniadis et al. (2015) gave optimality conditions and a polynomial-time algorithm for a generalization of the SMSEC problem variant with a speed-scalable machine and preemptive processing.

## 2 Mathematical description of the problem

As mentioned in the introduction, we refer to the problem that we study in this paper as the *single machine scheduling problem with electricity costs* (the SMSEC problem). In this problem, there is a time-of-use (TOU) tariff scheme represented by a set of time periods $\mathcal{P} = \{1, 2, \ldots, K\}$, with each period $k \in \mathcal{P}$ having an electricity price $c_k$ per unit energy, a start time $a_k$, and a duration $d_k$. We assume that the periods are distinct and contiguous starting at time 0, and ordered such that $0 = a_1 < a_2 < \cdots < a_K$. In other words, period $k$ is $[a_k, a_k + d_k)$ for $k = 1, \ldots, K$ and $a_{k+1} = a_k + d_k$ for $k = 1, \ldots, K-1$. For the sake of simplicity, we define $T = \sum_{k \in \mathcal{P}} d_k$ as the length of the time horizon. Figure 2.1 depicts a general TOU tariff scheme. In addition, there is a set of jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ that need to be scheduled on a single machine. Each job $j \in \mathcal{J}$ has a required workload $w_j$ and required power demand $q_j$. We assume that the workload of jobs and the durations of time periods are given as integers.

We define a *feasible schedule* as a schedule in which all of the jobs are processed within the time horizon. The objective of the SMSEC problem is to find a feasible schedule that minimizes

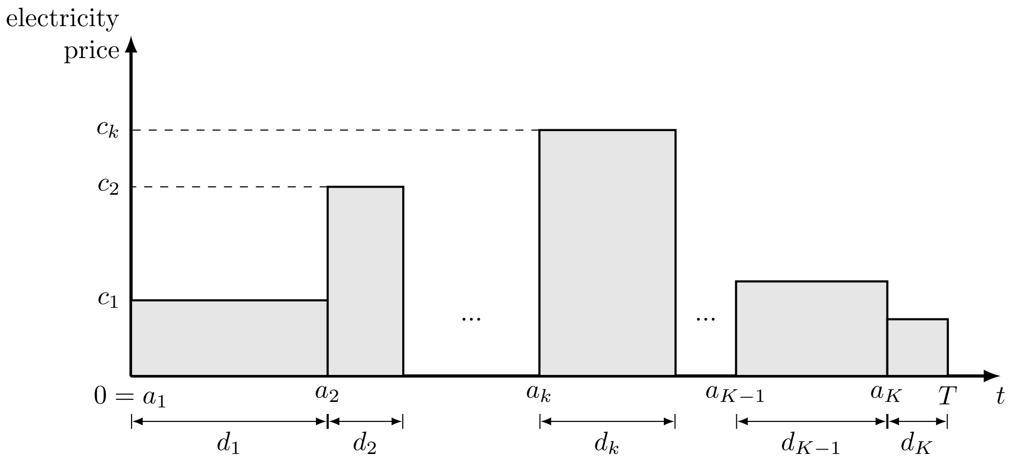

Figure 2.1: An example of TOU tariffs.

the total electricity cost, which is calculated based on the consumed energy over time, taking into account that each time period has a corresponding electricity price per unit energy consumed. For instance, since energy is power integrated over time, if we process job $j \in \mathcal{J}$ for $t$ time units in period $k \in \mathcal{P}$, then we incur an electricity cost of $c_k q_j t$. Depending on the machine environment and job processing requirements – i.e., uniform-speed or speed-scalable machine, non-preemptive or preemptive processing – we define the following variants of the SMSEC problem.

**Problem U.** Jobs must be processed non-preemptively at a uniform speed; that is, once a job has started, it must be processed until its completion. Each job $j \in \mathcal{J}$ has a processing time $p_j = w_j$ and power demand $q_j$. The relationship between processing time and power demand is arbitrary.

**Problem U-pmtn.** Jobs can be processed preemptively at a uniform speed; that is, a job can be paused and resumed at a later point in time. Each job $j \in \mathcal{J}$ has a processing time $p_j = w_j$ and power demand $q_j$. The relationship between processing time and power demand is arbitrary.

In a typical TOU tariff scheme, the peak electricity costs occur in the afternoon, while the off-peak costs occur in the morning and evening (Braithwait et al. 2007, see Figure 1.1). In other words, over a time horizon of one day, a typical TOU tariff scheme has a pyramidal structure. So, we also consider the following special case of Problem U in which the TOU tariff scheme has a pyramidal structure. In this special case, we also assume that all jobs have the same workload.

**Problem U-pyr.** Jobs must be processed non-preemptively at a uniform speed. Each job $j \in \mathcal{J}$ has a processing time $p_j = p$ and power demand $q_j$: all jobs have the same processing time, but not necessarily the same power demand. The relationship between processing time and power demand is arbitrary. The TOU tariff scheme is *pyramidal*: that is, $c_1 < c_2 < \cdots < c_{h-1} < c_h > c_{h+1} > \cdots > c_K$ for some $h \in \mathcal{P}$.

Unlike the uniform-speed cases in Problems U, U-pmtn and U-pyr, in which we only consider scheduling to exploit the variable pricing of electricity, we also consider the following variants of the SMSEC problem that incorporate speed scaling. As mentioned above, in a speed scaling environment, the power demand for processing a job depends on the speed at which the machine runs: the higher the speed, the higher the power demand. For simplicity of analysis with little loss of applicability, it is typical to assume that power demand is an exponential function of speed (e.g. Brooks et al. 2000; Mudge 2001). For these problems, we also add the mild assumption that the electricity prices are strictly positive.

**Problem S.** Jobs must be processed non-preemptively at an arbitrary speed. Specifically, each job must be processed at a single speed from start to completion, and different jobs can be processed at different speeds. Each job $j \in \mathcal{J}$ has workload $w_j$. When job $j$ is processed at speed $s$, its processing time is $p_j = w_j/s$, and its power demand is $q_j = s^\alpha$ for some fixed constant $\alpha > 1$. The electricity prices $c_k$ for $k \in \mathcal{P}$ are assumed to be strictly positive.

Note that our definition of non-preemption in the speed-scaling case is stronger than the classical definition, which only requires that the processing of a job be uninterrupted. We also consider the following preemptive version of Problem S, in which a job can be processed in multiple segments, with a different speed in each segment.

**Problem S-pmtn.** Jobs can be processed preemptively at an arbitrary speed. Specifically, different jobs can be processed at different speeds, and when a job is processed in several parts, the speed used to process the different parts of the job can vary. When one part of job $j \in \mathcal{J}$ is processed in period $\ell \in \mathcal{P}$ at speed $s$ with workload $w_{j\ell}$, its processing time in period $\ell$ is $p_{j\ell} = w_{j\ell}/s$, and its power demand in period $\ell$ is $q_j = s^\alpha$ for some fixed constant $\alpha > 1$. The electricity prices $c_k$ for $k \in \mathcal{P}$ are assumed to be strictly positive.

It turns out that the problem under the classical definition of non-preemption is equivalent to Problem S-pmtn; we will discuss this briefly at the end of Section 4.1.

## 3 Uniform-speed case

In this section, we consider the SMSEC problem in a uniform-speed machine environment. We first prove that Problem U is strongly NP-hard and in fact inapproximable within a constant factor, unless P = NP. This result reveals some of the intrinsic difficulties of the SMSEC problem. Then, we investigate the structural properties of an optimal schedule for Problem U-pmtn. Finally, we give an exact polynomial-time algorithm for Problem U-pyr.

### 3.1 NP-hardness and inapproximability of Problem U

**Theorem 3.1.** *Problem U is strongly NP-hard. In fact, there does not exist a $\rho$-approximation algorithm for any constant $\rho > 1$ for this problem, unless $P = NP$.*

*Proof.* We reduce any instance of the 3-PARTITION problem, which is strongly NP-hard (Garey and Johnson 1979), to an instance of Problem U. The 3-PARTITION problem is described as follows: Given a set $\mathcal{S} = \{1, 2, \ldots, 3m\}$ and positive integers $A_1, A_2, \ldots, A_{3m}, B$ such that $B/4 < A_j < B/2$ for all $j \in \mathcal{S}$ and $\sum_{j \in \mathcal{S}} A_j = mB$, does there exist a partition of $\mathcal{S}$ with $m$ 3-element subsets $S_1, \ldots, S_m$ such that $\sum_{j \in \mathcal{S}_i} A_j = B$ for all $i = 1, 2, \ldots, m$?

Given any instance $\mathcal{I}_1$ of 3-PARTITION, we construct an instance $\mathcal{I}_2$ of Problem U as follows. There are $m$ periods of duration $B$ and $m - 1$ periods of duration 1; in particular,

$$\begin{aligned} d_{2k-1} &= B, \quad c_{2k-1} = 0 \qquad \text{for } k = 1, \ldots, m; \\ d_{2k} &= 1, \qquad c_{2k} = 1 \qquad \text{for } k = 1, \ldots, m-1. \end{aligned}$$

The TOU tariff is shown in Figure 3.1.

The set of jobs is $\mathcal{J} = \{1, \ldots, 4m - 1\}$. The processing time and power demand for each job $j \in \mathcal{J}$ is as follows:

$$\begin{aligned} p_j &= 1, \qquad\qquad q_j = 0 \qquad \text{for } j = 1, 2, \ldots, m-1; \\ p_j &= A_{j-m+1}, \quad q_j = 1 \qquad \text{for } j = m, m+1, \ldots, 4m-1. \end{aligned}$$

Now fix some $\rho > 1$, and assume that there exists a $\rho$-approximation algorithm for Problem U.

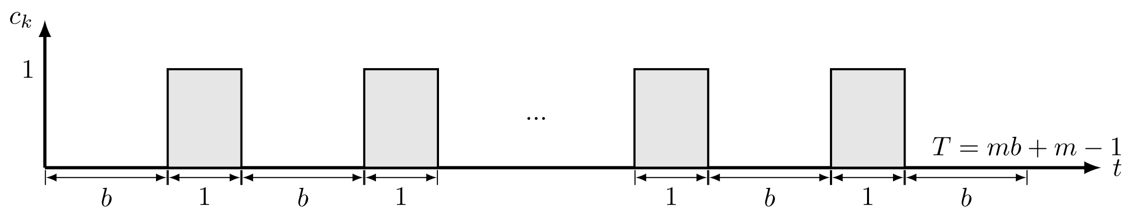

Figure 3.1: The TOU tariff used in the proof of Theorem 3.1.

We show that the output of this algorithm used on instance $\mathcal{I}_2$ allows us to conclude whether there is a solution to instance $\mathcal{I}_1$ in polynomial time. Let $E_{algo}$ be the cost of the solution output by the $\rho$-approximation algorithm for $\mathcal{I}_2$. Therefore, $E_{opt} \leq E_{algo} \leq \rho E_{opt}$, where $E_{opt}$ is the optimal cost. We consider the following two cases.

1. $E_{algo} = 0$. In this case, $E_{opt} = 0$. That is, every job $j \in \{1, 2, \ldots, m-1\}$ is processed in the $m-1$ intervals of duration 1, and the remaining jobs are partitioned over the $m$ intervals of duration $B$. Therefore, there exists a solution to instance $\mathcal{I}_1$.

2. $E_{algo} > 0$. In this case, there is no solution to $\mathcal{I}_1$. Otherwise, we can partition $\mathcal{S}$ into $m$ 3-element subsets $S_1, \ldots, S_m$ such that $\sum_{j \in S_i} A_j = B$ for all $i = 1, \ldots, m$. Then we can process the jobs $m, m+1, \ldots, 4m-1$ in the $m$ periods of duration $B$ by putting all the jobs in each subset $S_i$ in one of these periods, and we can process the other jobs in the $m-1$ periods of duration 1. The total electricity cost of this solution is 0, i.e. $E_{opt} = 0$. Since $E_{algo} \leq \rho E_{opt}$, we have $E_{algo} = 0$, which is a contradiction.

Therefore, we can conclude in polynomial time whether there is a solution to the instance $\mathcal{I}_1$ of 3-PARTITION from the output of the $\rho$-approximation algorithm for $\mathcal{I}_2$, and so Problem U is not only strongly NP-hard, but also inapproximable within a constant factor, unless P = NP. □

## 3.2 Structure of optimal schedules for Problem U-pmtn

The following lemma describes a useful structural property of optimal schedules for Problem U-pmtn.

**Lemma 3.2.** *For any optimal schedule of Problem U-pmtn, if one unit of job $i$ is processed in period $\ell$, one unit of job $j$ is processed in period $k$, and $c_\ell < c_k$, then $q_i \geq q_j$.*

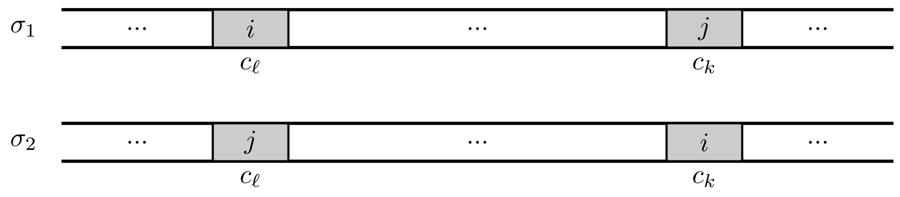

Figure 3.2: A pairwise interchange on two units of jobs $i$ and $j$.

*Proof.* By contradiction. Let $\sigma_1$ be an optimal schedule. If $\sigma_1$ does not satisfy $q_i \geq q_j$, i.e. $q_i < q_j$, then we perform a pairwise interchange on the two units of jobs $i$ and $j$, while maintaining the schedule of all the other units. Denote this new schedule as $\sigma_2$; see Figure 3.2.

Denote the total electricity cost for the two units of job $i$ and $j$ in $\sigma_1$ and $\sigma_2$ as $E_1$ and $E_2$, respectively. Then $E_1 = q_i c_\ell + q_j c_k$ and $E_2 = q_j c_\ell + q_i c_k$. Therefore, $E_1 - E_2 = (q_i - q_j)(c_\ell - c_k)$. Since $c_\ell < c_k$ and $q_i < q_j$, we obtain that $E_1 > E_2$, which contradicts $\sigma_1$ being optimal. □

The above lemma implies that in an optimal preemptive schedule, the jobs with higher power demand are assigned to time periods with a lower electricity price. Therefore, we can construct an optimal preemptive schedule using Algorithm 3.1 described below.

**Algorithm 3.1** Exact polynomial-time algorithm for Problem U-pmtn

```
1: Sort the periods in nondecreasing order of electricity cost:
     Find a permutation $\psi$ of $\mathcal{P}$ so that $c_{\psi(1)} \leq c_{\psi(2)} \leq \cdots \leq c_{\psi(K)}$.
2: Sort the jobs in nonincreasing order of power demand:
     Find a permutation $\varrho$ of $\mathcal{J}$ so that $q_{\varrho(1)} \geq q_{\varrho(2)} \cdots \geq q_{\varrho(n)}$.
3: Initialize:
     $k \leftarrow 1$    (current period is $\psi(k)$)
     $D \leftarrow d_{\psi(k)} = d_{\psi(1)}$    (remaining idle time in current period)
     $S \leftarrow a_{\psi(k)}$    (start time of next processing)
4: for $j = 1$ to $n$ do
5:   while $p_{\varrho(j)} > 0$ do
6:     if $D > 0$ then
7:       $t \leftarrow \min\{D, p_{\varrho(j)}\}$.
8:       Schedule $t$ units of job $\varrho(j)$ in period $\psi(k)$ starting at time $S$.
9:       Update: $p_{\varrho(j)} \leftarrow p_{\varrho(j)} - t$, $D \leftarrow D - t$, $S \leftarrow S + t$
10:    else
11:      Move to the next period: $k \leftarrow k + 1$, $D \leftarrow d_{\psi(k)}$, $S \leftarrow a_{\psi(k)}$
```

**Theorem 3.3.** *Algorithm 3.1 constructs an optimal schedule for Problem U-pmtn in polynomial time.*

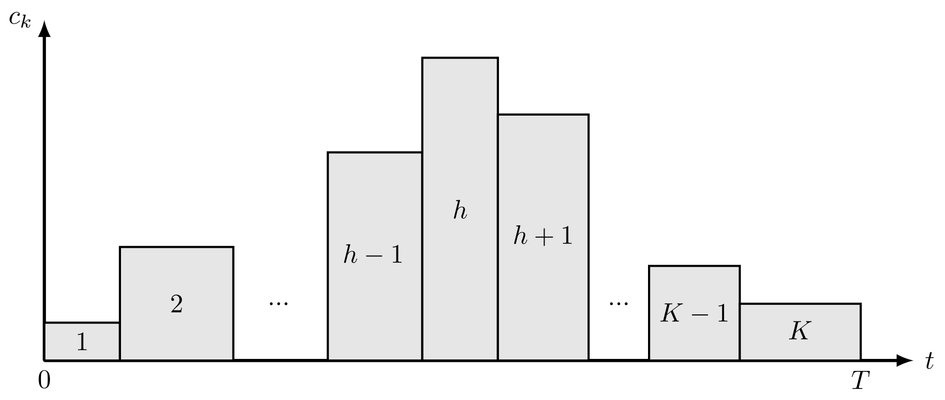

Figure 3.3: An example of pyramidal TOU tariffs.

Table 3.1: An instance with pyramidal TOU tariffs

(a) A pyramidal TOU tariff

| Period ($k$) | Duration ($d_k$) | Price ($c_k$) |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 3 | 4 |
| 3 | 2 | 7 |
| 4 | 2 | 6 |
| 5 | 2 | 3 |
| 6 | 3 | 2 |

(b) Jobs for this instance

| Job ($j$) | Processing time ($p_j$) | Power demand ($q_j$) |
|---|---|---|
| 1 | 2 | 6 |
| 2 | 2 | 5 |
| 3 | 2 | 4 |
| 4 | 2 | 3 |
| 5 | 2 | 2 |
| 6 | 2 | 1 |

## 3.3 An exact polynomial-time algorithm for Problem U-pyr

In this section we consider Problem U-pyr, in which all the jobs have the same workload, and the electricity prices are *pyramidal*; that is, the electricity prices satisfy $c_1 < c_2 < \cdots < c_{h-1} < c_h > c_{h+1} > \cdots > c_K$ for some $h \in \mathcal{P}$. Figure 3.3 gives a graphical example of pyramidal TOU tariffs.

It is easy to check if a given set of jobs can be scheduled within the given time horizon, so in this paper, we will always assume that a feasible solution exists. Before describing our exact polynomial-time algorithm for Problem U-pyr in detail, we consider a small instance of this problem to illustrate what an optimal schedule might look like.

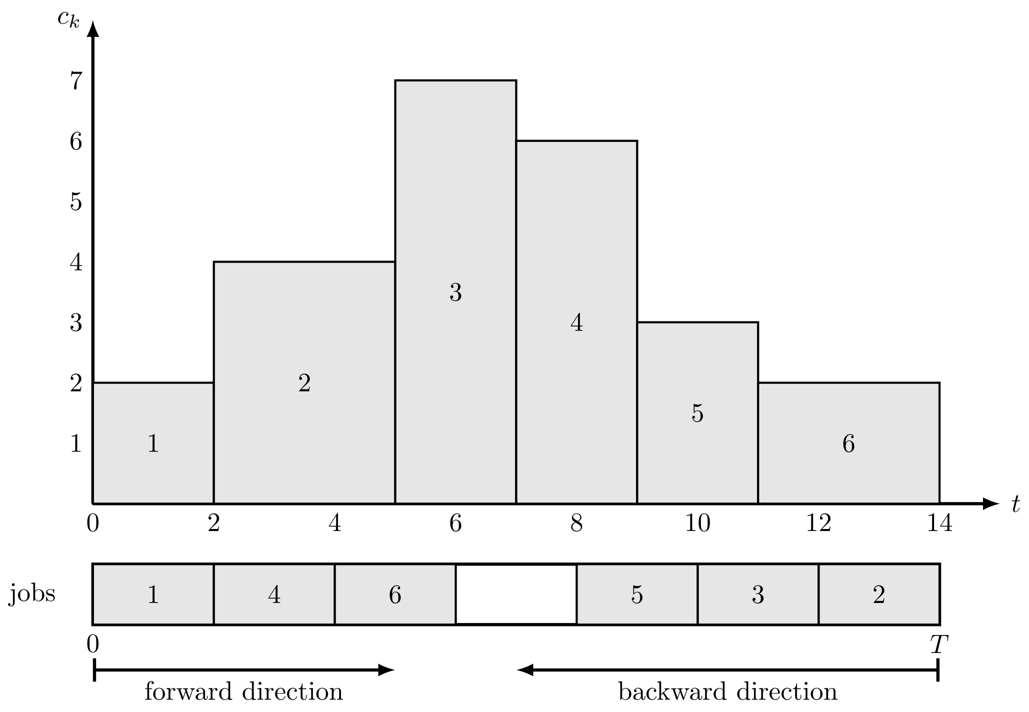

Figure 3.4: An optimal schedule for the instance given in Example 3.1.

**Example 3.1.** Table 1(a) gives a six-period pyramidal TOU tariff, and Table 1(b) gives the processing time and power demand of six jobs. Figure 3.4 illustrates an optimal schedule for the given instance. For simplicity, we define the *forward direction* as the direction in which time increases, and the *backward direction* as the direction in which time decreases. Through some examination, we find that there always exists an optimal schedule as follows: the jobs processed before the period with highest electricity price – period 3 in this example – are processed continuously in the forward direction starting from time 0 in order of nonincreasing power demand. Likewise, the jobs processed after the period with highest electricity price are processed continuously in the backward direction starting from time $T$ in order of nonincreasing power demand.

Inspired by this example, we propose an algorithm that schedules jobs simultaneously in the forward direction starting at time 0 and in the backward direction starting at time $T$. Suppose we are in the middle of scheduling jobs in this manner, and let $t_f$ and $t_b$ be the earliest idle time in the forward and backward direction respectively. In other words, $t_f$ and $T - t_b$ are the total processing time of jobs scheduled up to this point in the forward and backward direction respectively. Since we assume that a feasible solution exists, we always have $t_f + (T - t_b) \leq T$, or $t_f \leq t_b$. Since all the jobs have the same processing time, we define $A(t_f)$ as the electricity cost per unit of power demand for a job starting at $t_f$ in the forward direction; that is, if we schedule a job with power demand $q$ starting at $t_f$ in the forward direction, then the electricity cost incurred by this job is $A(t_f)q$. Likewise, we define $\tilde{A}(t_b)$ as the electricity cost per unit of power demand for a job starting at $t_b$ in the backward direction. Note that $A(t_f)$ and $\tilde{A}(t_b)$ are job-independent, since $p_j = p$ for all $j \in \mathcal{J}$.

To illustrate, in Example 3.1, after we schedule jobs 1 and 2, $t_f = 0 + p_1 = 2$ and $t_b = T - p_2 = \sum_{k=1}^{6} d_k - p_2 = 14 - 2 = 12$. Job 3 has a power demand of $q_3 = 4$ and is processed starting at time $t_b = 12$ in the backward direction, with one unit of time in period 6 and one unit of time in period 5. So, $\tilde{A}(12) = (1)(c_5) + (1)(c_6) = 5$, and the electricity cost incurred by job 3 is $\tilde{A}(12)q_3 = 5(4) = 20$.

Algorithm 3.2 simultaneously schedules jobs in the forward and backward direction in nonincreasing order of power demand, choosing the cheaper direction – comparing $A(t_f)$ and $\tilde{A}(t_b)$ – at each iteration.

**Algorithm 3.2** Exact polynomial-time algorithm for Problem U-pyr

```
1: Sort the jobs so that $q_1 \geq q_2 \geq \cdots \geq q_n$.
2: Initialize: $t_f \leftarrow 0, t_b \leftarrow T$.
3: for $j = 1$ to $n$ do
4:     if $A(t_f) < \tilde{A}(t_b)$ then
5:         Schedule job $j$ in the forward direction starting at $t_f$.
6:         Update: $t_f \leftarrow t_f + p$.
7:     else
8:         Schedule job $j$ in the backward direction starting at $t_b$.
9:         Update: $t_b \leftarrow t_b - p$.
```

It turns out that Algorithm 3.2 is an exact polynomial-time algorithm for Problem U-pyr. To show this, we first show that the values of $A(t_f)$ and $\tilde{A}(t_b)$ computed at each step of the algorithm are monotonic. Let $\mathcal{J}_f$ and $\mathcal{J}_b$ be the set of jobs that are processed in the forward and backward directions, respectively, according to Algorithm 3.2. Let job $\ell$ be the last job scheduled in $\mathcal{J}_f$, and let job $r$ be the last job scheduled in $\mathcal{J}_b$. In addition, let $S_j$ denote the time when job $j \in \mathcal{J}_f$ starts processing in the forward direction and job $j \in \mathcal{J}_b$ starts processing in the backward direction. Similarly, let $C_j$ denote the time when job $j \in \mathcal{J}_f$ completes processing in the forward direction and job $j \in \mathcal{J}_b$ completes processing in the backward direction. See Figure 3.5 for an illustration of these notions.

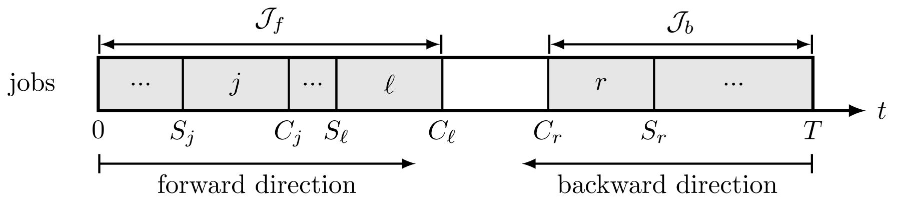

Figure 3.5: A schedule generated by Algorithm 3.2.

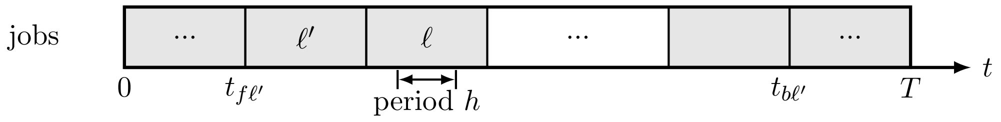

Figure 3.6: Case 2 in the proof of Lemma 3.4.

**Lemma 3.4.** *For any schedule generated by Algorithm 3.2, $A(S_j)$ for $j \in \mathcal{J}_f$ is monotonically nondecreasing in the forward direction. In addition, $\tilde{A}(S_j)$ for $j \in \mathcal{J}_b$ is monotonically nondecreasing in the backward direction.*

*Proof.* First, we consider the properties of $A(S_j)$ in the forward direction.

Case 1: $C_\ell \le \sum_{k=1}^{h} d_k$. In other words, job $\ell$ completes before the end of period $h$. Since $c_1 < c_2 \cdots < c_h$ and each job has the same processing time, the lemma follows directly.

Case 2: $C_\ell > \sum_{k=1}^{h} d_k$. First, it must be the case that $S_\ell \le \sum_{k=1}^{h-1} d_k$. Suppose instead that $S_\ell > \sum_{k=1}^{h-1} d_k$, and let $t_{f\ell}$ and $t_{b\ell}$ be the earliest idle time for scheduling job $\ell$ in the forward and backward direction, respectively. Therefore, $t_{f\ell} = S_\ell > \sum_{k=1}^{h-1} d_k$, and because $c_h > c_{h+1} > \cdots > c_K$, we have $A(t_{f\ell}) = \tilde{A}(T - (t_{f\ell} + p)) \ge \tilde{A}(t_{b\ell})$. As a result, according to Algorithm 3.2, job $\ell$ must be processed in the backward direction, which contradicts job $\ell \in \mathcal{J}_f$.

Now suppose job $\ell'$ is processed immediately before job $\ell$. See Figure 3.6 for a schematic. Since $C_{\ell'} = S_\ell \le \sum_{k=1}^{h-1} d_k$, we have that $A(S_i) \le A(S_j)$ for $1 \le i < j \le \ell'$. We still need to show that $A(S_{\ell'}) \le A(S_\ell)$. Right before we schedule job $\ell'$ in Algorithm 3.2, let $t_{f\ell'}$ and $t_{b\ell'}$ be the earliest idle time in the forward and backward direction, respectively. Since $\ell' \in \mathcal{J}_f$, we know that $A(S_{\ell'}) = A(t_{f\ell'}) < \tilde{A}(t_{b\ell'})$. Note that this implies $A(t) \ge A(t_{f\ell'})$ for any time $t \in [t_{f\ell'} + p, t_{b\ell'}]$. Therefore, we have $A(S_{\ell'}) \le A(S_\ell)$.

Showing the analogous properties of $\tilde{A}(S_j)$ in the backward direction follows in a similar manner. □

**Theorem 3.5.** *Algorithm 3.2 is an exact polynomial-time algorithm for Problem U-pyr.*

*Proof.* For any schedule generated by Algorithm 3.2, using the property in Lemma 3.4, it is straightforward to show that it is impossible to lower the total electricity cost by performing a pairwise interchange on any two jobs in $\mathcal{J}_f$ or any two jobs in $\mathcal{J}_b$. As a result, we only need to prove that it is also impossible to lower the total electricity cost by performing a pairwise interchange with one job in $\mathcal{J}_f$ and one in $\mathcal{J}_b$.

Suppose jobs $1, \ldots, j-1$ are already scheduled by Algorithm 3.2, and the current earliest idle time in the forward and backward directions are $t_f$ and $t_b$, respectively. Without loss of generality, we assume that the next job $j$ is scheduled in the forward direction; that is, $A(t_f) < \tilde{A}(t_b)$. We show that once all of the jobs have been scheduled according to Algorithm 3.2, the total electricity cost cannot be lowered by performing a pairwise interchange with job $j$ and any other job in $\mathcal{J}_b$.

Suppose job $k$ is scheduled in the backward direction starting at $t_b$, and fix job $k'$ to be some job that was scheduled in the backward direction before $t_b$, and job $k''$ to be some job that was scheduled in the backward direction after $t_b$. Denote the start time of jobs $k'$ and $k''$ in the backward direction as $t'_b$ and $t''_b$ respectively. See Figure 3.7 for a schematic. Since job $k'$ was scheduled prior to job $j$, it must be the case that $\tilde{A}(t'_b) \leq A(t'_f)$ for some $t'_f \leq t_f$ by the construction of Algorithm 3.2. Therefore, by Lemma 3.4, we have $\tilde{A}(t'_b) \leq A(t_f) < \tilde{A}(t_b) \leq \tilde{A}(t''_b)$. Denote $E^1_{jk}$ as the total electricity cost of jobs $j$ and $k$ in the schedule output by Algorithm 3.2, and $E^2_{jk}$ the total electricity cost of jobs $j$ and $k$ in the schedule obtained by performing a pairwise interchange on them, while keeping all the other jobs in the same positions. Then we have the following.

1. $E^1_{jk} \leq E^2_{jk}$. According to Algorithm 3.2, we know that $q_j \geq q_k$. Since $E^1_{jk} = A(t_f)q_j + \tilde{A}(t_b)q_k$ and $E^2_{jk} = A(t_f)q_k + \tilde{A}(t_b)q_j$, we obtain that $E^1_{jk} - E^2_{jk} = (A(t_f) - \tilde{A}(t_b))(q_j - q_k) \leq 0$; that is, we cannot reduce the electricity cost by a pairwise interchange on jobs $j$ and $k$.

2. $E^1_{jk'} \leq E^2_{jk'}$. Since job $k'$ is processed in the backward direction before $t_b$, we know that job $k'$ is processed earlier than job $j$ in Algorithm 3.2, and so we have $q_{k'} \geq q_j$. Similarly we have $E^1_{jk'} - E^2_{jk'} = (A(t_f) - \tilde{A}(t'_b))(q_j - q_{k'}) \leq 0$; that is, we cannot reduce the electricity cost by a pairwise interchange on jobs $j$ and $k'$.

3. $E^1_{jk''} \leq E^2_{jk''}$. Since job $k''$ is processed in the backward direction later than $t_b$, we know that

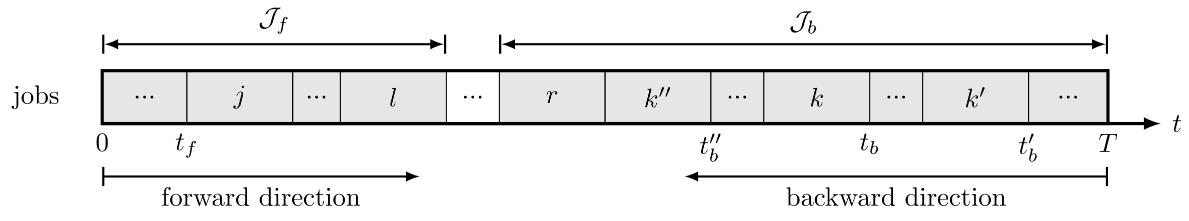

Figure 3.7: Schematic for proof of Theorem 3.5.

$q_j \geq q_{k''}$. Similarly we have $E^1_{jk''} - E^2_{jk''} = (A(t_f) - \tilde{A}(t_b''))(q_j - q_{k''}) \leq 0$; that is, we cannot reduce the electricity cost by a pairwise interchange on jobs $j$ and $k''$.

Using the same arguments as above, we also cannot reduce the total electricity cost by performing a pairwise interchange with job $k \in \mathcal{J}_b$ and any other job in $\mathcal{J}_f$. □

## 4 Speed-scalable case

In this section, we examine the SMSEC problem under the assumption that jobs can be processed at an arbitrary continuous speed. As we mentioned earlier, when job $j$ with a workload of $w_j$ is processed at speed $s$, its processing time is $w_j/s$, and its power demand is $s^\alpha$, for some fixed constant $\alpha > 1$. We also assume that the electricity prices $c_k$ for $k \in \mathcal{P}$ are strictly positive. We first investigate the structure of optimal schedules on a speed-scalable machine for Problem S-pmtn. Then we prove that Problem S is strongly NP-hard by exploiting this structure. Finally, we propose and analyze approximation algorithms for Problem S based on transforming preemptive schedules into non-preemptive ones.

### 4.1 Properties of optimal schedules for Problem S-pmtn

As we mentioned before, in Problem S-pmtn we assume that jobs can be processed preemptively. Specifically, when a job is processed in several parts, the speeds used to process each part can be different. We let $w_{j\ell}$ denote the workload of job $j$ assigned to period $\ell \in \mathcal{P}$ and $v_\ell$ denote the total workload assigned to period $\ell \in \mathcal{P}$, i.e., $v_\ell = \sum_{j \in \mathcal{J}} w_{j\ell}$.

The following lemma describes some structural properties of an optimal preemptive schedule in a given period.

**Lemma 4.1.** *In an optimal schedule for Problem S-pmtn, suppose that period $\ell$ processes a total workload of $v_\ell$. Then*

(a) *the total processing time of jobs in period $\ell$ must be $d_\ell$, and*

(b) *each job in period $\ell$ is processed at the same speed of $v_\ell/d_\ell$.*

*Proof.* (a) By contradiction. Suppose that in an optimal schedule, the total processing time of jobs in period $\ell$ is less than $d_\ell$; that is, there is positive idle time in period $\ell$. Then we can create a new schedule by lowering the speed of one job scheduled in period $\ell$ – say, job $j$ – while keeping the speeds of all the other jobs the same, until there is no idle time remaining in period $\ell$. Let $s_j$ and $s'_j$ be the speed of job $j$ in period $\ell$ in the original and new schedule, respectively. The only part of the total electricity cost that changes is the electricity cost for job $j$ in period $\ell$, which is $c_\ell w_{j\ell}(s_j)^{\alpha-1}$ in the original schedule, and $c_\ell w_{j\ell}(s'_j)^{\alpha-1}$ in the new schedule. Since $s'_j < s_j$, this contradicts the optimality of the original schedule.

(b) Suppose jobs $j_1, \ldots, j_k$ are processed in period $\ell$ in an optimal schedule, with workloads of $w_{j_i\ell} > 0$ for $i = 1, \ldots, k$. Without loss of generality, we assume that jobs are processed in the order $j_1, \ldots, j_k$. Let $T_{j_1j_2}$ be the total processing time of the interval in which jobs $j_1$ and $j_2$ are processed. See Figure 4.1 for a schematic. Then the speeds of job $j_1$ and job $j_2$ in the interval of length $T_{j_1j_2}$ can be determined by the following optimization problem:

$$\underset{s_{j_1}, s_{j_2}}{\text{minimize}} \quad c_\ell(w_{j_1\ell}s_{j_1}^{\alpha-1} + w_{j_2\ell}s_{j_2}^{\alpha-1}) \tag{4.1a}$$

$$\text{subject to} \quad \frac{w_{j_1\ell}}{s_{j_1}} + \frac{w_{j_2\ell}}{s_{j_2}} = T_{j_1j_2}, \tag{4.1b}$$

$$s_{j_1}, s_{j_2} > 0. \tag{4.1c}$$

For the sake of simplicity, we let $s_{j_1} = x, s_{j_2} = y, w_{j_1l} = a, w_{j_2l} = b$, and $T_{j_1j_2} = c$. From (4.1b), we obtain that

$$y = \frac{b}{c - \frac{a}{x}} = \frac{bx}{cx - a}.$$

Since $x > 0$ and $y > 0$, we must have $x > a/c$. Now we define

$$f(x) = ax^{\alpha-1} + b\left(\frac{bx}{cx - a}\right)^{\alpha-1}.$$

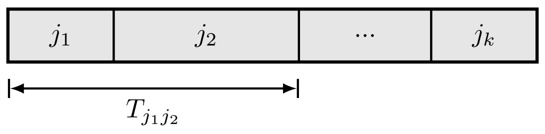

Figure 4.1: Jobs in period $\ell$ in an optimal preemptive schedule.

Then we have

$$f'(x) = \frac{a(\alpha-1)x^{\alpha-2}[(cx-a)^\alpha - b^\alpha]}{(cx-a)^\alpha}.$$

Let $f'(x^*) = 0$. Since $x > 0$, we obtain that $x^* = (a+b)/c$. In addition, when $a/c < x < x^*$, we have $f'(x) < 0$, and when $x > x^*$, we have $f'(x) > 0$. As a result, we know that $x^* = (a+b)/c$ is a unique global minimum for $f(x)$ within $(a/c, \infty)$; that is, the optimal solution to (4.1) is $s^*_{j_1} = s^*_{j_2} = (w_{j_1\ell} + w_{j_2\ell})/T_{j_1j_2}$. Similarly, for any two adjacent jobs processed in period $\ell$, using the same arguments we can prove that their speeds must be the same. Since the total workload in period $\ell$ is $v_\ell$, we obtain that each job in period $\ell$ is processed at the same speed of $v_\ell/d_\ell$. □

Let $E_{pmtn}$ be the optimal total electricity cost for Problem S-pmtn, and let $W$ be the total workload of all the jobs, i.e., $W = \sum_{j\in\mathcal{J}} w_j$. Suppose we restrict our schedule to $m$ time periods, $\ell_1, \ell_2, \ldots, \ell_m$. Note that these periods are not necessarily contiguous. The following lemma gives the relationship between the workload, speeds and the selected time periods in an optimal preemptive schedule.

**Lemma 4.2.** *In any optimal preemptive schedule for Problem S-pmtn restricted to periods $\ell_1, \ldots, \ell_m$, the workload assigned in period $\ell_k$ is*

$$v^*_{\ell_k} = \frac{W d_{\ell_k}}{c_{\ell_k}^{1/(\alpha-1)} \left( \displaystyle\sum_{i=1}^{m} \frac{d_{\ell_i}}{c_{\ell_i}^{1/(\alpha-1)}} \right)}, \tag{4.2}$$

*the speed of each job in period $\ell_k$ is*

$$s^*_{\ell_k} = \frac{W}{c_{\ell_k}^{1/(\alpha-1)} \left( \displaystyle\sum_{i=1}^{m} \frac{d_{\ell_i}}{c_{\ell_i}^{1/(\alpha-1)}} \right)}, \tag{4.3}$$

*and the total electricity cost is*

$$\frac{W^\alpha}{\left(\displaystyle\sum_{i=1}^m \frac{d_{\ell_i}}{c_{\ell_i}^{1/(\alpha-1)}}\right)^{\alpha-1}}. \tag{4.4}$$

*Proof.* By Lemma 4.1, we know that in an optimal schedule, each job scheduled in period $\ell_k$ is processed at the same speed of $v_{\ell_k}/d_{\ell_k}$. As a result, the electricity cost in period $\ell_k$ is $c_{\ell_k} d_{\ell_k} (v_{\ell_k}/d_{\ell_k})^\alpha$, and an optimal schedule can be obtained by solving the following optimization problem:

$$\begin{aligned}
\underset{v_{\ell_1},\ldots,v_{\ell_m}}{\text{minimize}} \quad & \sum_{k=1}^m c_{\ell_k} d_{\ell_k} \left(\frac{v_{\ell_k}}{d_{\ell_k}}\right)^\alpha && \text{(4.5a)}\\
\text{subject to} \quad & \sum_{k=1}^m v_{\ell_k} = W, && \text{(4.5b)}\\
& v_{\ell_k} \geq 0 \text{ for } k = 1, \ldots, m. && \text{(4.5c)}
\end{aligned}$$

Note that the objective function is convex when $v_{l_k} \geq 0$ for all $k = 1, \ldots, m$ and the feasible region is convex, so (4.5) is a convex program. In addition, note that the objective function and the constraints are continuously differentiable. Let

$$L(v_{\ell_1}, \ldots, v_{\ell_m}, \lambda, \mu_1, \ldots, \mu_k) = \sum_{k=1}^m c_{\ell_k} d_{\ell_k} \left(\frac{v_{\ell_k}}{d_{\ell_k}}\right)^\alpha - \lambda\left(\sum_{k=1}^m v_{\ell_k} - W\right) - \sum_{k=1}^m \mu_k v_{\ell_k}.$$

The KKT conditions are:

$$\begin{aligned}
& c_{\ell_k} d_{\ell_k}^{1-\alpha} \alpha (v_{\ell_k}^*)^{\alpha-1} = \lambda^* + \mu_k^* && \text{for } k = 1, \ldots, m, && \text{(4.6a)}\\
& v_{\ell_1}^* + v_{\ell_2}^* + \cdots + v_{\ell_m}^* = W, && && \text{(4.6b)}\\
& v_{\ell_k}^* \geq 0 && \text{for } k = 1, \ldots, m, && \text{(4.6c)}\\
& \mu_k^* \geq 0 && \text{for } k = 1, \ldots, m, && \text{(4.6d)}\\
& \mu_k^* v_{\ell_k}^* = 0 && \text{for } k = 1, \ldots, m. && \text{(4.6e)}
\end{aligned}$$

By the KKT conditions, we must have $\mu_k^* = 0$ for all $k = 1, \ldots, m$. For the purpose of deriving a contradiction, suppose $\mu_h^* > 0$ for some $h \in \{1, \ldots, m\}$. By (4.6e), we have $v_{\ell_h}^* = 0$, and so by (4.6a), we have $\lambda^* = -\mu_h^*$. Constraints (4.6a) and (4.6c) together imply that $\lambda^* + \mu_k^* \geq 0$, and therefore $\mu_k^* \geq \mu_h^* > 0$ for all $k = 1, \ldots, m$. Finally, it follows from (4.6e) that $v_{\ell_k}^* = 0$ for all $k = 1, \ldots, m$, which contradicts (4.6b).

In addition, we must also have $v^*_{\ell_k} > 0$ for all $k = 1, \dots, m$. For the purpose of deriving a contradiction, suppose $v^*_{\ell_h} = 0$ for some $h \in \{1, \dots, m\}$. Then by (4.6a) we have $\lambda^* = 0$, and so $v^*_{\ell_k} = 0$ for all $k = 1, \dots, m$, which contradicts (4.6b). As a result, (4.6) is equivalent to:

$$c_{\ell_k} d_{\ell_k}^{1-\alpha} \alpha (v^*_{\ell_k})^{\alpha-1} = \lambda^* \qquad \text{for } k = 1, \dots, m, \tag{4.7a}$$

$$v^*_{\ell_1} + v^*_{\ell_2} + \cdots + v^*_{\ell_m} = W, \tag{4.7b}$$

$$v^*_{\ell_k} > 0 \qquad \text{for } k = 1, \dots, m. \tag{4.7c}$$

It is straightforward to check that $v^*$ as defined in (4.2) is the unique solution that satisfies (4.7). Note that the linear independence constraint qualification (LICQ) for the convex program (4.5) holds at $v^*$, and so the claim follows. □

The following results follow easily from Lemma 4.2.

**Corollary 4.3.** *In any optimal preemptive schedule for the restriction of Problem S-pmtn in which jobs can only be processed in $m$ time periods, these $m$ periods are selected according to nonincreasing order of $d_k / c_k^{1/(\alpha-1)}$.*

*Proof.* By Lemma 4.2, the optimal preemptive schedule using periods $\ell_1, \dots, \ell_m$ has total electricity cost (4.4). Therefore, we can minimize this cost by choosing the $m$ periods with the largest values of $d_k / c_k^{1/(\alpha-1)}$. □

**Corollary 4.4.** *In any optimal preemptive schedule for Problem S-pmtn,*

$$\frac{s^*_\ell}{s^*_k} = \left( \frac{c_k}{c_\ell} \right)^{1/(\alpha-1)} \qquad \textit{for all } k, \ell \in \mathcal{P}.$$

**Theorem 4.5.** *The optimal electricity cost for Problem S-pmtn is*

$$\frac{W^\alpha}{\left( \displaystyle\sum_{i=1}^{K} \frac{d_i}{c_i^{1/(\alpha-1)}} \right)^{\alpha-1}}.$$

Note that in any optimal preemptive schedule, the workload and the speeds of jobs in each period $\ell$ depend only on the duration and electricity price of period $\ell$. Therefore, without loss of generality, in an optimal preemptive schedule, we can assume that the jobs are processed earliest to latest in the order $1, 2, \ldots, n$. Putting this all together, we can solve the restriction of Problem S-pmtn using Algorithm 4.1 described below.

**Algorithm 4.1** Exact polynomial-time algorithm for the restriction of Problem S-pmtn in which jobs can only be processed in $m$ periods

```
1: Determine the $m$ periods that will be used to process all the jobs based on Corollary 4.3.
2: Sort these $m$ periods in nondecreasing order of their start time and denote them as $\ell_1, \ell_2, \ldots, \ell_m$,
     i.e., $a_{\ell_1} < a_{\ell_2} < \cdots < a_{\ell_m}$.
3: Compute $s^*_{\ell_k}$ for $k = 1, 2, \ldots, m$ as in (4.3).
4: Initialize:
     $k \leftarrow 1$    (current period is $\ell_k$)
     $D \leftarrow d_{\ell_k} = d_{\ell_1}$    (remaining idle time in current period)
     $S \leftarrow a_{\ell_k}$    (start time of next processing)
5: for $j = 1$ to $n$ do
6:   while $w_j > 0$ do
7:     if $D > 0$ then
8:       $v \leftarrow \min\{D s^*_{\ell_k}, w_j\}$.
9:       Schedule $v$ units of job $j$'s workload at speed $s^*_{\ell_k}$ starting at time $S$.
10:      Update: $w_j \leftarrow w_j - v$, $D \leftarrow D - v/s^*_{\ell_k}$, $S \leftarrow S + v/s^*_{\ell_k}$.
11:    else
12:      Move to the next period: $k \leftarrow k + 1$, $D \leftarrow d_{\ell_k}$, $S \leftarrow a_{\ell_k}$
```

Note that when all $K$ periods are considered, Algorithm 4.1 constructs a schedule in which each job is processed without interruption, although possibly with multiple speeds. As a result, when jobs must be processed non-preemptively in the classical sense, the speed-scaling version of the SMSEC problem can be solved using Algorithm 4.1. However, as we will see next, when we assume that jobs must be processed without interruption and at a single speed from start to completion, the problem becomes harder.

## 4.2 NP-hardness of Problem S

Unlike the preemptive speed-scalable scheduling problem we discussed above, the corresponding non-preemptive problem is strongly NP-hard.

**Theorem 4.6.** *Problem S is strongly NP-hard.*

*Proof.* Given any instance $\mathcal{I}_1$ of 3-PARTITION with $\mathcal{S} = \{1, 2, \ldots, 3m\}$ and positive integers $A_1, \ldots, A_{3m}, B$ such that $B/4 < A_j < B/2$ for all $j \in \mathcal{S}$ and $\sum_{j\in\mathcal{S}} A_j = mB$, we construct an instance $\mathcal{I}_2$ of Problem S as follows. The TOU tariff is similar to the one depicted in Figure 3.1 in the proof of Theorem 3.1:

$$\begin{aligned} d_{2k-1} &= B, \quad c_{2k-1} = 1 && \text{for } k = 1, \dots, m; \\ d_{2k} &= 1, \qquad c_{2k} = 2^{\alpha-1} && \text{for } k = 1, \dots, m-1. \end{aligned}$$

The set of jobs is $\mathcal{J} = \{1, \dots, 4m-1\}$. The workload of each job $j \in \mathcal{J}$ is as follows:

$$\begin{aligned} w_j &= \frac{1}{2} && \text{for } j = 1, 2, \dots, m-1; \\ w_j &= A_{j-m+1} && \text{for } j = m, m+1, \dots, 4m-1. \end{aligned}$$

We first consider the corresponding optimal *preemptive* schedule for instance $\mathcal{I}_2$. Note that the total workload of jobs is $mB + (m-1)/2$. By Lemma 4.2, in any optimal preemptive schedule, the optimal workload $v_\ell^*$ and corresponding speed $s_\ell^*$ for periods $\ell = 1, \dots, 2m-1$ are

$$v_{2k-1}^* = B, \qquad s_{2k-1}^* = 1 \qquad \text{for } k = 1, \dots, m; \tag{4.8a}$$
$$v_{2k}^* = \frac{1}{2}, \qquad s_{2k}^* = \frac{1}{2} \qquad \text{for } k = 1, \dots, m-1, \tag{4.8b}$$

and the optimal total electricity cost is $E_{pmtn} = mB + (m-1)/2$.

Let $E^*$ be the optimal electricity cost for instance $\mathcal{I}_2$. Note that $E^* \geq E_{pmtn}$. Suppose there exists a non-preemptive schedule $\sigma_1$ for instance $\mathcal{I}_2$ with electricity cost $E^* = E_{pmtn}$. The workload and the speeds assigned to each period in $\sigma_1$ must be the same as in (4.8); otherwise, this would contradict Lemma 4.2. Note that the speeds are different between adjacent periods: $s_{2k-1}^* = 1$ for $k = 1, \dots, m$ and $s_{2k}^* = 1/2$ for $k = 1, \dots, m-1$. Therefore, since $\sigma_1$ is a non-preemptive schedule, each job in $\sigma_1$ must be completely processed within one period. This can be done if and only if every job $j \in \{1, 2, \dots, m-1\}$ can be scheduled in the $m-1$ time periods of duration 1, and the remaining jobs can be scheduled in the $m$ intervals of duration $B$. This can be done if and only if 3-PARTITION has a solution. Since 3-PARTITION is strongly NP-hard (Garey and Johnson 1979), the theorem follows. □

## 4.3 Approximation algorithms for Problem S

We propose and analyze approximation algorithms for Problem S that transform optimal schedules for Problem S-pmtn in various ways.

In Algorithm 4.2, we transform a preemptive schedule into a non-preemptive one by processing each job at a uniform speed while keeping the position – the start and completion times – of each job unchanged. Note that when we use Algorithm 4.1 for Problem S-pmtn on the entire set of periods $\mathcal{P} = \{1, \ldots, K\}$, each job is processed continuously, although possibly at multiple speeds.

**Algorithm 4.2** Approximation algorithm for Problem S

```
1: $\sigma_{pmtn} \leftarrow$ optimal schedule for Problem S-pmtn obtained by Algorithm 4.1 with $m = K$.
2: $T_j \leftarrow$ total processing time of job $j$ in $\sigma_{pmtn}$ for $j \in \mathcal{J}$.
3: for $j = 1$ to $n$ do
4:     Keep the position of job $j$ unchanged, and process job $j$ with uniform speed $s'_j = w_j/T_j$.
```

**Theorem 4.7.** *Algorithm 4.2 is a $\sum_{k=1}^{K} \left(c_k/\min_{\ell\in\mathcal{P}}\{c_\ell\}\right)^{\alpha/(\alpha-1)}$-approximation algorithm for Problem S.*

*Proof.* Let $\sigma_{pmtn}$ be an optimal preemptive schedule, and let $\sigma_{nonpmtn}$ be the non-preemptive schedule generated by Algorithm 4.2. Let the electricity cost for processing job $j \in \mathcal{J}$ in $\sigma_{nonpmtn}$ and $\sigma_{pmtn}$ be $E^j_{nonpmtn}$ and $E^j_{pmtn}$ respectively. We will show that

$$E^j_{nonpmtn} \leq \sum_{k=1}^{K} \left(\frac{c_k}{\min_{\ell\in\mathcal{P}}\{c_\ell\}}\right)^{\alpha/(\alpha-1)} E^j_{pmtn} \quad \text{for } j \in \mathcal{J}.$$

Fix a job $j \in \mathcal{J}$. Let $b$ be the period in which job $j$ begins processing, $e$ the period in which job $j$ ends processing, and $s^*_k$ the speed assigned to period $k$ in $\sigma_{pmtn}$. Let $T_{jb}$ and $T_{je}$ be the processing time of job $j$ in period $b$ and $e$ respectively. See Figure 4.2 for a schematic. Then we have that

$$E^j_{pmtn} = c_b T_{jb}(s^*_b)^\alpha + \sum_{k=b+1}^{e-1} c_k d_k (s^*_k)^\alpha + c_e T_{je}(s^*_e)^\alpha,$$

$$E^j_{nonpmtn} = c_b T_{jb}(s'_j)^\alpha + \sum_{k=b+1}^{e-1} c_k d_k (s'_j)^\alpha + c_e T_{je}(s'_j)^\alpha.$$

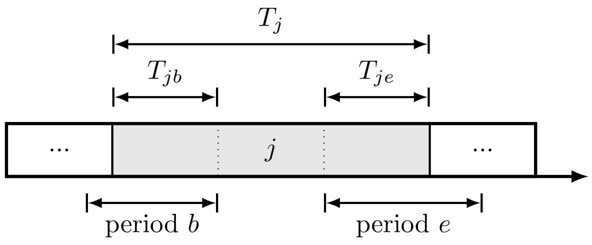

Figure 4.2: Job $j$ in an optimal preemptive schedule $\sigma_{pmtn}$.

Note that $(a+b)/(c+d) \le a/c + b/d$ for any $a, b, c, d > 0$. Therefore,

$$\begin{aligned}\frac{E^j_{nonpmtn}}{E^j_{pmtn}} &= \frac{c_b T_{jb}(s'_j)^\alpha + \sum_{k=b+1}^{e-1} c_k d_k (s'_j)^\alpha + c_e T_{je}(s'_j)^\alpha}{c_b T_{jb}(s^*_b)^\alpha + \sum_{k=b+1}^{e-1} c_k d_k (s^*_k)^\alpha + c_e T_{je}(s^*_e)^\alpha} \\ &\le \frac{c_b T_{jb}(s'_j)^\alpha}{c_b T_{jb}(s^*_b)^\alpha} + \sum_{k=b+1}^{e-1} \frac{c_k d_k (s'_j)^\alpha}{c_k d_k (s^*_k)^\alpha} + \frac{c_e T_{je}(s'_j)^\alpha}{c_e T_{je}(s^*_e)^\alpha} \\ &= \left(\frac{s'_j}{s^*_b}\right)^\alpha + \sum_{k=b+1}^{e-1} \left(\frac{s'_j}{s^*_k}\right)^\alpha + \left(\frac{s'_j}{s^*_e}\right)^\alpha.\end{aligned}$$

Since

$$T_{jb} s^*_b + \sum_{k=b+1}^{e-1} d_k s^*_k + T_{je} s^*_e = T_j s'_j,$$

it follows that $s'_j \le \max_{b \le k \le e}\{s^*_k\} \le \max_{k \in \mathcal{P}}\{s^*_k\}$. Note that a job can be scheduled in at most $K$ periods, so

$$\frac{E^j_{nonpmtn}}{E^j_{pmtn}} \le \sum_{k=1}^{K} \left(\frac{s'_j}{s^*_k}\right)^\alpha \le \sum_{k=1}^{K} \left(\frac{\max_{\ell \in \mathcal{P}}\{s^*_\ell\}}{s^*_k}\right)^\alpha.$$

By Corollary 4.4, we have

$$\frac{E_{nonpmtn}}{E_{opt}} \le \frac{E_{nonpmtn}}{E_{pmtn}} \le \sum_{k=1}^{K} \left(\frac{c_k}{\min_{\ell \in \mathcal{P}}\{c_\ell\}}\right)^{\alpha/(\alpha-1)},$$

where $E_{opt}$ is the total electricity cost of an optimal schedule for Problem S. □

By Theorem 4.5 we know that all of the $K$ periods are used to process the jobs in an optimal preemptive schedule. However, when constructing a non-preemptive schedule, it may be better to only use a subset of the $K$ periods to process jobs. The following lemma describes the relationship between the optimal cost when using a subset of periods and the optimal cost when using all the periods for Problem S-pmtn.

**Lemma 4.8.** *Let $E_m$ be the cost of an optimal preemptive schedule for the restriction of Problem S-pmtn in which jobs can only be processed in $m$ time periods, for $m = 1, \ldots, K$. Then $E_m \leq (K/m)^{\alpha-1} E_K$.*

*Proof.* Let $\ell_1, \ldots, \ell_m$ be the $m$ periods in which processing occurs, and $\ell'_1, \ldots, \ell'_{K-m}$ be the other $K - m$ periods. By Lemma 4.2 we know that

$$\frac{E_m}{E_K} = \frac{E_m}{E_{pmtn}} = \left( \sum_{i=1}^{m} \frac{d_{\ell_i}}{c_{\ell_i}^{1/(\alpha-1)}} \Bigg/ \sum_{k=1}^{K} \frac{d_k}{c_k^{1/(\alpha-1)}} \right)^{1-\alpha}.$$

Since $\alpha > 1$, in order to prove the lemma, we need to show that

$$\sum_{i=1}^{m} \frac{d_{\ell_i}}{c_{\ell_i}^{1/(\alpha-1)}} \Bigg/ \sum_{k=1}^{K} \frac{d_k}{c_k^{1/(\alpha-1)}} \geq \frac{m}{K},$$

or equivalently,

$$(K-m)\left( \sum_{i=1}^{m} \frac{d_{\ell_i}}{c_{\ell_i}^{1/(\alpha-1)}} \right) \geq m \left( \sum_{j=1}^{K-m} \frac{d_{\ell'_j}}{c_{\ell'_j}^{1/(\alpha-1)}} \right).$$

The above inequality holds because according to Corollary 4.3, the $m$ periods used for processing are selected according to nonincreasing value of $d_k / c_k^{1/(\alpha-1)}$ in an optimal preemptive schedule; that is,

$$\frac{d_{\ell'_j}}{c_{\ell'_j}^{1/(\alpha-1)}} \leq \min_{i=1,\ldots,m} \left\{ \frac{d_{\ell_i}}{c_{\ell_i}^{1/(\alpha-1)}} \right\} \quad \text{for } j = 1, \ldots, K-m. \qquad \square$$

In Algorithm 4.3, we transform optimal schedules for the restriction of Problem S-pmtn in which all jobs are processed in $m$ periods into non-preemptive schedules, for all $m = 1, \ldots, K$. Note that the $m$ periods in which jobs are processed in an optimal preemptive schedule may not be contiguous, and as a result, we cannot transform these preemptive schedules into non-preemptive ones as we did in Algorithm 4.2. Instead, we process the entirety of each job in the period in which the job has maximum processing time.

**Theorem 4.9.** *Algorithm 4.3 is a $\max_{1 \leq m \leq K} \beta_m$-approximation algorithm for Problem S, where*

$$\beta_m = m K^{\alpha-1} \left( \frac{\max_{k \in \mathcal{P}_m}\{c_k\}}{\min_{k \in \mathcal{P}_m}\{c_k\}} \right)^{\alpha/(\alpha-1)} \quad \textit{for } m = 1, \ldots, K.$$

**Algorithm 4.3** Approximation algorithm for Problem S

```
1: for $m = 1$ to $K$ do
2:     $\mathcal{P}_m \leftarrow$ set of periods $k \in \mathcal{P}$ with $m$ largest values of $d_k/c_k^{1/(\alpha-1)}$.
3:     $\sigma_{pmtn}^m \leftarrow$ optimal schedule for the restriction of Problem S-pmtn in which all jobs
               are processed in $\mathcal{P}_m$, obtained by Algorithm 4.1.
4:     for $j = 1$ to $n$ do
5:         $T_{jk} \leftarrow$ processing time of job $j$ in period $k$ in $\sigma_{pmtn}^m$.
6:         $k' \leftarrow \arg\max_{k \in \mathcal{P}_m} \{T_{jk}\}$.
7:         Keep the position of job $j$ in period $k'$ unchanged, and process the entire job $j$ in period $k'$
             with uniform speed $s'_j = w_j/T_{jk'}$.
8:     Denote this non-preemptive schedule as $\sigma_{nonpmtn}^m$ and its total electricity cost as $E_{nonpmtn}^m$.
9: $m^* \leftarrow \arg\min_{1 \le m \le K} \{E_{nonpmtn}^m\}$.
10: Output schedule $\sigma_{nonpmtn}^{m^*}$.
```

*Proof.* Let $E_{pmtn}^m$ and $E_{nonpmtn}^m$ be the total electricity cost of schedule $\sigma_{pmtn}^m$ and $\sigma_{nonpmtn}^m$, respectively. Similarly, let $E_{pmtn}^{m,j}$ and $E_{nonpmtn}^{m,j}$ be the electricity cost of job $j$ in $\sigma_{pmtn}^m$ and $\sigma_{nonpmtn}^m$, respectively. We will prove that $E_{nonpmtn}^{m,j} \le \beta_m E_{pmtn}^{m,j}$ for all $j \in \mathcal{J}$ and $m = 1, \ldots, K$, which implies that the performance guarantee of Algorithm 4.3 is at most $\max_{1 \le m \le K} \beta_m$.

Suppose job $j \in \mathcal{J}$ is processed in periods $\ell_1, \ldots, \ell_{h_j} \in \mathcal{P}_m$ in $\sigma_{pmtn}^m$. Let $T_{j\ell_k}$ be the processing time of job $j$ in period $\ell_k$, and $s_{\ell_k}^*$ be the optimal speed of period $\ell_k$ in $\sigma_{pmtn}^m$ for $k = 1, \ldots, h_j$. Also, let $T_{j\ell_{k'}} = \max_{k=1,\ldots,h_j} T_{j\ell_k}$ be the maximum processing time of job $j$ within the $h_j$ periods. See Figure 4.3 for a diagram. Algorithm 4.3 keeps the position of job $j$ in period $\ell_{k'}$ unchanged, and processes the entire job $j$ in this position. Therefore, $E_{pmtn}^{m,j} = \sum_{i=1}^{h_j} c_{\ell_i} T_{j\ell_i} (s_{\ell_i}^*)^\alpha$ and $E_{nonpmtn}^{m,j} = c_{\ell_{k'}} T_{j\ell_{k'}} (s'_j)^\alpha$, where $s'_j = w_j/T_{j\ell_{k'}}$. Since $\sum_{i=1}^{h_j} T_{j\ell_i} s_{\ell_i}^* = T_{j\ell_{k'}} s'_j$ and $T_{j\ell_{k'}} \ge T_{j\ell_i}$ for $i = 1, \ldots, h_j$, we have

$$s'_j \le \sum_{i=1}^{h_j} s_{\ell_i}^* \le h_j \max_{k \in \mathcal{P}_m} \{s_k^*\} \le m \max_{k \in \mathcal{P}_m} \{s_k^*\}.$$

Note that $c_{\ell_{k'}} T_{j\ell_{k'}} \le \sum_{i=1}^{h_j} c_{\ell_i} T_{j\ell_i}$, and so

$$\begin{aligned} \frac{E_{nonpmtn}^{m,j}}{E_{pmtn}^{m,j}} &\le \frac{c_{\ell_{k'}} T_{j\ell_{k'}} (m \max_{k \in \mathcal{P}_m} \{s_k^*\})^\alpha}{\sum_{i=1}^{h_j} c_{\ell_i} T_{j\ell_i} (\min_{k \in \mathcal{P}_m} \{s_k^*\})^\alpha} \\ &= \frac{(m \max_{k \in \mathcal{P}_m} \{s_k^*\})^\alpha}{(\min_{k \in \mathcal{P}_m} \{s_k^*\})^\alpha} \cdot \frac{c_{\ell_{k'}} T_{j\ell_{k'}}}{\sum_{i=1}^{h_j} c_{\ell_i} T_{j\ell_i}} \le m^\alpha \Big( \frac{\max_{k \in \mathcal{P}_m} \{s_k^*\}}{\min_{k \in \mathcal{P}_m} \{s_k^*\}} \Big)^\alpha. \end{aligned}$$

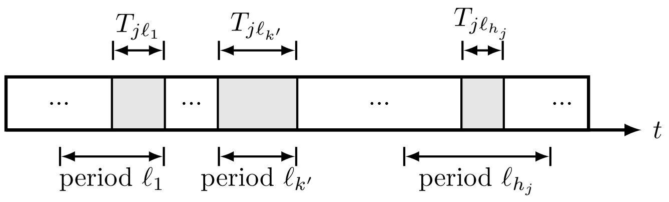

Figure 4.3: Job $j$ (the grey part) in a preemptive schedule $\sigma^m_{pmtn}$.

By Lemma 4.2 and Corollary 4.4, we know that

$$\frac{\max_{k\in\mathcal{P}_m}\{s^*_k\}}{\min_{k\in\mathcal{P}_m}\{s^*_k\}} = \Big(\frac{\max_{k\in\mathcal{P}_m}\{c_k\}}{\min_{k\in\mathcal{P}_m}\{c_k\}}\Big)^{1/(\alpha-1)}.$$

Therefore, by Lemma 4.8 we have

$$\frac{E^m_{nonpmtn}}{E_{opt}} \leq \frac{E^m_{nonpmtn}}{E_K} \leq \frac{E^m_{nonpmtn}}{E_m}\cdot\Big(\frac{K}{m}\Big)^{\alpha-1} \leq m^\alpha\Big(\frac{\max_{k\in\mathcal{P}_m}\{c_k\}}{\min_{k\in\mathcal{P}_m}\{c_k\}}\Big)^{\alpha/(\alpha-1)}\Big(\frac{K}{m}\Big)^{\alpha-1} = \beta_m,$$

where $E_{opt}$ is the total electricity cost of an optimal schedule for Problem S, and $E_m$ is the cost of an optimal preemptive schedule for the restriction of Problem S-pmtn in which jobs can only be processed in $m$ time periods, for $m = 1, \dots, K$. □

The main idea of Algorithm 4.2 is to transform an optimal preemptive schedule into a non-preemptive schedule while maintaining the time used to process each job. As a result, some jobs in the non-preemptive schedule end up being processed in the periods with high electricity prices. In order to exploit the varying electricity prices across periods, Algorithm 4.3 chooses to process all of the jobs in only a subset of time periods. However, there are still shortcomings in Algorithm 4.3 since the time used to process each job is shortened in the corresponding non-preemptive schedule. It turns out that neither algorithm strictly dominates the other, as the following example demonstrates.

**Example 4.1.** Consider the following instance with 1 job, and a two-period TOU tariff scheme in which $d_1 = d_2 = d$. Let $E_{algo2}$ and $E_{algo3}$ be the total electricity cost of the schedules output by Algorithms 4.2 and 4.3 respectively.

First, consider the case in which $c_2 = c_1$. In this case, $E_{algo2} = c_1(2d)\big(W/(2d)\big)^\alpha$ and $E_{algo3} = c_1 d\big(W/d\big)^\alpha$, and so $E_{algo3}/E_{algo2} = 2^{\alpha-1}$. In other words, Algorithm 4.2 performs better than Algorithm 4.3 in this case.

Now, consider the case in which $c_2 = 2^\alpha(2^\alpha - 1)c_1$. In this case, $E_{algo2} = c_1 d\big(W/(2d)\big)^\alpha + c_2 d\big(W/(2d)\big)^\alpha$ and $E_{algo3} = c_1 d\big(W/d\big)^\alpha$, and so

$$\frac{E_{algo2}}{E_{algo3}} = \frac{4^\alpha - 2^\alpha + 1}{2^\alpha} > 1.$$

That is, Algorithm 4.3 performs better than Algorithm 4.2 in this case.

For some special cases, we can take advantage of the benefits of both algorithms. For example, when the TOU tariffs satisfy $d_1/c_1^{1/(\alpha-1)} \geq \cdots \geq d_K/c_K^{1/(\alpha-1)}$, we know that $\mathcal{P}_m = \{1, 2, \ldots, m\}$ for any $m = 1, \ldots, K$. This happens, for instance, when $d_k = d$ for $k \in \mathcal{P}$ and $c_k$ is nondecreasing in $k$. As a result, when $m$ periods are used to process all of the jobs, these periods are consecutive, and so there exists a preemptive schedule in which jobs are processed contiguously, which Algorithm 4.1 constructs. Based on this idea, we propose the following algorithm.

**Algorithm 4.4** Approximation algorithm for Problem S with $d_1/c_1^{1/(\alpha-1)} \geq \cdots \geq d_K/c_K^{1/(\alpha-1)}$

```
1: Define $\mathcal{P}_m$ and $\sigma^m_{pmtn}$ as in Algorithm 4.3.
2: Define $T_j$ as in Algorithm 4.2.
3: for $m = 1$ to $K$ do
4:     for $j = 1$ to $n$ do
5:         Keep the position of job $j$ unchanged, and process job $j$ with uniform speed $s'_j = w_j/T_j$.
6:     Denote this non-preemptive schedule as $\sigma^m_{nonpmtn}$ and its total electricity cost as $E^m_{nonpmtn}$.
7: $m^* \leftarrow \arg\min_{1 \leq m \leq K}\{E^m_{nonpmtn}\}$.
8: Output schedule $\sigma^{m^*}_{nonpmtn}$.
```

The following theorem follows directly from Theorem 4.7 and Lemma 4.8.

**Theorem 4.10.** *Algorithm 4.4 is a* $\min_{m=1,\ldots,K} \mu_m$*-approximation algorithm for Problem S with* $d_1/c_1^{1/(\alpha-1)} \geq \cdots \geq d_K/c_K^{1/(\alpha-1)}$*, where*

$$\mu_m = \left(\frac{K}{m}\right)^{\alpha-1} \sum_{k=1}^{m} \left(\frac{c_k}{\min_{\ell=1,\ldots,m}\{c_\ell\}}\right)^{\alpha/(\alpha-1)} \quad \text{for } m = 1, \ldots, K.$$

## 4.4 Computational experiments

We empirically tested the performance of Algorithms 4.2, 4.3 and 4.4 on randomly generated instances of Problem S. Such testing gives us additional insight into how these algorithms may behave in practice, especially because the worst-case performance guarantees we have been able to obtain for all of these algorithms are not constant and depend on the number of periods as well as the electricity prices. We conducted two experiments. The first experiment examined the impact of varying electricity prices from period to period on the behavior of the three algorithms. The second experiment compared the performance of Algorithms 4.2 and 4.3.

Since it is difficult in general to obtain optimal schedules for instances of Problem S, to evaluate the performance of Algorithms 4.2, 4.3 and 4.4, we used the total electricity cost $E_{pmtn}$ of an optimal schedule for Problem S-pmtn as a lower bound on the total electricity cost $E_{opt}$ of an optimal schedule for Problem S.

To perform these experiments, we used the Python programming language to implement all the algorithms and computations on a computer with a 1.6GHz Intel Core i5 processor and 2GB of RAM running the OS X 10.10.2 operating system. For simplicity, we used $\alpha = 3$ in all our experiments.

### 4.4.1 Experiment 1: the effect of varying electricity prices

In this experiment, we studied the impact of varying electricity prices as well as the number of jobs in a two-period TOU tariff scheme. For simplicity, we let $\theta = c_2/c_1$. We considered the following combinations of the value of $\theta$ and the number of jobs $n$:

$$\{(\theta, n) : \theta \in \{1, 4, 64, 256, 2048, 32768\}, n \in \{5, 10, 20, 40, 70, 150\}\}.$$

We randomly generated 50 instances for each combination $(\theta, n)$, for a total of $6 \times 6 \times 50 = 1800$ instances. Each instance consists of durations $d_1, d_2$, and workload $w_1, \ldots, w_n$; each of these quantities was randomly generated from the uniform distribution on $\{1, \ldots, 20\}$. Note that $c_1 \leq c_2$. All of these quantities were generated independently, except that we required $d_1 \geq d_2$ to ensure $d_1/c_1^{1/(\alpha-1)} \geq d_2/c_2^{1/(\alpha-1)}$, so that we could implement Algorithm 4.4.

Let $E_{algo2}$, $E_{algo3}$ and $E_{algo4}$ denote the total electricity cost of the schedule computed by Algorithms 4.2, 4.3 and 4.4, respectively. The CPU time for Algorithms 4.2, 4.3 and 4.4 were minimal: the entire experiment – all 1800 instances – took less than 1 minute to run.

Table A.1 (on page 36) shows the average values of $E_{algo2}/E_{pmtn}$, $E_{algo3}/E_{pmtn}$, and $E_{algo4}/E_{pmtn}$ over all instances with $\theta \in \{1, 4, 64, 256, 2048, 32768\}$. Table A.1 shows that when $\theta$ was small (i.e. $\theta = 1, 4$), Algorithm 4.2 performed better than Algorithm 4.3. On the other hand, when $\theta$ became

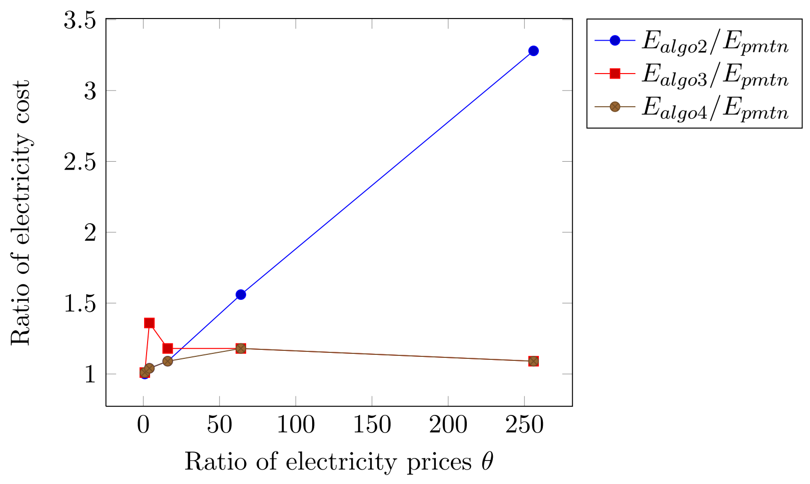

Figure 4.4: Ratios of electricity costs obtained by Algorithms 4.2, 4.3 and 4.4 for a particular set of 10 jobs.

large enough, Algorithm 4.2 performed rather poorly, while Algorithm 4.3 obtained schedules with near-optimal total electricity costs. In addition, when $\theta$ was small, Algorithm 4.4 performed similarly to Algorithm 4.2, but when $\theta$ became larger, Algorithm 4.4 performed similarly to Algorithm 4.3. In addition, Algorithm 4.4 always performed better than both Algorithms 4.2 and 4.3 for these specially structured randomly generated instances.

For the purposes of illustration, we also compared the electricity costs obtained by Algorithms 4.2, 4.3, 4.4 and the corresponding optimal preemptive schedule directly for a specific set of 10 jobs for $\theta \in \{1, 4, 16, 64, 256\}$; see Figure 4.4. This figure shows that when $\theta$ was small, Algorithm 4.2 performed better than Algorithm 4.3. When $\theta$ became large enough, the total electricity cost of the schedule obtained by Algorithm 4.2 increased dramatically, while Algorithms 4.3 and 4.4 obtained schedules with near-optimal total electricity costs.

In addition, we observed that the number of jobs also affected the performance of these three algorithms. To illustrate, Figures 4.5 and 4.6 show how the average ratio between the total electricity costs obtained by the algorithms and an optimal preemptive schedule changes as the number of jobs changes when $\theta$ is small ($\theta = 4$) and large ($\theta = 2048$). The larger the number of jobs, the lower the ratio, although the effect of the number of jobs on the particular algorithms appeared to depend considerably on the value of $\theta$.

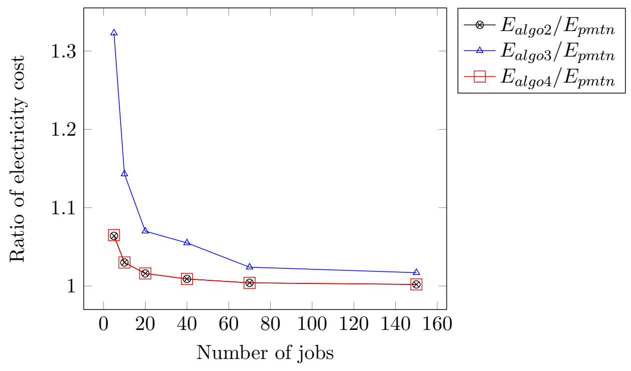

Figure 4.5: Average ratio between total electricity costs of schedules obtained by Algorithms 4.2, 4.3, 4.4 and an optimal preemptive schedule, for instances with $\theta = 4$.

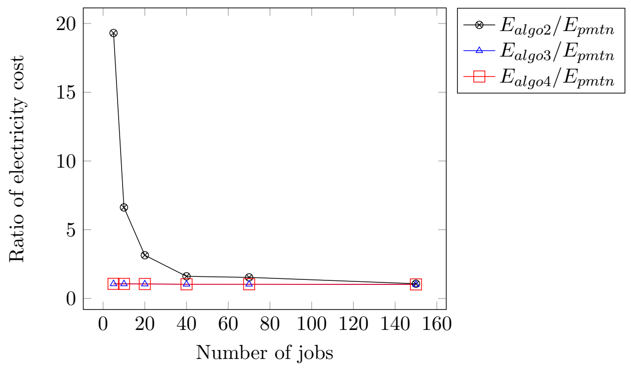

Figure 4.6: Average ratio between total electricity costs of schedules obtained by Algorithms 4.2, 4.3, 4.4 and an optimal preemptive schedule, for instances with $\theta = 2048$.

### 4.4.2 Experiment 2: Algorithm 4.2 vs. Algorithm 4.3

From Experiment 1, we found that the variance of the electricity prices played an important role in the performance of Algorithms 4.2, 4.3 and 4.4. In Experiment 2, we focused on their performance on general instances, not just instances in which $d_1/c_1^{1/(\alpha-1)} \geq \cdots \geq d_K/c_K^{1/(\alpha-1)}$. This disqualified Algorithm 4.4 from consideration in this experiment.

In this experiment, we considered the following combinations of the number of periods $K$ and the number of jobs $n$:

$$\{(K, n) : K \in \{2, 6, 10, 20, 30, 50\}, n \in \{5, 10, 20, 40, 70, 150\}\}.$$

We randomly generated 50 instances for each combination $(K, n)$, for a total of $6 \times 6 \times 50 = 1800$ instances. In each instance, $d_1, \ldots, d_K$ were randomly generated from the uniform distribution on $\{1, \ldots, 20\}$, and electricity prices $c_1, \ldots, c_K$ were generated from the uniform distribution on $[0.05, 1]$. In addition, workloads $w_1, \ldots, w_n$ were randomly generated from the uniform distribution on $\{1, \ldots, 20\}$. All quantities were generated independently. Note that $0.05 \leq c_k \leq 1$ for $k = 1, \ldots, K$; that is, the ratio of electricity prices between any two different periods can be at most 20. This is reasonable, since most of the TOU tariffs are pre-determined legislatively by the states within a fixed range of prices to ensure the stability of the retail electricity market and reduce customer resistance on price uncertainty (Braithwait et al. 2007). Again, the CPU time used to run Algorithms 4.2 and 4.3 was minimal; the entire experiment – all 1800 instances – took less than 1 minute to run.

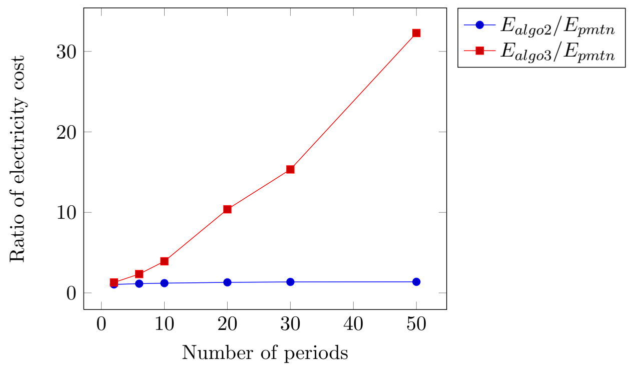

Figure 4.7: Average ratio between total electricity costs of schedules obtained by Algorithms 4.2, 4.3 and an optimal preemptive schedule, for instances with $n = 5$.

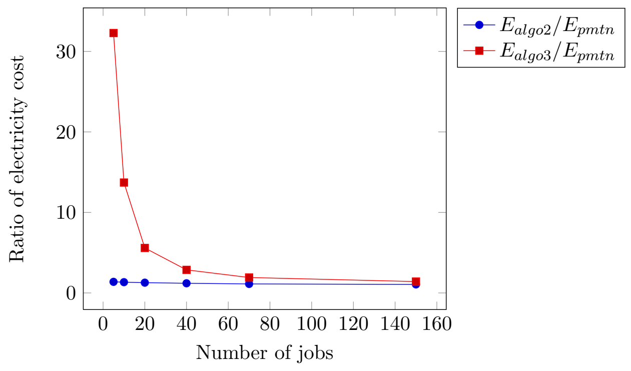

Figure 4.8: Average ratio between total electricity costs of schedules obtained by Algorithms 4.2, 4.3 and an optimal preemptive schedule, for instances with $K = 50$.

Tables A.2 and A.3 report values of $E_{algo2}/E_{pmtn}$ and $E_{algo3}/E_{pmtn}$: the ratio between the total electricity costs of the schedules obtained by Algorithms 4.2 and 4.3 and an optimal preemptive schedule. We found that the maximum average ratio between the total electricity costs of a schedule obtained by Algorithm 4.2 and an optimal preemptive schedule was 1.375, while the same ratio for Algorithm 4.3 was 32.287. Although the performance guarantee of Algorithm 4.2 is not constant, it provided good results on instances in which the ratios of electricity prices between different periods were within a small bounded range. Table A.4 gives values of $E_{algo3}/E_{algo2}$: the ratio between the total electricity costs of schedules obtained by Algorithms 4.2 and 4.3. Table A.4 shows that these average ratios were greater than 1 for all combinations of $(K, n)$, which suggests that Algorithm 4.2 readily outperformed Algorithm 4.3 in our empirical testing.

We also found that the number of periods and the number of jobs also affected the performance of these two algorithms: for a fixed number of jobs, as the number of periods increased, the quality of the schedules output by these algorithms decreased; see Figure 4.7. On the other hand, for a fixed number of periods, as the number of jobs increased, the quality of the schedules output by these algorithms increased; see Figure 4.8.

## 5 Conclusion

To the best of our knowledge, our paper is one of the first to consider a single machine scheduling problem with the objective of minimizing total electricity cost under time-of-use tariffs. We considered the problem when jobs must be processed on a uniform-speed machine (i.e. Problems U, U-pmtn and U-pyr) and when jobs must be processed on a speed-scalable machine (i.e. Problems S-pmtn and S). We showed that Problem U is strongly NP-hard and in fact inapproximable within a constant factor, unless P = NP. We also gave exact polynomial-time algorithms for Problem U-pmtn, in which jobs can be processed preemptively, and Problem U-pyr, in which all the jobs have the same workload and the electricity prices follow a pyramidal structure. In addition, we proved that Problem S is strongly NP-hard, while Problem S-pmtn is solvable in polynomial time. Finally, we proposed different approximation algorithms for Problem S (i.e. Algorithms 4.2, 4.3 and 4.4), and empirically tested the performance of these approximation algorithms on randomly generated instances.

There are many directions for future research that extend from this work. One natural direction would be to investigate whether there exists an approximation algorithm for Problem S with a constant performance guarantee. Another direction would be to consider minimizing total electricity costs under TOU tariffs for other scheduling environments. It would also be interesting to investigate whether results for energy minimization problems on identical or heterogeneous parallel machines, possibly with migratable jobs (e.g. Bampis et al. 2013; Cohen-Addad et al. 2014), can be adapted to Problem S: machines in these environments correspond naturally to periods in Problem S, but constructing a transformation that captures period durations and the definition of non-preemption in used in this work appears to be non-trivial.

## References

S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010.

S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms*, 3(4), 2007.

A. Antoniadis and C. Huang. Non-preemptive speed scaling. *Journal of Scheduling*, 16(4):385–394, 2013.

A. Antoniadis, P. Kling, S. Ott, and S. Riechers. Speed scaling with variable electricity rates and speed limits. In *Proceedings of the 12th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2015)*, 2015.

E. Bampis, D. Letsios, I. Milis, and G. Zois. Speed scaling for maximum lateness. In J. Gudmundsson, J. Mestre, and T. Viglas, editors, *18th International Conference on Computing and Combinatorics*

*(COCOON 2012)*, volume 7434 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2012.

E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, and M. Sviridenko. Energy efficient scheduling and routing via randomized rounding. In A. Seth and N. K. Vishnoi, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *Leibniz International Proceedings in Informatics*, pages 449–460. Schloss Dagstuhl, 2013.

E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, and I. Nemparis. From preemptive to non-preemptive speed-scaling scheduling. *Discrete Applied Mathematics*, 181:11–20, 2015.

N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow time. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA07)*, pages 805–813, 2007.

S. Braithwait, D. Hansen, and M. O'Sheasy. Retail electricity pricing and rate design in evolving markets. Technical report, Edison Electric Institute, 2007.

D.M. Brooks, P. Bose, S.E. Schuster, H. Jacobson, P.N. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P.W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *Micro, IEEE*, 20(6):26–44, 2000.

P.M. Castro, I. Harjunkoski, and I.E. Grossmann. New continuous-time scheduling formulation for continuous plants under variable electricity cost. *Industrial & Engineering Chemistry Research*, 48(14):6701–6714, 2009.

V. Cohen-Addad, Z. Li, C. Mathieu, and I. Milis. Energy-efficient algorithms for non-preemptive speed-scaling. In *Proceedings of the 12th Workshop on Approximation and Online Algorithms (WAOA 2014)*, 2014.

P. Detti, C. Hurkens, A. Agnetis, and G. Ciaschetti. Optimal packet-to-slot assignment in mobile telecommunications. *Operations Research Letters*, 37(4):261–264, 2009.

K. Fang, N.A. Uhan, F. Zhao, and J.W. Sutherland. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems*, 30(4): 234–240, 2011.

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

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.

S. Irani and K.R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

R. Kannan and S. Wei. Approximation algorithms for power-aware scheduling of wireless sensor networks with rate and duty-cycle constraints. In P.B. Gibbons, T. Abdelzaher, J. Aspnes, and R. Rao, editors, *2nd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2006)*, volume 4026 of *Lecture Notes in Computer Science*, pages 463–479. Springer, 2006.

J. Kulkarni and K. Munagala. Algorithms for cost-aware scheduling. In T. Erlebach and G. Persiano, editors, *10th International Workshop on Approximation and Online Algorithms (WAOA 2012)*, volume 7846 of *Lecture Notes in Computer Science*, pages 201–214. Springer, 2013.

S. Mitra, I.E. Grossmann, J.M. Pinto, and N. Arora. Optimal production planning under time-sensitive electricity prices for continuous power-intensive processes. *Computers & Chemical Engineering*, 38:171–184, 2012.

J.-Y. Moon, K. Shin, and J. Park. Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. *International Journal of Advanced Manufacturing Technology*, 68(1-4):523–535, 2013.

G. Mouzon, M.B. Yildirim, and J. Twomey. Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research*, 45 (18-19):4247–4271, 2007.

T. Mudge. Power: A first-class architectural design constraint. *Computer*, 34(4):52–58, 2001.

C.W. Park, K.S. Kwon, W.B. Kim, B.K. Min, S.J. Park, I.H. Sung, Y.S. Yoon, K.S. Lee, J.H. Lee, and J. Seok. Energy consumption reduction technology in manufacturing – A selective review of policies, standards, and research. *International Journal of Precision Engineering and Manufacturing*, 10(5):151–173, 2009.

S. Shapiro and J. Tomain. Rethinking reform of electricity markets. *Wake Forest Law Review*, 40: 497–543, 2005.

C. Subai, P. Baptiste, and E. Niel. Scheduling issues for environmentally responsible manufacturing: The case of hoist scheduling in an electroplating line. *International Journal of Production Economics*, 99(1):74–87, 2006.

Z. Sun, S. Biller, F. Gu, and L. Li. Energy consumption reduction for sustainable manufacturing systems considering machines with multiple-power states. In *ASME 2011 International Manufacuring Science and Engineering Conference*, Corvallis, Oregon, USA, June 13-17 2011.

G. Wan and X. Qi. Scheduling with variable time slot costs. *Naval Research Logistics*, 57(2):159–171, 2010.

J. Wang, J. Li, and N. Huang. Optimal scheduling to achieve energy reduction in automotive paint shops. In *2009 ASME Manufacturing Science and Engineering Conference*, West Lafayette, Indiana, USA, October 4-7 2009.

F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 374–382, 1995.

## A Tables of results from computational experiments

| Average ratio of total electricity costs | Statistics | $\theta = c_2/c_1$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 4 | 64 | 256 | 2048 | 32768 |
| $E_{algo2}/E_{pmtn}$ | Mean | 1.000 | 1.021 | 1.325 | 1.866 | 5.549 | 74.395 |
| | Std. dev. | 0.000 | 0.030 | 0.577 | 1.907 | 12.986 | 230.589 |
| | Minimum | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | Maximum | 1.000 | 1.184 | 5.015 | 13.734 | 100.655 | 1900.848 |
| $E_{algo3}/E_{pmtn}$ | Mean | 1.086 | 1.106 | 1.137 | 1.121 | 1.045 | 1.016 |
| | Std. dev. | 0.150 | 0.208 | 0.247 | 0.183 | 0.056 | 0.026 |
| | Minimum | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | Maximum | 2.067 | 2.995 | 3.240 | 2. 641 | 1.385 | 1.221 |
| $E_{algo4}/E_{pmtn}$ | Mean | 1.000 | 1.021 | 1.114 | 1.113 | 1.045 | 1.016 |
| | Std. dev. | 0.000 | 0.030 | 0.156 | 0.146 | 0.055 | 0.026 |
| | Minimum | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | Maximum | 1.000 | 1.184 | 2.183 | 1.806 | 1.385 | 1.221 |

Table A.1: Average ratio between the total electricity costs of schedules obtained by Algorithms 4.2, 4.3, 4.4 and an optimal preemptive schedule.

| $K$ | Statistics | $n$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 10 | 20 | 40 | 70 | 150 |
| 2 | Mean | 1.053 | 1.014 | 1.007 | 1.004 | 1.003 | 1.001 |
| | Std. dev. | 0.092 | 0.030 | 0.009 | 0.006 | 0.004 | 0.002 |
| 6 | Mean | 1.145 | 1.080 | 1.046 | 1.022 | 1.015 | 1.007 |
| | Std. dev. | 0.138 | 0.069 | 0.049 | 0.024 | 0.018 | 0.005 |
| 10 | Mean | 1.211 | 1.127 | 1.071 | 1.036 | 1.020 | 1.009 |
| | Std. dev. | 0.130 | 0.099 | 0.041 | 0.023 | 0.015 | 0.010 |
| 20 | Mean | 1.308 | 1.238 | 1.137 | 1.080 | 1.041 | 1.023 |
| | Std. dev. | 0.142 | 0.132 | 0.077 | 0.040 | 0.021 | 0.013 |
| 30 | Mean | 1.363 | 1.296 | 1.211 | 1.116 | 1.071 | 1.032 |
| | Std. dev. | 0.160 | 0.134 | 0.085 | 0.046 | 0.031 | 0.012 |
| 50 | Mean | 1.375 | 1.336 | 1.274 | 1.198 | 1.117 | 1.056 |
| | Std. dev. | 0.094 | 0.091 | 0.094 | 0.073 | 0.030 | 0.017 |

Table A.2: Average ratio between total electricity costs of schedules obtained by Algorithm 4.2 and an optimal preemptive schedule.

| $K$ | Statistics | $n$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 10 | 20 | 40 | 70 | 150 |
| 2 | Mean | 1.298 | 1.106 | 1.127 | 1.037 | 1.023 | 1.015 |
| | Std. dev. | 0.369 | 0.113 | 0.136 | 0.043 | 0.028 | 0.021 |
| 6 | Mean | 2.338 | 1.609 | 1.331 | 1.176 | 1.102 | 1.053 |
| | Std. dev. | 0.750 | 0.291 | 0.156 | 0.095 | 0.059 | 0.029 |
| 10 | Mean | 3.926 | 2.117 | 1.563 | 1.282 | 1.177 | 1.085 |
| | Std. dev. | 1.894 | 0.397 | 0.160 | 0.081 | 0.048 | 0.033 |
| 20 | Mean | 10.384 | 4.172 | 2.288 | 1.600 | 1.317 | 1.160 |
| | Std. dev. | 3.779 | 1.055 | 0.293 | 0.145 | 0.065 | 0.034 |
| 30 | Mean | 15.339 | 6.780 | 3.229 | 1.925 | 1.518 | 1.244 |
| | Std. dev. | 4.864 | 1.441 | 0.412 | 0.153 | 0.079 | 0.040 |
| 50 | Mean | 32.287 | 13.713 | 5.584 | 2.865 | 1.914 | 1.411 |
| | Std. dev. | 14.796215 | 4.490 | 0.941 | 0.284 | 0.132 | 0.052 |

Table A.3: Average ratio between total electricity costs of schedules obtained by Algorithm 4.3 and an optimal preemptive schedule.

| $K$ | Statistics | $n$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 10 | 20 | 40 | 70 | 150 |
| 2 | Mean | 1.226 | 1.091 | 1.118 | 1.033 | 1.020 | 1.014 |
| | Std. dev. | 0.284 | 0.116 | 0.127 | 0.039 | 0.026 | 0.019 |
| 6 | Mean | 2.032 | 1.491 | 1.272 | 1.149 | 1.085 | 1.047 |
| | Std. dev. | 0.561 | 0.252 | 0.133 | 0.078 | 0.052 | 0.027 |
| 10 | Mean | 3.234 | 1.886 | 1.461 | 1.238 | 1.154 | 1.075 |
| | Std. dev. | 1.423 | 0.350 | 0.151 | 0.081 | 0.044 | 0.025 |
| 20 | Mean | 8.060 | 3.397 | 2.016 | 1.481 | 1.265 | 1.134 |
| | Std. dev. | 3.032 | 0.859 | 0.258 | 0.120 | 0.066 | 0.032 |
| 30 | Mean | 11.528 | 5.306 | 2.676 | 1.724 | 1.418 | 1.205 |
| | Std. dev. | 4.116 | 1.342 | 0.382 | 0.120 | 0.071 | 0.035 |
| 50 | Mean | 23.871 | 10.360 | 4.410 | 2.393 | 1.713 | 1.337 |
| | Std. dev. | 11.788 | 3.562 | 0.849 | 0.209 | 0.111 | 0.048 |

Table A.4: Average ratio between total electricity costs of schedules obtained by Algorithm 4.3 and Algorithm 4.2.