Computers & Industrial Engineering 64 (2013) 224–234

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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

Parallel-machine scheduling to minimize tardiness penalty and power cost ☆

Kuei-Tang Fang, Bertrand M.T. Lin *

Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Hsinchu 300, Taiwan

ARTICLE INFO

Article history:
Received 2 March 2012
Received in revised form 3 October 2012
Accepted 10 October 2012
Available online 17 October 2012

Keywords:
Parallel-machine scheduling
Total weighted tardiness
Dynamic voltage scaling
Particle swarm optimization

ABSTRACT

Traditional research on machine scheduling focuses on job allocation and sequencing to optimize certain objective functions that are defined in terms of job completion times. With regard to environmental concerns, energy consumption becomes another critical issue in high-performance systems. This paper addresses a scheduling problem in a multiple-machine system where the computing speeds of the machines are allowed to be adjusted during the course of execution. The CPU adjustment capability enables the flexibility for minimizing electricity cost from the energy saving aspect by sacrificing job completion times. The decision of the studied problem is to dispatch the jobs to the machines as well as to determine the job sequence and processing speed of each machine with the objective function comprising of the total weighted job tardiness and the power cost. We give a formal formulation, propose two heuristic algorithms, and develop a particle swarm optimization (PSO) algorithm to effectively tackle the problem. Since the existing solution representations do not befittingly encode the decisions involved in the studied problem into the PSO algorithm, we design a tailored encoding scheme which can embed all decisional information in a particle. A computational study is conducted to investigate the performances of the proposed heuristics and the PSO algorithm.

1. Introduction

In traditional scheduling research, the processing speeds of machines are assumed to be static. This paper investigates a scheduling problem with heterogeneous parallel machines that exhibits the flexibility to adjust their processing speeds for each individual job. The flexibility of machine speed adjustment comes from real world applications that invoke intense computing tasks on a cluster of computers. The clock-rate of a processor is determined by the frequency of an oscillator crystal. It was commonly accepted that a computer runs at its highest speed so as to attain the best performances. When a computer runs at a higher speed/frequency, it will not only consume more electricity but also produce more heat. To cool down the high temperature, the computer needs to activate extra devices, like fans, thus causing more energy consumption. In scheduling problems, most of the objective functions are regular, i.e. non-decreasing in terms of job completion times. No ground seems to exist for slowing down the machine speeds to defer the processing of jobs. Nevertheless, if the delivery is not urgent or the implied cost of deference is offset by the reduction of energy cost, it is reasonable to reduce the processing speeds of the processors.

Nowadays more and more industrials not only pursuit better work efficiency, but also focus on energy-saving issues. Slower paces are attracting more considerations in different industries. For example, in maritimes, ships with lower voyage speeds are much preferred due to a lower consumption of energy. Demands for lower electricity costs also arise in the IT industry. Cloud computing has emerged as a booming business model. While some service providers start developing a wide variety of services to attract more users or customers, others on the other hand focus on how to optimize work schedules for efficiency. Cloud computing service providers host a larger number of processors that are deployed to fulfill the orders of computing tasks placed by clients. In addition to customer satisfaction and efficient utilization of facilities, lower electricity consumption is also crucial to the service providers. Koomey (2007) showed that the electricity used for servers worldwide, including their associated cooling and auxiliary equipment, cost 7.2 billion US dollars in 2005. The cost in that year had doubled when compared with consumption in 2000. Therefore, the electricity issue is not negligible but critical because energy consumption plays one of the key roles in the overall cost structure. To attain energy saving objectives, the CPU frequency can be adjusted through the dynamic voltage scaling technique. The CPU adjustment technique allows for flexibility to seek for potential balance between job completion times and energy cost.

This paper considers the objective function with a weighted sum of the total job tardiness penalty and the total power cost. The former one is an efficiency indicator in traditional schedule

☆ This manuscript was processed by Area Editor T.C. Edwin Cheng.
* Corresponding author. Tel.: +886 3 5131472; fax: +886 3 5723792.
E-mail addresses: jacky0831@gmail.com (K.-T. Fang), bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

0360-8352/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2012.10.002

Nomenclature

Notation

$\mathcal{J}$	set of jobs $\{J_1, J_2, \ldots, J_n\}$ to be scheduled
j	index $1 \leqslant j \leqslant n$ of jobs
$\mathcal{M}$	set of machines $\{M_1, M_2, \ldots, M_m\}$ for scheduling
i	index $1 \leqslant i \leqslant m$ of machines
l	index $1 \leqslant l \leqslant n$ of positions on the machines
s	index $1 \leqslant s \leqslant \mu_i$ of speeds of machine M_i, where μ_i is the number of different speeds of machine M_i
ℓ_j	processing load of job J_j
d_j	due date of job J_j
C_j	completion time of job J_j
T_j	tardiness of job J_j, i.e. $\max\{C_j - d_j, 0\}$
w_j	penalty of unit-time tardiness
S_{is}	s-th processing speed of machine M_i
e_{is}	energy consumption (per time unit) of machine M_i at speed S_{is}
t_{jis}	execution time of job J_j if assigned to machine M_i at speed s; $t_{jis} = \ell_j / S_{is}$
p_{jis}	actual power consumption of job J_j if it is executed by machine M_i at speed S_{is}, i.e. $t_{jis} \times e_{is}$

Decision variables

x_{jils}	if job J_j is assigned to the lth position on machine M_i at speed S_{is}, then $x_{jils} = 1$; otherwise, $x_{jils} = 0$
b_{il}	starting time of the job at the lth position on machine M_i

problems to reflect both customer service (concerning average waiting time) and internal inventory management (concerning work-in-process parts). This weighted combinatorial objective function can significantly close to the business considerations in schedule planning. The machines exhibit the possibility to allow the shop floor manager to select their processing speeds by adjusting CPU frequencies. Higher speeds permit a shorter processing makespan of jobs, yet more power cost will be incurred. The scheduling policy thus rests in the trade-off between the performance measures defined in job completion times and the total power cost. To be more precisely, we have to (a) dispatch the jobs to the machines; (b) sequence the jobs on each machine; and (c) select the processing speed for each job.

The remainder of the paper is organized as follows: In Section 2, we first give a formal definition of the studied problem and present the notations that will be used throughout this paper. An integer linear programming (ILP) model follows. Literature review on related scheduling problems and the concept of energy saving is also presented. Since the studied problem is strongly NP-hard, we adopt approximation approaches to produce approximate schedules in a reasonable time. Two heuristic algorithms are proposed in Section 3. Then, in Section 4, we introduce the meta-heuristic particle swarm optimization (PSO). The development of a PSO algorithm, especially the tailor-designed encoding scheme, for the studied problem is introduced in Section 5. Section 6 is dedicated to a computational study for appraising the effectiveness of the proposed algorithms. Concluding remarks and suggestions for several future research subjects are given in Section 7.

2. Problem statements and literature review

This section presents a formal definition of the allocation and scheduling of jobs from the aspect of parallel-machine scheduling. Following the problem definition is an integer linear programming (ILP) model that describes the studied problem.

2.1. Problem definition and ILP

Consider a set of n jobs $\{J_1, J_2, \ldots, J_n\}$ to process on a set of m heterogeneous machines $\{M_1, M_2, \ldots, M_m\}$. The jobs can be processed on any machine. Each job J_j is characterized by a processing load ℓ_j and a due date d_j, before which the job is assumed to be finished, and a weight indicating the penalty of unit-time tardiness w_j. Due to the operating characteristics, we can select the machine frequency to adjust the processing speed for the jobs. A higher frequency allows the machines to run faster at the cost of more energy consumption. The actual processing time of a job J_j thus depends on the speed selected for the machine in charge of it. The decision is to assign and schedule the jobs to the machines as well as to select appropriate execution speeds for processing the jobs. The objective function to minimize is the weighted sum of the tardiness penalties of jobs and the energy consumption of the machines. Total weighted tardiness and energy consumption cost may not be added up in a natural way for they are not of the same type of measures. Yet it is possible to adopt a normalization through a common measurement, say a monetary base. The weighted sum of two criteria adopted in this paper is for demonstrating the significance of the proposed models and solution approaches.

In the following, we introduce the notations that will be used throughout the paper and the integer programming formulation.

In the following, we present an integer linear programming model of the studied problem, which is denoted as the WTPC (Weighted Tardiness and Power Consumption) problem.

Problem WTPC:

$$\text{Minimize} \quad \sum_{i=1}^{n} w_j T_j + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{s=1}^{\mu_i} t_{jis} e_{is} x_{jils} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{m}\sum_{l=1}^{n}\sum_{s=1}^{\mu_i} x_{jils} = 1, \ 1 \leqslant j \leqslant n; \tag{2}$$

$$\sum_{j=1}^{n}\sum_{s=1}^{\mu_i} x_{jils} \leqslant 1, \ 1 \leqslant i \leqslant m, \ 1 \leqslant l \leqslant n; \tag{3}$$

$$b_{i,l+1} - b_{i,l} = \sum_{j=1}^{n}\sum_{s=1}^{\mu_i} x_{jils} t_{jis}, \ 1 \leqslant i \leqslant m, \quad 1 \leqslant l \leqslant n-1;$$

$$T_j \geqslant b_{i,l} + t_{jis} x_{jils} - (1 - x_{jils})M - d_j, \ 1 \leqslant i \leqslant m, 1 \leqslant s \leqslant \mu_i, \quad 1 \leqslant j, \ l \leqslant n; \tag{5}$$

$$T_j \geqslant 0, \ 1 \leqslant j \leqslant n; \tag{6}$$

$$x_{jils} \in \{0, 1\}, \ 1 \leqslant i \leqslant m, \ 1 \leqslant s \leqslant \mu_i, \ 1 \leqslant j, \ l \leqslant n. \tag{7}$$

The objective function of Eq. (1) consists of two parts. The first part is concerned about the total tardiness penalty, and the second part gives the total power consumption resulted from the dispatching decision. Constraints (2) confine each job to be processed on exactly one position of some machine running at a specific speed. Constraints (3) reflect the fact that each position of any machine can accommodate at most one job. Constraints (4) define the starting time of the job in each position on the machines. The computation of the tardiness $T_j = \{C_j - d_j, 0\}$ of each job is given in constraints (5) and (6). We would elaborate on constraints (5) in more details. If job J_j is not scheduled at the lth position on machine i with speed s, then $x_{jils} = 0$ and the inequality is automatically satisfied due to the negative value $-M$. On the other hand, when $x_{jils} = 1$, the value

of T_j will reflect the difference between the job completion time and the job due date.

2.2. Literature review

As aforementioned in Section 1, the subject of energy saving techniques draws considerable research attention. Adjusting CPU speeds, by sacrificing job completion times, has been proven to be one of the most significant and effective approaches (Bunde, 2006). The main idea of CPU speed adjustment is supported by dynamic voltage scaling (DVS), which equips the machines with the ability to change their computing states or clock frequencies. With the possibility to change the processing speeds of machines, many of the traditional scheduling problems can be discussed again. Some researchers have combined the idea of energy saving and DVS with traditional scheduling problems. Zhu, Melhem, and Childers (2003) proposed two algorithms for task scheduling with and without precedence constraints on multiprocessor systems. Zhu, Mosse, and Melhem (2004) designed a greedy slack stealing scheme to solve the AND/OR model of real-time applications and showed that their scheme outperforms other existing ones. Ge, Feng, and Cameron (2005) proposed distributed performance-directed DVS scheduling strategies for use in scalable power-aware HPC clusters and obtained significant energy savings (36%) without increasing the required execution time. Kumar and Palani (2012) applied the DVS technique to reduce the energy consumption of the embedded system technology to mobile systems. They used genetic algorithm for solving the problem of minimizing schedule length subject to an energy consumption constraint and the problem of minimizing energy consumption subject to a schedule length constraint. Rizvandi, Zomaya, Lee, Boloori, and Taheri (2012, chap. 17) addressed the energy issue with task scheduling and presented the MFS-DVFS algorithm to reduce energy consumption. Akgul, Puschini, Lesecq, Miro-Panades, and Benoit (2012) applied DVFS techniques to mobile computing platforms where performance constraints, such as task deadlines, are given. They recast the problem in a linear program and solve the problem by the simplex algorithm.

The objective function considered in this paper comprises the minimization of total weighted tardiness and power cost. For the single-machine scheduling problem, Lawler (1977) designed a pseudo-polynomial time algorithm for solving the $1||\sum T_j$ problem. The complexity status remained open until Du and Leung (1990) gave an NP-hardness proof to confirm the intractability of the $1||\sum T_j$ problem. The weighted version of this problem, $1||\sum w_j T_j$, is known to be NP-hard in the strong sense (Lenstra, Rinnooy Kan, & Brucker, 1977). In the case of parallel machines, Lenstra et al. (1977) showed the problem with two machines to be ordinary NP-hard. For the case with a fixed number of machines, Garey and Johnson (1978) gave an proof of ordinary NP-hard. The case with an arbitrary number of machines turns to be NP-hard in the strong sense (Garey & Johnson, 1978). Hence, the complexity of the total weighted tardiness of parallel machines is also NP-hard in the strong sense. Some well-known meta-heuristics are applied to solve parallel-machine total weighted tardiness problems (Anghinofi & Paolucci, 2007; Runwei, Mitsuo, & Tatsumi, 1995).

The studied WTPC problem is concerned about (a) assigning the jobs to the machines, (b) determining the processing sequence of the jobs on each machine, and (c) selecting the processing speed for each job, such that the weighted sum of tardiness penalty and power cost in minimum. From the intrinsic complexity of the scheduling problem with the objective function of total tardiness, WTPC is obviously strongly NP-hard. As a consequence, it is very unlikely to design polynomial or pseudo-polynomial time algorithms to solve the problem to optimality. This paper develops two heuristics and a meta-heuristic algorithm to derive approximate solutions.

3. Heuristics

This section introduces two construction heuristics, based upon the earliest due date (EDD) rule, and the weighted shortest processing time (WSPT) rule, respectively. The jobs are then dispatched to the machines by the list scheduling algorithm (Graham, 1966), which assigns the first unscheduled job to the machine with the shortest completion time until all jobs are dispatched.

The EDD rule, proposed by Jackson (1955), guarantees the optimality in the minimization of the maximum lateness on a single machine. It has been widely adopted to deal with due-date related objective functions. For example, Baker (1974) showed that based on the EDD rule, if there is at most one tardy job, the minimum total tardiness is guaranteed. Therefore, we adopt the EDD rule and expect that it will work reasonably well when the number of tardy jobs is small. This first heuristic algorithm is outlined in the following:

Due-date-based heuristic H_{EDD}.

Step 1. Sort the jobs in non-decreasing order of due dates.
Step 2. Remove the first job, say J_j, from the list and assign it to the machine which has the shortest completion time.
Step 3. Select the machine speed that minimizes the contributions (weighted tardiness and power cost) that J_j will make to the objective function.
Step 4. Execute job J_j by using the speed selected in Step 3. Update the completion time of the machine.
Step 5. Repeat Steps 2–4 until all jobs are processed.

To solve the single-machine scheduling problem of minimizing the total weighted completion time, Smith (1956) proposed the WSPT rule that sequences the jobs in non-decreasing order of the ratio between job processing time and job weight. In view of a special case where all jobs have a common due date, the total weighted tardiness is minimized by the WSPT rule. This stimulates the adoption of the WSPT rule for dealing with the problem of minimizing the total weighted tardiness, especially when most jobs tend to be tardy.

WSPT-based heuristic H_{WSPT}

Step 1. Sort the jobs in non-decreasing order of the ratio between processing load and weight, ℓ_j/w_j.
Step 2. Remove the first job, say J_j, from the list and assign it to the machine which has the shortest completion time.
Step 3. Select the machine speed that minimizes the contributions (weighted tardiness and power cost) that J_j will make to the objective function.
Step 4. Execute job J_j by using the speed selected in Step 3. Update the completion time of the machine.
Step 5. Repeat Steps 2–4 until all jobs are processed.

The comparison between the two heuristics would be carried out in Section 6. In the next section, we develop a meta-heuristic to produce better solutions at the cost of longer computing time.

4. Particle swarm optimization

In the nature, some animals exhibit social behavior, like a flock of birds find food sources or a school of fish avoid predators. These kinds of animals can transfer their own messages to each other and achieve some goals in a collective way. Particle swarm optimization

(PSO) is a swarm-based intelligence algorithm. Kennedy and Eberhart (1995) studied the above-mentioned behavior of swarms in the nature, such as birds, and fish and developed the PSO framework for dealing with hard optimization problems. PSO has been widely applied to many scheduling problems. Dye and Ouyang (2011) deployed PSO algorithms for solving joint pricing and lot-sizing problem. Önüt, Tuzkaya, and Doğaç (2008) studied multiple-level warehouse layout design problem by applying PSO algorithms. Ai and Kachitvichyanukul (2009) used PSO algorithms for solving the capacitated vehicle routing problem.

In PSO algorithms, each solution is represented as a particle, which is analogous to a bird flying through the solution space. Each particle is associated with two key parameters, *current position* and *current velocity*. The current position is used to evaluate the distance of the current solution away from the best position of any swarm particle so far. The current velocity, represented a vector, dictates the direction and the speed at which the bird is flying. At any time, each particle position is influenced by the best position that the particle has ever visited. The performances of the particles in different positions are measured by their fitness values, which reflect the objective function values of the studied problem. The general framework of the PSO algorithm is described below.

Let x_j^t denote the current position of particle j at iteration t, and v_j^t the velocity of particle j at iteration t. The velocity and position of solution j at the next iteration are computed by the following equations:

$$v_j^{t+1} = \omega v_j^t + c_1 \times rand_1 \times (pbest_j - x_j^t) + c_2 \times rand_2 \times (gbest - x_j^t) \quad (8)$$

$$x_j^{t+1} = x_j^t + v_j^{t+1} \quad (9)$$

where:

- ω: The inertia weight is a constant defined by the user to regulate the impact of the current velocity on the velocity of the next iteration.
- c_k: The acceleration coefficients for $k = 1, 2$, where c_1 is the acceleration coefficient indicating the attraction to the previous best position of the current particle, c_2 is the acceleration coefficient indicating the attraction to the previous best position of the swarm, and $rand_1$ and $rand_2$ are random numbers generated from the uniform interval [0,1].
- $pbest_i$: The best position ever visited in the historical course of particle i.
- $gbest$: The best position ever visited by the swarm.

As in Eq. (8), the current velocity of a particle is a linear combination of three types of velocities: (a) the inertia velocity which is related to its previous velocity; (b) the velocity to the best location found by the particle and (c) the velocity to the best location found by the swarm. Eq. (9) indicates that the current position of a particle is the previous position plus the adjustment induced by the current velocity. A larger ω value implies that the current velocity encounters a higher traction force from the previous velocity. Empirical studies of PSO with inertia weights have shown that a relatively larger ω value implies a more global search ability and on the other hand a relatively smaller ω value results in faster convergence. Larger c_1 values exploit more personal behavior, while larger c_2 values exploit more the global swarm behavior. The trade-off among the inertia weight (i.e. the previous velocity), the explorations (i.e. the global search) and the exploitations (i.e. local search) of search space is crucial to the efficiency and effectiveness of PSO algorithms. Through the adjustment of the position and the velocity of each particle iteratively, PSO algorithms could obtain near-optimal solutions when it reaches the convergence conditions or the stop criteria.

Most PSO algorithms start with a population of random solutions, each of which is associated a random velocity. Each particle keeps track of its potential best solution (*pbest*), which is the best solution in its movement history, and the global best solution (*gbest*), which is the best solution ever visited thus far by all particles of this swarm.

PSO Algorithm

Step 1. Set the particle dimension equal to the number of jobs n.
Step 2. Randomly generate a set of particle positions x_j and velocities v_j for initialization.
Step 3. For each generated particle, calculate its fitness value.
Step 4. Check if the fitness value is better than *pbest* or not. If yes, update *pbest*.
Step 5. If the minimum *pbest* in the swarm is better than *gbest*, then replace *gbest* with the minimum *pbest*.
Step 6. For each particle, use Eq. (8) to calculate the new velocity and Eq. (9) to update the position.
Step 7. Repeat from Step 3, until the stopping criteria are satisfied.

One of the key designs to the success of PSO algorithms is concerned about the representation of solutions. A proposed representation must reflect the characteristics of the studied problem as well as permit easy manipulation in the course of PSO execution. The WTPC problem involves decisions of allocation, sequencing and state tuning. It is not intuitive to construct a mapping between the particles of PSO and the solutions of WTPC. In general PSO algorithms, a particle can completely express the action of a solution, including the relative position in the solution space and the motion from the previous position to the current position. In the context of the WTPC problem, a solution should clearly indicate the machine each job is assigned to and the position and speed at which each job is processed on the machine it resides. The challenge is the necessity to encode at least four types of information, namely jobs, machines, positions and speeds in the representation of solutions.

A straightforward resolution approach is to construct a 4-dimensional array for encoding solutions and let each dimension denote one type of information. Nevertheless, there is no straightforward 4-dimensional velocity array befitting for the structure and inherent information of particles. The above-mentioned 4-dimensional array would not work when applied to Eq. (9). Another alternative design is to use a 4-level linked list to encode the solutions. Unfortunately, we cannot interpret the physical meaning of the acceleration coefficient times a linked list (Eq. (8)). Therefore, we need to design a more flexible solution encoding method, which is feasible and meaningful to the PSO algorithm, and also convenient for computing objective function values. In the next section, we will elaborate on the PSO solution encoding scheme.

5. The framework of PSO for WTPC

This section describes how PSO is adapted to solve the WTPC problem. As aforementioned, how the solutions are represented for further manipulations is crucial to the success of PSO deployment. Some solution representations applied to PSO for parallel machine scheduling problems are studied. Then, we analyze the solution structure of the WTPC problem. Since the existing solution representations are not necessarily suitable for WTPC, we need to design a new encoding method. An example solution is given for illustrating the design.

5.1. Existing solution representations

There are several types of solution representations in the deployment of PSO for coping with parallel-machine scheduling

problems. Kashan and Karimi (2009) proposed a discrete encoding approach to solve the parallel-machine scheduling problem of makespan minimization. An array with a length equal to the number of jobs is adopted to represent the solutions. The value stored in the jth position of the array corresponds to the machine to which the jth job is assigned.

Niu, Zhou, and Wang (2010) used a similar array structure and proposed a real number encoding scheme to tackle parallel-machine scheduling of minimizing the total tardiness. In each real number used, the integer part denotes the machine that the job is allocated to, and the fractional part stands for a relative execution order of the jobs on that machine.

5.2. The solution structure of WTPC problem

In the WTPC problem, besides for job dispatching and job sequencing, an additional solution encoding mechanism to describe the decision of machine speed selection. In previous papers, the integer part and the fractional part of real numbers in the solution encoding scheme are both used. One possibility for the speed selection is to use the positive sign and negative sign to express additional solution information. Yet it could be limited to binary choices, low speed or high speed. Therefore, we would like to extend the use of integer part and fractional part of real numbers to denote the selection of multiple machine speeds. Unlike the method of Niu et al. (2010) for encoding job sequencing, we need to know the exact execution speed of each job. Moreover, each machine could have distinct abilities to adjust its speeds. In other words, μ_i, the number of different speeds of machine M_i, varies across the machines. This flexibility demands an encoding mechanism which is flexible enough for precisely indicating the execution speed for each job.

5.3. New encoding method of problem WTPC

With reference to the above mentioned solution representations and difficulties encountered in the studied problem, we design a new solution encoding representation to overcome the difficulties. We first randomly generate n real numbers from a uniform distribution on the interval $[1, m+1)$ with a 3-digit fractional part. For example, if $m = 6$ machines are available, then we may generate such real numbers as $2.783, 3.513, 6.987$, and so forth. Such a real number is used to encode the information of the three decisions: job dispatching, job sequencing, and machine speed selection. The integer part stores the index of the machine to which the job is assigned. The fractional part represents the processing order of the job on its machine. For example, the job J_{j_1} annotated by the real number 2.415 and the job J_{j_2} annotated by the real number 2.873 are both processed on machine M_2, and the job J_{j_1} precedes the job J_{j_2} because $.415 < .873$. Moreover, the last 2 digits of the fractional part are also used to indicate the speed selected for the job execution. Suppose the last 2 digits of the fractional part are q_1 and q_2. For machine M_i that permits μ_i different speeds, if $\lfloor 100 \times (r-1)/\mu_i \rfloor \leqslant q_1 \times 10 + q_2 < \lceil 100 \times r/\mu_i \rceil$, then the job is executed on machine M_i at speed S_{ir} for $1 \leqslant r \leqslant \mu_i$. For example, if the last two digits of the factional part are 4 and 5 and there are three different speeds on the machines, then this job will be processed at the second speed.

5.4. Example of the new encoding method

The following numerical example solution is given to illustrate the proposed encoding scheme. Suppose that there are 6 jobs, 3 machines, and 3 different speeds on each machine. The detail interpretation of the solution $\pi = [2.554, 1.263, 1.711, 3.487, 2.822, 2.176]$ is as follows:

Job J_1 is executed on machine M_2 at speed S_{22}.
Job J_2 is executed on machine M_1 at speed S_{12}.
Job J_3 is executed on machine M_1 at speed S_{11}.
Job J_4 is executed on machine M_3 at speed S_{33}.
Job J_5 is executed on machine M_2 at speed S_{21}.
Job J_6 is executed on machine M_2 at speed S_{23}.
The job sequence on machine M_1 is (J_2, J_3).
The job sequence on machine M_2 is (J_6, J_1, J_5).
The job sequence on machine M_3 is (J_4).

The proposed solution representation approach makes a good use of all the digits of the real numbers. Another advantage is that a PSO algorithm can easily compile the stored information of a real number to realize the corresponding schedule and to calculate its objective function value. For a given solution, we sort the array in non-decreasing order of the real numbers. Knowing the integer part of a job, we can assign the job to the corresponding machine. Based on the preprocessing calculation of the processing time of each job on each machine under distinct speeds, the completion times of all jobs on all machines can be computed in $O(n)$ time. Hence, the objective function values, total weighted tardiness and total power cost, of a solution in the proposed representation approach can be computed in $O(n \log n)$ time.

As mentioned above, PSO algorithms iterate for exploring solution space until the defined convergence conditions or the stopping criteria are met. Considering the efficiency without sacrificing solution quality, we use limits on the elapsed CPU time and the number of iterations as the stopping criteria in the PSO algorithm.

Table 1
Comparison of solutions produced by CPLEX and two heuristics.

R	Time	CPLEX	H_{EDD}	Dev. (%)	H_{WSPT}	Dev. (%)
	$m = 3,\ n = 5$					
0.1	195.32	1387.44	1871.75	25.87	1932.96	28.22
0.2	45.86	872.74	1193.90	26.90	943.95	7.54
0.3	4.08	915.89	1088.31	15.84	1242.29	26.27
0.4	24.22	647.95	869.33	25.47	764.51	15.25
0.5	283.40	893.68	1091.66	18.14	1157.01	22.76
0.6	12.30	558.32	590.45	5.44	623.17	10.41
0.7	7.08	505.32	544.85	7.26	811.75	37.75
0.8	5.47	1060.59	1241.56	14.58	1252.18	15.30
0.9	13.69	596.64	672.91	11.33	709.08	15.86
1.0	19.22	993.15	1094.14	9.23	1246.23	20.31
	$m = 3,\ n = 10$					
0.1	60 min	1877.00	2327.45	19.35	3096.92	39.39
0.2	60 min	1757.94	2437.20	27.87	3049.67	42.36
0.3	60 min	1536.94	2274.49	32.43	2786.02	44.83
0.4	60 min	1315.97	1778.45	26.00	2465.03	46.61
0.5	60 min	1640.72	2876.04	42.95	3149.07	47.90
0.6	60 min	1982.68	2567.91	22.79	3836.15	48.32
0.7	60 min	2118.40	2984.72	29.03	3733.19	43.25
0.8	60 min	2004.27	3042.41	34.12	2819.48	28.91
0.9	60 min	1810.99	2386.82	24.13	3408.94	46.88
1	60 min	1623.79	2031.01	20.05	3006.34	45.99
	$m = 5,\ n = 10$					
0.1	60 min	1429.50	1786.19	19.97	2218.51	35.56
0.2	60 min	1441.12	1935.80	25.55	2376.91	39.37
0.3	60 min	1472.24	2676.17	44.99	2400.67	38.67
0.4	60 min	1297.86	2101.38	38.24	2224.99	41.67
0.5	60 min	1144.63	1805.87	36.62	2164.71	47.12
0.6	60 min	1660.90	2297.67	27.71	3066.80	45.84
0.7	60 min	1974.28	3387.65	41.72	3794.63	47.97
0.8	60 min	1422.05	1936.61	26.57	2225.21	36.09
0.9	60 min	1112.78	1586.97	29.88	2052.13	45.77
1.0	60 min	1454.65	2093.35	30.51	2786.00	47.79

That is, the PSO algorithm will keep searching for optimal solutions until the specified limits are reached.

6. Computational experiments

This section presents a computational study for examining the performances of the designed solution algorithms. The test instances were generated as follows. The processing loads l_j of the jobs were randomly generated by the uniform distribution $U(1,100)$. The tardiness penalty weights w_j of the jobs were drawn from $U(1,10)$. The generation of due dates followed the scheme proposed by Hall and Posner (2000). A parameter R was to adjust the period of due dates to reflect relative tightness. The mean value of due dates is equal to a half of the average total processing load on each machine divided by the average machine processing speed, namely

$$\overline{D} = \frac{1}{2} \times \frac{\sum_j l_j}{m} \div \frac{\sum_i \sum_s S_{is}}{\sum_i \mu_i}.$$

Therefore, $\overline{D}$ is an estimate of the makespan of the jobs and machine characteristics in the generated instance. The due dates were drawn from the uniform distribution $U[\overline{D}(1-R), \overline{D}(1+R)]$, where R is a

Table 2
Summary of the experiments setting.

	(c_1, c_2)	n	m	ω, R	Solution Approaches
E_1	(1,1)				
E_2	(1,2)	{30,50}	{3,5}	$\{0.1, 0.2, \ldots, 1.0\}$	PSO
E_3	(2,1)				
E_4	min of $\{E_1, E_2, E_3\}$	{30,50}	{3,5}		H_{EDD}, H_{WSPT}
E_5	max of $\{E_1, E_2, E_3\}$	{30,50}	{3,5}		H_{EDD}, H_{WSPT}

Table 3
Results of experiment $E_1, (c_1, c_2) = (1,1)$.

R	$\omega = 0.2$		$\omega = 0.4$		$\omega = 0.6$		$\omega = 0.8$		$\omega = 1.0$	
	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)
	30-3									
0.1	8120.4	1.95	8066.7	1.28	7964.7	0.00	8305.9	4.28	10432.2	30.98
0.2	8779.2	17.27	7901.0	5.54	7486.1	0.00	8235.7	10.01	9927.5	32.61
0.3	7996.3	8.02	8053.1	8.79	7402.4	0.00	8928.9	20.62	11394.7	53.93
0.4	8490.9	21.05	8432.3	20.22	7014.3	0.00	8568.3	22.15	10416.5	48.50
0.5	8938.5	22.68	7716.4	5.90	7286.2	0.00	8722.5	19.71	10605.9	45.56
0.6	9044.5	35.29	8414.6	25.87	6685.3	0.00	8877.2	32.79	10287.3	53.88
0.7	9110.4	31.65	8995.9	29.99	6920.3	0.00	10535.1	52.23	10271.6	48.43
0.8	8521.5	11.51	8068.8	5.58	7642.2	0.00	8348.3	9.24	10367.4	35.66
0.9	9028.0	23.85	8427.2	15.61	7289.2	0.00	8480.9	16.35	10487.6	43.88
1.0	8549.9	20.13	8770.3	23.22	7117.4	0.00	9923.2	39.42	11140.0	56.52
	30-5									
0.1	19868.8	12.84	18852.4	7.07	17607.4	0.00	22801.9	29.50	27369.0	55.44
0.2	18939.4	7.40	18485.8	4.83	17633.7	0.00	23799.4	34.97	29842.0	69.23
0.3	20991.3	15.30	18760.4	3.05	18205.3	0.00	22856.9	25.55	29639.2	62.81
0.4	19318.3	8.33	18248.4	2.33	17832.3	0.00	23391.0	31.17	27584.0	54.69
0.5	20594.9	6.29	19375.7	0.00	19598.6	1.15	24707.6	27.52	29014.3	49.75
0.6	20655.9	17.02	17923.1	1.53	17652.2	0.00	24426.1	38.37	27609.9	56.41
0.7	20721.7	20.78	18269.8	6.49	17157.1	0.00	23803.5	38.74	27689.0	61.39
0.8	20068.1	22.37	19129.0	16.64	16399.7	0.00	23371.2	42.51	26896.4	64.01
0.9	20405.6	12.35	18162.2	0.00	20780.6	14.42	21725.7	19.62	28124.9	54.85
1.0	19948.1	15.96	18701.5	8.71	17203.1	0.00	24334.6	41.45	28532.0	65.85
	50-3									
0.1	7555.9	13.62	6810.7	2.41	6650.3	0.00	8626.2	29.71	10655.7	60.23
0.2	7655.4	13.11	7822.2	15.57	6768.3	0.00	8306.3	22.72	9701.1	43.33
0.3	7756.7	14.11	7422.7	9.20	6797.3	0.00	8430.8	24.03	11174.8	64.40
0.4	7888.0	27.44	6818.9	10.17	6189.5	0.00	7291.3	17.80	9450.2	52.68
0.5	8021.5	15.71	6932.6	0.00	7071.5	2.00	9186.9	32.52	9529.0	37.45
0.6	7826.4	19.90	7117.9	9.05	6527.2	0.00	7707.0	18.08	10197.6	56.23
0.7	7309.6	8.89	6986.0	4.07	6712.6	0.00	7910.2	17.84	10624.2	58.27
0.8	7578.9	7.55	7046.7	0.00	7325.1	3.95	8478.9	20.32	10281.9	45.91
0.9	7974.0	21.25	7017.5	6.70	6576.6	0.00	7833.9	19.12	9568.2	45.49
1.0	7204.7	5.53	6827.0	0.00	6933.3	1.56	9068.8	32.84	10787.4	58.01
	50-5									
0.1	17011.5	19.60	15200.0	6.86	14224.1	0.00	20242.7	42.31	21936.7	54.22
0.2	15241.5	2.81	14825.3	0.00	15462.5	4.30	21361.2	44.09	20901.2	40.98
0.3	15959.8	1.49	15725.5	0.00	16063.0	2.15	21862.5	39.03	22804.7	45.02
0.4	16790.9	13.97	14732.2	0.00	15872.8	7.74	20498.7	39.14	21537.0	46.19
0.5	15719.0	8.73	14457.0	0.00	15327.9	6.02	22554.5	56.01	23236.4	60.73
0.6	15941.2	14.22	14810.8	6.12	13956.7	0.00	22144.2	58.66	20698.5	48.31
0.7	17484.1	17.28	14907.4	0.00	16353.4	9.70	19592.2	31.43	21991.1	47.52
0.8	16311.8	13.73	15781.3	10.03	14342.8	0.00	21014.7	46.52	23523.7	64.01
0.9	18199.5	24.69	15653.0	7.25	14595.3	0.00	23087.1	58.18	23057.5	57.98
1.0	17588.7	29.20	13613.5	0.00	15900.3	16.80	22640.6	66.31	23003.1	68.97

control parameter of the due-date ranges and selected from $\{0.1, 0.2, \ldots, 1.0\}$. Regarding the feature of CPU clock frequency adjustment, we assumed three distinct CPU frequencies. The CPU frequencies on distinct machines were assigned from the uniform distribution $U(1,2)$. The energy consumption of each machine was given to the third power of the corresponding machine frequency. It is expected that using a higher execution speed incurs a higher power consumption per time unit.

All the proposed algorithms were coded in C and executed on a personal computer equipped with an Intel Core 2 Quad Q8200 2.33 GHz CPU and 2 GB of RAM. The commercial software ILOG CPLEX 7.0 was used to solve the integer linear programming model for small-size problems. The derived solutions are used for comparisons with the heuristic solutions. The experiment results are shown in Table 1. We set the time limit as 60 min in the CPLEX runs. Once the CPLEX solution-finding process reached the time limit, it aborted and output the incumbent best solution. The first column of the table contains different due-date parameters, R. We show the computational times and solutions obtained from CPLEX in the next two columns. The two heuristics, H_{EDD} and H_{WSPT}, took less than 0.01s to get the objective function values. Therefore, we only list the output objective values. We also show the percentage deviation from the solutions obtained by CPLEX and two heuristics in the row (dev.%). The percentage deviations (dev.%) are defined as

$$\frac{obj(H_{EDD}) - obj(CPLEX)}{obj(H_{EDD})} \times 100\%.$$

and

$$\frac{obj(H_{WSPT}) - obj(CPLEX)}{obj(H_{WSPT})} \times 100\%.$$

With regard to the required computing time, the deviation between CPLEX and two proposed heuristics varies in a great range. In the case of $(m,n) = (5,10)$, the deviation even grows up to more than 40% in some instances. For the cases with a higher deviation between the heuristic solutions and the CPLEX solutions, we examine the derived schedules and find that there is one machine totally vacant. This is due to the fact that all scalable processing speeds of that machine are relatively higher than those on other machines. If jobs are assigned to that machine with relatively higher processing speeds, the power cost of that machine is much larger than other machines. Thus, that machine would not be selected by the

Table 4
Results of experiment $E_2, (c_1, c_2) = (1,2)$.

R	$\omega = 0.2$		$\omega = 0.4$		$\omega = 0.6$		$\omega = 0.8$		$\omega = 1.0$	
	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)
	30-3									
0.1	7004.1	0.00	8292.3	18.39	10227.4	46.02	10925.4	55.99	11543.5	64.81
0.2	7344.8	0.00	8030.8	9.34	9399.6	27.98	9624.7	31.04	10565.7	43.85
0.3	6878.1	0.00	7964.1	15.79	9001.5	30.87	10312.6	49.93	12537.8	82.29
0.4	7393.3	0.00	8347.7	12.91	8689.0	17.53	10157.8	37.39	11313.5	53.02
0.5	7667.5	5.93	7238.5	0.00	9084.9	25.51	10901.8	50.61	11726.7	62.00
0.6	7411.5	0.00	8237.0	11.14	8572.0	15.66	11401.3	53.83	11353.7	53.19
0.7	7507.7	0.00	8708.0	15.99	8982.1	19.64	11097.0	47.81	12250.4	63.17
0.8	6940.9	0.00	7190.1	3.59	9391.4	35.31	10102.7	45.55	11349.2	63.51
0.9	6451.5	0.00	8098.3	25.53	9605.7	48.89	11053.9	71.34	11965.1	85.46
1.0	6554.6	0.00	8308.4	26.76	9341.4	42.52	11788.6	79.85	12755.7	94.61
	30-5									
0.1	15631.9	0.00	19519.1	24.87	23384.6	49.60	26942.7	72.36	28996.9	85.50
0.2	14489.1	0.00	20964.6	44.69	25485.9	75.90	27033.3	86.58	30910.3	113.33
0.3	15856.9	0.00	20082.0	26.65	24023.4	51.50	25578.4	61.31	28779.5	81.50
0.4	14450.6	0.00	19530.8	35.16	23556.5	63.01	25928.2	79.43	27525.2	90.48
0.5	15058.5	0.00	20405.0	35.50	26020.1	72.79	27937.3	85.53	28226.4	87.44
0.6	15638.4	0.00	19226.7	22.95	25513.1	63.14	26869.5	71.82	27360.0	74.95
0.7	14821.6	0.00	19803.0	33.61	25679.2	73.26	27168.1	83.30	27453.0	85.22
0.8	13840.1	0.00	21160.5	52.89	25424.3	83.70	27358.1	97.67	28878.3	108.66
0.9	14448.9	0.00	20326.2	40.68	27525.3	90.50	25796.1	78.53	27316.5	89.06
1.0	13744.5	0.00	19803.7	44.08	25054.6	82.29	28484.6	107.24	28881.6	110.13
	50-3									
0.1	6690.4	0.00	7200.2	7.62	8737.5	30.60	9578.4	43.17	9757.2	45.84
0.2	6387.2	0.00	8507.7	33.20	8577.7	34.30	9183.7	43.78	10457.8	63.73
0.3	6427.6	0.00	7809.4	21.50	8820.4	37.23	9914.2	54.24	10804.4	68.09
0.4	6599.7	0.00	7240.7	9.71	8193.9	24.16	8863.9	34.31	9675.1	46.60
0.5	6689.2	0.00	7153.4	6.94	9473.5	41.62	9486.1	41.81	9504.0	42.08
0.6	6416.2	0.00	7898.9	23.11	8534.9	33.02	9563.8	49.06	9805.0	52.82
0.7	6205.4	0.00	7249.6	16.83	8793.1	41.70	9598.8	54.68	10807.6	74.16
0.8	6341.8	0.00	7464.0	17.70	9532.1	50.31	9960.6	57.06	10063.2	58.68
0.9	6466.0	0.00	7443.1	15.11	8773.5	35.69	9533.1	47.43	10263.9	58.74
1.0	6133.2	0.00	7392.1	20.53	9411.1	53.45	10302.8	67.98	10999.1	79.34
	50-5									
0.1	12909.4	0.00	17090.5	32.39	20412.5	58.12	20098.9	55.69	22119.9	71.35
0.2	11858.5	0.00	17010.7	43.45	21849.0	84.25	20806.2	75.45	20585.5	73.59
0.3	12102.3	0.00	18053.2	49.17	21703.2	79.33	22062.0	82.30	22342.5	84.61
0.4	12201.6	0.00	17088.0	40.05	20431.0	67.45	21992.1	80.24	22200.1	81.94
0.5	12215.9	0.00	16781.2	37.37	20972.5	71.68	21730.3	77.89	22606.8	85.06
0.6	11976.6	0.00	16974.4	41.73	20446.1	70.72	23537.7	96.53	20274.2	69.28
0.7	14023.5	0.00	17116.6	22.06	21060.3	50.18	21405.2	52.64	20786.2	48.22
0.8	12676.9	0.00	17965.1	41.72	20511.4	61.80	21489.8	69.52	24135.3	90.39
0.9	12742.9	0.00	18299.5	43.61	20205.1	58.56	21917.0	71.99	21499.3	68.72
1.0	12817.4	0.00	15848.9	23.65	20933.3	63.32	23612.7	84.22	23207.1	81.06

heuristics. However, the proposed heuristics, which is based upon list scheduling, would anyhow allocate at least one job on each machine. Thus, encountering such special instance patterns, the proposed heuristic will deteriorate due to the processing with higher power consumption costs. The hardness of the studied problem lies in job scheduling, job dispatching issues as well as in speed selection. The EDD rule and the WSPT rule reflect the intuitive ideas behind the scheduling and dispatching parts. Since the speed selection issue is addressed in the heuristic without retrospective corrections, the performances could be then less favorable. This inspires the necessity in the development of meta-heuristics that address all decision issues and provide iterative correction mechanisms.

For the large-size data, we only compared the proposed heuristics and the PSO algorithm because CPLEX failed to report a decent solution before the time limit is elapsed. This experiment consists of two parts. The first part is carried out by considering the relative values of the acceleration coefficients used in PSO. Three settings are of interest, namely, $(c_1, c_2) = (1, 1), (c_1, c_2) = (1, 2)$, and $(c_1, c_2) = (2, 1)$. The three settings address the relative importance of the two parameters. In each of the three settings, another PSO coefficient ω varies within $\{0.1, 0.2, \ldots, 1.0\}$ for all tests. The second part is designed to compare the solutions provided by the proposed PSO algorithm and the two heuristics.

The settings of the computational experiments are summarized in Table 2. Settings E_1, E_2 and E_3 are designed to examine the performances of the proposed PSO under the three combinations of the values of (c_1, c_2). For each of the setting, instances were generated with different inertia weights, different numbers of machines and different numbers of jobs. In the setting E_4, we compare the best solution yielded in among E_1, E_2 and E_3 with the solutions produced by the two heuristics. For further contrast, we conducted setting E_5 to compare the worst solutions yielded in among E_1, E_2 and E_3 with the solutions produced by the two heuristics.

Since a certain kind of randomness exists in the execution of the PSO algorithm, in each scenario of the experiment we invoked the PSO Algorithm 10 times and computed the average values.

Tables 3–5 summarize the computational results. The tables are headlined with the number of jobs and the number of machines. For example, "**30-3**" indicates that $n = 30$ jobs were to be dispatched to $m = 3$ machines. The number in each cell was obtained through 10 repetitions of the PSO algorithm. The effectiveness of the PSO algorithm is measured by the average objective function values reported. Each row corresponds to a specific value of the due-date range control parameter, *R*. The performance of the PSO algorithm under a specific inertia ω value is indicated by two values, namely, the average objective function values (obj.) and the

Table 5
Results of experiment $E_3, (c_1, c_2) = (2, 1)$.

R	$\omega = 0.2$		$\omega = 0.4$		$\omega = 0.6$		$\omega = 0.8$		$\omega = 1.0$	
	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)
	30-3									
0.1	6944.8	0.00	8709.4	25.41	9984.5	43.77	10402.1	49.78	11691.4	68.35
0.2	7108.9	0.00	7853.6	10.48	9737.0	36.97	9760.8	37.30	10896.9	53.29
0.3	6500.5	0.00	8287.5	27.49	9046.6	39.17	10248.0	57.65	12068.8	85.66
0.4	7127.1	0.00	9075.5	27.34	8627.4	21.05	10246.0	43.76	11741.1	64.74
0.5	7300.0	0.00	7602.3	4.14	9027.4	23.66	10722.4	46.88	11864.6	62.53
0.6	6856.8	0.00	8723.8	27.23	8802.9	28.38	10901.3	58.99	11466.3	67.23
0.7	7454.4	0.00	9084.7	21.87	9081.8	21.83	11310.0	51.72	11530.0	54.67
0.8	6773.0	0.00	8152.6	20.37	9251.9	36.60	10260.7	51.49	11481.0	69.51
0.9	6285.8	0.00	8241.5	31.11	9578.0	52.38	11194.7	78.10	12039.4	91.53
1.0	6415.2	0.00	8440.8	31.58	9601.7	49.67	11708.2	82.51	12431.0	93.77
	30-5									
0.1	15285.3	0.00	20242.0	32.43	22697.3	48.49	25301.6	65.53	28799.5	88.41
0.2	14189.8	0.00	20958.2	47.70	24526.9	72.85	27800.2	95.92	30344.8	113.85
0.3	14892.6	0.00	21962.9	47.48	24431.5	64.05	25342.4	70.17	30147.6	102.43
0.4	14788.2	0.00	19996.9	35.22	22495.5	52.12	25410.3	71.83	28171.3	90.50
0.5	14537.0	0.00	20785.7	42.98	25759.3	77.20	28312.1	94.76	29065.0	99.94
0.6	14654.6	0.00	19832.1	35.33	24195.9	65.11	26564.4	81.27	28412.8	93.88
0.7	14585.6	0.00	19123.8	31.11	24409.8	67.36	26309.6	80.38	27080.5	85.67
0.8	13251.8	0.00	21694.6	63.71	25629.5	93.40	26922.8	103.16	27723.1	109.20
0.9	14816.4	0.00	19963.3	34.74	27867.8	88.09	25025.5	68.90	28547.5	92.68
1.0	13620.7	0.00	19826.2	45.56	23538.0	72.81	27699.4	103.36	27597.0	102.61
	50-3									
0.1	6744.7	0.00	7268.5	7.77	8462.6	25.47	9552.8	41.63	10091.9	49.63
0.2	6384.0	0.00	8718.3	36.56	8547.4	33.89	9126.5	42.96	9957.6	55.98
0.3	6428.4	0.00	8028.8	24.90	8591.2	33.64	9592.9	49.23	10847.6	68.74
0.4	6425.0	0.00	7425.1	15.57	8125.1	26.46	8395.7	30.67	9001.5	40.10
0.5	6617.3	0.00	7421.1	12.15	9238.1	39.61	9647.7	45.80	9671.5	46.15
0.6	6553.7	0.00	7851.6	19.80	8202.1	25.15	9010.1	37.48	10232.0	56.13
0.7	5977.5	0.00	7376.5	23.40	8376.6	40.14	9448.2	58.06	11014.5	84.27
0.8	6400.3	0.00	7559.9	18.12	9270.7	44.85	9870.7	54.22	9988.8	56.07
0.9	6529.6	0.00	7313.7	12.01	8683.1	32.98	9401.2	43.98	9416.8	44.22
1.0	5736.9	0.00	7196.6	25.44	9193.6	60.25	9986.1	74.07	10565.8	84.17
	50-5									
0.1	12988.5	0.00	16884.1	29.99	19685.3	51.56	20124.6	54.94	21914.0	68.72
0.2	12449.2	0.00	16390.5	31.66	21152.1	69.91	21149.4	69.89	21785.2	74.99
0.3	12044.0	0.00	18162.3	50.80	21117.5	75.34	20646.5	71.43	20846.6	73.09
0.4	12314.6	0.00	16696.2	35.58	19767.7	60.52	20158.5	63.70	22120.2	79.63
0.5	12833.9	0.00	17019.1	32.61	20917.3	62.98	22050.7	71.82	23112.4	80.09
0.6	11383.3	0.00	17036.0	49.66	20099.1	76.57	22960.8	101.71	20466.7	79.80
0.7	13439.8	0.00	17052.6	26.88	21425.4	59.42	21653.8	61.12	20896.2	55.48
0.8	12306.6	0.00	17830.2	44.88	19534.2	58.73	21405.2	73.93	22155.1	80.03
0.9	13089.4	0.00	17928.2	36.97	19512.7	49.07	23614.3	80.41	22567.5	72.41
1.0	12612.8	0.00	16006.3	26.91	20281.5	60.80	21972.3	74.21	23446.8	85.90

percentage deviation from the best objective function values in this row (dev.%). The percentage deviation (dev.%) of the PSO algorithm with the inertia weight $\omega' \in \{0.2, 0.3, \ldots, 1.0\}$ is defined as

$$\frac{obj(PSO_{\omega'}) - \min_{\omega \in \{0.2,\ldots,1.0\}}\{obj(PSO_{\omega})\}}{obj(PSO_{\omega'})} \times 100\%.$$

The execution of the PSO algorithm was terminated once either 1,000 iterations were reached or the specified convergence status was encountered.

The results of the setting $(c_1, c_2) = (1, 1)$ are summarized in Table 3. Through each combination of due date variable R, the number of jobs and the number of machines, we investigate the influence caused by the different values of the PSO coefficient ω. In terms of the average objective function values, we can find that in most of the runs, the best solutions are obtained at $\omega = 0.6$. In two runs of **30-5**, three runs of **50-3** and six runs of **50-5**, the PSO algorithm with $\omega = 0.4$ delivered the minimum objective function values. This suggests that setting the inertia weight ω around 0.5 can better exhibit the strength of the proposed PSO algorithm. On the other hand, for most of the cases, the inferior solutions were encountered when the inertia weight ω is 1.0. Moreover, in one run of **30-3** and three runs of **50-5**, the setting $\omega = 0.8$ leads to the maximum objective function values. Larger inertia weight indicates larger the adoption of previous velocity when determining the new velocity. This highlights the phenomenon that each particle moves independently by it own velocity, resulting in a type of straight-line routes. Diversity required for probing different areas is diminished and thus the solution quality is unfavorable.

Table 4 reports the results of the experiment framework E_2. The PSO coefficients $(c_1, c_2) = (1, 2)$ reflect that the global best solutions have double impact on the new velocity of each particle than the local optimal solution of the particle itself. From the numerical values, we can easily see that the setting $\omega = 0.2$ leads to favorable solutions with the minimum objective function values, yet the setting $\omega = 1.0$ again results in the worst performance in most of the tests. Especially, for the cases of **30-3**, **30-5** and **50-3**, a clear trend shows that the average objective function values increase as ω values become larger.

The results of setting E_3 are summarized in Table 5. The PSO coefficients are $(c_1, c_2) = (2, 1)$ to reenforce the impact of local optimal solutions of the particles. For all test cases, the best solutions were reported by the setting with $\omega = 0.2$. In most of the cases, the setting $\omega = 1.0$ still leads to the worst solutions. In the runs of **50-5** with $R = 0.6$, the results of $\omega = 0.8$ and $\omega = 1.0$ are rather inferior.

The second part of the computational study is to investigate the relative performances of the PSO algorithm and the two heuristics. In Table 6, each row shows the objective function values derived by the two heuristics and the best results delivered by the PSO algorithm under three different settings of c_1 and c_2. The numerical results clearly show the superiority of heuristic H_{EDD} over heuristic H_{WSPT}. The PSO algorithm under different settings can produce schedules whose objective function values are much smaller than those produced by the heuristics. One of the most major reasons behind the results could be the following observation: In the heuristics, once a job is assigned to a machine at a favorable processing speed, no adjustment is allowed during the remaining execution course of the heuristics. For the jobs that are prone to be early, they will be processed at the lowest possible speeds so as to save the power cost. This policy on the other hand may induce inevitable tardiness to the unscheduled jobs.

In Table 7, we show a comparison among two heuristics and the worst results delivered by the PSO algorithm under three different settings of c_1 and c_2. In the test of instances with 30 jobs and 50 jobs, the PSO algorithm even performed at least 11.27% and 43.37% better than two heuristics. Also the parameter setting $(c_1, c_2) = (2, 1)$ indices a higher probability of producing inferior solutions in the 30-job instances. However, for the 50-job problem instances, the parameter settings, $(c_1, c_2) = (1, 1)$ and $(c_1, c_2) = (1, 2)$ are prone to be unfavorable.

For each job-machine combination as displayed by columns of Table 8, we list the best objective function values and the parameter settings of (c_1, c_2, ω) corresponding to these results. The experiment setting with $\omega = 0.2$, which performs better than others in most of experiments, is suggested to applied as the number of machines and jobs increase. Due to the fact that the random value of the coefficients of parameters c_1 and c_2 may affect the process of evolutionary solution significantly. The setting of parameters c_1 and c_2 are hard to justify which one is better than another. However, the experiment results suggest that using different values of c_1 and c_2 can obtain better solutions than using the same values.

Only under the case of 30 jobs with 3 machines, the PSO algorithm achieves convergence with less than 1,000 iterations. For the rest of job-machine combinations, the objective function values are the incumbent best solutions while achieving 1000

Table 6
Comparison of the best solutions of PSO with the solutions of two heuristics.

(c_1, c_2)	(1,1)	(1,2)	(2,1)	H_{EDD}		H_{WSPT}	
R	min	min	min	Obj.	Dev. (%)	Obj.	Dev. (%)
	30-3						
0.1	7964.7	7004.1	6944.8	16085.6	131.62	24478.6	252.47
0.2	7486.1	7344.8	7108.9	14739.4	107.34	22374.9	214.74
0.3	7402.4	6878.1	6500.5	14045.1	116.06	23886.4	267.45
0.4	7014.3	7393.3	7127.1	13922.5	98.49	23505.2	235.10
0.5	7286.2	7238.5	7300.0	13937.5	92.55	24019.1	231.82
0.6	6685.3	7411.5	6856.8	14479.6	116.59	24459.2	265.87
0.7	6920.3	7507.7	7454.4	15684.6	126.65	24715.5	257.14
0.8	7642.2	6940.9	6773.0	14750.3	117.78	23147.9	241.77
0.9	7289.2	6451.5	6285.8	14942.8	137.72	25299.5	302.49
1.0	7117.4	6554.6	6415.2	15122.2	135.72	24785.1	286.35
	30-5						
0.1	7964.7	7004.1	6944.8	36350.9	423.43	59237.7	752.98
0.2	7486.1	7344.8	7108.9	34392.9	383.80	59390.6	735.44
0.3	7402.4	6878.1	6500.5	39033.6	500.47	62421.5	860.26
0.4	7014.3	7393.3	7127.1	33507.6	377.70	59850.1	753.26
0.5	7286.2	7238.5	7300.0	34395.3	375.17	59654.2	724.12
0.6	6685.3	7411.5	6856.8	36756.6	449.81	59566.6	791.01
0.7	6920.3	7507.7	7454.4	36260.4	423.97	63760.2	821.35
0.8	7642.2	6940.9	6773.0	35852.2	429.34	57949.2	755.59
0.9	7289.2	6451.5	6285.8	38218.3	508.01	57064.7	807.84
1.0	7117.4	6554.6	6415.2	34093.6	431.45	61600.0	860.22
	50-3						
0.1	7964.7	7004.1	6944.8	17224.2	148.02	25808.6	271.62
0.2	7486.1	7344.8	7108.9	19863.2	179.41	29136.8	309.86
0.3	7402.4	6878.1	6500.5	17520.9	169.53	28814.5	343.27
0.4	7014.3	7393.3	7127.1	16830.5	139.95	27462.5	291.52
0.5	7286.2	7238.5	7300.0	18825.5	160.07	26371.7	264.33
0.6	6685.3	7411.5	6856.8	18963.6	183.66	28098.1	320.30
0.7	6920.3	7507.7	7454.4	15791.2	128.19	26303.3	280.09
0.8	7642.2	6940.9	6773.0	18136.4	167.77	27139.1	300.70
0.9	7289.2	6451.5	6285.8	17398.3	176.79	27118.7	331.43
1.0	7117.4	6554.6	6415.2	16978.0	164.65	27588.2	330.04
	50-5						
0.1	7964.7	7004.1	6944.8	51831.3	646.33	83221.2	1098.32
0.2	7486.1	7344.8	7108.9	52488.1	638.34	77014.6	983.35
0.3	7402.4	6878.1	6500.5	48300.9	643.03	73473.5	1030.27
0.4	7014.3	7393.3	7127.1	50368.2	618.08	80980.7	1054.51
0.5	7286.2	7238.5	7300.0	45309.8	525.96	73926.1	921.29
0.6	6685.3	7411.5	6856.8	49253.7	636.75	78208.4	1069.86
0.7	6920.3	7507.7	7454.4	50374.8	627.93	80297.6	1060.32
0.8	7642.2	6940.9	6773.0	50327.1	643.05	80561.8	1089.46
0.9	7289.2	6451.5	6285.8	51171.0	714.07	79596.4	1166.29
1.0	7117.4	6554.6	6415.2	50185.4	682.29	81320.5	1167.62

Table 7
Comparison of the worst solutions of PSO with the solutions of two heuristics.

(c_1, c_2)	(1,1)	(1,2)	(2,1)	H_{EDD}		H_{WSPT}	
R	max	max	max	Obj.	Dev. (%)	Obj.	Dev. (%)
	30-3						
0.1	10432.2	11543.5	11691.4	16085.6	37.58	24478.6	109.37
0.2	9927.5	10565.7	10896.9	14739.4	35.26	22374.9	105.33
0.3	11394.7	12537.8	12068.8	14045.1	12.02	23886.4	90.52
0.4	10416.5	11313.5	11741.1	13922.5	18.58	23505.2	100.20
0.5	10605.9	11726.7	11864.6	13937.5	17.47	24019.1	102.44
0.6	10287.3	11401.3	11466.3	14479.6	26.28	24459.2	113.31
0.7	10535.1	12250.4	11530.0	15684.6	28.03	24715.5	101.75
0.8	10367.4	11349.2	11481.0	14750.3	28.48	23147.9	101.62
0.9	10487.6	11965.1	12039.4	14942.8	24.12	25299.5	110.14
1.0	11140.0	12755.7	12431.0	15122.2	18.55	24785.1	94.31
	30-5						
0.1	27369.0	28996.9	28799.5	36350.9	25.36	59237.7	104.29
0.2	29842.0	30910.3	30344.8	34392.9	11.27	59390.6	92.14
0.3	29639.2	28779.5	30147.6	39033.6	29.47	62421.5	107.05
0.4	27584.0	27525.2	28171.3	33507.6	18.94	59850.1	112.45
0.5	29014.3	28226.4	29065.0	34395.3	18.34	59654.2	105.24
0.6	27609.9	27360.0	28412.8	36756.6	29.37	59566.6	109.65
0.7	27689.0	27453.0	27080.5	36260.4	30.96	63760.2	130.27
0.8	26896.4	28878.3	27723.1	35852.2	24.15	57949.2	100.67
0.9	28124.9	27525.3	28547.5	38218.3	33.88	57064.7	99.89
1.0	28532.0	28881.6	27699.4	34093.6	18.05	61600.0	113.28
	50-3						
0.1	10655.7	9757.2	10091.9	17224.2	61.64	25808.6	142.20
0.2	9701.1	10457.8	9957.6	19863.2	89.94	29136.8	178.61
0.3	11174.8	10804.4	10847.6	17520.9	56.79	28814.5	157.85
0.4	9450.2	9675.1	9001.5	16830.5	73.96	27462.5	183.85
0.5	9529.0	9504.0	9671.5	18825.5	94.65	26371.7	172.67
0.6	10197.6	9805.0	10232.0	18963.6	85.34	28098.1	174.61
0.7	10624.2	10807.6	11014.5	15791.2	43.37	26303.3	138.81
0.8	10281.9	10063.2	9988.8	18136.4	76.39	27139.1	163.95
0.9	9568.2	10263.9	9416.8	17398.3	69.51	27118.7	164.21
1.0	10787.4	10999.1	10565.8	16978.0	54.36	27588.2	150.82
	50-5						
0.1	21936.7	22119.9	21914.0	51831.3	134.32	83221.2	276.23
0.2	21361.2	21849.0	21785.2	52488.1	140.23	77014.6	252.49
0.3	22804.7	22342.5	21117.5	48300.9	111.80	73473.5	222.19
0.4	21537.0	22200.1	22120.2	50368.2	126.88	80980.7	264.78
0.5	23236.4	22606.8	23112.4	45309.8	94.99	73926.1	218.15
0.6	22144.2	23537.7	22960.8	49253.7	109.25	78208.4	232.27
0.7	21991.1	21405.2	21653.8	50374.8	129.07	80297.6	265.14
0.8	23523.7	24135.3	22155.1	50327.1	108.52	80561.8	233.79
0.9	23087.1	21917.0	23614.3	51171.0	116.69	79596.4	237.07
1.0	23003.1	23612.7	23446.8	50185.4	112.54	81320.5	244.39

Table 8
Best parameter settings of PSO.

R	30-3		30-5		50-3		50-3	
	Obj.	(c_1, c_2, ω)	Obj.	(c_1, c_2, ω)	Obj.	(c_1, c_2, ω)	Obj.	(c_1, c_2, ω)
0.1	6944.8	(2,1,0.2)	15285.3	(2,1,0.2)	6650.3	(1,1,0.6)	12909.4	(1,2,0.2)
0.2	7108.9	(2,1,0.2)	14189.8	(2,1,0.2)	6384.0	(2,1,0.2)	11858.5	(1,2,0.2)
0.3	6500.5	(2,1,0.2)	14892.6	(2,1,0.2)	6427.6	(1,2,0.2)	12044.0	(2,1,0.2)
0.4	7014.3	(1,1,0.6)	14450.6	(1,2,0.2)	6189.5	(1,1,0.6)	12201.6	(1,2,0.2)
0.5	7238.5	(1,2,0.4)	14537.0	(2,1,0.2)	6617.3	(2,1,0.2)	12215.9	(1,2,0.2)
0.6	6685.3	(1,1,0.6)	14654.6	(2,1,0.2)	6416.2	(1,2,0.2)	11383.3	(2,1,0.2)
0.7	6920.3	(1,1,0.6)	14585.6	(2,1,0.2)	5977.5	(2,1,0.2)	13439.8	(2,1,0.2)
0.8	6773.0	(2,1,0.2)	13251.8	(2,1,0.2)	6341.8	(1,2,0.2)	12306.6	(2,1,0.2)
0.9	6285.8	(2,1,0.2)	14448.9	(1,2,0.2)	6466.0	(1,2,0.2)	12742.9	(1,2,0.2)
1.0	6415.2	(2,1,0.2)	13620.7	(2,1,0.2)	5736.9	(2,1,0.2)	12612.8	(2,1,0.2)

iterations. The PSO algorithm produced better solutions when $\omega = 0.2$ (34 of the 40 cases as shown Table 7). Moreover, for the cases of 30 jobs on 5 machines, 50 jobs on 3 machines and 50 jobs on 5 machines, the PSO parameters (c_1, c_2, ω) are either $(1,2,0.2)$ or $(2,1,0.2)$, which implies that distinct values of c_1 and c_2 achieves better solutions than the setting with identical values of c_1 and c_2.

Here we summarize the experiment results. The comparison results between CPLEX and the two heuristics on small-size

instances show that the performance of the heuristics is highly related to the machine speeds of the test instances. CPLEX needs more than 60 min to solve even the small-size problem instances. The experiment has already demonstrated the proposed PSO algorithm could obtain quality results but took more computing times, when compared to the heuristics. The computing time required by PSO algorithms could be acceptable if the benefits of less tardiness penalty and lower power cost can be realized. Another important point to highlight is that cloud computing systems usually have more than thousands of computers in the real world. Taking the effectiveness and the efficiency of scheduling simultaneously in the context of cloud computing, the proposed PSO algorithm can be considered as a viable solution approach.

7. Conclusions

This paper investigated a scheduling problem with heterogeneous unrelated parallel machines to minimize the weighted sum of tardiness penalty and power consumption cost. The power consumption cost varies due to the flexibility of adjustment of CPU frequencies (dynamic voltage scaling) for processing the jobs. Jobs executed at a higher machine speed for time saving incurs more energy cost. We formulated an integer linear programming model. Since the studied scheduling problem is NP-hard in the strong sense, we designed two heuristics and a PSO algorithm to produce approximate solutions. A major contribution of the PSO design is the encoding scheme proposed to represent a schedule as a particle. We carried out a computational study to assess the performances of the proposed algorithms and using CPLEX to solve the studied problem. The two heuristics and the PSO algorithm under distinct settings of PSO coefficients were tested for larger-size data of the problems. Statistics showed that the PSO algorithm can provide quality solutions in an acceptable time. The proposed encoding scheme of the PSO algorithm can be easily applied to other processing contexts with adjustable machine speeds.

For further research, it could be interesting to extend the proposed approach to dealing with other performance criteria. Further attempts to use Pareto front based method to evaluate the fitness. Some interesting metrics as hyper-volume are very suitable for multi-objective optimization based on Pareto front (Zitzler & Thiele, 1998). We can also introduce the concept of CPU frequency adjustment to the traditional machine scheduling problems by allowing speed adjustment during the execution of each individual job. Moreover, applying the proposed encoding method to more complex machine environments might be an interesting and challenging direction. Machine eligibility is an example in which not all but only specific machines are eligible for processing each specific job. This may introduce a type of complexity in the manipulation of particles. It is also worthwhile to design other versions of PSO algorithms to continue pursuing the best performance in solving flow-shop scheduling problems. Hybrid PSO algorithms with other metaheuristics (Xia & Wu, 2005; Zhang, Shao, Li, & Gao, 2009) or integrating PSO algorithms with discrete Lagrange multipliers (Nezhad & Mahlooji, 2011) or designing new PSO features, e.g. dynamically adjusting acceleration coefficients as the course of iterations continues, are other possible research directions.

Acknowledgements

This research was partially supported by the National Science Council of Taiwan under Grant NSC-100-2410-H-009-MY2. The authors are also grateful to the reviewers for their valuable comments.

References

Ai, J., & Kachitvichyanukul, V. (2009). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering, 56*(1), 380–387.

Akgul, Y., Puschini, D., Lesecq, S., Miro-Panades, I., Benoit, P., & Torres L. et al. (2012). Power mode selection in embedded systems with performance constraints. In *Proceedings of the faible tension faible consommation conference*. IEEE, June 6–8 (pp. 1–4).

Anghinofi, D., & Paolucci, M. (2007). Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers & Operations Research, 34*(11), 3471–3490.

Baker, KR. (1974). *Introduction to sequencing and scheduling*. New York: Wiley.

Bunde, DP. (2006). Power-aware scheduling for makespan and flow. *Journal of Scheduling, 12*(5), 489–500.

Du, J., & Leung, JY. (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research, 15*(3), 483–495.

Dye, CY., & Ouyang, LY. (2011). A particle swarm optimization for solving joint pricing and lot-sizing problem with fluctuating demand and trade credit financing. *Computers & Industrial Engineering, 60*(1), 127–137.

Garey, MR., & Johnson, DS. (1978). "Strong" NP-completeness results: Motivation, examples, and implications. *Journal of the Association for Computing Machinery, 25*(3), 499–508.

Ge, R., Feng, X., & Cameron, K. W. (2005). Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In *Proceedings of the ACM/IEEE conference on supercomputing*, November 12–18, Seattle, Washington (pp. 34–44).

Graham, RL. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal, 45*, 1563–1581.

Hall, NG., & Posner, ME. (2000). Generating experimental data for computational testing with machine scheduling applications. *Operations Research, 49*(6), 854–865.

Jackson, J. R. (1955). *Scheduling a production line to minimize maximum tardiness*. Research report 43. Management Science Research Project, UCLA.

Kashan, A., & Karimi, B. (2009). A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers & Industrial Engineering, 56*(1), 216–223.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (pp. 1942–1948).

Koomey, J. G. (2007). *Estimating total power consumption by servers in the US and the world*. <http://enterprise.amd.com/Downloads/svrpwrusecompletefinal.pdf>.

Kumar, P. R., & Palani, S. (2012). A dynamic voltage scaling with single power supply and varying speed factor for multiprocessor system using genetic algorithm. In *Proceedings of the international conference on pattern recognition, informatics and medical engineering*, March 21–23 (pp. 342–346).

Lawler, E. L. (1977). A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics, 1*, 331–342.

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics, 1*, 343–362.

Nezhad, A. M., & Mahlooji, H. (2011). A revised particle swarm optimization based discrete Lagrange multipliers method for nonlinear programming problems. *Computers & Operations Research, 38*(8), 1164–1174.

Niu, Q., Zhou, T., & Wang, L. (2010). A hybrid particle swarm optimization for parallel machine total tardiness scheduling. *International Journal of Advanced Manufacturing Technology, 49*, 723–739.

Önüt, S., Tuzkaya, R. U., & Doğaç, B. (2008). A particle swarm optimization algorithm for the multiple level warehouse layout design problem. *Computers & Industrial Engineering, 54*(4), 783–799.

Rizvandi, N. B., Zomaya, A. Y., Lee, Y. C., Boloori, A. J., & Taheri, J. (2012). *Multiple frequency selection in DVFS-enabled processors to minimize energy consumption*. arXiv:1203.5160v1, ed.

Runwei, C., Mitsuo, G., & Tatsumi, T. (1995). Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. *Computers & Industrial Engineering, 29*(1), 513–517.

Smith, WE. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly, 3*, 59–66.

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering, 48*(2), 409–425.

Zhang, GH., Shao, XY., Li, PG., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering, 56*(4), 1309V1318.

Zhu, D., Melhem, R., & Childers, BR. (2003). Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. *IEEE Transaction on Parallel and Distributed Systems, 14*(7), 686–700.

Zhu, D., Mosse, D., & Melhem, R. (2004). Power-aware scheduling for AND/OR graphs in real-time systems. *IEEE Transactions on Parallel and Distributed Systems, 15*(9), 849–864.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms – A comparative case study. In *Proceedings of the 5th conference on parallel problem solving from nature (PPSN5). Lecture notes in computer science* (Vol. 1498, pp. 292–301). Springer Verlag.