Computers and Operations Research 85 (2017) 172–183

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Energy-efficient bi-objective single-machine scheduling with power-down mechanism

Ada Che[a], Xueqi Wu[a], Jing Peng[b], Pengyu Yan[c,*]

[a] School of Management, Northwestern Polytechnical University, Xi'an 710072, China
[b] School of Management, University of Science and Technology of China, Hefei 230026, China
[c] School of Management and Economics, University of Electronic Science and Technology of China, Chengdu 610054, China

ARTICLE INFO

Article history:
Received 11 June 2016
Revised 1 February 2017
Accepted 11 April 2017
Available online 12 April 2017

Keywords:
Energy-efficient scheduling
Single machine
Bi-objective scheduling
Power-down mechanism
ε–constraint method

ABSTRACT

This paper considers a single-machine scheduling problem with power-down mechanism to minimize both total energy consumption and maximum tardiness. The aim is to find an optimal processing sequence of jobs and determine if the machine should be executed a power-down operation between two consecutive jobs. To formulate the problem, a mixed-integer linear programming (MILP) model is developed. Then a basic $\varepsilon-$constraint method is proposed to obtain the complete Pareto front of the problem. Considering the particularity of the problem, we also develop local search, preprocessing technique and valid inequalities to strengthen the basic $\varepsilon-$constraint method. Finally, to obtain approximate Pareto fronts for large-size problems, we utilize the method of cluster analysis to divide the jobs into several sorted clusters according to their release times and due dates. Any job in a preceding cluster must be processed before all jobs in a subsequent cluster. Thus, the solution space is reduced significantly. Computational experiments on benchmark and randomly generated instances demonstrate the effectiveness of the proposed exact and approximation approaches.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

Energy has made our lives easier. Ever since people began to utilize energy, the world has been changed significantly. With rapid industrialization and development of civilization, energy is obviously one of the crucial elements that affects the overall economic performance of almost all businesses. Since the majority of resources we utilize to generate energy are nonrenewable (Tacconi, 2000) and energy consumption has increased by 300% in the last 50 years due to massive usage (Park et al., 2010), energy has become more scarce and precious than ever before. More importantly, the process of generating energy is polluting the environment and accelerating global warming. Based on a recent survey conducted by the International Energy Agency (2015), the worldwide demand for energy will increase by 37% by 2040. Accordingly, the total emitted CO_2 will increase by around 80% in the next 40–50 years (Marchal & Dellink, 2011). With the aim of achieving sustainable development, it is essential to reduce energy consumption and improve the quality of environment.

As the backbone of the industrialized society, manufacturing sectors consume half of the energy that is available globally and emit the most greenhouse gases (Ross, 1992; Jovane et al., 2008). To achieve the goal of sustainable development, the manufacturing sectors should focus on the endeavor to reduce resource usage as well as energy consumption (Sudarsan et al., 2010). However, they have concentrated most of their attention on the efficiency and quality of production systems over the past 60 years (e.g. Altiok, 1997; Li and Meerkov, 2010). With the ever-increasing demand for ecological resources protection and energy saving, it is of vital importance that the manufacturing sectors control their energy consumption during the production process in an effective manner (Rajemi et al., 2010).

Various methods have been proposed to reduce energy consumption and greenhouse gases emission, such as implementing management tools, innovative technologies and new policies (Abdelaziz et al., 2011). As a widely used planning and management method, scheduling is an effective approach in controlling energy consumption. In general, scheduling is a strategy to allocate limited resources to a number of tasks in an efficient way. Specifically, machine scheduling decides when and how to allocate machines to jobs. In other words, it is used to determine the starting times of processing jobs on the machines. Ever since the first

* Corresponding author.
E-mail address: yanpy@uestc.edu.cn (P. Yan).

http://dx.doi.org/10.1016/j.cor.2017.04.004
0305-0548/© 2017 Elsevier Ltd. All rights reserved.

publication of scheduling in 1954 (Johnson, 1954), more attention has been paid to use the machine scheduling as a strategy to control energy consumption. This is mainly because that the machine scheduling can exert significant impact on the production performance in terms of energy saving.

As for single-machine scheduling, the majority of researches focus on time-related objectives, such as makespan (Kasap et al., 2006; Xu et al., 2013), total completion time (Xu et al., 2015) and tardiness (Kolliopoulos and Steiner, 2006). These objectives are identified to reflect the efficiency of the production process. With the increasing concern over the energy consumption and environmental issues, the objective related to energy-saving has been identified recently. To achieve the objective of energy-saving by means of machine scheduling, several directions can be pursued, among which power-down mechanism is quite an effective approach (e.g., Dai et al., 2013; Pach et al., 2014). The idea of the power-down mechanism is that if the total energy consumption will be reduced, then a machine will be shut down instead of running idle when it is not processing any jobs. For example, for one Wichita, Kansas aircraft supplier of small parts, it was reported (Drake et al., 2006) that the machines are kept idle for 16% of an eight-hour shift. According to their research, by simply turning off the machines rather than keeping them running idle, one could achieve at least 13% energy savings. Mouzon and Yildirim (2008) proposed that if the duration of machine idleness lasts longer than a breakeven duration, significant energy savings can be achieved by shutting down the machine. In a nutshell, the power-down mechanism determines whether and when to shut down a machine or to leave it idle in order to improve energy efficiency during the course of production. The power-down mechanism has broader applicability. For example, Swaminathan and Chakrabarty (2003) discovered that a control system installed in computer and portable devices for changing the state of battery can save energy and extend its lifespan.

We now conduct a literature review for single-machine scheduling with power-down mechanism. Mouzon et al. (2007) proposed several dispatching rules for single-machine scheduling with power-down mechanism to minimize energy consumption. In particular, these rules are set to shut down non-bottleneck machines when they are kept idle for a certain duration in order for minimizing the energy consumption and total completion time. Mouzon and Yildirim (2008) developed a framework based on a greedy randomized adaptive search metaheuristic to resolve a single-machine scheduling problem aimed at minimizing total energy consumption and total tardiness. Yildirim and Mouzon (2012) developed a mathematical model and a multi-objective genetic algorithm to minimize total energy consumption and total completion time. Dominance rules and a heuristic are used in their study to increase the efficiency of the algorithm. Similarly, Liu et al. (2014) proposed a multi-objective optimization model to minimize the total carbon dioxide emissions and total completion time. They assumed that the jobs to be processed follow the First-Come-First-Serve rule and developed a non-dominated sorting genetic algorithm (NSGA) II to obtain an approximate Pareto front of the problem. Shrouf et al. (2014) proposed a mathematical model for single-machine scheduling with power-down mechanism under time-of-use electricity tariffs to minimize the total energy consumption cost. They assumed that the processing sequence of jobs is fixed and applied genetic algorithm to obtain the near-optimal solution of the problem.

Another useful technique to reduce energy consumption for machine scheduling problems is speed scaling (Yao et al., 1995). In this mechanism, the machine (or processor) can vary its processing speed, resulting in different processing times and energy consumption rates. The speed-scaling problems have been studied by many scholars (e.g. Bansal et al., 2007; Pruhs et al., 2008; Angel et al., 2011; Fang et al., 2011; Fang et al., 2013). Wierman et al., (2012) provided a comprehensive review on algorithms for speed-scaling problems. Another technique is based on the observation that the electricity price for industry users usually varies from hour to hour during a day (time-of-use tariffs). To tackle this new dynamic pricing scheme, several works were conducted to reduce the total electricity cost (e.g. Babu and Ashok, 2008; Luo et al., 2013; Fang et al., 2016; Che et al., 2016; Che et al., 2017a; Zeng et al., 2017).

In this paper, we investigate a single-machine bi-objective scheduling problem with power-down mechanism to minimize both total energy consumption and maximum tardiness. Different from total tardiness, maximum tardiness indicates the maximum degree to which the completion times of jobs exceed their due dates. As for the single-machine scheduling problem with power-down mechanism, to the best of our knowledge, the meta-heuristics, such as greedy randomized multi-objective adaptive search metaheuristic (Mouzon and Yildirim, 2008) and non-dominated sorting genetic algorithm II (Liu et al., 2014), can only achieve near-optimal Pareto front within reasonable time. There is still lack of research on finding the exact Pareto fronts of such problems. This study fills this gap.

To achieve this, we first formulate a mixed integer linear programming model for the single-machine scheduling problem with power-down mechanism. Then a basic $\varepsilon-$constraint method (Berube et al., 2009; Feng et al., 2014) is developed to obtain the exact Pareto front of the problem. Considering the specific properties of the problem, we develop local search, preprocessing technique and valid inequalities to strengthen the proposed method. To enable the proposed model to solve large-size problems, we use cluster analysis to divide jobs into several sorted clusters in terms of their release times and due dates. In this case, any job in a preceding cluster must be processed before the jobs in subsequent clusters. This significantly narrows down the solution space. Computational results show that the proposed exact approach is able to obtain the exact Pareto front and the clustering procedure can greatly enhance the effectiveness of our approach.

The rest of this paper is organized as follows. We present a clear description of the problem to be addressed and formulate a mathematical model in Section 2. In Section 3, we propose an advanced $\varepsilon-$constraint method combined with local search, preprocessing technique and valid inequalities to solve the model and introduce the procedure of job clustering. In Section 4, we use benchmark and randomly generated instances to test the proposed algorithms and present the computational results. Finally, a conclusion is presented in Section 5.

2. Problem description and formulation

2.1. Problem definition

The problem addressed in this paper is a single machine scheduling problem to process a set of n jobs with the objective of minimizing the total energy consumption and maximum tardiness. Each job to be processed on the machine has its unique release time r_i, processing time t_i and due date d_i, $1 \le i \le n$. For any job, it can be processed only after it is released. If the completion time of a job exceeds its due date, then tardiness appears. No preemption is allowed. When the machine is working (i.e. processing a job), its energy consumption rate is denoted as P_w, and when the machine stands idle (i.e. not processing any jobs but still running), it consumes P_v.

Due to different release times for different jobs, the machine may run idle for a period of time when it has finished processing a job and the next job is not yet available. When the machine is idle, it still consumes energy. So one might think of turning the

machine off during the idle period and then turning it on again when the next job needs to be processed to save energy. But since the operation of turning off and subsequently turning on the machine requires extra time and energy, there should be a breakeven period. Only when the idle duration lasts longer than this period, turning off and subsequently turning on the machine will save energy. Since turning off and turning on the machine always appear in pairs during the processing horizon, we consider them as an integrated turn-off-on operation. Let T_s be the time for a turn-off-on operation and E_s its amount of energy consumption.

To determine the duration of this breakeven period, two facets have to be considered. On one hand, if the idle time between two successive jobs is shorter than T_s, then it is impossible to perform a turn-off-on operation; on the other hand, if the energy consumed for the idle machine between two successive jobs is less than E_s, then a turn-off-on operation is not necessary in terms of energy-saving. So to define this breakeven period denoted by T_B, we have $T_B = max(\frac{E_s}{P_v}, T_s)$ (Mouzon et al., 2007). Note that we assume that r_i, t_i, d_i, P_v, E_s and T_B are all integers.

Generally, the total energy consumption during the whole processing course is comprised of four parts: energy required to turn on the machine initially and turn it off ultimately, energy consumed when it is processing jobs, energy consumed for the idle machine and energy required by all turn-off-on operations. Since the sum of the first two types of energy consumption is constant and not dependent on the jobs' processing sequence, we only focus on the last two types. That is, it is sufficient to minimize the amount of energy consumption for the idle machine and all turn-off-on operations.

There are two objectives in this study: minimizing the total energy consumption and maximum tardiness. Our aim is to find an optimal processing sequence of jobs as well as their starting and completion times and determine if the machine should be executed a turn-off-on operation between two consecutive jobs. Generally, there are two approaches to formulating such optimization problems: discrete and continuous time formulations (Mendez et al., 2006). For discrete time formulations, the time horizon is divided into many time units and then binary variables are defined with respect to each time unit. As for continuous time formulations, two main methods are often used to formulate the job processing sequence: immediate and general precedence (Mendez et al., 2006). The two methods determine if a job is processed immediately before or after another job and simply it is processed before or after another job, respectively. In this study, we formulate the processing sequence of jobs based on the position assignment (Tseng and Stafford, 2008), which directly identifies the processing position of a job in a processing sequence. Our initial computational experiments demonstrate that this formulation significantly outperforms the formulation with immediate or general precedence in terms of computational time.

2.2. MILP model based on position assignment

In our model, each job is assigned to a position in the processing sequence. The decision variables in this model are described as follows:

- x_{ik}: Binary variable. If job i is processed in position k, then $x_{ik}=1$; otherwise, $x_{ik}=0$, for $1 \le i \le n,\ 1 \le k \le n$
- y_k: Binary variable. If there is a turn-off-on operation immediately after the kth job (i.e. the job processed in position k), then $y_k=1$; otherwise $y_k=0$, for $1 \le k \le n-1$
- C_k: The completion time of the kth job, for $1 \le k \le n$
- D_k: The tardiness of the kth job, for $1 \le k \le n$
- E_k: The energy consumed by the machine between the completion of the kth job and the start of the $(k+1)$th job, for $1 \le k \le n-1$

With the notations above, the MILP model can be presented as follows:

$$Min \sum_{k=1}^{n-1} E_k \tag{1}$$

$$Min\ D_{max} \tag{2}$$

subject to

$$C_k - \sum_{i=1}^{n} x_{ik} * t_i \ge \sum_{i=1}^{n} x_{ik} * r_i,\ \text{for}\ 1 \le k \le n \tag{3}$$

$$D_k \ge C_k - \sum_{i=1}^{n} x_{ik} * d_i,\ \text{for}\ 1 \le k \le n \tag{4}$$

$$D_{max} \ge D_k,\ \text{for}\ 1 \le k \le n \tag{5}$$

$$C_k \ge C_{k-1} + \sum_{i=1}^{n} x_{ik} * t_i + y_{k-1} * T_B,\ \text{for}\ 2 \le k \le n \tag{6}$$

$$E_{k-1} \ge \left(C_k - C_{k-1} - \sum_{i=1}^{n} x_{ik} * t_i \right) * P_v - y_{k-1} * M,\ \text{for}\ 2 \le k \le n \tag{7}$$

$$E_k \ge E_s * y_k,\ \text{for}\ 1 \le k \le n-1 \tag{8}$$

$$\sum_{i=1}^{n} x_{ik} = 1,\ \text{for}\ 1 \le k \le n \tag{9}$$

$$\sum_{k=1}^{n} x_{ik} = 1,\ \text{for}\ 1 \le i \le n \tag{10}$$

$$D_k \ge 0,\ \text{for}\ 1 \le k \le n \tag{11}$$

Eqs. (1) and (2) formulate the two objectives of minimizing total energy consumption and maximum tardiness, respectively. We now identify the relation between the two objectives. Suppose that $D_{\max}$ is fixed in the model. It is understandable that constraint (5) becomes tighter when $D_{\max}$ decreases and all the other constraints of the model remain the same. This implies that the optimal value of the total energy consumption shows a non-decreasing trend as $D_{\max}$ decreases. Therefore, there is a trade-off between the two objectives.

Eq. (3) ensures that the kth job should be processed after it is released. Eqs. (4) and (5) ensure that the maximum tardiness is well defined. Eq. (6) states that if a turn-off-on operation is carried out after the $(k-1)$th job, the starting time of the kth job should be no less than the completion time of the kth job plus the breakeven duration T_B; when there is no turn-off-on operation, it should be no less than the completion time of the previous job. If $y_{k-1}=0$, Eq. (7) means that the energy consumed between the $(k-1)$th job and the kth job should be no less than P_v multiplied by the duration between the completion time of the $(k-1)$th job and the starting time of the kth job. Note that if $y_{k-1}=1$, Eq. (7) becomes redundant, where M is a very large number. Eq. (8) tells that if a turn-off-on operation is performed, then the energy consumption at this position should be no less than E_s. Finally, Eqs. (9) and (10) ensure that each position can be assigned with only one job and each job can be only processed in one position, respectively.

Note that there is no constraint in the model to ensure $C_{k+1}-C_k-t_{k+1} < T_B$ holds when $y_k=0$. In fact, such a constraint can be relaxed as stated below.

Theorem 1. *For any optimal solution to the MILP model, there exists no k, $1 \le k \le n-1$, such that $y_k=0$ and $C_{k+1}-C_k-t_{k+1} \ge T_B$ hold at the same time.*

Proof. We prove the theorem by contradiction. Suppose that there exists an optimal solution such that $y_k=0$ and $C_{k+1}-C_k-t_{k+1} \ge T_B$. The fact that $y_k=0$ means there is no turn-off-on operation executed immediately after the kth job. On the other hand, relation $C_{k+1}-C_k-t_{k+1} \ge T_B$ means the idle time between the kth job and the $(k+1)$th job lasts longer than the breakeven period T_B. For this reason, if we change the value of y_k from 0 to 1 (i.e. the machine is shut down between the two jobs), we can obtain a solution with lower energy consumption. This is in contradiction with the optimality of the solution. This ends the proof. □

Corollary 1. For any optimal solution to the model, if $y_k=0$, then $C_{k+1}-C_k-t_{k+1} < T_B$ must hold and vice versa.

Therefore, there is no need to formulate an extra constraint to guarantee that if $y_k=0$, then $C_{k+1}-C_k-t_{k+1} < T_B$. Note that if such a disjunctive constraint is included in the model, then a big M must be introduced to transform the constraint to a linear one. It is understandable that an MILP model without a big M is tighter than the one with a big M. The less the number of big M's in an MILP model, generally the tighter the model will be. Thus, the model becomes more compact without such a disjunctive constraint. Computational experiment also demonstrates that the model runs faster after the removal of such a constraint with a big M while the optimal solutions remain unchanged.

3. An advanced ε–constraint method

3.1. Basic ε–constraint method

The problem considered in this paper is a bi-objective combinatorial optimization problem, which can be generally formulated as below:

$$\begin{aligned} & min \quad \{f_1(x), \quad f_2(x)\}, \\ \text{subject to} \quad & x \epsilon X, \end{aligned}$$

where x is the solution vector and X stands for the space of solution vectors that satisfy the constraint set.

Since the two objectives are conflicting, there is no method of optimizing both of them simultaneously. Hence our aim is to find a reasonable trade-off between the two objectives such that no better solution exists. The following are some basic definitions related to bi-objective optimization (e.g. Miettinen, 1998).

Definition 1. For any pair of solutions x_1 and $x_2 \in X$, we say x_1 dominates x_2 if $f_1(x_1) \le f_1(x_2)$ and $f_2(x_1) < f_2(x_2)$ hold or $f_1(x_1) < f_1(x_2)$ and $f_2(x_1) \le f_2(x_2)$ hold.

Definition 2. A Pareto optimal solution is the one that no solution $x \in X$ dominates it.

Definition 3. All the Pareto optimal points over the objective space for a problem are called its Pareto front.

There have been many popular approaches to solving bi-objective optimization problems, such as the weighted sum scalarization method and the ε–constraint method (Berube et al., 2009; Feng et al., 2014; Wu et al., 2015; Che et al., 2017b). In the weighted sum scalarization method, different objectives are multiplied by different weights and transformed into a single objective by calculating the sum. The key step of this method is to change the weights of different objectives in order to obtain the whole Pareto front. Since this method only changes the weights of objectives and no additional constraints are imposed, the sub-problems can be easily solved as long as the corresponding single-objective optimization problem can be easily solved. Different from the weighted sum scalarization method, the ε–constraint method transforms a bi-objective optimization problem into a single-objective problem by making one objective to be minimized a constraint bounded from above by a parameter ε and forms the so-called ε–constraint problem. By varying the value of ε according to some specific rule, we may obtain all the Pareto optimal points for the problem. To implement the ε–constraint method, for simplicity, we use $P_E(\varepsilon_1)$ (or $P_D(\varepsilon_2)$) to denote the ε–constraint problem:

$min\, E \;\; (or \;\; min\, D)$,

subject to

$x \epsilon X$,
$D \le \varepsilon_1 \;\; (or \;\; E \le \varepsilon_2)$,

and $P'_E(\varepsilon_1)$ (or $P'_D(\varepsilon_2)$) to denote the problem:

$min\; E \;\; (or \;\; min\, D)$,

subject to

$x \epsilon X$,
$D = \varepsilon_1 \;\; (or \;\; E = \varepsilon_2)$,

For a given ε_1, we use $E(\varepsilon_1)$ (or $E'(\varepsilon_1)$) to represent the objective value obtained by solving $P_E(\varepsilon_1)$ (or $P'_E(\varepsilon_1)$). Similarly, for a given ε_2, we use $D(\varepsilon_2)$ (or $D'(\varepsilon_2)$) to represent the objective value obtained by solving $P_D(\varepsilon_2)$ (or $P'_D(\varepsilon_2)$).

Theorem 2. *For given ε_1 and ε'_1 (resp. ε_2 and ε'_2), if $\varepsilon_1 < \varepsilon'_1$ (resp. $\varepsilon_2 < \varepsilon'_2$), then $E(\varepsilon_1) \ge E(\varepsilon'_1)$ (resp. $D(\varepsilon_2) \ge D(\varepsilon'_2)$) must hold.*

Proof. If $\varepsilon_1 < \varepsilon'_1$ (resp. $\varepsilon_2 < \varepsilon'_2$), all the constraints in the formulation of $P_E(\varepsilon_1)$ (resp. $P_D(\varepsilon_2)$) remain the same except $D \le \varepsilon_1$ (resp. $E \le \varepsilon_2$), which becomes tighter as ε_1 (resp. ε_2) decreases. Therefore, the optimal objective value of $P_E(\varepsilon_1)$ (resp. $P_D(\varepsilon_2)$) should be no less than $P_E(\varepsilon'_1)$ (resp. $P_D(\varepsilon'_2)$). This ends the proof. □

To better fix the interval of ε, we give the definitions of ideal point and Nadir point (Berube et al., 2009), which depict the precise area we need to explore in order to obtain the whole Pareto front.

Definition 4. The ideal point of the bi-objective optimization problem addressed is defined as $(E^I, D^I)=\{(E,\ D)|E=min_{x\,\epsilon\,X}(E),\ D=min_{x\,\epsilon\,X}(D)\}$. The Nadir point is defined as $(E^N, D^N)=\{(E,\ D)|E=E'(D^I), D=D'(E^I)\}$.

Theorem 3. *The lower bound of E (i.e. E^I) is zero.*

Generally, the lower bound of E can be obtained by solving the MILP model without constraint (5) to minimize E. However, for our case, we can derive the lower bound without solving an MILP model as explained below. Note that E represents the total energy consumed for the idle machine and for all turn-off-on operations. Let us consider a specific situation (schedule) where the first job is processed at a sufficiently late time such that all the other jobs can be processed immediately after its previous job has been processed without any delay. In this way, the machine will not stand idle or experience any turn-off-on operations in the whole processing course. Naturally in this case, the total energy consumed between all pairs of consecutive jobs (i.e. E) is zero. Thus, the lower bound of E must be zero.

Corollary 2. $(0, D'(0))$ is a Pareto optimal point of the problem.

If we remove the objective function of total energy consumption and constraints (7) and (8) from the model and then solve the

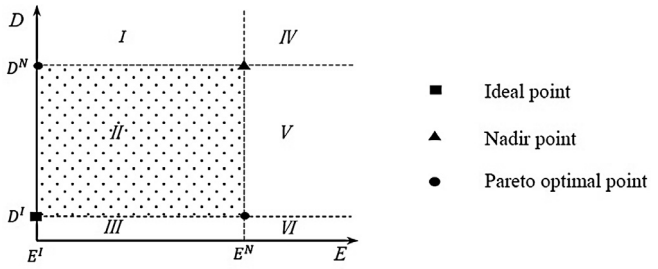

Fig. 1. Area division with ideal point and Nadir point.

corresponding model, we can obtain the smallest objective value of maximum tardiness, which we denote as D_{min}. It is clear that D_{min} is the lower bound of the maximum tardiness for this problem.

Corollary 3. $(E'(D_{min}), D_{min})$ is a Pareto optimal point of the problem.

From what has been discussed above, the ideal point of the problem is $(0,\ D_{min})$ while the Nadir point is $(E'(D_{min}), D'(0))$, which are depicted in Fig. 1. We can see from Fig. 1 that the whole area (i.e. $E \geq 0,\ D \geq 0$) is divided into six segments. Obviously, points in segments *III* and *VI* are infeasible ones. Points in segment *I* are dominated by (E^I, D^N), and points in segment *V* are dominated by (E^N, D^I). For points in segment *IV*, they are dominated by both (E^I, D^N) and (E^N, D^I). Finally, segment *II* contains all the Pareto optimal points. In other words, this segment is the only one we need to consider.

In the problem addressed in this paper, we use (E^0, D^0) to represent a Pareto optimal point over the objective space, which denotes the total energy consumption and maximum tardiness, respectively.

To implement the ε–constraint method, we first need to conduct a property analysis for our problem. For a Pareto optimal point (E^0, D^0) of the problem, we check its corresponding optimal solution, and divide the n jobs into m groups using the following rule. In the same group, the starting time of any job is equal to the completion time of its previous job (except the first job in each group). We denote the group as G_i, $1 \leq i \leq m$, where m is the number of groups. Let N_i be the number of jobs in group G_i, and G_p^q represent the job in the qth position of the pth group. For example, for group G_h, the completion time of G_h^i is equal to the starting time of G_h^{i+1}, $1 \leq i \leq N_h - 1$. We now give the definition of an effective group.

Definition 5. An effective group G_e in the solution corresponding to (E^0, D^0) is the one that satisfies:

1) The starting time of G_e^1 exceeds the completion time of $G_{e-1}^{N_{e-1}}$ by no less than T_B;
2) The starting time of G_{e+1}^1 exceeds the completion time of $G_e^{N_e}$ by less than T_B;

An illustration of an effective group is shown in Fig. 2. For the first group (i.e., G_1) in a solution, we regard it as an effective group if it satisfies conditions 2) of Definition 5. As will be discussed later, postponing the starting times of the jobs in an effective group will lead to decreased total energy consumption. Note that the last group (i.e., G_m) in a solution is not an effective one because the above postponing operation performed on the last group will not decrease the total energy consumption.

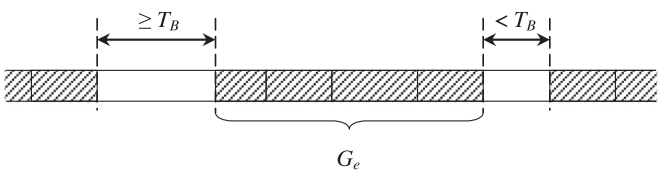

Fig. 2. Illustration of an effective group.

Theorem 4. *In the solution corresponding to the Pareto optimal point (E^0, D^0), there must exist a job with the maximum tardiness D^0 in any effective group.*

Proof. We prove the theorem by contradiction. Suppose that in an effective group, say G_e, the maximum tardiness of its jobs is less than D^0. According to Definition 5, the starting time of G_e^1 exceeds the completion time of $G_{e-1}^{N_{e-1}}$ by no less than T_B, which means that the machine is shut down between the two jobs; and the starting time of G_{e+1}^1 exceeds the completion time of $G_e^{N_e}$ by less than T_B, which means the machine is idle. If we postpone the starting time of each job in G_e by some time units such that the maximum tardiness of jobs in G_e is exactly D^0. In this case, the energy consumed between $G_{e-1}^{N_{e-1}}$ and G_e^1 remains the same (i.e. E_s), while the energy consumed between $G_e^{N_e}$ and G_{e+1}^1 is reduced. This implies that we have obtained a new solution that dominates (E^0, D^0). This contradicts the optimality of (E^0, D^0). This ends the proof. □

Theorem 5. *If there exists an effective group in the solution corresponding to (E^0, D^0), then for any arbitrarily small value of θ, there must exist a feasible point $(E^0 - \theta * P_v,\ D^0 + \theta)$ that cannot be dominated by (E^0, D^0).*

Proof. Suppose that G_e is an effective group in the solution corresponding to (E^0, D^0). Similar to the proof of Theorem 4, if we postpone the starting time of each job in this group by θ time unit, then the tardiness of the job with the maximum tardiness D^0 before postponement is increased by θ time unit, which is $D^0 + \theta$. At the same time, the energy consumed between $G_{e-1}^{N_{e-1}}$ and G_e^1 remains the same, which is E_s, while the energy consumed between $G_e^{N_e}$ and G_{e+1}^1 is reduced by $\theta * P_v$. That is, we obtain a feasible point $(E^0 - \theta * P_v, D^0 + \theta)$ which cannot be dominated by (E^0, D^0). This implies that for any arbitrarily small value of θ, there must exist a Pareto optimal solution either with the energy consumption $E^0 - \theta * P_v$ or with the maximum tardiness $D^0 + \theta$ among the solution space. This ends the proof. □

Corollary 4. If there exists an effective group in any Pareto optimal point of the problem, then it has infinite number of Pareto optimal points.

From Corollary 4, since the number of the Pareto optimal points is infinite, it is impossible to obtain them all. We only consider the situation where E and D are restricted to integer values for the following three reasons. First, with such a restriction, we can find all Pareto optimal integer points via an exact ε-constraint method by decreasing the value of ε by one each time. Second, such a restriction also allows us to devise and incorporate specific local search algorithms to enhance the basic ε-constraint method. Third, the difference between an optimal integer value and an optimal real value can be made arbitrarily small by scaling on input data. As to how to implement the ε–constraint method to solve our problem, we have to choose one objective to be the primary one, and use an ε-constraint to represent the other one. Here, we set maximum tardiness as the primary objective and use the ε-constraint to formulate the energy consumption, because in this way, local search and preprocessing technique, which will be introduced later, can be much easier to apply and detailed reasons will be presented later.

To obtain the whole integer Pareto front, we use the following basic ε–constraint method and make ε decrease by one each time to search for the next optimal point from an obtained one.

Algorithm A1 :

1: Compute the interval $[0, E'(D_{min})]$ and set $\varepsilon = E'(D_{min})$;
2: Set $\varepsilon = \varepsilon - 1$;
3: Solve problem $P_D(\varepsilon)$ and obtain the corresponding $D(\varepsilon)$;
4: If $D(\varepsilon)$ is integer, go to Step 5; otherwise, set $D(\varepsilon) = \lfloor D(\varepsilon) \rfloor + 1$, where $\lfloor D(\varepsilon) \rfloor$ is the largest integer no more than $D(\varepsilon)$, and then go to Step 5;
5: Solve problem $P'_E(D(\varepsilon))$ to obtain the corresponding $E'(D(\varepsilon))$, and set $\varepsilon = E'(D(\varepsilon))$;
6: If $\varepsilon = 0$, stop; otherwise go to Step 2.

Note that $D(\varepsilon)$ in Step 5 must be integer. Hence, the obtained $E'(D(\varepsilon))$ must be integer as well due to the structure of the MILP model. With Algorithm A1, we can obtain the exact integer Pareto front of the problem. But with it alone, we can only solve small-scale problems and its computational efficiency is generally low. Considering the particularity of the problem, we propose local search, preprocessing technique and valid inequalities to enhance the basic $\varepsilon-$constraint method, which will significantly improve its speed.

3.2. Local search

Due to the particularity of the problem, we develop local search to strengthen Algorithm A1. In the algorithm, each iteration involves solving two MILP problems, which consumes considerable amount of time. The local search improves the algorithm by solving only one MILP problem in each iteration while keeping the Pareto optimality of the solutions obtained.

The proposed local search starts from a Pareto optimal point, say (E^0, D^0), and tries to find the next feasible point without missing out any of the integer Pareto optimal points. Specifically, there are two types of operations for local search, which are applied when there exist effective groups in the Pareto optimal solution and no effective group exists, respectively.

For the condition where there exist effective groups in the solution corresponding to (E^0, D^0), the first type of local search lies in postponing the starting times of all jobs in each effective group by one time unit. Then we can obtain a feasible solution whose maximum tardiness is larger than that of the previous Pareto optimal point by one (i.e. D^0+1) and smaller energy consumption by at least P_v.

Theorem 6. *If there exists only one effective group in the solution corresponding to (E^0,D^0), then we can obtain the feasible point (E^0-P_v,D^0+1) with local search.*

Proof. To prove the theorem, we have to use the concept of effective group introduced in Definition 5. If we find an effective group, say G_e, in the solution corresponding to (E^0,D^0), then we can postpone the starting times of all jobs in G_e by one time unit. Similar to the proof of Theorem 5, the energy consumed between $G_{e-1}^{N_{e-1}}$ and G_e^1 after the postponement will remain the same while that between $G_e^{N_e}$ and G_{e+1}^1 will be reduced by P_v. At the same time, the maximum tardiness is increased to D^0+1 due to the postponement. Therefore, we obtain the feasible point (E^0-P_v,D^0+1). This ends the proof. □

Theorem 7. *If there exist k_1 ($k_1 \geq 1$) effective groups in the solution corresponding to (E^0,D^0), then we can obtain the feasible point $(E^0-k_1{}^*P_v,D^0+1)$ with local search and no integer Pareto optimal point is omitted from (E^0,D^0) to $(E^0-k_1{}^*P_v,D^0+1)$.*

Proof. As is discussed in the proof of Theorem 6, if we postpone the starting times of all jobs in each of the k_1 effective groups, we can achieve a reduction of E^0 by $k_1{}^*P_v$ contributed by the postponement of each effective group and make the previous maximum tardiness D^0 increase by only one unit, which corresponds to the feasible point $(E^0-k_1{}^*P_v,D^0+1)$. Any point $(E^0-\delta{}^*P_v,D^0+1)$ for $1 \leq \delta \leq k_1-1$ is dominated by $(E^0-k_1{}^*P_v,D^0+1)$. Therefore, there is no integer Pareto optimal point being omitted from (E^0,D^0) to $(E^0-k_1{}^*P_v,D^0+1)$. This ends the proof. □

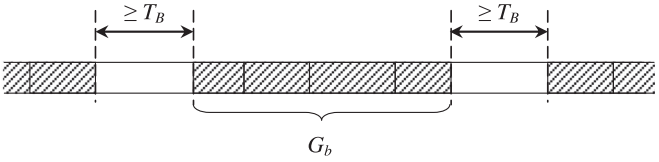

Fig. 3. Illustration of a blocked group.

The next step after postponement is to check the Pareto optimality of the point $(E^0-k_1{}^*P_v,D^0+1)$. To achieve this purpose, we solve $P'_E(D^0+1)$ and obtain $E'(D^0+1)$. It is clear that $(E'(D^0+1), D^0+1)$ is the next Pareto optimal point. Note that the above local search fails when there exists no effective group in the solution. To tackle such condition, we first give the definition of blocked group, on which the second type of local search is based. Similarly, for a blocked group, say group G_b, its number of jobs is denoted by N_b, and G_b^q represents the job in its qth position.

Definition 6. A blocked group G_b is the one that satisfies:

1) The starting time of G_b^1 exceeds the completion time of $G_{b-1}^{N_{b-1}}$ by no less than T_B;
2) The starting time of G_{b+1}^1 exceeds the completion time of $G_b^{N_b}$ by no less than T_B;

Fig. 3 illustrates a blocked group. Obviously, the turn-off-on operation is executed both before and after a blocked group. For the first group (i.e., G_1) in a solution, we regard it as a blocked group if it satisfies condition 2) of Definition 6. Note that the last group (i.e., G_m) in a solution is not a blocked group due to the same reason explained for the effective group. Particularly, if there is only one group in a solution, then it is neither a blocked group nor an effective group. As a result, there is no need to carry out the local search on it. Therefore, in what follows, we only consider the solutions with at least two groups.

Theorem 8. *If no effective group exists in the solution corresponding to the Pareto optimal point (E^0,D^0), there must exist at least one blocked group.*

Proof. Let us consider the first group (i.e., G_1) in the solution. By assumption, there are at least two groups in the solution and G_1 is not the last group. Since G_1 is not an effective group, it follows that the starting time of G_2^1 minus the completion time of $G_1^{N_1}$ is no less than T_B, which means that condition 2) of Definition 6 must hold for G_1. By definition, G_1 must be a blocked group. This ends the proof.

Under the condition where no effective group exists, the second type of local search lies in postponing the starting times of all jobs in a blocked group, say G_b, until the starting time of G_{b+1}^1 minus the completion time of $G_b^{N_b}$ is equal to $\frac{E_s}{P_v}-1$ time units. Specifically, for the condition where $\frac{E_s}{P_v} > T_s$, the breakeven duration $T_B = \frac{E_s}{P_v}$. Hence after executing the second type of local search, the machine is kept idle between the processing of $G_b^{N_b}$ and G_{b+1}^1. Therefore, the energy consumed between $G_b^{N_b}$ and G_{b+1}^1 is reduced to E_s-P_v. Similar analysis conducted under the conditions where $\frac{E_s}{P_v} < T_s$ and $\frac{E_s}{P_v} = T_s$ show that the energy consumed between $G_b^{N_b}$ and G_{b+1}^1 are both reduced by P_v after the second type of local search. Note that the energy consumed between $G_{b-1}^{N_{b-1}}$ and G_b^1 remains the same, since the machine is still shut down between the

processing of them after the postponement. Hence the total energy consumption becomes smaller than E^0 by P_v. At the same time, the maximum tardiness D^0 is increased by at least one time unit. Note that in the second type of local search, we only need to postpone one blocked group.

Theorem 9. *For any optimal schedule corresponding to (E^0,D^0), after executing the second type of local search, the job with maximum tardiness must lie in the postponed blocked group.*

Proof. We prove the theorem by contradiction. Suppose that after postponement, the job with maximum tardiness lies in other groups, which remains D^0 since they are not postponed, then we can obtain a feasible solution corresponding to (E^0-P_v,D^0), which dominates (E^0,D^0). This contradicts the assumption that (E^0,D^0) is a Pareto optimal point. This ends the proof. □

Detailed reasons of deciding which blocked group to postpone will be presented later. Similar to the first type of local search, the next step after postponement is to check the Pareto optimality of the point (E^0-P_v,D^0+k_2), where k_2 is the time units by which the chosen blocked group has been postponed. Similarly, to achieve this purpose, we solve $P'_D(E^0-P_v)$ and obtain $D'(E^0-P_v)$. Obviously, $(E^0-P_v,D'(E^0-P_v))$ is the next Pareto optimal point.

The whole procedure of local search can be summarized as:

The procedure of local search

1: Divide all the jobs in a Pareto optimal solution into groups
2: **If** there exist effective groups, **then**
  2.1: Locate all the k_1 effective groups and
    2.1.1: Postpone the starting times of all jobs in each effective group by one time unit and update their completion times and tardiness;
    2.1.2: Check the Pareto optimality of $(E^0-k_1{*}P_v,\ D^0+1)$;
3: **Else** choose a blocked group G_b and
  3.1: Postpone the starting times of all jobs in G_b until the starting time of G^1_{b+1} minus the completion time of $G^{N_b}_b$ is equal to $\frac{E_s}{P_v}-1$ time units;
  3.2: Check the Pareto optimality of (E^0-P_v,D^0+k_2), where k_2 is the time units by which the chosen blocked group has been postponed

3.3. Dynamic preprocessing technique and valid inequalities

In the MILP model, the binary variable x_{ik} formulates the processing sequence of jobs while y_k represents whether there should be a turn-off-on operation between the kth and $(k+1)$th job. Naturally, the number of binary variables in an MILP model can affect its computational time to a large extent. If we can pre-determine the values of some binary variables, then the computational time may be reduced significantly. Inspired by this idea, we propose the following theorems.

Theorem 10. *Given maximum tardiness D^*, for any pair of jobs i and j, if $r_i \geq d_j+D^*-t_j$ holds, then job i must be processed after job j.*

Proof. By definition, r_i is the earliest starting time for the machine to process job i, since each job can only be processed after it is released. Note that $d_j+D^*-t_j$ is the latest starting time for the machine to process job j in the solution with the maximum tardiness being D^*. If the relation $r_i \geq d_j+D^*-t_j$ is satisfied, then the earliest starting time of job i lies after the latest starting time of job j. This suggests that job i must be processed after job j. This ends the proof.

Corollary 5. In the MILP model, $x_{ik}=0$ for all $1 \leq k \leq \alpha$, where α represents the number of jobs that must be processed before job i. That is, $r_i \geq d_k+D^*-t_k$ holds for all $1 \leq k \leq \alpha$.

Corollary 6. In the MILP model, $x_{jk}=0$ for all $n-\beta+1 \leq k \leq n$, where β is the number of jobs processed after job j. That is $r_k \geq d_j+D^*-t_j$ holds for all $n-\beta+1 \leq k \leq n$.

Theorem 11. *At most $\frac{E^*}{E_s}$ turn-off-on operations exist in the solution with the total energy consumption being E^*.*

Proof. For given total energy consumption E^*, since each turn-off-on operation consumes E_s, the total times of turn-off-on operations must be no more than $\frac{E^*}{E_s}$. This ends the proof. □

Corollary 7. Given total energy consumption E^*, $\sum_{k=1}^{n-1} y_k \leq \frac{E^*}{E_s}$ holds, which provides an upper bound on the total number of turn-off-on operations.

Since r_i, d_j, t_j and E_s in equations $r_i \geq d_j+D^*-t_j$ and $\sum_{k=1}^{n-1} y_k \leq \frac{E^*}{E_s}$ are fixed while D^* and E^* are dynamically changed with the iteration going on, the relations may vary as well. In Algorithm A1, the dynamic value of maximum tardiness (i.e.D^*) is increasing while that of energy consumption (i.e.E^*) decreasing.

For the relation $r_i \geq d_j+D^*-t_j$, smaller value of D^* yields more equations described in Corollaries 5 and 6, which can accelerate the algorithm more. This idea is also the basis of deciding which blocked group to be postponed in the second type of local search. When multiple blocked groups can be found in the solution corresponding the Pareto optimal point (E^0,D^0), postponing different blocked groups brings about different values of maximum tardiness. If we choose the one with the smallest maximum tardiness, then preprocessing can be the most powerful when solving $P'_D(E^0+P_v)$.

Similarly, for the relation $\sum_{k=1}^{n-1} y_k \leq \frac{E^*}{E_s}$, smaller value of E^* will make the valid inequalities more efficient. In particular, if there exist k_1 $(k_1 \geq 1)$ effective groups in the solution corresponding to (E^0,D^0), which means that we can obtain a feasible solution corresponding to $(E^0-k_1{*}P_v,D^0+1)$, then $\sum_{k=1}^{n-1} y_k \leq \frac{E^0-k_1{*}P_v}{E_s}$ can be applied when solving $P'_E(D^0+1)$.

With the preprocessing technique and valid inequalities stated above, the original algorithm can be accelerated significantly. The procedure of obtaining a new Pareto optimal point from a known one based on the advanced ε–constraint method combined with local search, preprocessing technique and valid inequalities is summarized in Fig. 4. Note that k_1 represents the number of effective groups in the solution corresponding to (E, D) while k_2 is the time units by which the chosen blocked group is postponed.

We now explain why we choose the maximum tardiness as the primary objective from the perspective of facilitating local search and preprocessing, respectively.

In terms of local search, if we set E as the primary objective, then the crucial step in this situation is to advance the starting time of certain jobs in order to make the maximum tardiness reduced by one. The problem is that in a Pareto optimal solution, there may exist multiple jobs with the maximum tardiness D^0. In this case, we have to reduce the maximum tardiness of all these jobs by one time unit while at the same time the total energy consumption is allowed to increase by only P_v. This can be very hard to achieve.

From the perspective of preprocessing, as is introduced before, it is based on a known maximum tardiness, which means if we have no idea of the value of D before solving $P_D(E)$, preprocessing cannot be applied. In terms of the $\varepsilon-$constraint method, the first step is to calculate the effective interval of the secondary objective. If we set E as the primary objective, then we have to calculate the effective interval of D (i.e. $[D_{min},D'(E_{min})]$), where preprocessing cannot be applied. However, if we set D as the primary objective, we can utilize preprocessing to calculate the effective interval of E, which may accelerate the algorithm to some extent.

3.4. Clustering of jobs to obtain approximate Pareto front

With the procedure described in Fig. 4, as will be presented in the next section, the problem with up to 25 jobs can be solved to

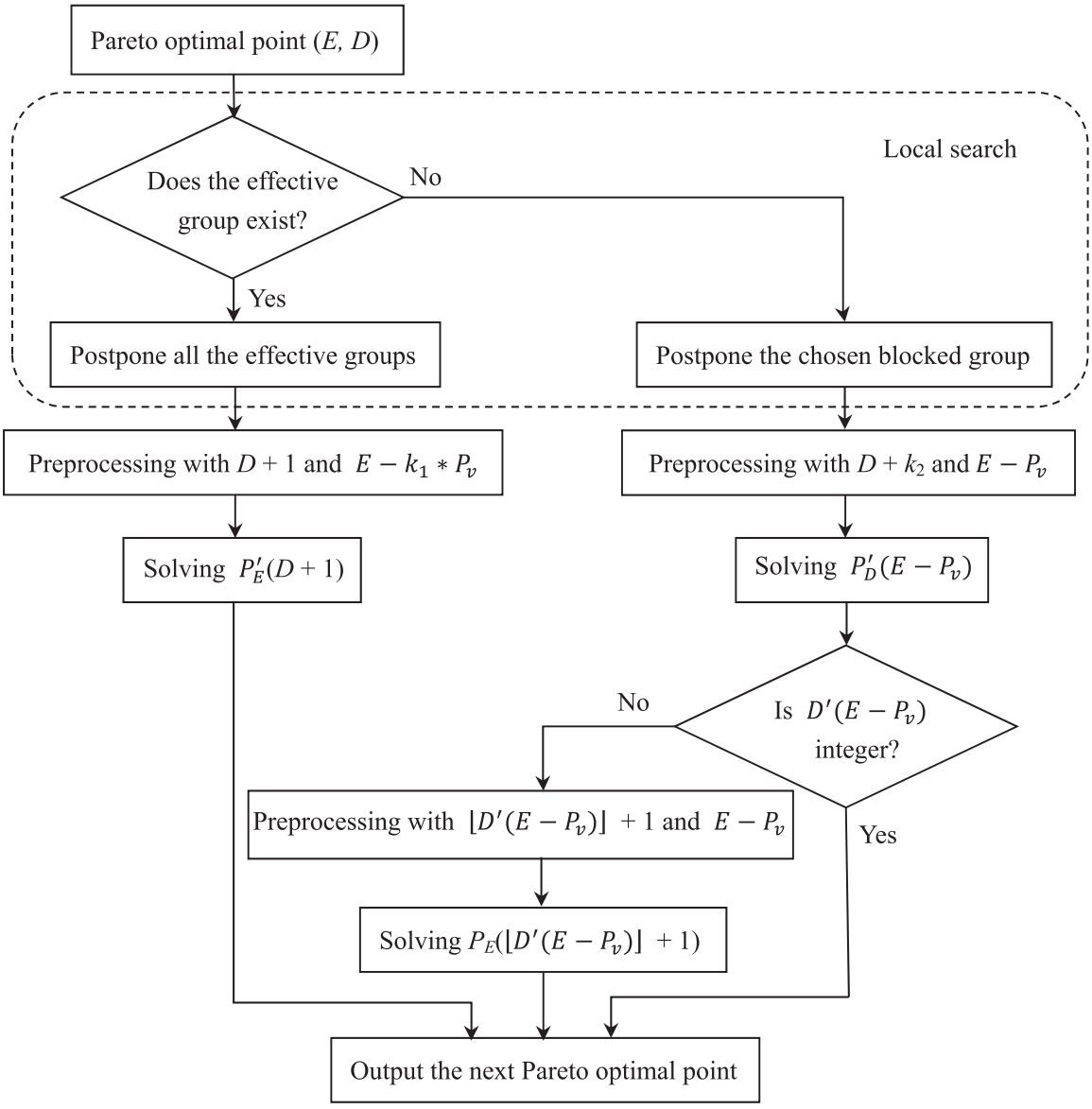

Fig. 4. The procedure of obtaining a new Pareto optimal point from a known one.

optimality within reasonable time. Further analysis of optimal solutions reveals that the jobs with smaller values of release times and due dates tend to be processed earlier. On the other hand, in terms of preprocessing technique, a given maximum tardiness always restricts that jobs with large (or small) values of release times and due dates cannot be processed at the earliest (or at the latest). Inspired by that, to obtain approximate Pareto front for large-size problems, we utilize the method of cluster analysis to divide the jobs into several sorted clusters based on their release times and due dates. Any job in a preceding cluster must be processed before all the jobs in subsequent clusters, thus reducing the solution space significantly. Although by clustering of jobs, the Pareto front we obtained is approximate, the problem scale that our algorithm is able to solve increases significantly.

Specifically, to divide n jobs into m ($m \leq n$) clusters, we propose the clustering procedure based on the following analysis. As is discussed above, the release time and due date of each job play the most crucial role when deciding its processing sequence. Hence, we define the distance between any pair of jobs i and j according to their release times and due times.

Definition 7. The distance of job i and j, $L_{i,j} = \sqrt{(r_i - r_j)^2 + (d_i - d_j)^2}$.

Let Θ_a, $1 \leq a \leq m$, represents the set of indices of jobs assigned to cluster a. The distance between two clusters, say a and b, is defined as the shortest distance between all the jobs in them, which is given in Definition 8.

Definition 8. The distance of clusters a and b, $L_{a,b} = min_{i\in\Theta_a,\ j\in\Theta_b}\{L_{i,j}\}$.

Note that the total number of clusters generated needs to be pre-determined. The proposed clustering procedure is an iterative process. In each iteration, two clusters with the shortest distance are merged into one until the number of clusters is equal to m. For simplicity, let Ψ be the set of indices of non-empty clusters. With the notations above, the clustering procedure of dividing n jobs into m ($m \leq n$) clusters can be summarized as below.

The clustering procedure

1: Initialization: $\Psi \Leftarrow \emptyset$ and **For** $a=1$ to n **do**
  1.1: $\Theta_a \Leftarrow \{a\}$
  1.2: $\Psi \Leftarrow \Psi \cup \{a\}$
2: **For** each $a \in \Psi$ **do**
  2.1: **For** each $b \in \Psi$ $(b > a)$ **do**
    2.1.1: **For** each $i \in \Theta_a$ and $j \in \Theta_b$ calculate $L_{i,j}$
    2.1.2: Select the least value of all the obtained $L_{i,j}$'s and denote it as $L_{i',j'}$
    2.1.3: $L_{a,b} \Leftarrow L_{i',j'}$
3: Select the least value among all $L_{a,b}$'s and denote it as $L_{a',b'}$
4: $\Psi \Leftarrow \Psi - \{b'\}$, $\Theta_{a'} \Leftarrow \Theta_{a'} \cup \Theta_{b'}$
5: **If** Ψ contains exactly m elements, **then** go to Step 7
6: **Else**, go to Step 2
7: Output Θ_a for $a \in \Psi$

To explain why the above clustering procedure is able to accelerate the proposed approach, we assume that the sth cluster contains N_s jobs, $1 \le s \le m$. Since the processing sequence of clusters is fixed, all jobs in cluster s must be processed before those in cluster $s+1$ for any $1 \le s \le m$-1. That is, the processing position for some job in a certain cluster, say job h in cluster s, can only be chosen from the positions to which cluster s corresponds, which are positions $\sum_{j=1}^{s-1} N_j + 1$,..., $\sum_{j=1}^{s} N_j$. Hence, job h cannot be processed in the positions of the clusters before cluster s, which means that $x_{hz} = 0,\ z = 1, \ldots, \sum_{j=1}^{s-1} N_j$ must hold. Similarly, it cannot be processed in positions of the clusters after cluster s, which implies that $x_{hz} = 0,\ z = \sum_{j=1}^{s} N_j + 1, \ldots, \sum_{j=1}^{m} N_j$ must hold. Therefore, after examining all the jobs, the values of considerable quantity of binary variables can be fixed, which significantly reduces the computational time for the algorithm to solve the problem. Note that with less number of clusters, we will fix the values of less binary variables, which leads to more precise Pareto front while consuming more computational time.

4. Computational results

4.1. Benchmark instances

To test the proposed algorithm, we use several benchmark instances taken from the literature, for which approximate Pareto fronts were known. Though the objectives of their models may be different from ours, we can apply our algorithm to obtain the exact Pareto front of their problems with slight changes in our model. All the model formulations in this paper are implemented in C++ solved by the MIP solver of CPLEX (Version 12.5). The experiments are performed on a HP PC with an Intel Core i5-2400 CPU operating at 3.10 GHz.

Mouzon and Yildirim (2008) presented a two-job benchmark instance with the objective of minimizing both total energy consumption and total tardiness. The release times of the two jobs are 0 and 4 time units, the processing times being 2 and 1 time unit, and the due dates being 3 and 6 time units, respectively. The machine consumes 2 units of power when it is processing jobs and 1 unit when idle. The energy and time for a turn-off-on operation are 1.5 units of energy and 2 time units, respectively. In their work, the energy required to turn on the machine initially and turn it off ultimately and energy consumed when it is processing jobs are also taken into account. Since they use a greedy randomized multi-objective adaptive search metaheuristic, two nearly non-dominated points (7.5, 0) and (6, 1) are obtained. Using our algorithm, we can obtain two exact Pareto integer points: (7, 0) and (6, 1).

Yildirim and Mouzon (2012) gave a three-job instance to minimize both total energy consumption and total completion time. In their instance, the processing times of job 1 and job 3 are both 1 time unit while that of job 2 is 2 time units. They assume that job 1 and job 2 have zero release times and the release time of job 3 is 4 time units. The machine consumes 1 unit of power when it is idle and 2 units when processing jobs. A turn-off-on operation takes 1 time unit and costs 1.5 units of energy. With our algorithm, we can obtain the same exact integer Pareto optimal points as Yildirim and Mouzon (2012): (9, 9) and (8, 11), which only takes about 0.4 s.

Table 1
Processing data of the five jobs.

Jobs	1	2	3	4	5
Release time (h)	0	30	50	80	150
Processing time (h)	10	20	20	30	10
Due date (h)	200	200	200	200	200

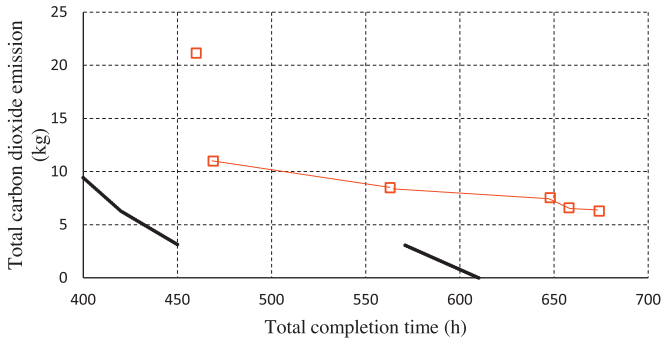

Fig. 5. The Pareto front of a five-job instance.

Liu et al. (2014) presented a five-job instance. Their aim is to minimize the total carbon dioxide emission and the total completion time simultaneously. The carbon dioxide emission is equal to the amount of energy consumption multiplied by the carbon conversion coefficient of electricity λ, which takes the value of 0.785 $kgCO_2$/kwh in their work. The total completion time can be denoted by $\sum_{k=1}^{n} C_k$. The processing of the five jobs follows the First-Come-First-Serve rule, whose processing data are shown in Table 1.

In their instance, the machine consumes 0.4 kw when it is idle and 3 kw when working. Each turn-off-on operation takes 12 h and requires 4 kwh of electricity. To implement our ε – constraint method, we set the carbon dioxide emission as the primary objective and the total completion time is decreased by one hour in each iteration. In Fig. 5, the Pareto front obtained by our ε – constraint method is depicted as the black points while Liu et al.'s is given with the red square points. We can see from Fig. 5 that our algorithm outperforms theirs significantly in terms of the quality and number of the Pareto optimal points. The six Pareto optimal points reported in Liu et al. (2014) are all dominated by those obtained using our algorithm.

4.2. Randomly generated instances

To further evaluate the proposed algorithm, we utilize randomly generated instances. The way in which our data are generated is the same as that by Mouzon and Yildirim (2008). The processing time of each job follows a uniform distribution between 1 and 5 time units. The release time of each job follows an exponential distribution with a mean of 10 time units. The due date of each job is the sum of three parts: its release time, processing time and a slack time (s_i), which follows a uniform distribution between 0 and $\beta \times \sum_{i=1}^{n} t_i$. The parameter β reflects the degree of how far the due date deviates from the release time. As the value of β increases, the due dates become more spread over time.

In this section, we test in total three algorithms. The first two are Algorithm A1 and Algorithm A1 combined with local search, preprocessing technique and valid inequalities, called A1LP below. Note that Berube et al. (2009) also proposed an exact ε – constraint method for the bi-objective optimization problem and we

Table 2
Comparison of computational times for the three algorithms.

		Number of jobs			
		10	15	20	25
$\beta=0.05$	A1LP	5.88	14.53	62.65	1854.46
	A1	14.53	51.09	364.02	–
	A2P	26.38	78.14	266.83	–
$\beta=0.1$	A1LP	6.51	11.54	58.19	944.61
	A1	13.36	48.85	424.28	–
	A2P	33.10	72.84	218.56	–
$\beta=0.15$	A1LP	5.29	15.59	53.94	3483.60
	A1	13.57	48.18	269.22	–
	A2P	29.08	68.73	347.28	–
$\beta=0.2$	A1LP	6.75	12.04	73.82	2769.21
	A1	14.54	45.67	422.27	–
	A2P	29.03	87.66	290.62	–

–: Problems that cannot be solved to optimality within reasonable time.

Table 3
Numbers of Pareto optimal solution for test instances.

	Average number of jobs			
	10	15	20	25
$\beta=0.05$	15.17	20.90	24.00	30.60
$\beta=0.10$	15.67	19.37	23.67	30.20
$\beta=0.15$	14.17	19.53	24.37	26.30
$\beta=0.20$	15.33	17.80	23.00	27.40

also compare it with our algorithms. Note that due to the particularity of our problem, fractional optimal points may be obtained by their algorithm. Hence we need to slightly modify Berube et al.'s algorithm and remove the non-integer points from the obtained Pareto front. In their algorithm, suppose the two objectives to be optimized is z_1 and z_2, without loss of generality, $P_{z_1}(\epsilon)$ is solved in each iteration and then ε is assigned with the corresponding value of z_2^* minus one. Since the obtained z_2^* may be non-integer, the modification here is to set $\epsilon = \lfloor z_2^* \rfloor$ in this case. There is no guarantee that the solution obtained in each iteration of their algorithm is non-dominated. Since the local search proposed in this paper embarks on a Pareto optimal solution, there is no way that we can apply local search to it. For simplicity, we denote this algorithm combined with preprocessing technique and valid inequalities as A2P.

We test the three algorithms using the instances with the number of jobs being 10, 15, 20, 25 and the value of β being 0.05, 0.10, 0.15, 0.20 respectively. All the instances are generated in the same rule, and each setting has been tested for 10 instances and we calculate their means. Table 2 shows the computational times for the three algorithms, and Table 3 gives the average number of Pareto optimal points for each setting of instances.

We can see from Table 2 that, in terms of computational time, A1LP outperforms greatly than A2P. To be more specific, the average computational time consumed by A2P is at least four times of that by A1LP. The comparison between A1 and A1LP reveals that the proposed local search, preprocessing technique and valid inequalities have significantly accelerated the basic $\varepsilon-$constraint method. Table 3 shows that the number of Pareto optimal points does not vary significantly with β.

4.3. Approximate Pareto front based on cluster analysis

To enable our algorithm to solve large-scale problems, we utilize the clustering technique introduced in Section 3. The solution performance measurement of our algorithm (i.e., A1LP) under different clustering circumstances is based on the observation that less clusters will lead to more precise Pareto front while consuming more computational time.

Table 4
Computational time of A1LP for small-size problems.

		Number of jobs			
		10	15	20	25
$\beta=0.05$	No clustering	5.88	14.53	62.65	1854.46
	$n/3$ clusters	5.29	8.92	13.29	21.93
	$n/2$ clusters	4.92	8.24	10.92	17.43
$\beta=0.1$	No clustering	6.51	11.54	58.19	944.61
	$n/3$ clusters	5.75	7.46	13.04	22.40
	$n/2$ clusters	5.25	6.87	10.82	17.69
$\beta=0.15$	No clustering	5.29	15.59	53.94	3483.60
	$n/3$ clusters	4.74	9.43	15.41	20.34
	$n/2$ clusters	4.29	8.61	11.74	15.25
$\beta=0.2$	No clustering	6.75	12.04	73.82	2769.21
	$n/3$ clusters	5.97	7.51	13.93	22.53
	$n/2$ clusters	5.42	7.15	10.86	15.47

Table 5
Solution performance of A1LP for small-size problems.

		Number of jobs			
		10	15	20	25
$\beta=0.05$	$n/3$ clusters	100%	100%	100%	100%
	$n/2$ clusters	100%	100%	100%	100%
$\beta=0.1$	$n/3$ clusters	100%	100%	100%	100%
	$n/2$ clusters	99.13%	100%	100%	100%
$\beta=0.15$	$n/3$ clusters	100%	100%	100%	100%
	$n/2$ clusters	100%	99.57%	100%	100%
$\beta=0.2$	$n/3$ clusters	100%	100%	100%	99.67%
	$n/2$ clusters	100%	98.46%	100%	99.67%

For small-size problems, we test the algorithm with the job number being 10, 15, 20, 25 under three circumstances. We set the number of clusters as 1, $n/2$ and $n/3$, respectively. Obviously, when there is only one cluster (i.e. no clustering is conducted), we will obtain the exact Pareto front of the problem. To evaluate the algorithm under the other two circumstances, we propose the following rule. For any instance, we use Ω_i, $i=1, 2, 3$ to denote the Pareto front obtained when there is 1, $n/2$ and $n/3$ clusters, respectively. Let Ω_i', $i=2, 3$ be the sets of points in Ω_2 and Ω_3 that cannot be dominated by any point in Ω_1, respectively. The solution performance of A1LP when there are $n/2$ and $n/3$ clusters is measured as

$$per^i = \frac{\Omega_i'}{\Omega_1} \times 100\%, \quad i = 2, 3$$

Their computational time and corresponding solution performance are shown in Table 4 and Table 5, respectively.

From Table 4, we can see that the computational time has been significantly reduced when we apply the clustering procedure. On the other hand, Table 5 reveals that the solutions obtained by A1LP under different settings are almost the same (i.e., all non-dominated). That is, its solution performance is hardly affected by job clustering. However, with the help of the clustering procedure our algorithm has been accelerated greatly.

For large-size problems with the job number being 30, 35, 40, 45 and 50, we test three circumstances, namely there are $n/3$, $n/2$ and $2n/3$ clusters, respectively. As is introduced before, less clusters lead to more precise Pareto front. Hence we use the Pareto front obtained when there are $n/3$ clusters to evaluate the algorithm under the other two circumstances utilizing the same rule for small-size problems. The computational time and solution performance are shown in Tables 6 and 7, respectively.

Tables 6 and 7 indicate that the computational time of A1LP can be significantly reduced when the number of clusters increases,

Table 6
Computational time of A1LP for large-size problems.

		Number of jobs				
		30	35	40	45	50
$\beta=0.05$	$n/3$	32.24	47.56	111.64	118.38	178.30
	$n/2$	22.02	29.33	42.52	57.91	70.00
	$(2\times n)/3$	18.37	23.40	34.20	42.37	50.64
$\beta=0.1$	$n/3$	29.74	51.97	79.53	138.59	424.11
	$n/2$	21.83	32.46	41.75	61.22	78.68
	$(2\times n)/3$	17.61	24.41	33.01	41.11	55.66
$\beta=0.15$	$n/3$	32.42	44.46	103.78	156.05	1179.38
	$n/2$	22.25	27.09	47.07	66.30	93.81
	$(2\times n)/3$	19.03	19.13	34.54	41.76	54.11
$\beta=0.2$	$n/3$	35.09	59.42	95.22	301.53	1464.25
	$n/2$	24.44	33.89	44.68	80.70	142.73
	$(2\times n)/3$	18.36	25.32	32.67	45.03	57.15

Table 7
Solution performance of A1LP for large-size problems.

		Number of jobs				
		30	35	40	45	50
$\beta=0.05$	$n/2$	100%	100%	100%	100%	100%
	$(2\times n)/3$	100%	99.21%	100%	100%	100%
$\beta=0.1$	$n/2$	100%	99.63%	100%	100%	98.47%
	$(2\times n)/3$	100%	94.40%	96.48%	89.80%	95.93%
$\beta=0.15$	$n/2$	99.29%	98.40%	98.73%	96.38%	85.94%
	$(2\times n)/3$	97.88%	74.34%	84.65%	78.95%	70.47%
$\beta=0.2$	$n/2$	100%	88.18%	90.32%	92.09%	74.80%
	$(2\times n)/3$	87.05%	79.18%	77.37%	71.39%	55.36%

but the solutions we obtain are less precise. Another observation from the two tables is that when dealing with problems with lower values of β, the A1LP with the clustering procedure can obtain more precise Pareto front of the problem. Generally speaking, the proposed clustering procedure has significantly accelerated the algorithm while leaving its solution performance only slightly affected. For example, even for 50-job problems with $n/2$ clusters and $\beta=0.2$, the algorithm can still reach 74.80% solution performance. That is only 25.20% of the Pareto optimal points obtained with $n/2$ clusters are dominated by those with $n/3$ clusters.

5. Conclusion

This paper has addressed a single-machine scheduling problem with power-down mechanism to minimize total energy consumption and maximum tardiness simultaneously. An MILP model based on position assignment has been developed to formulate the problem. To obtain the exact Pareto front of the problem, we proposed a basic $\varepsilon-$constraint method and developed local search, preprocessing technique and valid inequalities to enhance the proposed model. To solve large-size problems, we applied the cluster analysis method to divide jobs into several sorted clusters based on their release times and due dates, which significantly reduces the solution space of the problem.

The computational results on several benchmark instances showed that the proposed exact algorithm is able to obtain the exact Pareto fronts and outperforms the existing approaches in terms of the quality of the Pareto front. In addition, the computational experiments on randomly generated instances demonstrated that the proposed clustering procedure enables the algorithm to solve larger-size problems with slight loss in solution performance. In its practical significance, the proposed model and methods provide production managers with a decision support tool to make reasonable trade-offs between energy consumption and maximum tardiness in production scheduling.

Future works on this issue might focus on multi-machine environments, in which the complexity of the problem will be greatly increased. Another extension of this study is to identify other objectives and develop new methods to obtain the exact Pareto fronts.

Acknowledgments

The authors are grateful to the Area Editor and two anonymous reviewers for their constructive comments and suggestions, which helped us greatly improve the presentation of the paper. This work was partially supported by the National Natural Science Foundation of China under grant nos. 71471145, 71571032 and 71071129.

References

Abdelaziz, E.A., Saidur, R., Mekhilef, S., 2011. A review on energy saving strategies in industrial sector. Renewable and Sustainable Energy Rev. 15 (1), 150–168.
Altiok, T., 1997. Performance Analysis of Manufacturing Systems. Springer.
Angel, E., Bampis, E., Kacem, F., Letsios, D., 2011. Speed scaling on parallel processors with migration. Lect. Notes Comput. Sci. 7484 (1), 128–140.
Babu, C.A., Ashok, S., 2008. Peak load management in electrolytic process industries. IEEE Trans. Power Syst. 23 (2), 399–405.
Bansal, N., Kimbrel, T., Pruhs, K., 2007. Speed scaling to manage energy and temperature. J. ACM 54 (1), 347–357.
Berube, J.F., Gendreau, M., Potvin, J.Y., 2009. An exact -constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. Eur. J. Oper. Res. 194 (1), 39–50.
Che, A., Zeng, Y., Lyu, K., 2016. An efficient greedy insertion heuristic for energy-conscious single machine scheduling problem under time-of-use electricity tariffs. J. Clean. Prod. 129, 565–577.
Che, A., Zhang, S., Wu, X., 2017a. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. J. Clean. Prod. doi:10.1016/j.jclepro.2017.04.018, Published online.
Che, A., Zhang, Y., Feng, J., 2017b. Bi-objective optimization for multi-floor facility layout problem with fixed inner configuration and room adjacency constraints. Comput. Ind. Eng. 105, 265–276.
Dai, M., Tang, D., Giret, A., Salido, M.A., Li, W.D., 2013. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Rob. Comput. Integr. Manuf. 29 (5), 418–429.
Drake, R., Yildirim, M.B., Twomey, J.M., Whitman, L.E., Ahmad, J.S., Lodhia, P., 2006. Data collection framework on energy consumption in manufacturing. *Institute of Industrial Engineering Research Conference*, May 2006.
Fang, K., Uhan, N., Zhao, F., Sutherland, J.W., 2011. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. J. Manuf. Syst. 30 (4), 234–240.
Fang, K., Uhan, N.A., Zhao, F., Sutherland, J.W., 2013. Flow shop scheduling with peak power consumption constraints. Ann. Oper. Res. 206 (1), 115–145.
Fang, K., Uhan, N.A., Zhao, F., Sutherland, J.W., 2016. Scheduling on a single machine under time-of-use electricity tariffs. Ann. Oper. Res. 238 (1-2), 199–227.
Feng, J., Che, A., Wang, N., 2014. Bi-objective cyclic scheduling in a robotic cell with processing time windows and non-Euclidean travel times. Int. J. Prod. Res. 52 (9), 2505–2518.
International Energy Agency (IEA), 2015. World Energy Investment Outlook. IEA, Paris.
Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included. Naval Res. Logist. Q. 1 (1), 61–68.
Jovane, F., Yoshikawa, H., Alting, L., Boer, C.R., Westkamper, E., Williams, D., Paci, A.M., 2008. The incoming global technological and industrial revolution towards competitive sustainable manufacturing. CIRP Ann.Manuf. Tech. 57 (2), 641–659.
Kasap, N., Aytug, H., Paul, A., 2006. Minimizing makespan on a single machine subject to random breakdowns. Oper. Res. Lett. 34 (1), 29–36.
Kolliopoulos, S.G., Steiner, G., 2006. Approximation algorithms for minimizing the total weighted tardiness on a single machine. Theor. Comput. Sci. 355 (3), 261–273.
Li, J., Meerkov, S.M., 2010. Production Systems Engineering. Betascript Publishing.
Liu, C., Yang, J., Lian, J., Li, W., Evans, S., Yin, Y., 2014. Sustainable performance oriented operational decision-making of single machine systems with deterministic product arrival time. J. Cleaner Prod. 85, 318–330.
Luo, H., Du, B., Huang, G.Q., Chen, H., Li, X., 2013. Hybrid flow shop scheduling considering machine electricity consumption cost. Int. J. Prod. Econ. 146 (2), 423–439.
Marchal, V., Dellink, R., 2011. OECD Environmental Outlook to 2050: Climate Change Chapter. Organization for Economic Cooperation and Development (OECD) Publishing.
Mendez, C.A., Cerda, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Comput. Chem. Eng. 30 (6-7), 913–946.
Miettinen, K., 1998. Nonlinear Multiobjective Optimization, 12. Springer, US.

Mouzon, G., Yildirim, M.B., 2008. A framework to minimise total energy consumption and total tardiness on a single machine. Int. J. Sustainable Eng. 1 (2), 105–116.
Mouzon, G., Yildirim, M.B., Twomey, J., 2007. Operational methods for minimization of energy consumption of manufacturing equipment. Int. J. Prod. Res. 45 (18-19), 4247–4271.
Pach, C., Berger, T., Sallez, Y., Bonte, T., Adam, E., Trentesaux, D., 2014. Reactive and energy-aware scheduling of flexible manufacturing systems using potential fields. Comput. Ind. 65 (3), 434–448.
Park, C.W., Kwon, K.S., Kim, W.B., Min, B.K., Park, S.J., Sung, I.H., Seok, J., 2010. Energy consumption reduction technology in manufacturing — A selective review of policies, standards, and research. Int. J. Precis. Eng. Manuf. 10 (5), 151–173.
Pruhs, K., Stee, R.V., Uthaisombut, P., 2008. Speed scaling of tasks with precedence constraints. Theory Comput. Syst. 43 (1), 67–80.
Rajemi, M.F., Mativenga, P.T., Aramcharoen, A., 2010. Sustainable machining: selection of optimum turning conditions based on minimum energy considerations. J. Cleaner Prod. 18 (10–11), 1059–1065.
Ross, M., 1992. Efficient energy use in manufacturing. Proc. Natl. Academy Sci. United States Am 89 (3), 827–831.
Shrouf, F., Ordieres-Meré, J., García-Sánchez, A., Ortega-Mier, M., 2014. Optimizing the production scheduling of a single machine to minimize total energy consumption costs. J. Cleaner Prod. 67 (6), 197–207.
Sudarsan, R., Sriram, R.D., Narayanan, A., Sarkar, P., Lee, J.H., Lyons, K.W., Kemmerer, S.J., 2010. Sustainable manufacturing: Metrics, standards, and infrastructure-workshop summary. Int. J. Sustainable Manuf. 2 (2/3), 144–149.
Swaminathan, V., Chakrabarty, K., 2003. Energy-conscious, deterministic I/O device scheduling in hard real-time systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 22 (7), 847–858.
Tacconi, L., 2000. Biodiversity and ecological economics: Participation, values and resource management. Biolo. Conserv. 107, 259–260.
Tseng, F.T., Stafford, E.F., 2008. New MILP models for the permutation flowshop problem. J. Oper. Res. Soc. 59 (10), 1373–1386.
Wierman, A., Andrew, L.L.H., Lin, M., 2012. Speed scaling: An algorithmic perspective. In: Handbook of Energy, pp. 385–406.
Wu, P., Che, A., Chu, F., Zhou, M., 2015. An improved exact epsilon-constraint and cut-and-solve combined method for biobjective robust lane reservation. IEEE Trans. Intell. Transp. Syst. 16 (3), 1479–1492.
Xu, D., Liu, M., Yin, Y., Hao, J., 2013. Scheduling tool changes and special jobs on a single machine to minimize makespan. Omega-Int. J. Manage. Sci. 41 (2), 299–304.
Xu, D., Wan, L., Liu, A., Yang, D.-L., 2015. Single machine total completion time scheduling problem with workload-dependent maintenance duration. Omega-Int. J. Manage. Sci. 52, 101–106.
Yao, F., Demers, A., Shenker, S., 1995. A scheduling model for reduced CPU energy. In: Foundations of Computer Science Annual Symposium on, pp. 374–382.
Yildirim, M.B., Mouzon, G., 2012. Single-machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm. IEEE Trans. Eng. Manage. 59 (4), 585–597.
Zeng, Y., Che, A., Wu, X., 2017. Bi-objective scheduling on uniform parallel machines considering electricity cost. Eng. Optim. doi:10.1080/0305215X.2017.1296437, Published online.