A Heuristic Algorithm for the Scheduling Problem of Parallel Machines with Mold Constraints

TZUNG-PEI HONG[1], PEI-CHEN SUN[2], and SHIN-DAI LI[2]
[1]Department of Computer Science and Information Engineering
National University of Kaohsiung
Kaohsiung, 811, Taiwan, R.O.C.
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, 80424, Taiwan, R.O.C.
tphong@nuk.edu.tw
[2]Graduate Institute of Information and Computer Education
National Kaohsiung Normal University
Kaohsiung, 802, Taiwan, R.O.C.
sun@nuknucc.edu.tw, cursor@icemail.nknu.edu.tw

Abstract: - This paper addresses the scheduling problem of parallel machines with mold constraints. Each machine has to load one kind of molds to process a specific job in the production environment. Because it takes lots of time to change one mold to another on a same machine for producing jobs with different types, it will be efficient to put all similar jobs together as a batch production. This way will, however, result in the total tardiness of jobs increasing due to the different due dates of jobs. This kind of problems is an NP-hard problem. In this paper, we build a model to describe the problem and present a heuristic algorithm to solve it.

Key-Words: - Scheduling, parallel machines, mold constraints, heuristic.

1 Introduction

In this paper, we discuss a scheduling problem in which there are n jobs and m unrelated parallel machines with l molds. Each job requires a single operation which is processed on one machine equipped with one mold. A job is the minimal unit and can't be separated. Each job has its type and a fixed due date denoted by d_j. Each job's release time is current.

Each machine only can load no more than one mold at a time and a mold is only loaded on one machine at a time. If the type of the current job is different with the type of the following jobs scheduled on the same machine, it needs a setup time for changing the online mold to next mold for forming the following job. The setup time is fixed, machines- and sequence- independent, denoted by S.

Unrelated machine means that a machine may process different jobs in different speeds. For example, machine A processes job i at a low speed, but it may process job j at a high speed. On the other hand, machine B processes job i at a high speed, but it processes job j at a low speed.

If all machines have the same speed, then the environment is identical to the identical machines in parallel. Due to unrelated parallel machines, the precessing time of a job depends on which machine it works on.

2 Related research

The parallel machines' problem is divided into two groups by the type of machines: identical parallel machines and unrelated parallel machines. The problem of parallel machines with setup time is an independent subject to discuss.

For identical parallel machines with setup time, Wang et al. [1] considered that parallel machine with modulo constraints, in which the setup time is independent from last modulo but only dependent on the next modulo. He presented a heuristic algorithm based on list scheduling and then used NBR (net benefit of relocation) algorithm to adjust the sequence of jobs on each machine for minimizing total tardiness. Schutten and Leussink [2] saw setup as setup jobs with release dates, due dates and processing times. They developed a branch-and-bound algorithm to solve the problem for minimizing the maximum tardiness of any job. Lee and Pinedo [3] considered that jobs were weighted and proposed a three-phase heuristic to minimize total weighted tardiness. First, factors or statistics were computed. Second, a sequence was constructed by a dispatching rule. Third, a simulated annealing method was applied to improve the solution.

For unrelated parallel machines with setup time, several studies discuss sequence- or machine-dependent setup time.

Chen and Wub [4] developed a heuristic based on threshold-accepting methods and tabu lists to minimize total tardiness. Tamaki et al. [5] transformed the scheduling problem to a mathematical programming problem and used SA method and genetic algorithm to solve it. Besides, Kim et al. [6] considered that each job may refer to a lot composed of different items while every item within each job has an identical processing time with a common due date. They used SA to determine a scheduling policy so as to minimize total tardiness. Although scheduling of unrelated parallel machine with setup time has been studied in recent years, the most studies hypothesize that dies or molds can be used on any parallel machine in the scheduling. In this paper, we deal with the scheduling problem in which molds do not be allowed to load on any machine.

3 Problem definition

After describing the shecduling problem in section 1, we present the problem in mathematical formulas. The symbols we defined are shown as follows.

j: index of jobs (j = 1, 2, …, n),
h: index of molds (h = 1, 2, …, l),
k: index of machines (k = 1, 2, …, m).
The parameters are associated with the job j:
d_j: the due date of job j,
c_j: the complete time of job j,
p_j: the processing time of job j which is processed in the standard speed,
v_{jk}: the speed that machine k can process job j at, it is relative to the standard speed,
p_{jk}: the processing time of job j on machine k and $p_{jk} = p_j/v_{jk}$.

We take an example to explain the relation of processing time of jobs and speeds of machines. There is a job j with quantity q_j to product, and the standard speed v = 1. So the processing time of job j are $p_j = q_j/v = 6/1 = 6$. Now we have three machines 1, 2 and 3. Their processing speeds for job j are v_{j1}, v_{j2} and v_{j3}, respectively. If $v_{j1} = v_{j2} = v_{j3} = v = 1$, the processing times of job j on three machines will be $p_{j1} = p_{j2} = p_{j3} = p_j = 6$. Now we set $v_{j1} = 1$, $v_{j2} = 2$ and $v_{j3} = 3$, which means speed of machine 1 is equal to standard speed, speed of machine 2 is two times than standard speed, and speed of machine 3 is three times than standerd speed. The processing time of job j on machine 1, machine 2 and machine 3 are $p_{j1} = p_j/v_{j1} = (q_j/v)/v_{j1} = 6/(1*1) = 6$, $p_{j2} = 6/(1*2) = 3$ and $p_{j3} = 6/(1*3) = 2$, respectively.

3.1 Fit molds and fit machines

Every mold has a range of type which it can form. The forming range of each mold may be partial covered. A subset of molds which can form the required type of the job is called fit molds of the job. Due to the equipment and attributes of machines, a mold is not allowed to equip on any machine. A subset of machines which can equip the mold is called fit machines of the mold. Hence each job is restricted to a specific subset of machines. Figure 1(a) shows the relation of jobs, molds, and machines. For example, job 3 can be formed by mold 4, mold 5 and mold 6. The mold 4 can be loaded on machine 2 and 3. The mold 5 can be loaded on machine 2 and 4. The mold 6 can be loaded on machine 3 and 4. The Figure 1(b) shows the relation of this example.

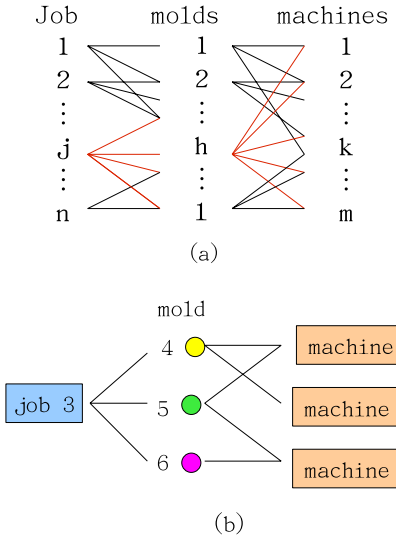

Figure 1: (a) The relation of jobs, molds, and machines. (b) The relation of this example.

We define the fit molds and fit machines by formula (1) and formula (2).

$$H_{jh} = \begin{cases} 1, \text{ if job } j \text{ can be formed by mold } h \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

$$K_{hk} = \begin{cases} 1, \text{ if mold } h \text{ can be loaded on machine } k \\ 0, \text{ otherwise} \end{cases} \quad (2)$$

For any job j and mold h, the value of H_{jh} is 1 or 0. For any mold h and machine k, the value of K_{hk} is 1 or 0. Obviously, these two definitions have to satisfy constraints of formula (3) and formula (4), respectively.

$$\sum_{h=1}^{l} H_{jh} \le l \quad (3)$$

$$\sum_{k=1}^{m} K_{hk} \le m \quad (4)$$

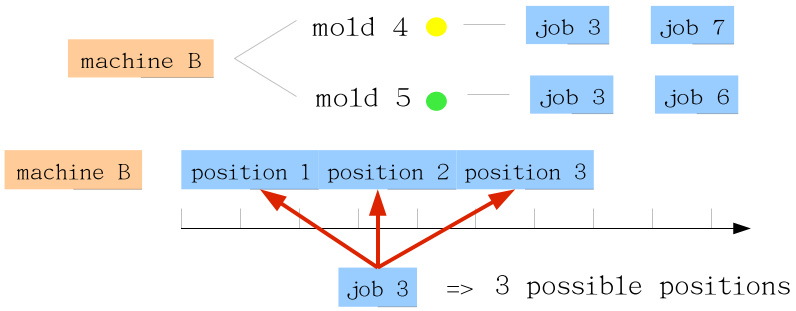

Figure 2: An example of positions.

3.2 The positions of a job in the schedule

To model the problem, we want to know that all possible positions of a job in the entire feasible schedule, which are depended on its fit molds and fit machines of its fit molds. Firstly, we should know all possible positions of a job in a fit machine. Because the amount of jobs which may be processed on a machine is the amount of possible positions of job j on that machine, we want to know how many jobs can be scheduled on a fit machine.For example, machine 1 can load mold 4 and 5. The mold 4 is one of fit molds of Job 3 and job 7, and the mold 5 is one of fit molds of Job 3 and job 6. Hence there are three possible jobs which may be scheduled on machine 1. That means job 3 has three possible positions on machine 1. See Figure 2.

We denote the amount of jobs which may be processed on machines k by n_k. For any machine k, the n_k is derivated by the formula (5).

$$\sum_{j=1}^{n} K_{1k} H_{j1} \vee K_{2k} H_{j2} \vee, \ldots, \vee K_{lk} H_{jl} = n_k \le n * l * m \tag{5}$$

3.3 The model of the problem

We define the symbol X_{jkuh} to represent the position which job j is scheduled on. For any job j, mold h, machine k, and position u of machine k, the value of X_{jkuh} is 1 or 0. The values of sequence $\{X_{jkuh}\}$ are a feasible schedule.

$$X_{jkuh} = \begin{cases} 1, & \text{if job } j \text{ is scheduled as } u-th \text{ position on machine } k \text{ equipped mold } h \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The model of the problem is represented by the complete time c_j of a job j in a feasible schedule, as shown in the formula (7).

$$c_j = \sum_{1 \le j \le n} c_{ku} X_{jkuh}, \quad h = 1, 2, \ldots, l,\ k = 1, 2, \ldots, m,\ u = 1, 2, \ldots, n_k \tag{7}$$

The symbol c_{hku} is defined in the formula (8). The symbol n_k represents the amount of jobs which may be processed on machine k.

$$c_{ku} = \begin{cases} c_{k,(u-1)} + \sum_{1 \le j \le n} p_{jk} X_{jkuh}, & \text{if } M(k, u-1) = M(k, u) \\ c_{k,(u-1)} + \sum_{1 \le j \le n} (p_{jk} + S) X_{jkuh}, & \text{otherwise} \end{cases} \tag{8}$$
$$h = 1, 2, \ldots, l,\ k = 1, 2, \ldots, m,\ u = 1, 2, \ldots, n_k$$

There are some constraints on X_{jkuh} in the feasible schedule, as shown in the formula (9) and formula (10).

$$\sum_{1 \le j \le n} X_{jhku} = 1, \quad h = 1,2,\ldots,l,\ k = 1,2,\ldots,m,\ u = 1,2,\ldots,n_K \tag{9}$$

$$\sum_{\substack{1 \le h \le l \\ 1 \le k \le m \\ 1 \le u \le n_k}} X_{jhku} = \{1, 0\},\ j = 1, 2, \ldots, n \tag{10}$$

Finally, we define symbol Y_{hkt} to represent the machine which mold h is equipped on at time t, as shown in formula (11). And the formula (12) shows the constraint of a mold in the feasible schedule.

$$Y_{hkt} = \begin{cases} 1, & \text{if mold } h \text{ is loaded on machine } k \text{ at time } t \\ 0, & \text{o.w.} \end{cases} \tag{11}$$

$$\sum_{1 \le h \le l} Y_{hkt} = \{1, 0\},\ k = 1, 2, \ldots, m,\ t = \text{any time} \tag{12}$$

4 The heuristic algorithm

In this section, we present a heuristic algorithm to solve the scheduling problem. Table 1 illustrates the algorithm briefly. We explain details of each step in the following subsections.

Table 1: Heuristic algorithm

STEP 1 Sort job set J^s by EDD (Earliest Due Date) rule.
Find the job with earliest due date in J^s, say job j.

STEP 2 Find all pairs (h, k) of job, where h is a fit mold of job j and k is a fit machine of h.

STEP 3 For each $(h, k)_r$
3.1 Find its cmg_r^j and sort each of them by EDD rule.
3.2 Find enough points and the interrupt point of the cmg_r^j.
3.3 Calculate the total tardiness or total earliness of cmg_r^j and put them into average set A^s.

STEP 4
4.1 Get the minimum A^* of A^s.
4.2 Check the mold h^* of A^* is available. If no, delete A^* from A^s and goto step 4.1.

STEP 5
5.1 Schedule cmg_r^j.
5.2 Delete all jobs of cmg_r^j from J^s.
If an interruption was occurred in step 5.1, put the interrupted jobs back to J^s.
5.3 Go to step 1 until J^s is empty.

4.1 The first step and the second step

In the first step, the job set J^s contains all jobs not scheduled yet. The EDD rule will list the job with increasing due date, and the one with the earliest due date will be listed in the most front. In the second step, the order pair (h, k)'s of job j represents all combinations of fit molds of job j and fit machines of the fit molds. They are all possible choices of job j and are indexed with r. We will examine each pair to decide which one is a better choice.

4.2 The third step

When a machine is on the setup time of the changing molds, it can not process any job. If the setup time is longer, the cost is greater. To reduce times of changing molds, the jobs which can be formed by the same mold should be gathered to process. So, for each pair $(h, k)_r$ of job j, we find its cmg_r^j (common mold group) and sort it by the EDD rule. The group collects jobs of J^s, which can be processed on the machine k equipped with the mold h. Obviously, the job j is the leading job of its groups.

If a *cmg* is put on a machine loaded with a fit mold to process, the loaded mold should be used to form enough jobs before it is unloaded for making the setup time cost-effective. So, in the step 3.2, we find the enough point of each cmg^j. We add the processing time of jobs of cmg_r^j one by one, until the accumulative processing time (*APT*) is greater than α. The last added job is called enough point of cmg_r^j. The α is a threshold parameter set by experts according to the cost-benefit analysis, which means how many jobs in cmg_r^j are processed, so that the setup time of cmg_r^j will be cost-effective.

The jobs in cmg_r^j can be processed by the same mold, but their due dates may be different greatly. If the difference of due dates of two continual jobs in a cmg_r^j is great enough, an interruption may be considered. That is, the latter job with later due date and all jobs in cmg_r^j after it can wait for the next chance to be scheduled and another *cmg* with earlier due date of its leading jog can be processed first. The jobs in cmg_r^j after the latter one have later due dates then it due to EDD sorting in the step 3.1. From the job of enough point to penultimate job of cmg_r^j, if the difference of due dates between a job and its succeeded job is greater than β, we let the front job be an interrupted point of cmg_r^j. It must be noticed that interruptions may increase times of changing molds. In different situations, the consideration will be different. The β is another parameter set by experts. The pseudocode in Table 2 illustrates how to find the enough point and interrupted points.

When we try to assign cmg_r^j of the pair $(h, k)_r$ to the machine k, one of the following three conditions will occur.

- The machine h has other scheduled *cmg* and the mold of the last scheduled *cmg*, say cmg^i is different from the mold h of the pair $(h, k)_r$.
- The machine h has other scheduled *cmg* and the mold of the last scheduled *cmg*, say cmg^i is the same with the mold h of the pair $(h, k)_r$.
- There is no other scheduled *cmg* on the machine k.

In the first condition, we have two strategies: interrupt cmg^i and succeed cmg^i.

For the interrupted strategy, we interrupt the scheduled cmgi at one of its interrupted points,

Table 2: Pseudocode of finding the enough point and interrupted points

```
input: α, β, cmg_r^j, (h, k)_r
APT = 0;
for w is an index from first job of cmg_r^j to the last job of cmg_r^j do
    APT = APT + p_wk , where k is of (h, k)_r;
    if APT ≥ α then
       let the job with index w be the enough job of cmg_r^j;
       break for;
    end
    if w is on the last job of cmg_r^j and the enough
    point is not set yet then
       let the last job be the enough point of cmg_r^j;
    end
end
for the job of enough point of cmg_r^j to the penultimate job of cmg_r^j do
    if the difference of due dates between the job and its succeeded job ≥ β then
       let the job be an interrupted point of cmg_r^j;
    end
end
```

change molds, and let cmg_r^j succeed the job of interrupted point of cmg^i to produce. Because cmg^i may have several interrupted points which are indexed with t, we use average tardiness as a criterion. For every interrupted point ip_t of cmg^i, we calculate the average tardiness of cmg^i, denoted by $irT_r^j(ip_t^i)$. We add the average tardiness of cmg^i from the job of the interrupted point to the last job and the total tardiness of cmg_r^j from the first job to the job of its enough point, then divide the sum by the total number of jobs involved in the tardiness calculation to find the average tardiness. See the formula (13).

$$irT_r^j(ip_t^i) = \frac{1}{n(ep^j) + n(ip_t^i)} \left[\sum_{a=1}^{n(ep^j)} \max(0, c_a - d_a) + \sum_{b=1}^{n(ip_t^i)} \max(0, c_b - d_b) \right] \quad (13)$$

$$irE_r^j(ip_t^i) = \frac{1}{n(ep^j) + n(ip_t^i)} \left[\sum_{a=1}^{n(ep^j)} (c_a - d_a) + \sum_{b=1}^{n(ip_t^i)} (c_b - d_b) \right] \quad (14)$$

The $n(ep^j)$ is the amount of jobs of cmg_r^j from the first job to the job of its enough point. The $n(ip_t^i)$ is the amount of jobs of cmg^i from the job of its interrupted point to its last job. The $n(cmg_r^j)$ is the amount of jobs of whole cmg_r^j.

If the average tardiness is zero, the average earliness is substituted for the average tardiness to finding better interrupted point of this strategy. The formula of average earliness is similar to the average tardiness, as shown in the formula (14).

For the not-interrupted strategy, we do not interrupt the scheduled cmg^i, but succeed it. The average tardiness still is a criterion. We calculate the average tardiness of cmg_r^j from the first job to the job of its enough point, which is denoted by $nirT_r^j$ (see formula (15)). The parameters and indexes of this symbol mean cmg_r^j do not interrupt the scheduled cmg^i. If the average tardiness is zero, average earliness $nirE_r^j$ is calculated to substitute for the average tardiness as preceding strategy (see formula (16)).

$$nirT_r^j = \frac{1}{n(ep^j)} \sum_{a=1}^{n(ep^j)} \max(0, c_a - d_a) \quad (15)$$

$$nirE_r^j = \frac{1}{n(ep^j)} \sum_{a=1}^{n(ep^j)} (c_a - d_a) \quad (16)$$

In the second condition, the mold of the pair $(h, k)_r$ is same as an online mold. It is the most reasonable strategy that cmg_r^j succeeds cmg^i to process. We also calculate the average tardiness scT_r^j of the cmg_r^j, which only is influenced by the jobs in the cmg_r^j, as shown in formula (17). If scT_r^j is zero, the average earliness scE_r^j will also replace it. The formula of scE_r^j is formula (18).

$$scT_r^j = \frac{1}{n(cmg_r^j)} \sum_{a=1}^{n(cmg_r^j)} \max(0, c_a - d_a) \quad (17)$$

$$scE_r^j = \frac{1}{n(cmg_r^j)} \sum_{a=1}^{n(cmg_r^j)} (c_a - d_a) \quad (18)$$

In the third condition, the only strategy is one that cmg_r^j is directly assigned on the machine k of the pair $(h, k)_r$, being the the first *cmg* of the machine. In reality there is the first setup time, but we leave it out to simplify problem. The formulas of average tardiness and earliness of this strategy is just like those of the second condition, but we give them different symbols to discriminate, denoted by T_r^j and E_r^j, respectively.

Once the average tardiness or earliness is calculated, we put it into the average set A^s for comparison in step 4. Table 3 illustrates the pseudocode of calculating.

Table 3: Computating average tardiness and average earliness

```
input: cmg^i, cmg_r^j, (h, k)_r
 if another cmg^i is scheduled on the machine k then
     if the mold used by cmg^i is same as mold h
     then
         foreach interrupted jobs of cmg^i do
             calculate irT and nirT ;
             if irT or nirT are zero then
                 calculate irE or nirE;
                 put irE or nirE into A^s;
             else
                 put irT or nirT into A^s;
             end
         end
     else
         calculate scT ;
         if scT is zero then
             calculate scE; put scE into A^s;
         else
             put scT into A^s;
         end
     end
 else
     calculate T ;
     if T is zero then
         calculate E; put E into A^s;
     else
         put T into A^s;
     end
end
```

4.3 The fourth step and the fifth step

We take the minimal value of the average set A^s, denoted by A^*. The A^* decides which pair and what strategy of the pair will be adopted. It means that if we adopt the strategy of A^* to schedule cmg^j on the machine and load the mold designated by the pair of A^*, denoted by $h^*$and k^*, the average tardiness will be smallest. We determine whether mold h^* is available or not at that time by checking whether mold h^* is used on another fit machine. If the mold is not available, we delete the A^* from A^s and find the minimal of A^s again.

If the mold is available, we schedule cmg_r^j on machine k^* and load mold h^* and adopt the strategy decided by A^*. Then we delete jobs of cmg_r^j from job set J^s. If an interruption is occurred in the step 5.1, we put the interrupted jobs back to J^s. The algorithm will repeat until J^s is empty.

4.4 No starvation

Because *cmg* can be interrupted, some jobs may be putted back into J^s many times. Each job has a fixed due date, so any job will be the leading job of its *cmg* during a finite time. Even there are no other jobs which have same mold with the job and the complete time of the job is not greater then α , the algorithm still set it being the enough point of the *cmg* containing the only job itself.

In the step 4.2., the procedure of checking mold will delete the choice with not available mold. Would it happen that there is no any choice after the deleting procedure? The answer is no. Assume cmg^j has only one mold h. For pair (h, k_1), the mold h is not available, because it is equipped on another fit machine k_2 at the same time. Due to the step 2.1, the pair (h, k_2) is also one choice of cmg^j and h is available certainly for this pair.

5 An example of algorithm

Here an example is used to illustrate the algorithm. There are 11 jobs, 4 unrelated parallel machines, and 6 molds. Table 4 shows the data which influence the scheduling in this example included processing time (p_{jk}), due date (d_j), fit molds of jobs (H_{jh}), fit machines of molds (K_{hk}), and speed of machines for each job (v_{ik}).

The J^s is sorted by EDD and job 1 has the earliest due date. We find all pairs of job 1 and their *cmg*s, As shown in the Figure 3.

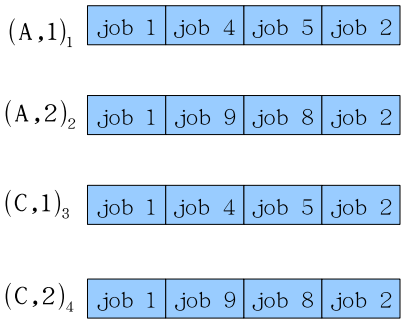

Figure 3: All pairs of job 1 and their cmgs.

Then we give $\alpha = 15$ and $\beta = 15$ to find the enough point and interrupt points. For cmg_1^1 of pair $(A, 1)_1$, the processing time of job 1 on machine A is $p_{1A} = p_1/v_{1A} = 6/1 = 6$, so the complete time of job 1 is c1 =

Table 4: Data of an example

jobs	1	2	3	4	5	6	7	8	9	10	11
processe	6	18	20	10	10	12	30	11	17	16	12
due date	8	45	27	24	26	30	35	15	10	23	44

job	1	2	3	4	5	6	7	8	9	10	11
mold 1	v	v		v	v						
mold 2	v	v						v	v		
mold 3				v	v				v	v	v
mold 4			v				v				
mold 5			v			v					
mold 6			v			v					

mold	1	2	3	4	5	6
machine A	v	v	v			
machine B				v	v	
machine C	v	v		v		
machine D					v	v

job	1	2	3	4	5	6	7	8	9	10	11
machine A	1	1	–	2	2	–	–	3	2	2	2
machine B	–	–	3	–	–	1	3	–	–	–	–
machine C	2	2	2	1	1	–	2	1	2	–	–
machine D	–	–	1	–	–	1	–	–	–	3	3

6. For α is 15, job 1 is not enough, neither is the job 4. Until to job 5, the complete time of job 5 is $c_5 = 16 > 15$, so job 5 is the enough point of cmg_1^1. The complete time is the accumulative processing time. Besides, the difference of due dates of job 5 and job 2 is 19 which is greater than β (=15). We set job 5 be a job of in interrupted point of cmg_1^1. The illustration is shown in Figure 4.

Because there is no other *cmg* on machine *A*, the strategy we take is scheduling cmg_1^1 on it directly. Then the average tardiness of the strategy is $T_1^1 = 1/4$ [max (0, 6−8) + max (0, 11−24) + max (0, 16−26) + max (0, 34−45)] = 0. Since the value is zero, we calculate the average earliness E_1^1 of cmg_1^1 to substitute T_1^1, which is 1/4[(6 − 8) + (11 −24) + (16 −26) + (34 − 45)] = −25 and put it into average set A^s.

As the same procedure, we calculate the average tardiness of cmg_2^1 of $(A, 2)_2$, average earliness of cmg_3^1 of $(C, 1)_3$, and average tardiness of cmg_4^1 of $(C, 2)_4$, which are $T_2^1 = 1.9$, $E_3^1 = -4.25$, and $T_4^1 = 2.25$, respectively. The average set A^s is {−25, 1.9, −4.25, 2.25} and the minimum A^* is $E_1^1 = -25$, which means the pair $(A, 1)$ is the better choice and the cmg_1^1 can be scheduled directly on machine *A*. For this is the first run of the algorithm, mold 1 should be available. We schedule the cmg_1^1 which contains job 1, job 4, job 5 and job 2 on machine *A* equipped with mold 1 and delete those jobs form J^s.

In the next run, the job with earliest due date is job 9 and its all pairs are $(A, 2)_1$, $(A, 3)_2$, and $(C, 2)_3$. See Figure 5(a).

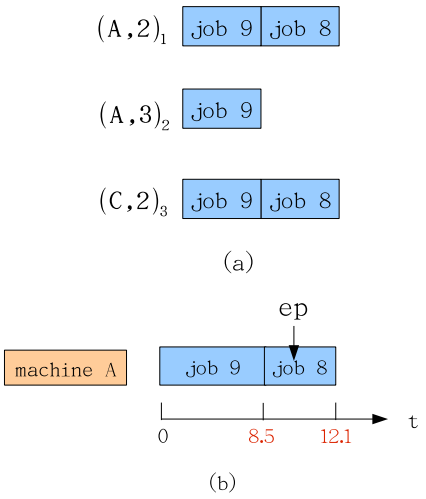

Figure 5: (a) All pairs and *cmg*s of job 9. (b) The enough point of cmg_1^9.

For the cmg_1^9 of $(A, 2)_1$, we try it on machine *A* to find its enough point and interrupted points, as shown in Figure 5(b). Obviously, even for the last job of cmg_1^9, that is job 8, the complete time is smaller than 15. In this situation, we let the last job be the enough point and the *cmg* has no interrupted point.

Because there is a scheduled *cmg* on machine *A*, we have two strategies: interrupt or not-interrupt. The

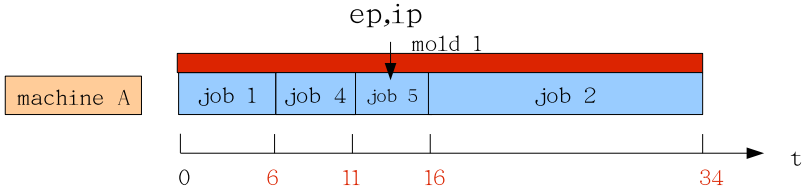

Figure 4: The enough point and interrupt points of cmg_1^1.

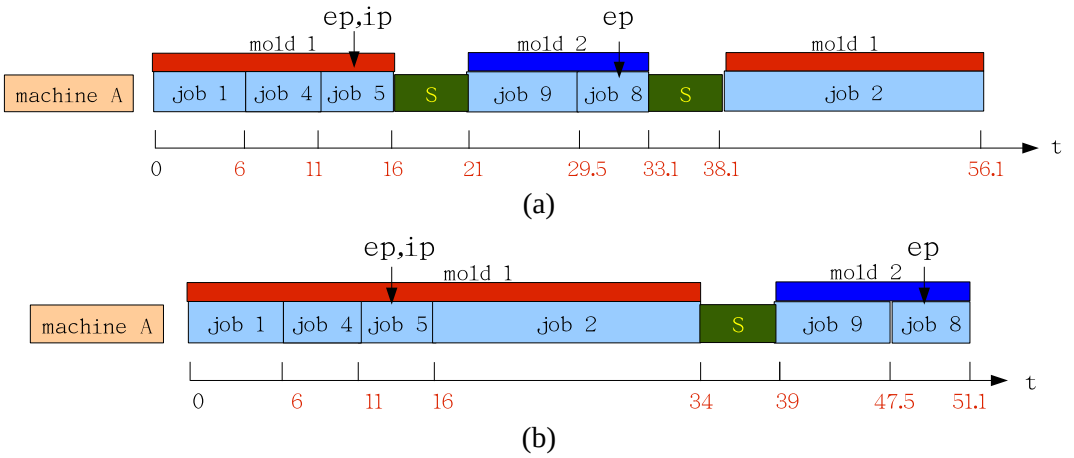

Figure 6: Two strategies of cmg_1^9: (a) interrupt the cmg_1^1. (b) not interrupt the cmg_1^1.

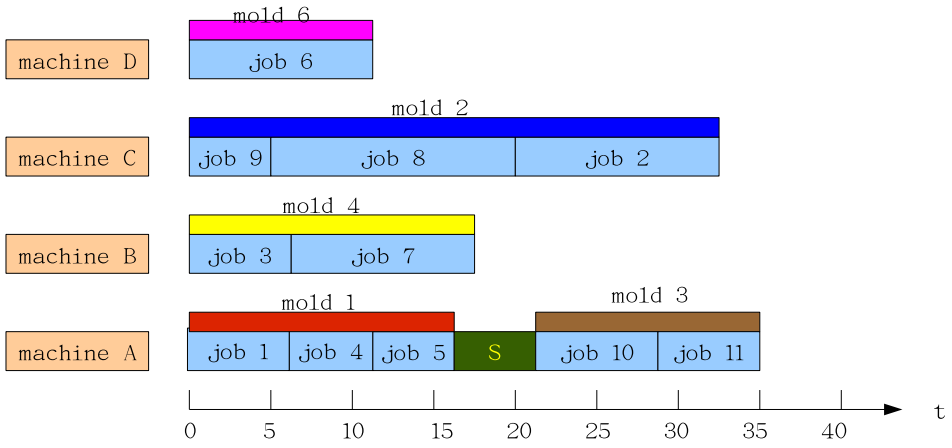

Figure 7: the outcome of the algorithm for this example.

average tardiness of interrupted strategy is irT_1^9 = 16.23 and the average tardiness of not-interrupted strategy is $nirT_1^9$= 36.8. Figure 6 illustrates those two strategies.

For the $(A, 3)_2$, and $(C, 2)_3$, we also calculate their average tardiness by the same procedure. Finally average set A^s is {16.23, 36.8, 13.5, 37.5, 2.25} and the minimum of A^s is T_3^9 = 2.25. The pair $(C, 2)_3$ is chosen, and cmg_3^9 can be scheduled on machine C directly. Since mold 2 is available, the schedule of cmg_3^9 is done and the jobs of cmg_3^9 are deleted from job set J^s. The rest runs of the algorithm are similar. Figure 7 shows the outcome of the algorithm for this example.

6 Experiments

In reality, this is the scheduling problem of a steel tube production company. A steel tube is formed from a steel sheet by a forming process on a tube forming machine. The company has 10 parallel unrelated tube forming machines and 206 molds. A job we called in this paper is a work order of steel tubes. Since this problem is mapping from a realistic problem, we compare the efficiency of heuristic algorithm with that of manual processing.

We simulate the reality production environment by speeds of 10 unrelated parallel machines for jobs (v_{jk}), the data of fit molds of jobs (H_{jh}) and fit machines of molds (K_{hk}). And we simulate the job set of the day before the first traced day which have 332 not processed jobs. The traced days are days on which the traced jobs import into the job set of scheduling system. We take 95 jobs which are imported jobs of five continual work days as evaluated data. We trace the evaluated data in the scheduling process.

Because a work day has only eight work hours, traced jobs which even are schedeuled may not be

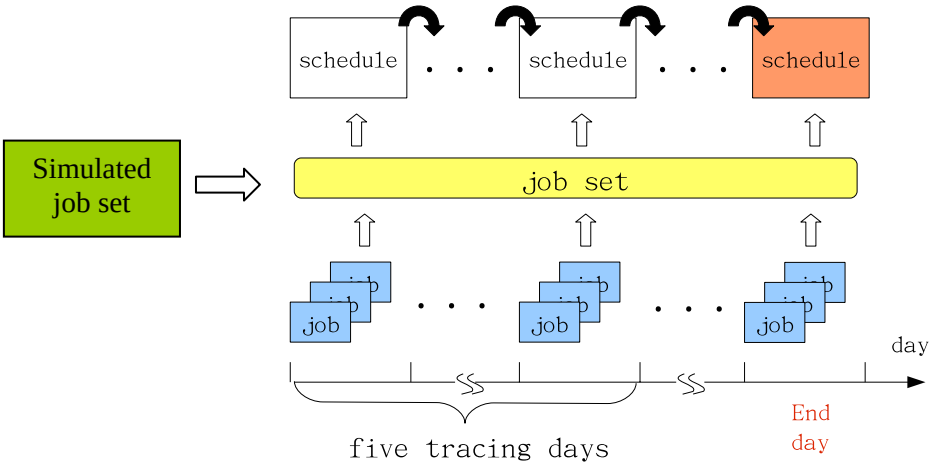

Figure 8: The design of experiments.

processed in a day. Moreover new jobs import into job set day by day, those jobs will influence the schedule of traced jobs. Those jobs are called extend jobs. In this scheduling process of this experiment, there are 578 extend jobs from the first traced day to the end day on which all traced jobs are be processed. The design of experiments is illustrated in Figure 8.

Accodrding to the scheduling export of the company, the setup time of changing mald cost about 3 hours for once. So a mold should be on a fit machine at least 2 days, which means α is 16 (hours) for eight work hours per day. Moreover, if there is an interrupted is need, it takes helf day (3 hours of 8 hours per day) to change mold, and at least 2 day to process new cmg, and then another helf day to change previous mold back. The total needed time is 3 day, so the β is set by 5 day (40 hours) for theoretical safety. But in reality, there are many unpredictable factors for changing mold, the scheduling export usually avoid unexpected changing mold. Hance we are suggested to set the condition of interruption more strictly. We take β by 5 days, 7 day and 10 day to experiment respectively.

We compare the total tardiness, the number of tardy jobs and the average tardiness of evaluated jobs scheduled by the scheduling system with those scheduled by manual scheduling. The results show that the proposed approach can greatly impove the scheduling results.

7. Conclusions and future work

The purpose of this paper is to solve a reality scheduling problem. The problem is modeled in mathematical expressions. We provide a heuristic algorithm as a solution and design an experiment to compare with the manual. There are many aspects not taken into account in this solution like material supplication. We will continue to extend the scheduling problem to solve the reality problem comprehensively in the future.

References:

[1]Cheng-Yao Wang, Lin Gao, Ding-Wei Wang, Zhi-Song Yin, Shu-Ning Wang. Minimize Total Tardiness of the Parallel Machine with Modulo Constraint. *Journal of Systems Engineering*, Vol.14, No. 4, 1999, pp. 345-350.

[2]J.M.J. Schutten, R.A.M. Leussink. Parallel Machine Scheduling with Release dates, Due dates and Family Setup Times. *International Journal of Production Economics*, Vol. 46-47, 1996, pp. 119-125.

[3]Young-Hoon Lee, Michael Pinedo. Scheduling Jobs on Parallel Machines with Sequence-dependent Setup Times. *European Journal of Operational Research* Vol. 100, 1997, pp. 464-474.

[4]Jeng-Fung Chen, Tai-His Wub. Total Tardiness Minimization on Unrelated Parallel Machine Scheduling with Auxiliary Equipment Constraints. *Omega*, Vol. 34, 2006, pp. 81-89

[5]Hisashi Tamaki, Yoshishige asegawa, Junji Kozasa, Mituhiko Araki. Application of Search Methods to Scheduling Problem in Plastics Forming Plant: A Binary Representation Approach. *The 32nd IEEE Conference on Decision and Control*, 1993, pp. 45-50.

[6]Dong-Won Kim, Kyong-Hee Kim, Wooseung Jang, F. Frank Chen. Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing. *Robotics and Computer-Integrated Manufacturing*, Vol. 18, 2002, pp. 223-231.