Scheduling on a two-machine permutation flow shop under time-of-use electricity tariffs

Shijin Wang, Zhanguo Zhu, Kan Fang, Feng Chu, Chengbin Chu

Scheduling on a two-machine permutation flow shop under time-of-use electricity tariffs

Shijin Wang[a], Zhanguo Zhu[b], Kan Fang[c], Feng Chud[e] and Chengbin Chua[f]

[a]School of Economics and Management, Tongji University, Shanghai, China
[b]College of Economics and Management, Nanjing Agricultural University, Nanjing, China
[c]College of Management and Economics, Tianjin University, Tianjin, China
[d]Laboratory IBISC, University of Evry-Val dÉssonne, Evry, France
[e]Management Engineering Research Center, Xihua University, Chengdu, China
[f]Laboratoire Génie Industriel, Centrale Supélec, Université Paris-Saclay, Châtenay-Malabry, France

We consider a two-machine permutation flow shop scheduling problem to minimise the total electricity cost of processing jobs under time-of-use electricity tariffs. We formulate the problem as a mixed integer linear programming, then we design two heuristic algorithms based on Johnson's rule and dynamic programming method, respectively. In particular, we show how to find an optimal schedule using dynamic programming when the processing sequence is fixed. In addition, we propose an iterated local search algorithm to solve the problem with problem-tailored procedures and move operators, and test the computational performance of these methods on randomly generated instances.

Keywords: scheduling; flow shop; electricity cost; dynamic programming; iterated local search

1. Introduction

Due to the increasing energy cost, growing public concerns about climate change, and the gradual implementation of new taxes and regulations related to carbon emissions, energy consumption management has been a primary for manufacturing enterprises to succeed in a competitive market. To create energy efficient manufacturing systems, various approaches have been proposed in both academia and industry, such as developing machines and equipments that are more energy efficient, designing products with recycled materials, and so on. However, these energy saving strategies mainly focus on the machine and product levels, and usually require considerable large capital investments as well as a comparative long time to implement, which makes many companies, especially small and medium ones, not able to afford the cost of machine reconfiguration and process re-engineering (Ding, Song, and Wu 2016b).

During the past decade, researchers have observed that the energy consumption of manufacturing enterprises can also be effectively reduced using alternative *operational* strategies, such as scheduling jobs in a 'smarter' way to avoid extra energy consumption caused by machine start-up and/or machine idling. Compared with the strategies at machine and product levels, the operational strategies have the benefit of requiring fairly small capital investments and little time on replacing and installing equipments, and have become one of the most attractive research areas in sustainable manufacturing. We give a few examples. Mouzon, Yildirim, and Twomey (2007) considered a CNC machine in a shop making small aircraft parts and investigated scheduling jobs on this machine to minimise total energy consumption, in which the energy consumption on machine idling, start-up, and shutdown were analysed. Fang et al. (2011, 2013) considered scheduling jobs on flow shop machines to minimise the peak power load, energy consumption and carbon footprint, in which jobs can be processed at varying speeds. Dai et al. (2013) studied a bi-objective flexible flow shop scheduling problem to simultaneously minimise the makespan and energy consumption, in which they proposed an energy-saving model to determine on/off status of machines after a certain amount of machine idling time. Mansouri, Aktas, and Besikci (2016) considered a two-machine flow shop scheduling problem with variable processing times to minimise the makespan and the total energy consumption. Che et al. (2017a) investigated the problem of scheduling jobs on a single machine with power-down mechanism to minimise total energy consumption and maximum lateness. For further pointers on the ever-growing literature on energy-efficient scheduling in manufacturing, we refer the reader to the comprehensive surveys by Giret, Trentesaux, and Prabhu (2015) and Gahm et al. (2016).

On the other hand, we know that most manufacturing enterprises use electricity as their main energy source (Park et al. 2009), and the electricity cost 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. Fang et al. (2016) have shown that

*Corresponding author. Email: chengbin.chu@ecp.fr

minimising total energy consumption and minimising total electricity cost can be quite different, especially when varying pricing of electricity is implemented, such as *time-of-use (TOU) electricity tariffs*. Figure 1 shows the electricity market price for industry during the non-summer time in Shanghai. We can see that the electricity price varies hourly, the higher the customer demand during the peak period, the higher the electricity price charged on customers. Therefore, such price structure may provide a great opportunity for electricity-intensive manufacturing enterprises to save costs by shifting their electricity demand from peak hours to off-peak hours.

Recently, there has been an increasing amount of work on scheduling problems under time-varying electricity prices in both computer science and manufacturing, in which most of them focused on the single or parallel machine environments. For example, in the single machine environments, Kulkarni and Munagala (2013) studied the online problem of scheduling jobs on a single machine to minimise the total weighted flow time and electricity cost, in which the electricity prices can only take two values (low and high) and the power requirement of each job is unit-size. Shrouf et al. (2014) proposed a genetic algorithm to minimise total energy cost of scheduling jobs on a single machine with on/off switching under variable energy prices. Fang et al. (2016) investigated the complexity results of scheduling jobs on a single machine under TOU tariffs using the technology of dynamic speed scaling, and proposed different approximation algorithms to solve the problems. Che, Zeng, and Lyu (2016) proposed an efficient greedy insertion heuristic to minimise the total electricity cost of processing jobs on a single machine under TOU tariffs. Burcea et al. (2016) considered an offline scheduling problem on a single machine to minimise the total electricity cost, in which the power requirement and the duration of jobs are both unit-size, and each job can only be assigned to a predetermined subset of time slots. Antoniadis et al. (2017) considered the problem of scheduling jobs preemptively on a speed-scalable machine under variable electricity prices to minimise the total electricity cost, in which jobs have release dates, deadlines and speed limits. Cheng et al. (2017) studied a single machine batch scheduling problem with machine on/off switching under TOU tariffs to simultaneously minimise total electricity cost and makespan.

In the parallel machine environments, Ding et al. (2016a) proposed a mixed integer programming formulation for the problem of scheduling jobs on unrelated parallel machine under TOU tariffs to minimise the total electricity cost, and designed a column generation heuristic algorithm to solve the problem. Che, Zhang, and Wu (2017b) proposed a two-stage heuristic algorithm to schedule jobs on unrelated parallel machine under TOU tariffs to minimise the total electricity cost with bounded makespan. Zeng, Che, and Wu (2017) investigated a bi-objective scheduling on uniform parallel machines to minimise the total electricity cost and the number of machine used simultaneously under TOU tariffs, and developed an iterative search framework to obtain the Pareto fronts.

As observed by Panwalkar, Dudek, and Smith (1973), more than 77% of practical problems in manufacturing are rather close to some kind of *shop environments*, in which jobs often need to be processed on multiple machines in some order. Thus, much of the previous studies on single and parallel machines may not be directly applicable to the shop scheduling problems faced by manufacturing enterprises. According to our literature review, research on scheduling jobs in shop environments under TOU tariffs appears to be rather sparse. One exception is the work by Zhang et al. (2014), who considered scheduling jobs on flow shop machines to minimise total electricity cost while ensuring prefixed production throughput. However, since they used a time-indexed mixed integer program to search for the optimal solutions, only few instances that with fairly small number of jobs can be solved within a reasonable computational time.

In this work, we study a two-machine flow shop scheduling problem to minimise the total electricity cost under TOU tariffs, motivated by a practical shop floor that with the processes of smelting and casting of injection die for automobile components, in which the job casting process is high power-consumption and can take more than 50 hours. In particular, we assume that jobs must be processed non-preemptively in a permutation way. This is reasonable, since the injection die in the shop floor is very heavy (sometimes more than 10 tons), and it is very difficult for the manufacturing enterprises to interrupt the processing of a job, or to break the processing sequence of jobs across machines. For simplicity, we call this problem the *two-machine permutation flow shop scheduling problem with electricity costs*, or the TMPFSEC problem for short.

To the best of our knowledge, we are one of the first to study the TMPFSEC problem as defined here. To solve this problem, we first propose a mixed integer programming formulation (MILP) for the problem in Section 2. Then, in Section 3, we design heuristic algorithms to obtain feasible solutions for the TMPFSEC problem. In particular, when the job sequence is fixed, we provide a dynamic programming approach to search for the optimal schedules. Furthermore, in Section 4, we propose an iterated local search method to tackle with the general case, and test the computational performance of our proposed methods on randomly generated instances in Section 5. Finally, we present the conclusions and future research directions in Section 6.

2. Mathematical description of the problem

As we mentioned in the introduction, we refer to the problem that we study in this paper as the *two machine permutation flow shop scheduling problem with electricity costs* (or the TMPFSEC problem for short). An instance of the TMPFSEC problem

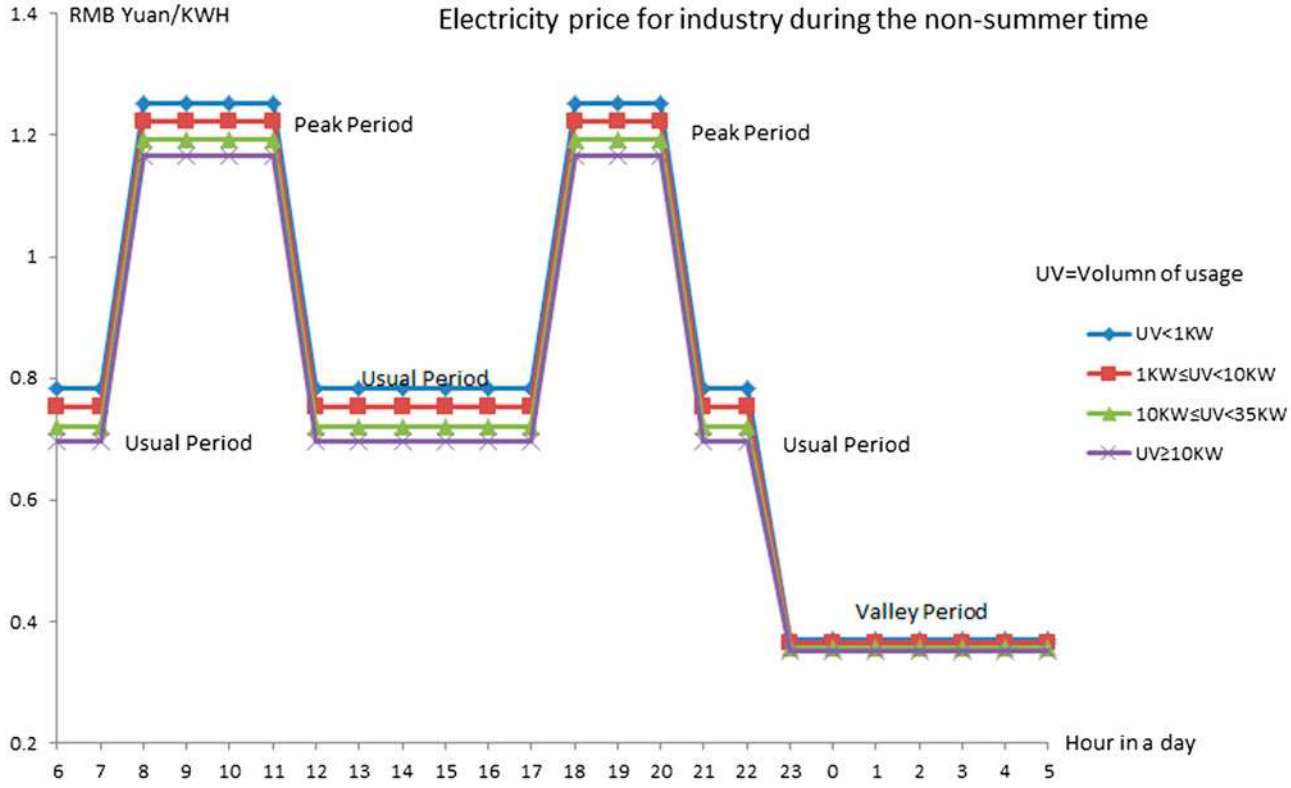

Figure 1. Electricity market price for industry during the non-summer time in Shanghai, China (Source: Shanghai Electricity Price, 2015, http://www.sh.sgcc.com.cn/).

consists of a set $\mathcal{J} = \{1, 2, \ldots, n\}$ of jobs and a set $\mathcal{M} = \{1, 2\}$ of machines. Each job $j \in \mathcal{J}$ on machine $i \in \mathcal{M}$ has a processing time p_{ij}, and must be processed non-preemptively first on machine 1, then on machine 2. In addition, there is a TOU tariff scheme, which is represented by a set $\mathcal{T} = \{1, 2, \ldots, T\}$ of time periods. Each period has an associated electricity price $c_t > 0$ per unit energy, and a duration of unit size, that is, period t is $[t-1, t)$ for $t = 1, 2, \ldots, T$. We denote b_i and d_i as the amount of energy consumed per unit time when machine i is busy (i.e. processing some job) and idle respectively. Moreover, we assume that each machine will start 'working' (busy or idle) at time instant 0, and can be shut down only if all the jobs on it have been completed. The objective of the TMPFSEC problem is to find a feasible schedule that minimises the total electricity cost E.

In what follows, we propose a mixed integer programming formulation for the TMPFSEC problem. We define the following decision variables:

- S_{ij} is the start time of job j on machine i;
- C_{ij} is the completion time of job j on machine i;
- x_{it} is equal to 1 if machine i is 'working' (either busy or idle) during time period t, and 0 otherwise (i.e. shutdown);
- y_{it} is equal to 1 if machine i is idle during time period t, and 0 otherwise;
- z_{ijt} is equal to 1 if job j is in processing on machine i during time period t, and 0 otherwise;
- δ_{jk} is equal to 1 if job j precedes job k, and 0 otherwise;
- F_i is the shutdown time of machine i.

Then the TMPFSEC problem can be modelled as follows:

$$\text{minimize} \quad \underbrace{\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} c_t b_i z_{ijt}}_{\text{busy energy consumption}} + \underbrace{\sum_{i\in\mathcal{M}}\sum_{t\in\mathcal{T}} c_t d_i y_{it}}_{\text{idle energy consumption}} \tag{2.1a}$$

$$\text{subject to} \quad \sum_{j\in\mathcal{J}} z_{ijt} \le 1 \qquad \text{for } i=1,2;\ t\in\mathcal{T}; \tag{2.1b}$$

$$\sum_{j\in\mathcal{J}} z_{ijt} + y_{it} = x_{it} \qquad \text{for } i=1,2;\ t\in\mathcal{T}; \tag{2.1c}$$

$$\sum_{t\in\mathcal{T}} z_{ijt} = p_{ij} \qquad \text{for } i=1,2;\ j\in\mathcal{J}; \tag{2.1d}$$

$$S_{ij} \le t + T(1-z_{ijt}) \qquad \text{for } i=1,2;\ j\in\mathcal{J};\ t\in\mathcal{T}; \tag{2.1e}$$

$$C_{ij} \ge t + 1 - T(1-z_{ijt}) \qquad \text{for } i=1,2;\ j\in\mathcal{J};\ t\in\mathcal{T}; \tag{2.1f}$$

$$C_{ij} = S_{ij} + p_{ij} \qquad \text{for } i=1,2;\ j\in\mathcal{J}; \tag{2.1g}$$

$$C_{ij} \le S_{ik} + T(1-\delta_{jk}) \qquad \text{for } i=1,2;\ j,k\in\mathcal{J},\ j\ne k; \tag{2.1h}$$

$$C_{ik} \le S_{ij} + T\delta_{jk} \qquad \text{for } i=1,2;\ j,k\in\mathcal{J},\ j\ne k; \tag{2.1i}$$

$$\delta_{jk} + \delta_{kj} = 1 \qquad \text{for } j,k\in\mathcal{J},\ j\ne k; \tag{2.1j}$$

$$C_{1j} \le S_{2j} \qquad \text{for } j\in\mathcal{J}; \tag{2.1k}$$

$$F_i \ge C_{ij} \qquad \text{for } i=1,2;\ j\in\mathcal{J}; \tag{2.1l}$$

$$F_i \le t + Tx_{it} \qquad \text{for } i=1,2;\ t\in\mathcal{T}; \tag{2.1m}$$

$$F_i \ge t + 1 - T(1-x_{it}) \qquad \text{for } i=1,2;\ t\in\mathcal{T}; \tag{2.1n}$$

$$F_i \le T \qquad \text{for } i=1,2; \tag{2.1o}$$

$$x_{it}, y_{it}, z_{ijt}, \delta_{jk} \in \{0,1\} \qquad \text{for } i=1,2;\ j,k\in\mathcal{J};\ t\in\mathcal{T}; \tag{2.1p}$$

The objective (2.1a) calculates the total electricity costs including the energy consumption when machines are busy and idle. Constraints (2.1b) ensure that each machine can at most process one job during each period. Constraints (2.1c) ensure that machine i is either busy or idle if the machine is not shut down. Constraints (2.1d) ensure that each job will be entirely processed. Constraints (2.1e)–(2.1g) ensure that jobs are processed non-preemptively. Constraints (2.1h)–(2.1j) are the disjunctive constraints between any two jobs j and k, which ensure that job j precedes job k or vice versa. Constraints (2.1k) ensure that jobs are processed in a flow shop way. Constraints (2.1l)–(2.1o) define the shutdown time on each machine. For simplicity, we denote the above model (2.1) as the MILP model.

3. Heuristic algorithms for the TMPFSEC problem

As we know, Fang et al. (2016) have shown that a special case of the TMPFSEC problem, i.e. the problem of scheduling jobs on a single machine under TOU tariffs, is strongly NP-hard, and in fact inapproximable within a constant factor. Thus, it is expected that medium and large size instances of the TMPFSEC problem cannot be efficiently solved by the MILP model proposed in Section 2 within a reasonable computational time. In this section, we develop a Johnson's rule-based heuristic algorithm, and a dynamic programming-based heuristic algorithm for the TMPFSEC problem.

3.1 *A Johnson's rule-based heuristic algorithm*

Note that most of the TOU tariffs implemented in reality often restrict the electricity prices within a fixed range by either the regulatory authorities or the electricity suppliers, so as to ensure the stability of the retail electricity market and reduce the customer resistance on price uncertainty (Braithwait et al. 2007). For example, in Figure 1 (on page 3175), we can see that the ratio of highest and lowest electricity market prices in Shanghai is less than 6. Moreover, as we mentioned before, each machine will start working at time instant 0, and can be shut down only if all the jobs on it have been completed. Thus, we expect that the smaller the makespan of the TMPFSEC problem, the lower the energy consumption caused by machine idling, which may be useful to reduce the total electricity cost when the variance of the electricity prices is within a small range.

It is well-known that Johnson's rule (i.e. SPT(1)-LPT(2)) can solve the problem of minimising makespan on a two-machine flow shop in $O(n \log n)$ time (Johnson 1954). Inspired by the above observations, we propose a Johnson's rule-based heuristic algorithm (Algorithm 3.1) for the TMPFSEC problem. For simplicity, we denote this algorithm as Algorithm JR, and the derived total electricity cost as E(JR).

Algorithm 3.1. A Johnson's rule-based heuristic for the TMPFSEC problem

1: Schedule jobs with Johnson's rule starting at time instant 0. Calculate the makespan $C^*_{\max}$.
2: **for** $t = 1$ to $T - C^*_{\max} + 1$ **do**
3: Start processing jobs at time instant $t-1$ (i.e. the beginning of period t) with Johnson's rule. Let $E(t)$ be the corresponding total electricity cost.
4: Calculate $t^* = \arg\min\{E(t) | 1 \le t \le T - C^*_{\max} + 1\}$.
5: Schedule jobs starting at time instant $t^* - 1$ with Johnson's rule. Output $E(t^*)$.

Note that when we start processing jobs at time instant $t-1$ (i.e. the beginning of period t), an additional amount of energy consumption $2(t-1)\sum_{i=1}^{2} d_i$ will be incurred by machine idling before period t. We define $\theta = \max\{c_t\}/\min\{c_t\}$, then it is easy to obtain the following results.

Theorem 3.1 *Algorithm 3.1 is a θ-approximation algorithm for the TMPFSEC problem.*

Proof. For simplicity, we denote $\mathrm{E}^{\mathrm{opt}}$ as the optimal electricity cost for the TMPFSEC problem, $\mathrm{E}^{\max}$ and $\mathrm{E}^{\min}$ as the optimal electricity cost when we set the electricity prices of all periods to $\max\{c_t\}$ and $\min\{c_t\}$, respectively. Obviously, we know that $\mathrm{E}^{\min} \le \mathrm{E}^{\mathrm{opt}} \le \mathrm{E(JR)} \le \mathrm{E}^{\max}$. As a result, we have

$$\frac{\mathrm{E(JR)}}{\mathrm{E}^{\mathrm{opt}}} \le \frac{\mathrm{E}^{\max}}{\mathrm{E}^{\min}} = \theta,$$

and our claim follows. □

3.2 *A dynamic programming-based heuristic algorithm*

In this section, we first propose a dynamic programming algorithm to search for the optimal schedule of the TMPFSEC problem when the job sequence is fixed, and then use it to derive a heuristic for the general TMPFSEC problem.

Let σ be any given job sequence, and $\mathrm{E}(\sigma)$ be the optimal electricity cost with job sequence σ. We define $\underline{C}_{i,\sigma(k)}$ and $\overline{C}_{i,\sigma(k)}$ as the lower and upper bounds for the completion time of the kth job in σ on machine i, respectively. Then we have the following:

$$\underline{C}_{1,\sigma(k)} = \sum_{\ell=1}^{k} p_{1,\sigma(\ell)} \qquad \text{for } k \in \mathcal{J}; \tag{3.1}$$

$$\underline{C}_{2,\sigma(1)} = \sum_{i=1}^{2} p_{i,\sigma(1)} \tag{3.2}$$

$$\underline{C}_{2,\sigma(k)} = \max\{\underline{C}_{2,\sigma(k-1)}, \underline{C}_{1,\sigma(k)}\} + p_{2,\sigma(k)} \qquad \text{for } k \in \mathcal{J}\backslash\{1\}; \tag{3.3}$$

$$\overline{C}_{2,\sigma(k)} = T - \sum_{\ell=k+1}^{n} p_{2,\sigma(\ell)} \qquad \text{for } k \in \mathcal{J}; \tag{3.4}$$

$$\overline{C}_{1,\sigma(k)} = \min\{\overline{C}_{1,\sigma(k+1)} - p_{1,\sigma(k+1)}, \overline{C}_{2,\sigma(k)}\} - p_{2,\sigma(k)} \qquad \text{for } k \in \mathcal{J}\backslash\{n\}; \tag{3.5}$$

$$\overline{C}_{1,\sigma(n)} = T - p_{2,\sigma(n)} \tag{3.6}$$

For the sake of simplicity, we define $B_{it} = b_i \sum_{k=1}^{t} c_k$ and $D_{it} = d_i \sum_{k=1}^{t} c_k$, where B_{it} (D_{it}) represents the total electricity cost incurred by machine i when it keeps busy (idle) until the end of period t. It is easy to see that when machine i keeps busy from time instant t to time instant t', then the corresponding electricity cost incurred by machine i within time interval $[t, t']$ can be determined as $B_{it'} - B_{it}$, similar results hold for the electricity cost incurred by machine idling.

Given the above notations, we are now able to propose our dynamic programming algorithm for the TMPFSEC problem with fixed job sequences. We define $\mathrm{E}(k, \phi, \psi)$ as the minimum total electricity cost of scheduling the first k jobs in a way such that the kth job is completed at time instants ϕ on machine 1 and ψ on machine 2, respectively, in other words, we have $C_{1,\sigma(k)} = \phi$, $C_{2,\sigma(k)} = \psi$. In particular, we set $\mathrm{E}(0, 0, 0) = 0$. Note that in order to obtain a feasible schedule, the following two conditions must be satisfied: (i) $\underline{C}_{1,\sigma(k)} \le \phi \le \overline{C}_{1,\sigma(k)}$; (ii) $\max\{\phi + p_{2,\sigma(k)}, \underline{C}_{2,\sigma(k)}\} \le \psi \le \overline{C}_{2,\sigma(k)}$.

We define $\Omega(k, \phi, \psi)$ as the set of possible completion time for the $(k-1)$th job on two machines when the kth job satisfies $C_{1,\sigma(k)} = \phi$, $C_{2,\sigma(k)} = \psi$, that is,

$$\begin{aligned}\Omega(k, \phi, \psi) = \ & (\phi', \psi') \big| \underline{C}_{1,\sigma(k-1)} \le \phi' \le \min\{\overline{C}_{1,\sigma(k-1)}, \phi - p_{1,\sigma(k)}\}, \\ & \max\{\phi' + p_{2,\sigma(k-1)}, \underline{C}_{2,\sigma(k-1)}\} \le \psi' \le \min\{\overline{C}_{2,\sigma(k-1)}, \psi - p_{2,\sigma(k)}\} \ . \end{aligned} \tag{3.7}$$

In particular, when $\Omega(k, \phi, \psi) = \emptyset$, we set $\mathrm{E}(k, \phi, \psi) = +\infty$.

Then, we can compute the entries of $E(k, \phi, \psi)$ using the following recurrence:

$$\begin{aligned}\mathrm{E}(k, \phi, \psi) = \min_{(\phi', \psi')} \ & B_{1\phi} - B_{1,\phi - p_{1,\sigma(k)}} + B_{2\psi} - B_{2,\psi - p_{2,\sigma(k)}} + D_{1,\phi - p_{1,\sigma(k)}} - D_{1\phi'} \\ & + D_{2,\psi - p_{2,\sigma(k)}} - D_{2\psi'} + \mathrm{E}(k-1, \phi', \psi') \big| (\phi', \psi') \in \Omega(k, \phi, \psi) \ , \end{aligned} \tag{3.8}$$

and the optimal electricity cost $\mathrm{E}(\sigma)$ can be obtained as follows.

$$\mathrm{E}(\sigma) = \min_{(\phi, \psi)} \big\{ \mathrm{E}(n, \phi, \psi) \ \text{Conditions (i) and (ii) are satisfied} \ .$$

It is easy to see that computing $\Omega(k, \phi, \psi)$ takes $O(T^2)$ time, thus the entire entries of $\mathrm{E}(k, \phi, \psi)$ can be computed in $O(nT^4)$ time. For simplicity, we denote the above dynamic algorithm for a fixed job sequence as Algorithm DP.

In what follows, we propose a dynamic programming-based heuristic algorithm (Algorithm 3.2) to generate feasible solutions for the TMPFSEC problem. For simplicity, we denote Algorithm 3.2 as Algorithm DPK, where K is the size of the generated job sequences, and the associated total electricity cost as E(DPK). In particular, when we enumerate all the possible job sequences and use Algorithm DP to search for optimal schedules for each job sequence, we denote this algorithm as Algorithm EDP.

Algorithm 3.2. A dynamic programming-based heuristic for the TMPFSEC problem

1: Given a fixed number K.
2: Determine the job sequence σ_1 when Johnson's rule is implemented.
3: Generate $K-1$ job sequences other than σ_1 randomly, and denote them as $\sigma_2, \ldots, \sigma_K$.
4: **for** $k = 1$ to K **do**
5: Use Algorithm DP to calculate $\mathrm{E}(\sigma_k)$.
6: Calculate $\sigma^* = \arg\min\{\mathrm{E}(\sigma_k)\}$.
7: Choose σ^* as the job sequence and schedule jobs in the way generated by Algorithm DP.

4. An iterated local search algorithm

We know that the iterated local search (ILS) algorithm is a simple yet quite powerful tool for improving the quality of successive local optima, and there has been a great deal of research on various problems using the ILS approaches, such as aircraft landing problem (Sabar and Kendall 2015), vehicle routing problem (Silva, Subramanian, and Ochi 2015) and team orienteering problem (Vansteenwegen et al. 2009). In this section, we propose an ILS method with problem-tailored procedures to solve the TMPFSEC problem. Similarly, we call our proposed ILS algorithm as Algorithm ILS, and the corresponding total electricity cost as E(ILS).

For completeness sake, we provide an algorithmic scheme for the basic ILS methods below (Algorithm 4.1). For more details about ILS methods, we refer the reader to the chapter by Lourenço et al. (2010).

From Algorithm 4.1, we can see that an ILS algorithm mainly includes four procedures: First, `GenerateInitial Solution` generates an initial solution s_0 as the starting point, and a `LocalSearch` is applied to the initial solution to obtain a local optimal solution s. Then, at each iteration, a `Perturbation` of the obtained local optimal solution is carried out to generate new starting point s'' by the `LocalSearch`. The generated solution s'' is accepted if the `AcceptanceCriterion` can be met. This process iterates until a given stopping criterion (e.g. reaching a prefixed number of iterations) is satisfied.

Given the framework of the basic ILS method, we now specify the main procedures which are used to implement our proposed ILS algorithm as follows.

Algorithm 4.1. General algorithmic outline for the basic ILS method

1: Given a fixed number of iterations K.
2: $s_0 \leftarrow$ `GenerateInitialSolution`.
3: $s \leftarrow$ `LocalSearch`(s_0).
4: **for** $k = 1$ to K **do**
5: $\quad s' \leftarrow$ `Perturbation`(s, history)
6: $\quad s'' \leftarrow$ `LocalSearch`(s')
7: $\quad s \leftarrow$ `AcceptanceCriterion`(s, s'', history)
8: Return the best solution.

4.1 *Initial solution generation*

Before describing the procedure about generating initial solutions, we first introduce the representation of a schedule which will be used in our algorithmic implementation. As we know, within a given time horizon $[0, T]$, the total processing time of jobs on machine 1 is $P_1 = \sum_{j=1}^{n} p_{1j}$, that is, there are $T - P_1$ periods during which no job is processed on machine 1. In our ILS algorithm, we impose that each job on machine 2 must be processed as early as possible, that is, $S_{2,\sigma(j)} = \max\{C_{1,\sigma(j)}, C_{2,\sigma(j-1)}\}$. As a result, we can see that once the schedule of jobs on machine 1 is fixed, then the schedule of jobs on machine 2 is also determined.

Therefore, we can only consider the schedule of jobs on machine 1, and represent it as a $(n + T - P_1)$-dimension vector, in which the elements of '0' denote the periods during which no job is processed on machine 1, and others denote the indices of jobs.

Example 4.1 To illustrate the above representation method of a schedule, we consider the following example. Let $\mathcal{J} = \{1, 2, 3\}$ and $p_{11} = 2$, $p_{12} = 2$, $p_{13} = 3$, $p_{21} = 3$, $p_{22} = 2$, $p_{23} = 2$. We also set $T = 14$ with electricity price vector $c = (2, 2, 3, 4, 2, 4, 3, 4, 2, 3, 4, 2, 3, 6)$, where c_k denote the electricity price of period k. Then $s = (2, 3, 1, 0, 0, 0, 0, 0, 0, 0)$ represents a feasible schedule as shown in Figure 2. For the sake of simplicity, given any job sequence σ, if there is no idle time between any two adjacent jobs on machine 1, we simply put σ in s instead of listing each of the jobs explicitly. For example, let $\sigma' = (2, 3, 1)$, then s in Figure 2 can be described as $(\sigma', 0, 0, 0, 0, 0, 0, 0)$.

Let E_i be the electricity cost incurred on machine i. Then, if $b_1 = 4$, $b_2 = 6$, $d_1 = 2$, $d_2 = 3$, we have $\mathrm{E}_1 = 80$, $\mathrm{E}_2 = 156$, and $\mathrm{E} = \mathrm{E}_1 + \mathrm{E}_2 = 236$.

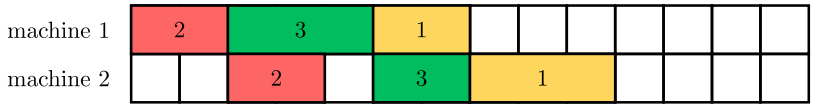

Figure 2. Gantt chart for Example 4.1.

We define a *feasible representation* as a representation in which the corresponding schedule is feasible, i.e. the completion time of the last job on machine 2 does not exceed T. Now we introduce the procedure of generating an initial solution as follows.

Algorithm 4.2. The `GenerateInitialSolution` procedure

1: Given a fixed number Z of the population size.
2: Determine the job sequence σ_0 when Johnson's rule is implemented.
3: Generate Z job sequences other than σ_0 randomly, and denote them as $\sigma_1, \ldots, \sigma_Z$.
4: **for** $z = 0$ to Z **do**
5: $\quad$ Schedule jobs according to the representation $(\sigma_z, 0, \ldots, 0)$.
6: $\quad$ Calculate the corresponding electricity cost $\mathrm{E}(\sigma_z)$.
7: Calculate $\sigma^* = \arg\min\{\mathrm{E}(\sigma_z)\}$.
8: Choose σ^* as the job sequence, output the initial solution $s_0 \leftarrow (\sigma^*, 0, \ldots, 0)$.

4.2 *Local search*

We know that local search is usually used to iteratively replace the current solution by a neighbour that improves the objective function. In our proposed ILS algorithm, we define the following move operator:

- `Swap(j,*)`: Swap job j and any other element in a given feasible representation s. Note that $*$ can be either a job or an element of '0'.
- `OPTSwap(j)`: Fix job j, perform swapping between job j and any other element in s, and output the swap that has a smallest value of electricity cost.

Example 4.2 Using the same setting as in Example 4.1, and fix job $j = 3$. Through some examination, we can find that `OPTSwap(3)` can be obtained either by swapping job 3 with job 1, or by swapping job 3 with job 2, with a same electricity cost $\mathrm{E}' = 224$. Figure 3 shows the corresponding schedule after performing the `Swap(3,1)` operator.

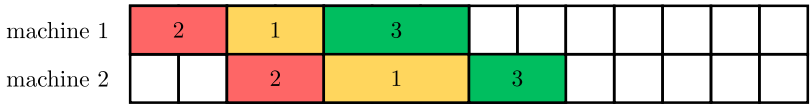

Figure 3. Gantt chart for the schedule after performing `Swap(3,1)` on s in Example 4.1.

During our local search procedure, we require that the generated representation of schedule after performing a move operator must also be feasible. Now we describe the `LocalSearch` procedure as follows.

Algorithm 4.3. The `LocalSearch` procedure

1: Given a feasible representation of schedule s'.
2: **for** $j \in \mathcal{J}$ **do**
3: Perform `OPTSwap(j)`, denote the new generated representation as $s''(j)$.
4: Calculate the corresponding electricity cost $\mathrm{E}(s''(j))$.
5: Calculate $j^* = \arg\min\{\mathrm{E}(s''(j))\}$. $s'' \leftarrow s''(j^*)$.

4.3 *Perturbation*

As we know, the perturbation procedure plays an important role in finding a good balance between intensification and diversification (Lourenço et al. 2010). In general, there are two important factors, i.e. the *type of perturbation operator* and the *perturbation strength*, can significantly affect the performance of the ILS method. The perturbation operator controls how to modify the current local optimal solution, and the perturbation strength controls how many times the selected perturbation operator should be applied.

In this paper, in addition to the `Swap(j,*)` operator, we also use the following move operator during our `Perturbation` procedure.

- `Move(j,*)`: Fix job j, randomly select a different position $*$ in s, and move job j to that position. For example, given $s = (1, 2, 3, 0, 0, 0, 0, 0)$, if we perform `Move(j,*)` on job 2 to move it to the fifth position, we obtain a new representation $s' = (1, 3, 0, 0, 2, 0, 0, 0)$.

At each iteration k, we define $n_p = \lceil (K - k)T/nK \rceil$ as the number of perturbation to be implemented during the `Perturbation` procedure, where k and K are defined as in Algorithm 4.1. Note that the value of n_p is time varying, which starts with a relatively high value when k is small, and gradually decreases as k increases. This way, we are able to adapt the ILS algorithm to derive more effective perturbations, since there is no single best size for the perturbation procedure (Lourenço et al. 2010).

The `Perturbation` procedure is described as follows.

Algorithm 4.4. The `Perturbation` procedure

Suppose the current iteration in Algorithm 4.1 is k, and the schedule to be perturbed is s.
Calculate n_p.
for $i = 1$ to n_p **do**
  Randomly choose `Swap(j,*)` or `Move(j,*)`, and perform it on any $j \in \mathcal{J}$ in s.
  Return the new schedule as s if it is feasible.
Output $s' \leftarrow s$.

4.4 *Acceptance criterion*

As shown in Algorithm 4.1, at each iteration, after implementing the `Perturbation` and `LocalSearch` procedures, we generate a 'neighbouring' solution s'' from a current solution s. The acceptance criterion is used to determine if s'' should be accepted, so as to balance the trade-off between the intensification and diversification of the search process.

In this work, we define an acceptance criterion as follows.

Algorithm 4.5. The `AcceptanceCriterion` procedure

Suppose the current iteration in Algorithm 4.1 is k, and the best found electricity cost is E^*.
Given two representations s and s'' with their associated electricity cost $\mathrm{E}(s)$ and $\mathrm{E}(s'')$.
Calculate the value of $p_a = e^{\mathrm{E}^* - \mathrm{E}(s)}$.
Generate a random number p_b within (0,1).
if $\mathrm{E}(s'') \leq \mathrm{E}(s)$ **then**
  Output s'' as the new schedule to be considered at iteration $k+1$.
else if $\mathrm{E}(s'') > \mathrm{E}(s)$ and $p_b < p_a$ **then**
  Output s'' as the new schedule to be considered at iteration $k+1$.
else
  Keep s as the new schedule to be considered at iteration $k+1$.

5. Computational experiments

In this section, we compare the performance of the different methods we proposed in Sections 3.1, 3.2 and 4. To be specific, we compare the performance of the MILP model, Algorithm JR, Algorithm DP11, Algorithm EDP and Algorithm ILS. To perform these experiments, we used ILOG CPLEX 12.5 to solve the MILP model on a computer with a 2.3 GHz Intel Core i7 processor and 16 GB of RAM running the OSX 10.9 operating system.

5.1 *Data generation*

Through some examination, we find that the computational performance of the above methods are mainly influenced by the following parameters: (i) the number of periods T. (ii) the electricity prices c_t; (iii) the energy consumption rates of machines, i.e. the values of b_i and d_i. To study the impact of these parameters, we consider the following parameter settings.

- Number of periods
  We first generate the processing time of jobs p_{ij} randomly from the uniform distribution on $\{1, 2, \ldots, 10\}$. Let $T = \lceil \lambda \times \sum_{i=1}^{2} \sum_{j \in \mathcal{J}} p_{ij} \rceil$. To reflect the flexibility of processing these jobs, we consider two λ levels of 0.8 and 1.2 to determine the number of periods. The higher the value of λ, the more flexibility the jobs can be processed since we have more choices of feasible schedules.
- Fluctuation of electricity prices
  The fluctuating electricity prices can significantly affect the performance of our proposed methods. Two $\theta = \max\{c_t\}/\min\{c_t\}$ levels of 3 and 6 are tested, which denote the stable and unstable electricity markets. Under different levels of θ, we generate the electricity prices at each period randomly from the uniform distribution on $\{1, \ldots, \theta\}$.
- Deviation of energy consumption rates

Table 1. Comparison of running time and number of optimal solutions solved between the MILP model and Algorithm EDP.

λ	θ	R	n	the MILP model			Algorithm EDP		
				Max time	Ave time	#OPT	Max time	Ave time	#OPT
0.8	3	R_1	6	37.6	6.8	30	0.2	0.1	30
0.8	3	R_1	8	59.6	13.9	30	33.9	13.7	30
1.2	3	R_1	6	13.4	5.7	30	4.8	1.7	30
1.2	3	R_1	8	759	41.9	30	830	386	30
0.8	6	R_1	6	78.6	9.7	30	0.1	0.1	30
0.8	6	R_1	8	1800	221	27	23.7	9.8	30
1.2	6	R_1	6	36.4	9.2	30	3.8	1.6	30
1.2	6	R_1	8	977	83.4	30	582	328	30
0.8	3	R_2	6	162	21.7	30	0.1	0.1	30
0.8	3	R_2	8	1800	443	27	26.0	12.7	30
1.2	3	R_2	6	102	16.0	30	3.8	1.5	30
1.2	3	R_2	8	1800	280	27	1012	399	30
0.8	6	R_2	6	573	47.1	30	0.1	0.1	30
0.8	6	R_2	8	1800	754	22	32.8	14.9	30
1.2	6	R_2	6	1800	148	29	4.7	2.3	30
1.2	6	R_2	8	1800	813	21	1340	489	30
0.8	3	R_3	6	45.9	14.0	30	0.1	0.1	30
0.8	3	R_3	8	1301	141	30	19.3	12.2	30
1.2	3	R_3	6	191	21.8	30	3.8	1.5	30
1.2	3	R_3	8	1800	372	26	835	391	30
0.8	6	R_3	6	171	22.6	30	0.1	0.1	30
0.8	6	R_3	8	1800	604	24	29.3	11.9	30
1.2	6	R_3	6	873	77.1	30	4.5	1.7	30
1.2	6	R_3	8	1800	784	22	1001	411	30

We consider the following three scenarios of energy consumption rates on machines: (i) R_1: $b_1 = 2, d_1 = 1, b_2 = 2, d_2 = 1$, that is, two machines are identical; (ii) R_2: $b_1 = 2, d_1 = 1, b_2 = 6, d_2 = 2$, that is, machine 2 is more energy-intensive; (iii) R_3: $b_1 = 6, d_1 = 2, b_2 = 2, d_2 = 1$, that is, machine 1 is more energy-intensive.

We consider the following combinations of the above parameters:

$$\big\{(\lambda, \theta, R, n) : \lambda \in \{0.8, 1.2\}, \theta \in \{3, 6\}, R \in \{R_1, R_2, R_3\}, n \in \{6, 8, 20, 30, 40, 50\}\ .$$

We randomly generate 30 instances for each combination (λ, θ, R, n), for a total of $2 \times 2 \times 3 \times 6 \times 30 = 2160$ instances. In particular, for small size instances, i.e. $n = 6, 8$, we compare the computational performance of all the proposed methods. For those medium and large size instances, we only compare the computational performance of Algorithm DP11 and Algorithm ILS, due to the computational intractability of the TMPFSEC problem. In addition, we set a 30 mintime limit on each instance.

5.2 *ILS parameter tuning*

As we know, there are two important parameters need to be prefixed in the ILS algorithm: the *number of iterations* (i.e. the value of K in Algorithm 4.1) and the *population size* (i.e. the value of Z in Algorithm 4.2). Different values of K and Z may lead to significant differences on the computation performance of our proposed algorithms. To determine the appropriate values of K and Z, we carried out a series of experiments on the following combinations:

$$\big\{(K, Z) : K \in \{100, 200, 300, 400, 500\}, Z \in \{100, 200, 500\}\ ,$$

which results in $5 \times 3 = 15$ combinations of parameter settings in total.

To calibrate the parameter values, we randomly generate 30 instances for each combination of $\{\lambda, \theta, R, n\}$ as mentioned in Section 5.1, where only $n \in \{8, 20, 30\}$ are considered. For each instance, we test the performance of our ILS algorithm with

Table 2. Comparison of optimality gap between Algorithm JR and Algorithm ILS.

λ	θ	R	n	Algorithm JR		Algorithm ILS	
				Max gap (%)	Ave gap (%)	Max gap (%)	Ave gap (%)
0.8	3	R_1	6	4.50	1.74	2.93	1.05
0.8	3	R_1	8	4.53	1.58	3.31	1.19
1.2	3	R_1	6	4.51	1.15	4.30	1.28
1.2	3	R_1	8	4.55	1.92	3.02	1.21
0.8	6	R_1	6	4.08	1.92	3.60	1.59
0.8	6	R_1	8	4.08	1.63	3.82	1.62
1.2	6	R_1	6	5.65	2.01	4.20	1.57
1.2	6	R_1	8	3.69	1.61	5.18	1.72
			Ave(R_1)	4.449	1.695	3.795	1.404
0.8	3	R_2	6	10.40	2.54	4.62	1.16
0.8	3	R_2	8	7.69	2.60	3.50	1.61
1.2	3	R_2	6	7.59	1.99	1.83	0.78
1.2	3	R_2	8	6.52	2.91	4.97	1.75
0.8	6	R_2	6	8.27	4.94	5.07	1.97
0.8	6	R_2	8	8.91	3.88	3.49	2.38
1.2	6	R_2	6	13.95	4.05	4.49	1.58
1.2	6	R_2	8	8.50	4.01	5.95	2.44
			Ave(R_2)	8.979	3.365	4.24	1.709
0.8	3	R_3	6	5.99	1.60	5.63	1.56
0.8	3	R_3	8	3.58	1.34	4.32	1.80
1.2	3	R_3	6	4.97	1.74	4.30	1.99
1.2	3	R_3	8	4.52	1.65	3.94	1.77
0.8	6	R_3	6	5.71	2.31	4.42	2.00
0.8	6	R_3	8	5.54	2.73	5.97	2.87
1.2	6	R_3	6	5.51	2.19	5.04	1.94
1.2	6	R_3	8	5.64	2.61	6.02	2.99
			Ave(R_3)	5.183	2.021	4.955	2.115

each combination (K, Z) of the parameter settings. The mean objective value of the 30 instances for each combination (K, Z) is recorded, and we choose the best combination as our parameter setting in Algorithm ILS. According to our computational results, we set $K = 400$ and $Z = 500$.

5.3 *Computational results*

We first focus on small size instances, i.e. $n = 6, 8$, to compare the performance of our proposed algorithms. As we know, solving the MILP model with ILOG CPLEX and using Algorithm EDP can both obtain the exact optimal schedule of the TMPFSEC problem. Thus, we first compare the performance of these two methods. Table 1 shows the CPU running time and number of optimal solutions obtained by these two methods within the time limit.

From Table 1, we obtain the following observations:

- The higher the value of λ, the more the CPU running time on solving the TMPFSEC problem, especially for Algorithm EDP. This is to be expected, since the value of T increases as λ increases, which as a result enlarges the search space to be explored in dynamic programming.
- The higher the value of θ, the higher the CPU running time, and the smaller the number of optimal solutions can be obtained by the MILP model. This reveals that when the electricity price market is unstable, it will be more difficulty for the manufacturing enterprises to make optimal decisions.

Table 3. Comparison of computational performance of Algorithms ILS, DP11 and ILSDP.

λ	θ	R	n	ILS	$ILSDP$	$DP11$	ILS v.s. DP11	ILSDP v.s. DP11
				Ave time (s)	Ave time (s)	Ave time (s)	Ave Dev(ILS) (%)	Ave Dev(ILSDP) (%)
0.8	3	R_1	20	0.1	0.2	0.2	2.68	–0.06
0.8	3	R_1	30	0.1	0.3	0.9	3.09	–0.07
0.8	3	R_1	40	0.4	1.1	4.1	3.63	–0.12
0.8	3	R_1	50	0.8	2.2	15.7	3.74	–0.47
1.2	3	R_1	20	0.1	0.8	4.9	2.01	–0.32
1.2	3	R_1	30	0.3	2.6	24.4	2.97	–0.02
1.2	3	R_1	40	1.8	14.8	128.6	0.78	–0.03
1.2	3	R_1	50	8.7	54.4	435.4	0.65	–0.02
0.8	6	R_1	20	0.1	0.1	0.2	2.42	–0.07
0.8	6	R_1	30	0.2	0.3	1.4	2.88	–0.06
0.8	6	R_1	40	0.8	1.9	11.1	1.43	–0.1
0.8	6	R_1	50	6.7	12.5	37.5	1.43	–0.1
1.2	6	R_1	20	0.1	0.7	3.7	2.74	–0.21
1.2	6	R_1	30	0.4	2.57	17.6	3.06	–0.28
1.2	6	R_1	40	0.8	10.6	104.4	2.19	–0.55
1.2	6	R_1	50	1.6	33.2	366.9	2.13	–0.71
0.8	3	R_2	20	0.3	0.4	1.4	2.45	–0.14
0.8	3	R_2	30	0.3	1.6	6.8	2.01	–0.17
0.8	3	R_2	40	0.7	4.6	32.5	2.57	–0.51
0.8	3	R_2	50	1.4	11.8	95.2	4.12	0.15
1.2	3	R_2	20	0.3	3.6	34.2	2.61	–0.27
1.2	3	R_2	30	0.7	21.9	240.3	1.92	–0.95
1.2	3	R_2	40	1.9	69.3	721.1	2.58	–0.59
1.2	3	R_2	50	15.3	135.8	1231.2	1.34	–0.31
0.8	6	R_2	20	0.1	0.2	0.8	1.36	–1.01
0.8	6	R_2	30	0.8	1.3	6.3	2.24	–0.07
0.8	6	R_2	40	0.9	3.8	123.6	2.45	–0.66
0.8	6	R_2	50	1.6	7.3	154.3	3.50	–0.51
1.2	6	R_2	20	0.1	0.5	3.5	2.02	–1.33
1.2	6	R_2	30	0.3	3.5	27.8	2.14	–0.84
1.2	6	R_2	40	1.1	11.4	113.3	3.01	–0.64
1.2	6	R_2	50	1.5	30.4	320.8	3.52	–0.51
0.8	3	R_3	20	0.1	0.1	0.2	0.05	–0.71
0.8	3	R_3	30	0.2	0.4	1.2	0.88	–0.09
0.8	3	R_3	40	0.4	0.7	2.8	2.19	0.003
0.8	3	R_3	50	0.7	1.7	9.5	0.97	–0.16
1.2	3	R_3	20	0.1	0.6	5.6	1.83	–0.11
1.2	3	R_3	30	0.4	4.1	34.8	1.93	–0.05
1.2	3	R_3	40	1.8	61.3	467.2	1.09	–0.42
1.2	3	R_3	50	9.3	101.5	821.0	1.21	–0.31
0.8	6	R_3	20	0.1	0.1	0.2	0.97	–0.75
0.8	6	R_3	30	0.2	0.4	2.1	0.84	–0.43
0.8	6	R_3	40	0.7	1.8	5.6	1.71	–0.28
0.8	6	R_3	50	1.4	6.5	16.8	0.43	–0.13
1.2	6	R_3	20	0.1	0.3	4.9	0.39	–0.37
1.2	6	R_3	30	0.3	4.9	33.8	0.35	–0.41
1.2	6	R_3	40	0.8	11.3	125.9	0.41	–0.22
1.2	6	R_3	50	1.5	24.5	260.7	1.12	–0.08

- It seems that using Algorithm EDP to search optimal solutions outperforms solving the MILP model, especially when λ is small. This observation comes from two folds: first, within the time limit, all of these small size instances can be solved by Algorithm EDP, while some of them cannot be solved by the MILP model; and second, the average CPU running time for solving the same instance by Algorithm EDP is comparatively smaller than the one by the MILP model.

We then compare the performance of Algorithm JR and Algorithm ILS for solving small size instances. Note that Algorithm EDP can solve all the small size instances optimally, so we do not need to consider Algorithm DP11 here. Table 2 shows the average optimality gap for solving these instances by Algorithm JR and Algorithm ILS, where their optimality gap is defined as $(\mathrm{E(JR)} - \mathrm{E^{opt}})/\mathrm{E^{opt}} \times 100\%$ and $(\mathrm{E(ILS)} - \mathrm{E^{opt}})/\mathrm{E^{opt}} \times 100\%$ respectively. The CPU running time for both Algorithm JR and Algorithm ILS were minimal, i.e. each instance can be solved within 1 s, thus we do not report them on the table.

From Table 2, we have the following observations:

- In terms of the maximum optimality gap to the optimal solutions, Algorithm ILS outperforms Algorithm JR for almost all of the small size instances.
- Unlike Algorithm EDP, the effect of λ is not obvious on the performance of these two methods. This is resonable, since these two algorithms do not need to explore the whole solution space, which is usually done in Algorithm EDP.
- The results show that the optimality gap for the instances with R_1 is smaller than those with R_2 and R_3 (see the values of Ave(R_i) for $i = 1, 2, 3$ in Table 1). This means that Algorithm JR and Algorithm ILS work relatively better on identical machines.

From our computational results, we notice that even when $n = 8$, some of the instances cannot be solved optimally within the time limit, and the CPU running time increases rapidly when n increases. Thus, the MILP model and Algorithm EDP will not be evaluated for the medium and large size instances. On the other hand, from Table 2, we know that Algorithm ILS usually outperforms Algorithm JR, therefore, we only compare the performance of Algorithm ILS and Algorithm DP11 for medium and large size instances. Moreover, we also consider a new algorithm which implement Algorithm DP on the final solution of Algorithm ILS. For simplicity, we call this new algorithm as Algorithm ILSDP, and the corresponding total electricity cost as E(ILSDP).

To compare the performance of Algorithms ILS, DP11 and ILSDP, we define $\mathrm{Dev(ILS)} = \big(\mathrm{E(ILS)} - \mathrm{E(DP11)}\big)/\mathrm{E(DP11)} \times 100\%$ and $\mathrm{Dev(ILSDP)} = \big(\mathrm{E(ILSDP)} - \mathrm{E(DP11)}\big)/\mathrm{E(DP11)} \times 100\%$ as the *performance deviation* of Algorithms ILS and ILSDP from Algorithm DP11, respectively. Table 3 shows the corresponding computational results.

From Table 3, we have the following observations:

- The CPU running time of Algorithm DP11 and ILSDP are highly affected by the values of λ, and increases much faster than the one of Algorithms ILS as n increases. One reason may be due to the global search efforts of the solution space in Algorithms DP11 and ILSDP.
- From the results of Dev(ILS), we see that Algorithm DP11 outperforms Algorithm ILS slightly. However, Algorithm ILS has a great advantage on the CPU running time, which means that Algorithm ILS can be used to find feasible solutions with acceptable solution quality for large size instances when the given time limit is very small.
- From the results of Dev(ILSDP), we see that Algorithm ILSDP outperforms Algorithm DP11 slightly. Moreover, the CPU running time of Algorithm ILSDP is much smaller than the one of Algorithm DP11. Hence, it may be promising to implement Algorithm ILSDP when the requirement of solution quality is high and the given time limit is within a comparative small range.

6. Conclusions and future work

In this paper, we studied the problem of minimising total electricity cost on a two-machine permutation flow shop under TOU tariffs (the TMPFSEC problem), in which each machine may have its own energy consumption rate. We proposed a mixed integer linear program as the first key step to understand the structure of the problem. Then, two heuristic algorithms based on Johnson's rule and the dynamic programming method are proposed and analysed. As a by-product, we showed how to obtain optimal schedules by dynamic programming when the job sequence is fixed. Furthermore, we proposed an iterated local search method that incorporates problem-tailored procedures to solve the TMPFSEC problem. Finally, we empirically tested the performance of these algorithms on randomly generated instances.

Several related topics might be considered as future work. First, the design and analysis of tight lower bounds remains an open area. Second, more general scheduling environments can be further considered, such as multiple machine flow shop, preemptive and non-permutation jobs, and so on. Third, an improved iterated local search method might be further analysed.

Last but not least, multi-objective optimisation problems might be considered and developed to balance the total electricity cost and production efficiency criteria (e.g. total completion time, maximum lateness and other objectives related to job due dates).

Acknowledgements

The authors would like to thank the editor and anonymous referees whose comments helped a lot in improving this paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Science Foundation of China (NSFC) [grant number 71571135], [grant number 71428002] and [grant number 71571134], [grant number 71701144]. This work was partially supported by the Cai Yuanpei Program between the French Ministries of Foreign and European Affairs and the Higher Education and Research and the Chinese Ministry of Education, [grant number 27927VE]. This work was also sponsored by the Fundamental Research Funds for the Central Universities.

References

Antoniadis, A., P. Kling, S. Ott, and S. Riechers. 2017. "Continuous Speed Scaling with Variability: A Simple and Direct Approach." *Theoretical Computer Science* 678: 1–13.

Braithwait, S., D. Hansen, and M. O'Sheasy. 2007. "Retail Electricity Pricing and Rate Design in Evolving Markets." Technical Report, Edison Electric Institute.

Burcea, M., W.-K. Hon, H.-H. Liu, P. W. H. Wong, and D. K. Y. Yau. 2016. "Scheduling for Electricity Cost in a Smart Grid." *Journal of Scheduling* 19 (6): 687–699.

Che, A., Y. Zeng, and K. Lyu. 2016. "An Efficient Greedy Insertion Heuristic for Energy Conscious Single Machine Scheduling Problem under Time-of-use Electricity Tariffs." *Journal of Cleaner Production* 129: 565–577.

Che, A., X. Wu, J. Peng, and P. Yan. 2017a. "Energy-efficient Bi-objective Single-machine Scheduling with Power-down Mechanism." *Computers & Operations Research* 85: 172–183.

Che, A., S. Zhang, and X. Wu. 2017b. "Energy-conscious Unrelated Parallel Machine Scheduling under Time-of-use Electricity Tariffs." *Journal of Cleaner Production* 156: 688–697.

Cheng, J., F. Chu, M. Liu, P. Wu, and W. Xia. 2017. "Bi-criteria Single-machine Batch Scheduling with Machine on/off Switching under Time-of-use Tariffs." *Computers & Industrial Engineering* 112: 721–734.

Dai, M., D. B. Tang, A. Giret, M. A. Salido, and W. D. Li. 2013. "Energy-efficient Scheduling for a Flexible Flow Shop using an Improved Genetic-simulated Annealing Algorithm." *Robotics and Computer-Integrated Manufacturing* 29: 418–429.

Ding, J. Y., S. Song, R. Chiong, and C. Wu. 2016a. "Parallel Machine Scheduling under Time-of-use Electricity Prices: New Models and Optimization Approaches." *IEEE Transactions on Automation Science and Engineering* 13 (2): 1138–1154.

Ding, J. Y., S. Song, and C. Wu. 2016b. "Carbon-efficient Scheduling of Flow Shops by Multi-objective Optimization." *European Journal of Operational Research* 248 (3): 758–771.

Fang, K., N. A. Uhan, F. Zhao, and J. W. Sutherland. 2011. "A New Approach to Scheduling in Manufacturing for Power Consumption and Carbon Footprint Reduction." *Journal of Manufacturing Systems* 30 (4): 234–240.

Fang, K., N. A. Uhan, F. Zhao, and J. W. Sutherland. 2013. "Flow Shop Scheduling with Peak Power Consumption Constraints." *Annals of Operations Research* 206 (1): 115–145.

Fang, K., N. A. Uhan, F. Zhao, and J. W. Sutherland. 2016. "Scheduling on a Single Machine under Time-of-use Electricity Tariffs." *Annals of Operations Research* 238 (1–2): 199–227.

Gahm, C., F. Denz, M. Dirr, and A. Tuma. 2016. "Energy-efficient Scheduling in Manufacturing Companies: A Review and Research Framework." *European Journal of Operational Research* 248 (3): 744–757.

Giret, A., D. Trentesaux, and V. Prabhu. 2015. "Sustainability in Manufacturing Operations Scheduling: A State of the Art Review." *Journal of Manufacturing Systems* 37 (1): 126–140.

Johnson, S. M. 1954. "Optimal Two-and Three-stage Production Schedules with Setup Times Included." *Naval Research Logistics* 1 (1): 61–68.

Kulkarni, J., and K. Munagala. 2013. "Algorithms for Cost-aware Scheduling." In *10th International Workshop on Approximation and Online Algorithms (WAOA 2012)*. Vol. 7846 of *Lecture Notes in Computer Science*, edited by T. Erlebach and G. Persiano, 201–214, Berlin, Heidelberg: Springer.

Lourenço, H. R., O. C. Martin, and T. Stützle. 2010. "Iterated Local Search: Framework and Applications." In *Handbook of Metaheuristics*, edited by M. Gendreau and J. Y. Potvin, 363–397. Boston, MA: Springer.

Mansouri, S. A., E. Aktas, and U. Besikci. 2016. “Green Scheduling of a Two-machine Flowshop: Trade-off Between Makespan and Energy Consumption.” *European Journal of Operational Research* 248 (3): 772–788.

Mouzon, G., M. B. Yildirim, and J. Twomey. 2007. “Operational Methods for Minimization of Energy Consumption of Manufacturing Equipment.” *International Journal of Production Research* 45 (18–19): 4247–4271.

Panwalkar, S. S., R. A. Dudek, and M. L. Smith. 1973. “Sequencing Research and the Industrial Scheduling Problem.” In *Symposium on the Theory of Scheduling and Its Applications*. Vol. 86 of *Lecture Notes in Economics and Mathematical Systems (Operations Research, Computer Science, Social Science)*, edited by S. E. Elmaghraby, 29–38, Lubbock, TX: Springer.

Park, C. W., 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. 2009. “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.

Sabar, N. R., and G. Kendall. 2015. “An Iterated Local Search with Multiple Perturbation Operators and Time Varying Perturbation Strength for the Aircraft Landing Problem.” *Omega* 56: 88–98.

Shrouf, F., J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier. 2014. “Optimizing the Production Scheduling of a Single Machine to Minimize Total Energy Consumption Costs.” *Journal of Cleaner Production* 67: 197–207.

Silva, M. M., A. Subramanian, and L. S. Ochi. 2015. “An Iterated Local Search Heuristic for the Split Delivery Vehicle Routing Problem.” *Computers & Operations Research* 53: 234–249.

Vansteenwegen, P., W. Souffriau, G. V. Berghe, and D. Van Oudheusden. 2009. “Iterated Local Search for the Team Orienteering Problem with Time Windows.” *Computers & Operations Research* 36 (12): 3281–3290.

Zeng, Y., A. Che, and X. Wu. 2017. “Bi-objective Scheduling on Uniform Parallel Machines Considering Electricity Cost.” *Engineering Optimization* 50: 19–36.

Zhang, H., F. Zhao, K. Fang, and J. W. Sutherland. 2014. “Energy-conscious Flow Shop Scheduling under Time-of-use Electricity Tariffs.” *CIRP Annals-Manufacturing Technology* 63 (1): 37–40.