Springer

Dear Author,

Here are the proofs of your article.

- You can submit your corrections **online**, via **e-mail** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- You can also insert your corrections in the proof PDF and **email** the annotated PDF.
- For fax submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Remember to note the **journal title**, **article number**, and **your name** when sending your response via e-mail or fax.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/ corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style.
  Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.
- Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible.**
- The **printed version** will follow in a forthcoming issue.

Please note

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: http://dx.doi.org/[DOI].
If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: http://www.springerlink.com.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us if you would like to have these documents returned.

Metadata of the article that will be visualized in OnlineFirst

ArticleTitle	Using heuristic algorithms to solve the scheduling problems with job-dependent and machine-dependent learning effects	
Article Sub-Title		
Article CopyRight		
Journal Name	Journal of Intelligent Manufacturing	
Corresponding Author	Family Name	**Wu**
	Particle	
	Given Name	**Hsien-Chung**
	Suffix	
	Division	Department of Mathematics
	Organization	National Kaohsiung Normal University
	Address	Kaohsiung , 802, Taiwan
	Email	hcwu@nknucc.nknu.edu.tw
Author	Family Name	**Lai**
	Particle	
	Given Name	**Peng-Jen**
	Suffix	
	Division	Department of Mathematics
	Organization	National Kaohsiung Normal University
	Address	Kaohsiung , 802, Taiwan
	Email	
	Received	13 February 2013
Schedule	Revised	
	Accepted	8 August 2013
Abstract	The multi-machine scheduling problems with job-dependent and machine-dependent learning effects are proposed in this paper. Since it is almost impossible to obtain the analytic results for this complicated multi-machine scheduling problems with learning effects, four heuristic algorithms are used to solve this newly proposed model, where the variants of well-known genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO) and particle swarm optimization (PSO) are coded in the commercial software MATLAB. The objective is to minimize the makespan of this new model. For this kind of scheduling problem, the numerical experiments show that the GA and SA outperform ACO and PSO.	
Keywords (separated by '-')	Scheduling problems - Genetic algorithm - Simulated annealing - Ant colony optimization - Particle swarm optimization - Learning effects	
Footnote Information		

Journal: 10845
Article: 827

Author Query Form

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your article, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query	Details required	Author's response
1.	Kindly check and confirm the edit in the paragraph 'This paper is organized as follows...' in page 2 of the manuscript.	
2.	Please provide a definition for the significance of bold in the inline tables.	
3.	References citation 'Lee et al., Kennedy and Eberhart, Spears and DeJong' have been changed to 'Lee and Wu, Eberhard and Kennedy, Speras and DeJong'. Please check and change accordingly.	

J Intell Manuf
DOI 10.1007/s10845-013-0827-x

Using heuristic algorithms to solve the scheduling problems with job-dependent and machine-dependent learning effects

Peng-Jen Lai · Hsien-Chung Wu

Received: 13 February 2013 / Accepted: 8 August 2013

Abstract The multi-machine scheduling problems with job-dependent and machine-dependent learning effects are proposed in this paper. Since it is almost impossible to obtain the analytic results for this complicated multi-machine scheduling problems with learning effects, four heuristic algorithms are used to solve this newly proposed model, where the variants of well-known genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO) and particle swarm optimization (PSO) are coded in the commercial software MATLAB. The objective is to minimize the makespan of this new model. For this kind of scheduling problem, the numerical experiments show that the GA and SA outperform ACO and PSO.

Keywords Scheduling problems · Genetic algorithm · Simulated annealing · Ant colony optimization · Particle swarm optimization · Learning effects

Introduction

The learning effects in scheduling problems have been widely studied recently. The main reasons come from the fact that the same kind of jobs will be repeatedly processed and the employees or workers can improve their skills after doing the same task for a long time.

To the best of our knowledge, the scheduling problems with learning effects coming from machines was seemingly not proposed in the literature. In practical situation, the different machines might own the different learning rates. In this paper, we consider the n-job and m-machine flow shop scheduling problems. The learning factors come from jobs and machines will be included simultaneously in the scheduling problem. Therefore, we can consider three kinds of scheduling problems with learning effects. Firstly, we may assume that only the job-dependent learning factor is taken into account in this problem; that is, the learning factor comes from machines will be ignored. This problem was considered by Moshiev and Sidney (2003). Secondly, suppose that only the machine-dependent learning factor is taken into account in this problem; that is, the learning factor comes from jobs will be ignored. Thirdly, in the general case, we shall consider the job-dependent and machine-dependent learning factors simultaneously. This kind of problem is really complicated such that it is almost impossible to obtain the analytic results. In this paper, we apply four heuristic algorithms that are genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO) and particle swarm optimization (PSO) to minimize the makespan of this problem.

This paper is organized as follows. In second section, we provide the brief review for the scheduling problems with learning effects. In third section, we introduce the new models that simultaneously consider the job-dependent and machine-dependent learning effects. In fourth section, we introduce four heuristic algorithms that will be used to solve the scheduling problems with job-dependent and machine-dependent learning effects. In fifth section, we provide the numerical experiments in order to minimize the makespan of this newly proposed model.

Review for scheduling problems with learning effects

We briefly review the frequently adopted scheduling problems with learning effects in the literature. Of course, the analytic results can be obtained for the single-machine prob-

P.-J. Lai · H.-C. Wu (✉)
Department of Mathematics, National Kaohsiung Normal University, Kaohsiung 802, Taiwan
e-mail: hcwu@nknucc.nknu.edu.tw

lems. The multi-machine scheduling problems with learning effects were seldom studied in the literature due to its complication, where the machine-dependent learning effects was also not considered. In other words, considering the machine-dependent learning effects will increase the complication of this kind of problem. Therefore, we can use the heuristic algorithms to solve this kind of new problem.

Single-machine scheduling problems

Suppose that there are n jobs available at time zero. We denote by p_i the normal processing time of job i. Because of the learning effects, the actual processing times of the later jobs in a schedule are smaller than their normal processing times. Therefore, Biskup (1999) proposed that the actual processing time of job i, when it is scheduled at the rth position in the schedule, can be given by

$$p_{ir} = p_i \cdot r^{\alpha}, \tag{1}$$

where $\alpha \le 0$ is the learning index. This can also be interpreted as the position-dependent learning effects.

Wang and Xia (2005) proposed that the actual processing time p_{ir} can be given by

$$p_{ir} = p_i \cdot (\beta - \alpha r), \tag{2}$$

where β and α denote a constant number and a learning ratio, respectively.

Koulamas and Kyparisis (2007) assumed that the actual processing time p_{ir} can be given by

$$p_{ir} = p_i \cdot \left(1 - \frac{\sum_{k=1}^{r-1} p_{[k]}}{\sum_{k=1}^{n} p_k}\right)^{\alpha} = p_i \cdot \left(\frac{\sum_{k=r}^{n} p_{[k]}}{\sum_{k=1}^{n} p_k}\right)^{\alpha}, \tag{3}$$

where $p_{[k]}$ denotes the normal processing time occupying the kth position in the schedule and $\alpha \ge 1$.

The volume-dependent processing time can also affect the learning effects. The learning effects on the processing time of a job were assumed to depend on the number of jobs that are processed before the current job. Cheng and Wang (2000) proposed that the actual processing time $\hat{p}_i$ of job i can be modelled as follows:

$$\hat{p}_i = p_i - \alpha_i \cdot \min\{n_i, n_{0i}\} \tag{4}$$

for $i = 1, \ldots, n$, where α_i is the learning coefficient, n_i is a nonnegative integer with $0 \le n_i \le n-1$ indicating the number of jobs processed before job i in the schedule (i.e., $n_i + 1$ is the position of job i), and n_{0i} is a nonnegative integer with $n_{0i} \le n-1$ indicating a threshold value.

Another volume-dependent learning effects based on the job processing times were also considered by Kuo and Yang (2006c,b). Since the employees or workers can learn more if they perform a job with a longer processing time; that is, the actual processing time of a job is affected by the total processing time of the previous jobs, they proposed that the actual processing times can be given by

$$p_{ir} = \begin{cases} p_i & \text{if } r = 1 \\ (p_{[1]} + p_{[2]} + \cdots + p_{[r]})^{\alpha} \cdot p_i & \text{if } r \ge 2, \end{cases} \tag{5}$$

or

$$p_{ir} = \begin{cases} p_i & \text{if} r = 1 \\ (1 + p_{[1]} + p_{[2]} + \cdots + p_{[r]})^{\alpha} \cdot p_i & \text{if} r \ge 2, \end{cases} \tag{6}$$

where $\alpha \le 0$ is a learning index, and $p_{[k]}$ denotes the normal processing time occupying the kth position in the schedule.

The learning effects presented in (1), (2) and (3) are job-independent. However, in the realistic situations, the improvement in the production process of some jobs may be faster than that of others, or the different jobs are affected depending on their positions in the schedule. Therefore, it is reasonable to study the scheduling problem with job-dependent learning effects. Moshieov and Sidney (2003) proposed that the actual processing time p_{ir} can be given by

$$p_{ir} = p_i \cdot r^{\alpha_i}, \tag{7}$$

where α_i is a job-dependent negative parameter. Bachman and Janiak (2004) also introduced the actual processing time p_{ir} that can be given by

$$p_{ir} = p_i - \alpha_i r, \tag{8}$$

where α_i denotes a learning ratio.

Cheng et al. (2008) took the product of the models proposed by Biskup (1999) and Koulamas and Kyparisis (2007), respectively, to introduce a model that considered the position-based and sum-of-processing-timed-based learning effects in which the actual processing time of a job is a function of the total normal processing times of the jobs already processed and of the job's scheduled position with the form given by

$$p_{i[r]} = p_i \cdot \left(1 - \frac{\sum_{k=1}^{r-1} p_{[k]}}{\sum_{k=1}^{n} p_k}\right)^{a_1} \cdot r^{a_2}.$$

The model proposed by Lee and Wu (2009) generalized the model of Kuo and Yang (2006b), which is given by

$$p_{i[r]} = p_i \cdot \left(q(r) + \sum_{k=1}^{r-1} p_{[k]}\right)^{a}. \tag{9}$$

Yin et al. (2009) also generalized the model proposed by Cheng et al. (2008), which is given by

$$p_{i[r]} = p_i \cdot f\left(\sum_{k=1}^{r-1} p_{[k]}\right) \cdot g(r), \tag{10}$$

where the functions f and g satisfy some suitable conditions. Recently, based on the model of Yin et al. (2009) and Lai and

Journal: 10845-JIMS MS: 0827 ☐TYPESET ☐DISK ☐LE ☐CP Disp.:2013/8/17 Pages: 11 Layout: Large

Lee (2011) proposed a more general model given by

$$p_{i[r]} = p_i \cdot f\left(\sum_{k=1}^{r-1} \beta_k \cdot p_{[k]}, r\right), \tag{11}$$

where the function f with two arguments satisfies some suitable conditions.

The goal of unrestricted common due date problem is to jointly minimize the weighted earliness, tardiness and completion time. Here the unrestricted common due date d is regarded as a decision variable whose value is going to be determined. Let C_i, $E_i = \max\{0, d - C_i\}$ and $T_i = \max\{0, C_i - d\}$ be the completion time, earliness and tardiness of job i, respectively. We also denote by w_1, w_2 and w_3 the per time unit penalties for earliness, tardiness and the completion time, respectively. Then we shall find a schedule π that minimizes the following objective function:

$$f(\pi) = \sum_{i=1}^{n} (w_1 E_i + w_2 T_i + w_3 C_i). \tag{12}$$

By introducing the leaning effects in (1), Biskup (1999) showed that the unrestricted common due date problem can be solved as an assignment problem which takes $O(n^3)$ time. In other words, the unrestricted common due date problem with learning effects is polynomially solvable. Moshieov (2001) considered the following objective function

$$f(d, \pi) = \sum_{i=1}^{n} (w_1 d + w_2 E_i + w_3 T_i) \tag{13}$$

with learning effect given in (7) and the objective function in (13). Also, the corresponding assignment problem is solved to obtain the optimal schedule.

Using the standard pair-wise interchange arguments, the following results were obtained.

- Moshieov (2001) showed that the makespan minimization problem with learning effects given in (1) can be optimized by the SPT rule.
- Wang and Xia (2005) showed that the makespan minimization problem with learning effects given in (2) can be optimized by the SPT rule.
- Koulamas and Kyparisis (2007) showed that the makespan minimization problem with learning effects given in (3) can be optimized by the SPT rule.

On the other hand, Bachman and Janiak (2004) showed that the makespan minimization problem with learning effects given in (1) can be solved in $O(n^3)$ times by an assigning procedure, and the optimal schedule considering the learning effects given in (8) can be found in $O(n \log n)$ times by sequencing jobs in nondecreasing order of the learning ratio α_i. Moshieov and Sidney (2003) considered the learning effects given in (7). Kuo and Yang (2006c) considered the learning effects presented in (5) and shows that the optimal schedule that minimizes the makespan satisfies the following condition: the sequence of all jobs except for the first processed job is the smallest processing time first (SPT rule). Bachman and Janiak (2004) showed that the problems $1|\zeta_i, p_{ir} = p_i - \alpha_i r|C_{\max}$ and $1|\zeta_i, p_{ir} = p_i \cdot r^{\alpha}|C_{\max}$ are strongly NP-hard.

One of the elementary results of single-machine scheduling problem is that the sum of flowtimes of all jobs is minimized by sequencing the jobs according to the SPT rule. Incorporating the learning effects into this problem, the following results were obtained.

- Biskup (1999) showed that the total completion time minimization problem with learning effects given in (1) is optimized by the SPT order.
- Wang and Xia (2005) showed that the total completion time minimization problem with learning effects given in (2) is optimized by the SPT rule.
- Koulamas and Kyparisis (2007) showed that the total completion time minimization problem with learning effects given in (3) is optimized by the SPT rule.
- Kuo and Yang (2006b) showed that the total completion time minimization problem with learning effects given in (5) is optimized by the SPT rule.

Using the job interchanging technique, Bachman and Janiak (2004) also proved many interesting results.

Moshieov and Sidney (2003) considered the learning effects given in (7). On the other hand, Wu (2006) used the branch-and-bound method to minimize the total weighted completion time under the learning effects given in (1). The objective is to find an optimal schedule π^* such that

$$\sum_{i=1}^{n} w_i C_i(\pi^*) \leq \sum_{i=1}^{n} w_i C_i(\pi)$$

for any schedule π, where w_i are positive real numbers for $i = 1, \ldots, n$.

Let d_i be the due date of job i. The lateness is defined by $L_i = C_i - d_i$. The maximum lateness is defined as $L_{\max} = \max\{L_1, \ldots, L_n\}$. The objective is to minimize the maximum lateness $L_{\max}$. It is well-known that the conventional maximum lateness minimization problem is solvable by the earliest due date rule (EDD rule). However, Cheng and Wang (2000) showed that, under the consideration of learning effects given in (4), this problem becomes NP-hard in strong sense. Cheng and Wang (2000) also showed that, although the general problem is NP-hard in the strong sense, there are two special cases that can be solved in polynomial time. Let d denote the common due date and U_i be a 0–1 variable, where $U_i = 1$ if job i is late, i.e., $C_i > d$, and

Journal: 10845-JIMS MS: 0827 TYPESET DISK LE CP Disp.:2013/8/17 Pages: 11 Layout: Large

$U_i = 0$ otherwise, i.e., $C_i \leq d$. Moshiev and Sidney (2005) also considered the single-machine scheduling problem to minimize the number of tardy jobs.

We denote by $TC = \sum_{i=1}^{n} C_i$ the total completion time and by $TADC = \sum_{i=1}^{n}\sum_{j=1}^{n} |C_i - C_j|$ the total absolute differences in completion times. Let $\delta \in [0, 1]$. The objective is to find a schedule that minimizes the following measure

$$f(\pi) = \delta \cdot TC + (1 - \delta) \cdot TADC.$$

Moshiev (2001) considered the learning effects specified in (1) and obtain the optimal schedule by solving its corresponding assignment problem. Under the learning effects given in (1), Lee et al. (2004) used the branch-and-bound algorithm to find a schedule that minimizes the some of total completion time and the maximum tardiness, i.e., to find a schedule that minimizes the following objective function

$$\min \lambda \cdot TC(\pi) + (1 - \lambda) \cdot T_{\max}(\pi),$$

where $0 < \lambda < 1$, $TC(\pi) = \sum_{i=1}^{n} C_i(\pi)$ is the total completion time and $T_{\max}(\pi)$ is the maximum tardiness of a schedule π.

Suppose that there are n jobs to be classified into m groups and to be processed on a single machine. All jobs are available at time zero. It is assumed that there is no setup time between any two consecutive jobs in the same group. However, the group setup times are required. The group setup times are assumed to be sequence-independent. Moreover, the normal processing time of a job is incurred if the job is scheduled first in a sequence of a certain group. Let J_{ij} denote the jth job in group G_i and p_{ijr} be the actual processing time of J_{ij} that is scheduled in the rth position in a sequence in group G_i. Kuo and Yang (2006a) considered the time-dependent learning effects defined by

$$p_{ijr} = (1 + p_{i[1]} + \cdots + p_{i[r-1]})^{\alpha_i} p_{ij}, \tag{14}$$

where α_i is a constant learning index in a certain group G_i, p_{ij} is the normal processing time of J_{ij} in the original sequence and $p_{i[k]}$ is the normal processing time of $J_{i[k]}$ that is scheduled in the kth position in a sequence in group G_i.

For the scheduling problems with deteriorating jobs, the actual processing time of a job in a schedule is modeled as an increasing function of its starting time due to deterioration effects. This model reflects a variety of real-life situations such as steel production, resource allocation, fire fighting, maintenance or cleaning, in which any delay in processing a job may result in an increasing effort to accomplish the job. In order to obtain the analytic results, most researchers model the actual processing time of a job as a linear or piecewise linear increasing function of its starting time. For example, the actual processing time can be assumed as $p_i + \beta_i t$, where p_i is the normal processing time, β_i is the growth rate of the processing time, and t is the starting time, of job i. Wang and Cheng (2007) incorporated the learning effects into this kind of problem. If job i is scheduled in position r in a sequence, then its actual processing time is given by

$$p_{ir}(t) = (p_0 + \beta_i t) \cdot r^{\alpha}, \tag{15}$$

where p_0 is a common normal processing time which is incurred if job i is scheduled first in a sequence, t is the starting time of job i to be processed, β_i is the growth rate of the processing time of job i, which is the amount of increase in the processing time of job i per unit delay in its starting time due to the deterioration effects, and α is the learning index. On the other hand, Lee (2004) and Wang (2007) also incorporated the learning effects into the scheduling problems with deteriorating jobs. If job i is started at time t and scheduled in position r in a sequence, then the actual processing time is given by

$$p_{ir}(t) = p_i \cdot (\zeta(t) + \beta \cdot r^{\alpha}),$$

where $\zeta(t)$ is an increasing function.

Multi-machine scheduling problems

Suppose that there are n jobs to be processed on two machines, where each job requires to be processed on machine 1 first and then on machine 2. We denote by a_i and b_i the normal processing times of job i on machine 1 and 2, respectively. For the job-position-based learning effects, Lee and Wu (2004) and Wu et al. (2007) proposed that the actual processing times can be given by

$$a_{ir} = a_i \cdot r^{\alpha} \text{ and } b_{ir} = b_i \cdot r^{\alpha} \tag{16}$$

with $\alpha < 0$. Thus the completion time of job scheduled in the rth position is given by

$$C_{[r]} = \max\left\{\sum_{j=1}^{r} a_{[j]} \cdot j^{\alpha}, C_{[r-1]}\right\} + b_{[r]} \cdot r^{\alpha}.$$

Under the learning effects given in (16), Wu et al. (2007) provided a heuristic algorithm using the SA approach to minimize the maximum tardiness, and Lee and Wu (2004) used the branch-and-bound algorithm to minimize the total completion time.

On the other hand, Koulamas and Kyparisis (2007) proposed that the actual processing times can be given by

$$a_{ir} = a_i \cdot \left(1 - \frac{\sum_{k=1}^{r-1} a_{[k]}}{\sum_{k=1} a_k}\right)^{\alpha} \text{ and } b_{ir} = b_i \cdot \left(1 - \frac{\sum_{k=1}^{r-1} b_{[k]}}{\sum_{k=1} b_k}\right)^{\alpha} \tag{17}$$

which $\alpha \geq 1$. Two special cases that are called ordered job processing times and proportional job processing times are also investigated.

Journal: 10845-JIMS MS: 0827 TYPESET DISK LE CP Disp.:2013/8/17 Pages: 11 Layout: Large

For the problem of ordered job processing times, we assume $a_i \leq b_i$ for all jobs $i = 1, \ldots, n$ and $b_j \leq b_k$ whenever $a_j \leq a_k$ for any two jobs j and k. In this case, the problem is denoted by $F2|LE, ord|\gamma(\pi)$, where γ is an objective function. For the problem of proportional job processing times, we assume $b_i = ca_i$ for all jobs $i = 1, \ldots, n$, where $c \geq 1$ is a constant factor. In this case, the problem is denoted by $F2|LE, prp|\gamma(\pi)$. Under these settings, Koulamas and Kyparisis (2007) also obtained many interesting results.

For the n-job and m-machine scheduling problems with learning effects, we denote by p_{ij} the normal processing time of job i on machine j and p_{ijr} the actual processing time of job i on machine j that is scheduled in position r. Wang and Xia (2005) proposed two models that are given by

$$p_{ijr} = p_{ij} \cdot (\beta - \alpha r) \text{ and } p_{ijr} = p_{ij} \cdot r^{\alpha} \tag{18}$$

for $i, r = 1, \ldots, n$ and $j = 1, \ldots, m$, where β is a constant number and α is a learning index. It is assumed that β is a positive integer. Since the processing time is positive, for model (18), it is also assumed that $\beta - (n + 1) \cdot \alpha > 0$.

For the problem $Fm||\sum C_i$, Gonzalez and Sahni (1978) provided an approximation algorithm in order of increasing $L_i = \sum_{j=1}^{m} p_{ij}$ and show that it has worst-case performance ratio, i.e., this algorithm is guaranteed to produce a schedule with cost no more than m times the cost of an optimal schedule. This heuristic algorithm will also be referred as SPT rule. Therefore, Wang and Xia (2005) used the SPT rule in order of L_i as an approximate algorithm for the problem $Fm|p_{ijr} = p_{ij} \cdot (\beta - \alpha r)|\sum C_i$, and obtained some interesting results. Wang and Xia (2005) also used the SPT rule as an approximate algorithm to problem $Fm|p_{ijr} = p_{ij} \cdot (\beta - \alpha r)|C_{\max}$, and obtained some other interesting results.

Parallel machine scheduling problems

The parallel machine scheduling with learning effect was studied by Moshieov (2001). We firstly consider n jobs to be processed on m parallel identical machines. We assume that $m < n$. Jobs are numbered such that $p_1 \leq p_2 \leq \cdots \leq p_n$. With no learning effects, the problem $Pm||C_{\max}$ is known to be NP-hard even for two machines. Clearly, for the learning effects given in (1), the problem $Pm|p_{ir} = p_i \cdot r^{\alpha}|C_{\max}$ is also NP-hard, since the special case $\alpha = 0$ is identical with the conventional version. However, minimizing flow time on parallel identical machines, i.e., $Pm||\sum C_i$, is solved by the SPT rule. When learning effect in (1) is assumed, Moshieov (2001) showed that an optimal schedule for $Pm|p_{ir} = p_i \cdot r^{\alpha}|\sum C_i$ consists of SPT sequences on each machine. The problem that n jobs are to be processed on m unrelated parallel machines was also studied by Moshieov and Sidney (2003) by formulating it as an assignment problem.

Multi-machine scheduling problems with learning effects

Now, we shall consider the n-job and m-machine flow shop scheduling problems with learning effects. Given n jobs and m machines, each job consists of m operations. The mth operation of each job has to be processed on the mth machine. The $(m + 1)$th operation starts only if the mth operation has been completed. Each machine is assumed to process one operation at a time with no precedence constraints between jobs. Operations are non-preemptive and are available for processing at time 0 on machine 1. Let p_{ij} be the normal processing time for job i on machine j, $i = 1, \ldots, n$ and $j = 1, \ldots, m$. In this paper, the learning factors come from jobs and machines will be included in the scheduling problem. Therefore, we can consider three kinds of scheduling problems with learning effects.

Job-dependent learning effects

Suppose that only the job-dependent learning factor is taken into account in this problem; that is, the learning factor comes from machines will be ignored. We denote by δ_i the job-dependent parameter for job $i = 1, \ldots, n$, where δ_i are negative real numbers. Then the actual processing time of job i on machine j scheduled in position r is given by

$$p_{ijr} = p_{ij} \cdot r^{\delta_i}. \tag{19}$$

This problem was considered by Moshieov and Sidney (2003).

Machine-dependent learning effects

Suppose that only the machine-dependent learning factor is taken into account in this problem; that is, the learning factor comes from jobs will be ignored. We denote by η_j the machine-dependent parameter for machine $j = 1, \ldots, m$, where η_i are negative real numbers. Then the actual processing time of job i on machine j scheduled in position r is given by

$$p_{ijr} = p_{ij} \cdot r^{\eta_j}. \tag{20}$$

To the best of our knowledge, this problem has not been investigated in scheduling problems with learning effects.

Job-dependent and machine-dependent learning effects

In the general case, we shall consider the job-dependent and machine-dependent learning factors simultaneously. We

Journal: 10845-JIMS MS: 0827 TYPESET DISK LE CP Disp.:2013/8/17 Pages: 11 Layout: Large

denote by λ_{ij} the job-machine-dependent parameter for job i on machine j, where λ_{ij} are negative real numbers. Then the actual processing time of job i on machine j scheduled in position r is given by

$$p_{ijr} = p_{ij} \cdot r^{\lambda_{ij}}. \tag{21}$$

For example, we may take $\lambda_{ij} = \delta_i + \eta_j$. This problem has also not been investigated in this research field so far.

For convenient discussions, Eqs. (19), (20) and (21) are unified as the following formula

$$p_{ijr} = p_{ij} \cdot r^{\zeta_{ij}}, \tag{22}$$

where ζ_{ij} is a learning factor defined below:

$$\zeta_{ij} = \begin{cases} \delta_i & \text{if only the job-dependent learning effects are considered} \\ \eta_j & \text{if only the machine-dependent learning effects are considered} \\ \lambda_{ij} & \text{if the job-dependent and machine-dependent learning effects are considered.} \end{cases} \tag{23}$$

Design of heuristic algorithms

We shall use four different heuristic algorithms that are SA, GA, ACO, and PSO to search for the "best solution" of the scheduling problems proposed in this paper. All of the algorithms adopted in this paper are also based on the concept of random keys proposed by Bean (1994) to generate the individuals.

Simulated annealing

The idea of SA algorithm arises from the physical annealing of solids, and it has been successfully applied to combinatorial problems by Kirkpatrick et al. (1983). SA has the advantage that it can avoid be trapped in a local optimum by occasionally allowing "hill-climbing moves". This algorithm, although it was invented long time ago, still works very well and very efficiently in many problems up to now. In literature, it is often used to compare with other more fashioned heuristic algorithms. In this paper, we adopt the standard type of SA algorithm. The reader can refer to Kirkpatrick et al. (1983) for the main steps of this algorithm. Nearchou (2004) and Mirsanei et al. (2011) used SA to solve some other scheduling problems.

Genetic algorithms

GA has a lot of formulation in literature. The main steps adopted in this paper are the elitism, uniform crossover, and immigration. We shall randomly generate N chromosomes in the initial population, and use the concept of random keys proposed by Bean (1994) to generate the chromosomes. Suppose that we consider the five-job problem. Then the length of chromosome will be five. Therefore, we generate five random numbers in (0, 1) for each chromosome. The mapping to the job sequence is accomplished by sorting the random numbers and sequencing the jobs in ascending order. For example, if we have obtained the random numbers (0.46, 0.91, 0.33, 0.75, 0.51), i.e., $1 \leftarrow 0.46$, $2 \rightarrow 0.91$, $3 \leftarrow 0.33$, $4 \leftarrow 0.75$ and $5 \rightarrow 0.51$, then it would represent the chromosome (job sequence) (3,1,5,4,2), since $0.33 < 0.46 < 0.51 < 0.75 < 0.91$.

For the crossover, we are going to invoke the parameterized uniform crossover proposed by Speras and DeJong (1991). Suppose that two chromosomes (0.46, 0.91, 0.33, 0.75, 0.51) and (0.84, 0.32, 0.64, 0.04, 0.48) are chosen randomly from the old population. At each gene, we toss a faired coin to select which parent will contribute the allele. We can also consider the biased coin to perform this crossover. For example, the probability of tossing a head may take as 0.7. In this paper, we take the probability of tossing a head as 0.5. Now we assume that a coin toss of head selects the allele from the first parent, and a tail chooses the allele from the second parent, which forms the first offspring. The second offspring is obtained in the reverse way as obtaining the first offspring. We provide a simple example given below:

Coin toss	T	H	T	H	T
Parent 1	0.46	0.91	0.33	0.75	0.51
Paremt 2	0.84	0.32	0.64	0.04	0.48
Offspring 1	0.84	0.91	0.64	0.75	0.48
Offspring 2	0.46	0.32	0.33	0.04	0.51

Then the two offsprings can be obtained by sorting the random numbers and sequencing the jobs in ascending order.

Instead of performing mutation, we employ the concept of immigration in this paper. In other words, at each generation, more new members of the population are randomly generated from the same distribution. In this paper, we take the uniform (0,1) random variate. The stopping criterion will be determined by specifying the maximal generation.

Finally, the reproduction is accomplished by using the elitist strategy. We choose the best chromosomes (e.g., 10 % of the population size) from one generation to the next. The elitist strategy is frequently adopted by different variants of GAs.

Now we briefly describe the entire evolution procedure. Let P_t be the family of chromosomes in the tth generation, and $|P_t|$ denote the population size of P_t. The next generation is made of $a\%$ best chromosomes from P_t, $b\%$ chromosomes for taking crossovers, and $c\%$ chromosomes generated randomly (i.e., performing immigration), where

Journal: 10845-JIMS MS: 0827 TYPESET DISK LE CP Disp.:2013/8/17 Pages: 11 Layout: Large

$a + b + c = 100$. The computational procedure is described as below:

- **Step 1**. Initialize the population by generating the random numbers.
- **Step 2**. Calculate the completion time of every job in each schedule selected from the population.
- **Step 3**. Calculate the fitness function $f(\pi)$ for each schedule π.
- **Step 4**. Choose $a \cdot 0.01 \cdot |P_t|$ best chromosomes as the members in the next generation.
- **Step 5**. Choose $b \cdot 0.01 \cdot |P_t|$ chromosomes to perform crossover and produce the members in the next generation.
- **Step 6**. Randomly generate $c \cdot 0.01 \cdot |P_t|$ chromosomes as the members in the next generation like performing immigration.
- **Step 7**. Save the best schedule and fitness value obtained so far.
- **Step 8**. If the maximal generation is reached, then STOP, otherwise go to Step 2. to perform another iteration.

Ant colony optimization

The ACO proposed by Dorigo and Stützle (2004) has also been recognized as an efficient algorithm to solve the combinatorial optimization problem. Therefore, a lot of different variants of ACO have been proposed based on the different purposes of combinatorial optimization problems. For example, Lai and Wu (2009) used one of the variants to solve the scheduling problems with fuzzy-valued processing times. Also, Arnaout et al. (2010) and Solano-Charris et al. (2011) used the ACO to solve some other scheduling problems.

The main steps adopted in this paper will be described below. The probability, currently at node i, for choosing next node j is given by

$$p_{ij}^{(k)} = \frac{\tau_{ij}}{\sum_{l \in N_i^{(k)}} \tau_{il}} \text{ if } j \in N_i^{(k)}, \tag{24}$$

where $N_i^{(k)}$ is the neighborhood of node i except for the predecessor of node i when ant k is staying at node i, and τ_{ij} denotes the amount of pheromone currently deposited in the edge (i, j).

Dorigo and Stützle (2004) modified the random proportional rule and proposed a so-called pseudo-random proportional rule that is given below: when an ant k is now located at city i, it moves to a city j according to the following rule

$$j = \begin{cases} \arg\max_{l \in N_i^{(k)}} \tau_{il} & \text{if} q \leq q_0 \\ J & \text{otherwise,} \end{cases} \tag{25}$$

where q is a random number, $q_0 \in [0, 1]$ is a parameter, and J is a random variable selected according to the probability distribution given by (24).

Only the best-so-far tour is allowed to deposit the pheromone after each iteration. Therefore, the pheromone update rule for the tour $T^{(\text{bs})}$ is given by

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{(\text{bs})} \tag{26}$$

for the edge (i, j) in $T^{(\text{bs})}$, where $\Delta\tau_{ij}^{(\text{bs})}$ is given by

$$\Delta\tau_{ij}^{(\text{bs})} = \begin{cases} \dfrac{\alpha}{\eta(\tilde{C}_{\max}^{(\text{bs})})} & \text{if edge } (i, j) \text{ is in } T^{(\text{bs})} \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

for some constant α in $\mathbb{R}$.

In addition to the global pheromone update rule in (26), Dorigo and Stützle (2004) also suggested a local pheromone update rule that will be applied after all K ants having finished the tour construction. For edge (i, j) in some tour T_k that is constructed by ant k, the update rule is given by

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi \cdot \tau_0, \tag{28}$$

where $\xi \in (0, 1)$ and τ_0 is set to be the initial pheromone.

Now, the computational procedure is summarized below.

- **Step 1**. Initialize K artificial ant tours by randomly generating the random numbers in $\{1, \ldots, n\}$.
- **Step 2**. Initialize the pheromone trails by depositing a constant value p on all edges.
- **Step 3**. Construct the artificial ant tours according to the rule presented in (25).
- **Step 4**. Evaluate the objective function values of schedules determined by the artificial ant tours.
- **Step 5**. Perform the local pheromone trails updating rule according to (28).
- **Step 6**. Identify the best-so-far tour.
- **Step 7**. Perform the global pheromone trails updating rule according to (26).
- **Step 8**. If the pre-determined maximal iteration is reached, then STOP and return the "optimal schedule"; otherwise we go to Step 4 to perform another iteration.

Particle swarm optimization

The PSO has also been used to solve the scheduling problems by referring to Tasgetiren et al. (2007) and the references therein. The PSO is based on the social interaction and communication such as bird flocking and fish schooling. The PSO is different from other evolutionary methods in a way that it does not use the filtering operation, e.g., crossover, mutation and so on. The members of the entire population are maintained through the search procedure so

Journal: 10845-JIMS MS: 0827 TYPESET DISK LE CP Disp.:2013/8/17 Pages: 11 Layout: Large

that the information can be socially shared among individuals to conduct the search direction towards the best position in the search space. The PSO was originally introduced by Eberhard and Kennedy (1995). Tasgetiren et al. (2007) introduced the smallest position value (SPV) rule borrowed from the random key representation in Bean (1994) to convert the continuous position value to a discrete job permutation.

The main steps adopted in this paper will be described below. In the initial population, every particle is a list of n random numbers between 0 and 1, where n is the job number. The SPV rule applies to each particle to find its corresponding permutation. The ith particle in the tth generation is denoted as

$$X_i^{(t)} = \left(x_{i1}^{(t)}, x_{i2}^{(t)}, \ldots, x_{in}^{(t)}\right).$$

The initial continuous position values of the particle is produced randomly:

$$x_{ij}^{(0)} = x_{\min} + (x_{\max} - x_{\min}) \cdot r_1$$

where $x_{\min} = 0$, $x_{\max} = 4$ and r_1 is a uniform random number between 0 and 1. Initial velocities are established in a similar way:

$$v_{ij}^{(0)} = v_{\min} + (v_{\max} - v_{\min}) \cdot r_2$$

where $v_{\min} = -4$, $v_{\max} = 4$ and r_2 is a uniform random number between 0 and 1. We denote by

$$P_i^{(t)} = \left(p_{i1}^{(t)}, p_{i2}^{(t)}, \ldots, p_{in}^{(t)}\right)$$

the personal best particle in the tth generation which is initialized by $p_{i1}^{(0)} = x_{i1}^{(0)}$ for $1 \le i \le n$. We also denote by

$$G^{(t)} = \left(g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)}\right)$$

the global best particle in the tth generation which is initialized as the best particle in the initial population.

The inertia weight needs to be updated according to the following rule

$$w^{(t)} = w^{(t-1)} \cdot \beta,$$

where $\beta = 0.975$ is the decrement factor and $w^{(0)}$ is set to be 0.9 and never decreases below 0.4. The velocity is updated according to the following rule

$$v_{ij}^{(t)} = w^{(t-1)} \cdot v_{ij}^{(t-1)} + c_1 \cdot r_1 \cdot \left(p_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right)$$
$$+ c_2 \cdot r_2 \cdot \left(g_j^{(t-1)} - x_{ij}^{(t-1)}\right)$$

where $c1$ and $c2$ are acceleration coefficients set to be 2, and $r1$ and $r2$ are uniform random numbers between 0 and 1. Finally, the position is updated according to the following rule

$$x_{ij}^{(t)} = x_{ij}^{(t-1)} + v_{ij}^{(t)}.$$

Now, the computational procedure is summarized below.

- **Step 1**. Initialization and evaluation.
- **Step 2**. Update iteration counter.
- **Step 3**. Update inertia weight.
- **Step 4**. Update velocity
- **Step 5**. Update position
- **Step 6**. Evaluation
- **Step 7**. Update personal best
- **Step 8**. Update global best
- **Step 9**. If the stopping criterion is satisfied then stop. Otherwise go to STEP 2.

Numerical examples

We consider n-job and m-machine flow shop scheduling problem. Recall that $C_{J_i j}$ is the completion time of job J_i on machine j. We denote by C_{J_i} the completion time of job J_i, i.e., $C_{J_i} = C_{J_i m}$, which means the completion time on the last machine. We sometimes simply write C_i as the completion time of job i. Now the *makespan* $C_{\max}$ is defined as the last job to leave the system, i.e., $C_{\max} = \max\{C_1, \ldots, C_n\}$. In other words, we see that $C_{\max} = C_{J_n m}$. The purpose is to minimize the makespan. Therefore, we want to solve the following problem

$$\min_{\pi \in \Pi} \quad f(\pi) = C_{\max}.$$

Now, we are in a position to perform the computational experiments. All of the algorithms are coded in the commercial software MATLAB and are executed in a personal computer with Intel(R)Core(TM)2 6300 1.86, 1.87 GHz and 1.99 GB RAM on Windows XP. We test the proposed algorithms on flowshop scheduling problem with job-dependent and machine-dependent learning effects. We consider three machines and three different numbers of jobs $n = 20$, $n = 50$ and $n = 100$.

Choosing the values of parameters is time-consuming and experience-depending. We first determine the range of possible values of each parameter based on the previous experience or well-known adoption in literature. For example, the ranges of parameters of GA and ASO refer to Lai and Wu (2008, 2009), and the ranges of parameters of PSO are adapted from Tasgetiren et al. (2007). Also, we still have to make preliminary trial among such ranges by testing different values for every parameter in order to determine the best suit among them. When the best fitness does not improve for 10 generations, the algorithm is stopped, which is the stopping

Journal: 10845-JIMS MS: 0827 TYPESET DISK LE CP Disp.:2013/8/17 Pages: 11 Layout: Large

criterion adopted in this paper. Now, the values of parameters are shown below.

(i) For the different job sizes, we take the same values of parameters of ACS, which are listed below: generation number is 300, population size is 200, ρ is 0.2, ξ is 0.6, pseudo-random number is 0.8, α is 40, and initial pheromone is 0.005.

(ii) For the different job sizes, we take the same values of parameters of GA, which are listed below: generation number is 2,000, population size is 500, crossover rate is 0.8, elitist rate is 0.1, and mutate rate is 0.1.

(iii) The values of parameters of PSO are listed below.

- job numbers $n = 20$: generation number is 1,000, population size is 500, the acceleration coefficient is 2.5, initial inertial weight is 0.8, and the decrement factor is 0.975.
- job numbers $n = 50$ and $n = 100$: generation number is 2,000, population size is 500, the acceleration coefficient is 2.5, initial inertial weight is 0.8, and the decrement factor is 0.975.

(iv) The values of parameters of SA are listed below.

- job numbers $n = 20$: generation number is 10, 000, $L = 0.1$ and

$$\lambda = \frac{\text{current iteration number}}{\text{generation number}}.$$

- job numbers $n = 50$ and $n = 100$: generation number is 100,000, $L = 0.1$ and

$$\lambda = \frac{\text{current iteration number}}{\text{generation number}}.$$

For each case of different job size, a set of 20 instances of job processing times associated with the job-dependent and machine-dependent learning indices are randomly generated.

- The job processing time on machines 1, 2 and 3 are generated from the uniform distribution between the integers 1 and 50.
- The job-dependent and machine-dependent learning indices are generated from the uniform distributions between −0.2 and 0.

Because the heuristic algorithms are kind of random search, each instance is run 5 times. We present the best one among 5 times and the mean of them. For each heuristic algorithm, we execute 20 experiments. The average CPU time for 20 experiments is reported. Now, the experimental results are shown in the following tables, where the CPU time is reported in average with second as unit for all experiments.

Job numbers 20								
Experiments	ACO		GA		PSO		SA	
	Mean	Min	Mean	Min	Mean	Min	Mean	Min
Exp.1	586	581	**578**	**578**	583	579	585	579
Exp.2	605	605	**603**	**603**	604	**603**	604	604
Exp.3	550	548	**542**	**542**	543	**542**	**542**	**542**
Exp.4	617	616	**612**	**612**	613	**612**	**612**	**612**
Exp.5	494	490	**480**	**480**	482	481	481	**480**
Exp.6	576	570	**565**	**565**	**565**	**565**	**565**	**565**
Exp.7	518	517	**513**	**513**	515	514	515	514
Exp.8	549	548	**546**	**546**	547	**546**	**546**	**546**
Exp.9	514	512	**498**	**498**	500	499	**498**	**498**
Exp.10	518	516	**499**	**499**	503	501	500	**499**
Exp.11	530	529	**524**	**524**	528	**524**	**524**	**524**
Exp.12	525	522	**494**	**494**	501	495	**494**	**494**
Exp.13	502	501	**499**	**499**	**499**	**499**	**499**	**499**
Exp.14	440	440	**436**	**435**	**436**	**435**	**436**	436
Exp.15	517	516	**511**	**511**	**511**	**511**	**511**	**511**
Exp.16	542	540	**531**	**531**	533	**531**	532	**531**
Exp.17	572	571	**564**	**564**	56	**564**	**564**	**564**
Exp.18	583	579	**555**	**555**	557	556	556	556
Exp.19	516	515	**514**	**514**	515	**514**	**514**	**514**
Exp.20	521	520	**518**	**518**	520	519	519	519
Average CPU time (s)	44.4		38.3626		36.9812		0.6532	

Job numbers 50								
Experiments	ACO		GA		PSO		SA	
	Mean	Min	Mean	Min	Mean	Min	Mean	Min
Exp.1	1,048	1,042	**1,011**	**1,010**	1,018	1,013	**1,011**	1,011
Exp.2	1,076	1,073	**1,057**	**057**	1,064	1,063	**1,057**	**1,057**
Exp.3	1,064	1,054	1,021	**1,019**	1,029	1,025	**1,019**	**1,019**
Exp.4	1,075	1,066	1,037	1,037	1,045	1,038	**1,034**	**1,034**
Exp.5	1,038	1,034	**1,004**	**1,004**	1,008	1,006	1,005	1,005
Exp.6	1,095	1,085	**1,055**	**1,055**	1,064	1,062	1,062	**1,055**
Exp.7	926	924	**894**	**893**	903	899	**894**	**893**
Exp.8	1,123	1,111	**1,087**	**1,087**	1,091	1,089	1,088	1,088
Exp.9	977	970	936	936	945	939	**935**	**934**
Exp.10	1,063	1,053	**1,033**	**1,033**	1,040	1,037	**1,033**	**1,033**
Exp.11	1,003	998	**972**	**971**	977	975	**972**	972
Exp.12	1,038	1,036	**1,013**	**1,013**	1,019	1,015	1,014	1,014
Exp.13	1,121	1,115	**1,078**	**1,077**	1,084	1,081	**1,078**	1,078
Exp.14	1,173	1,163	**1,123**	**1,122**	1,131	1,127	1,130	1,123
Exp.15	1,087	1,079	**1,060**	**1,060**	1,064	1,062	1,062	1,061
Exp.16	1,157	1,153	**1,131**	**1,128**	1,141	1,138	**1,131**	**1,128**
Exp.17	1,063	1,060	1,044	1,043	1,050	1,047	**1,043**	**1,042**
Exp.18	1,011	1,005	980	978	991	986	**977**	**977**
Exp.19	1,002	995	**969**	**969**	975	**969**	970	**969**
Exp.20	974	971	**950**	**949**	955	952	951	**949**
Average CPU time (s)	152.2905		94.6624		94.772		10.25	

Job numbers 100								
Experiments	ACO		GA		PSO		SA	
	Mean	Min	Mean	Min	Mean	Min	Mean	Min
Exp.1	2,019	2,010	**1,961**	**1,960**	1,982	1,968	**1,961**	1,961
Exp.2	1,843	1,838	1,782	**1,785**	1,803	1,795	**1,785**	**1,785**
Exp.3	1,896	1,889	**1,844**	1,843	1,865	1,860	1,847	**1,841**
Exp.4	1,992	1,988	1,921	1,921	1,945	1,941	**1,920**	**1,920**
Exp.5	1,751	1,744	**1,685**	1,683	1,708	1,701	1,689	**1,681**
Exp.6	1,944	1,936	**1,864**	1,863	1,883	1,878	**1,864**	**1,862**
Exp.7	1,817	1,809	1,747	1,745	1,766	1,756	**1,741**	**1,738**
Exp.8	2,080	2,073	**2,028**	**2,027**	2,053	2,042	2,034	2,028
Exp.9	1,995	1,992	**1,915**	**1,913**	1,939	1,928	**1,915**	1,914
Exp.10	1,776	1,771	1,722	1,721	1,742	1,735	**1,720**	**1,718**
Exp.11	1,820	1,812	1,753	1,751	1,775	1,768	**1,751**	**1,750**
Exp.12	1,745	1,735	1,673	1,671	1,697	1,691	**1,670**	**1,670**
Exp.13	1,872	1,857	1,815	**1,814**	1,832	1,827	**1,814**	**1,814**
Exp.14	1,807	1,794	**1,748**	1,748	1,769	1,756	**1,748**	**1,745**
Exp.15	1,751	1,742	1,688	1,685	1,710	1,699	**1,684**	**1,684**
Exp.16	1,965	1,961	**1,902**	**1,901**	1,918	1,906	**1,902**	19,02
Exp.17	1,947	1,943	**1,900**	**1,899**	1,917	1,909	1,908	1,900
Exp.18	1,876	1,871	**1,812**	**1,811**	1,829	1,822	**1,812**	**1,811**
Exp.19	1,832	1,829	1,781	**1,778**	1,794	1,791	**1,780**	1,779
Exp.20	1,779	1,764	**1,718**	1,716	1,736	1,727	**1,718**	**1,715**
Average CPU time (s)	449.0344		1,90.6218		190.8124		16.8968	

The experiments show that the GA outperforms the other heuristic algorithms for the job sizes of 20 and 50. However, for the job size of 100, the SA shows the best results in the search domain. Because the searched results of heuristic algorithms depend heavily on the initial values of parameters and the types of problems, the performance for the different heuristic algorithms presented in this paper cannot apply to the other combinatorial optimization problems. In other words, the efficiency of different heuristic algorithms is problem-dependent.

Conclusion

The main purpose of this paper is to propose a new model for the multi-machine scheduling problems by simultaneously considering the job-dependent and machine-dependent learning factors. Since this general problem is really complicated, we solve it by using four popular heuristic algorithms in literature, which are SA, GA, ACO and PSO etc.

The scheduling problems considered in this paper always assume that the resources are available and there is no deadlock issue. This may not be sensible in the reality. Owing to the competition for limited resources among several processes, the entire system might get stuck at deadlock. For this issue, we may refer to Hu and Li (2009a,b,c, 2010) and Hu et al. (2011). Therefore, in the future research, we can consider the scheduling problems with deadlock issue.

Lee (2004), Toksari and Güner (2010), Wang (2007), Wang and Cheng (2007) and Wu et al. (2012) simultaneously considered the deteriorating jobs and learning effects in scheduling problems. In the future research, we can also study the multi-machine scheduling problems by simultaneously considering the deteriorating jobs and the job-dependent and machine-dependent learning factors. We can also impose the job-dependent and machine-dependent learning factors upon the different models reviewed in second section in the future research. These considerations may be the challenge topic.

On the other hand, in the future research, it is also possible to propose different variants of the prototype of the multi-machine scheduling problems with job-dependent and machine-dependent learning effects in third section. For example, we may study the job-dependent and machine-dependent learning factors upon the sum-of-processing-time based problems. More precisely, we can extend (9) to the following formula

$$p_{ij[r]} = p_{ij} \cdot \left(q(r) + \sum_{k=1}^{r-1} p_{[k]} \right)^{\zeta_{ij}},$$

where ζ_{ij} is defined in (23). We can also extend (10) to the following formula

$$p_{ij[r]} = p_{ij} \cdot f_{ij} \left(\sum_{k=1}^{r-1} p_{[k]} \right) \cdot g_{ij}(r), \quad (29)$$

where the functions f_{ij} and g_{ij} satisfy some suitable conditions, and play the same roles as parameter ζ_{ij}. In general, we can extend (11) to the following formula

$$p_{ij[r]} = p_{ij} \cdot f_{ij} \left(\sum_{k=1}^{r-1} \beta_k \cdot p_{[k]}, r \right),$$

which also generalizes the setting in (29).

As we have mentioned before, the multi-machine scheduling problems by simultaneously considering the job-dependent and machine-dependent learning factors proposed in this paper is complicated. In the future research, we shall also develop some other more efficient heuristic algorithms to solve this complicated problem.

References

Arnaout, J.-P., Rabadi, G., & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *21*, 693–701.

Bachman, A., & Janiak, A. (2004). Scheduling jobs with position-dependent processing times. *Journal of Operational Research Society*, *55*, 257–264.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, *6*, 154–160.

Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research*, *115*, 173–178.

Cheng, T. C. E., Wu, C. C., & Lee, W. C. (2008). Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects. *Information Sciences*, *178*, 2476–2487.

Cheng, T. C. E., & Wang, G. (2000). Single machine scheduling with learning effect considerations. *Annals of Operations Research*, *98*, 273–290.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Eberhard, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science, Nagoya, Japan* (pp. 39–43).

Gonzalez, T., & Sahni, S. (1978). Flowshop and jobshop schedule: Complexity and approximation. *Operations Research*, *26*, 36–52.

Hu, H., & Li, Z. (2009a). Modeling and scheduling for manufacturing grid workflows using timed Petri nets. *The International Journal of Advanced Manufacturing Technology*, *42*, 553–568.

Hu, H., & Li, Z. (2009b). Liveness enforcing supervision in video streaming systems using siphons. *Journal of Information Science and Engineering*, *25*, 1863–1884.

Hu, H., & Li, Z. (2009c). Local and global deadlock prevention policies for resource allocation systems using partially generated reachability graphs. *Computers and Industrial Engineering*, *57*, 1168–1181.

Hu, H., & Li, Z. (2010). Synthesis of liveness enforcing supervisor for automated manufacturing systems. *Journal of Intelligent Manufacturing*, *21*, 555–567.

Hu, H., Li, Z., & Al-Ahmari, A. (2011). Reversed fuzzy Petri nets and their application for fault diagnosis. *Computers and Industrial Engineering*, *60*, 505–510.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Koulamas, C., & Kyparisis, G. J. (2007). Single-machine and two-machine flowshop scheduling with general learning functions. *European Journal of Operational Research*, *178*, 402–407.

Kuo, W.-H., & Yang, D.-L. (2006a). Single-machine group scheduling with a time-dependent learning effect. *Computers and Operations Research*, *33*, 2099–2112.

Kuo, W.-H., & Yang, D.-L. (2006b). Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect. *European Journal of Operational Research*, *174*, 1184–1190.

Kuo, W.-H., & Yang, D.-L. (2006c). Minimizing the makespan in a single machine scheduling problem with a time-based learning effect. *Information Processing Letter*, *97*, 64–67.

Lai, P.-J., & Lee, W.-C. (2011). Single-machine scheduling with general sum-of-processing time-based and position-based learning effects. *Omega*, *39*, 467–471.

Lai, P.-J., & Wu, H.-C. (2008). Using genetic algorithms to solve fuzzy flow shop scheduling problems based on possibility and necessity measures. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *16*, 409–433.

Lai, P.-J., & Wu, H.-C. (2009). Using ant colony optimization to minimize the fuzzy makespan and total weighted fuzzy completion time in flow shop scheduling problems. *The International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *17*, 559–584.

Lee, W. C. (2004). A note on deteriorating jobs and learning in single-machine scheduling problems. *International Journal of Business and Economics*, *3*, 83–89.

Lee, W.-C., & Wu, C.-C. (2004). Minimizing total completion time in a two-machine flowshop with a learning effect. *International Journal of Production Economics*, *88*, 85–93.

Lee, W.-C., Wu, C.-C., & Sung, H.-J. (2004). A bi-criterion single-machine scheduling problem with learning considerations. *Acta Informatica*, *40*, 303–315.

Lee, W.-C., & Wu, C.-C. (2009). Some single-machine and m-machine flowshop scheduling problems with learning considerations. *Information Sciences*, *179*, 3885–3892.

Mirsanei, H. S., Zandieh, M., Moayed, M. J., & Khabbazi, M. R. (2011). A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *22*, 965–978.

Moshieov, G. (2001). Scheduling problems with a learning effect. *European Journal of Operational Research*, *132*, 687–693.

Moshieov, G. (2001). Parallel machine scheduling with a learning effect. *Journal of Operational Research Society*, *52*, 1165–1169.

Moshieov, G., & Sidney, J. B. (2003). Scheduling with general job-dependent learning curves. *European Journal of Operational Research*, *147*, 665–670.

Moshieov, G., & Sidney, J. B. (2005). Note on scheduling with general learning curves to minimize the number of tardy jobs. *Journal of Operational Research Society*, *56*, 110–112.

Nearchou, A. C. (2004). Flow-shop sequencing using hybrid simulated annealing. *Journal of Intelligent Manufacturing*, *15*, 317–328.

Solano-Charris, E. L., Montoya-Torres, J. R., & Paternina-Arboleda, C. D. (2011). Ant colony optimization algorithm for a bi-criteria 2-stage hybrid flowshop scheduling problem. *Journal of Intelligent Manufacturing*, *22*, 815–822.

Speras, W. M. & DeJong, K. A. (1991). On the virtues of parameterized uniform crossover. In: *Proceedings of the fourth international conference genetic algorithms* (pp. 230–236).

Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, *177*, 1930–1947.

Toksari, M. D., & Güner, E. (2010). Parallel machine scheduling problem to minimize the earliness/tardiness costs with learning effect and deteriorating jobs. *Journal of Intelligent Manufacturing*, *21*, 843851.

Wang, J.-B. (2007). Single-machine scheduling problems with the effects of learning and deterioration. *Omega*, *35*, 397–402.

Wang, J.-B., & Xia, Z.-Q. (2005). Flow shop scheduling with a learning effect. *Journal of Operational Research Society*, *56*, 1325–1330.

Wang, X., & Cheng, T. C. E. (2007). Single-machine scheduling with deteriorating jobs and learning effects to minimize the makespan. *European Journal of Operational Research*, *178*, 57–70.

Wu, C.-C. (2006). The development of a solution to the single-machine total weighted completion time problem with a learning effect. *International Journal of Management*, *23*, 113–116.

Wu, C.-C., Lee, W.-C., & Wang, W.-C. (2007). A two-machine flowshop maximum tardiness scheduling problem with a learning effect. *International Journal of Advanced Manufacturing Technology*, *31*, 743–750.

Wu, W.-H., Cheng, S.-R., Wu, C.-C., & Yin, Y. (2012). Ant colony algorithms for a two-agent scheduling with sum-Of processing times-based learning and deteriorating considerations. *Journal of Intelligent Manufacturing*, *23*, 1985–1993.

Yin, Y.-Q., Xu, D.-H., Sun, K.-B., & Li, H.-X. (2009). Some scheduling problems with general position-dependent and time-dependent learning effects. *Information Sciences*, *179*, 2416–2425.

Journal: **10845-JIMS** MS: **0827** ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:**2013/8/17** Pages: **11** Layout: **Large**