Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 167073, 8 pages
http://dx.doi.org/10.1155/2014/167073

Research Article

Clustering and Genetic Algorithm Based Hybrid Flowshop Scheduling with Multiple Operations

Yingfeng Zhang, Sichao Liu, and Shudong Sun

Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi'an 710072, China

Correspondence should be addressed to Yingfeng Zhang; zhangyf@nwpu.edu.cn

Received 6 January 2014; Accepted 19 February 2014; Published 27 March 2014

Academic Editor: Gongnan Xie

This research is motivated by a flowshop scheduling problem of our collaborative manufacturing company for aeronautic products. The heat-treatment stage (HTS) and precision forging stage (PFS) of the case are selected as a two-stage hybrid flowshop system. In HTS, there are four parallel machines and each machine can process a batch of jobs simultaneously. In PFS, there are two machines. Each machine can install any module of the four modules for processing the workpeices with different sizes. The problem is characterized by many constraints, such as batching operation, blocking environment, and setup time and working time limitations of modules, and so forth. In order to deal with the above special characteristics, the clustering and genetic algorithm is used to calculate the good solution for the two-stage hybrid flowshop problem. The clustering is used to group the jobs according to the processing ranges of the different modules of PFS. The genetic algorithm is used to schedule the optimal sequence of the grouped jobs for the HTS and PFS. Finally, a case study is used to demonstrate the efficiency and effectiveness of the designed genetic algorithm.

1. Introduction

This research is motivated by a flowshop scheduling problem of our collaborative manufacturing company for aeronautic product. This company is composed of a set of multiple stages with each stage having parallel machines. According to our investigation, precision forging stage is the bottleneck because it has many WIP (work in progress) stock at different times. Therefore, we focus mainly on the scheduling method of this stage and its previous stage (heat-treatment).

The heat-treatment stage (HTS) combined with the precision forging stage (PFS) forms a two-stage hybrid flowshop system. Figure 1 describes the details of these two stages. As seen in Figure 1, the HTS has four parallel machines and the PFS has two parallel machines. The HTS can process five workpieces at one machine simultaneously, while the PFS can process one workpiece at one time. In PFS, four modules are used to process the different workpieces with different sizes, and the setup time and working time of each module have an effect on the efficiency of PFS. As a result, in PFS of collaborative company, the phenomenon that most of the modules' working time deviate their nominal working time often occurs. In addition, there are many WIP blocked for PFS because the multiple workpieces with different sizes can be simultaneously processed in each machine of HTS.

In contrast to general hybrid flowshop scheduling, some characteristics which emerged in this industrial case have substantiated the complexity of the problem. It is summarized as follows.

(1) In HTS, there are four parallel machines. Each machine can process a batch of jobs simultaneously. The method of loading job into these machines is unusual. It follows the principle that a new job can be loaded when one job is released. For each machine, only all the jobs which are loaded into the machine can be processed. It means that the start time of the batch of jobs is the loading time of the last loaded job of the machine. In this case, the capacity of each machine is five.

(2) In PFS of this case, there are two machines. Each machine can install any module of the four modules for processing the workpieces with different sizes, and the setup time and working time of each module should be considered.

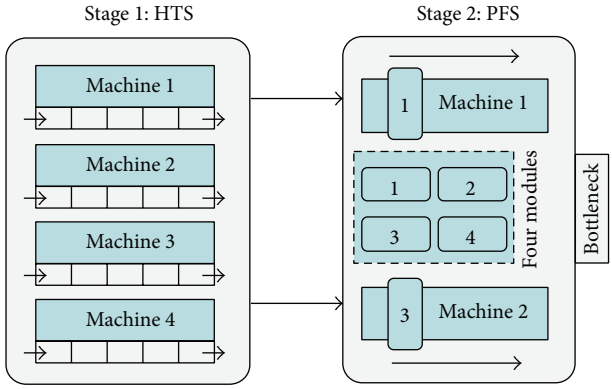

Figure 1: Two-stage hybrid flowshop scheduling within collaborating company.

Each module has a processing range and can be reused. The workpieces with different sizes can be dealt with under the processing range of the module. But for each module, it should not be continuously used exceeding its working time at one time. Therefore, it is better to make all the modules could be used near to their working time at each time and try to avoid the setup of modules at the two machines simultaneously.

(3) In a general flowshop problem, buffers can be used to store jobs for waiting for the next process. However, in this case, the released jobs from HTS should be immediately transported to PFS because the workpieces will be affected with a large temperature difference. If PFS is not ready for a new job, the jobs must be blocked inside the machines although it is finished.

In order to deal with the above special characteristics, in this paper, the clustering and genetic algorithm is designed to calculate the good solution for this two-stage hybrid flowshop problem. The clustering is used to group the jobs according to the processing ranges of the different modules of PFS. The genetic algorithm is used to schedule the optimal sequence of the grouped jobs for the HTS and PFS. The research will improve the productivity for the manufacturing companies with these two-stage hybrid flowshop problems.

The rest of this paper is organized as follows. Section 2 reviews the relevant literature. The mathematical model of the two-stage hybrid flowshop problem is established in Section 3. Section 4 presents analysis on the clustering algorithm of jobs. A case study which is used to demonstrate the efficiency and effectiveness of the designed genetic algorithm is proposed in Section 5. Finally, conclusions and recommendations for future work are summarized in Section 6.

2. Literature Review

Early researches majorly focused on mathematical methodologies and solution algorithms. Rahendran and Chaudhuri [1] propose a multistage parallel-processor flowshop problem with minimum flowtime. Based a hybrid three-stage flowshop problem, Riane et al. [2] propose an efficient heuristics to minimize makespan and branch and bound crossed with GA to solve hybrid flowshops. According to the literature about hybrid flow shop scheduling problems, Linn and Zhang [3] make a comprehensive survey. With the rapid development of information technology, computer technology and integrated manufacturing system, multiprocessor task scheduling has come into focus. Oğuz et al. [4] propose the concept of hybrid flowshop scheduling problems with multiprocessor task systems; for multiprocessor task scheduling in multistage hybrid flowshops, Ying and Lin [5] present an ant colony system approach. Alaykýran et al. [6] make in-depth research under the ant colony system approach and use an optimal ant colony algorithm to solve hybrid flow shop scheduling problems. Luo et al. [7] investigate a two-stage hybrid flowshop scheduling problem in a metal-working company and develop a genetic algorithm to obtain a near-optimal solution.

The optimal algorithm for hybrid flowshop scheduling is always a difficulty. Taking into account complexity and efficiency of algorithm, many methods have been applied in the solution of hybrid flowshop scheduling problems. At early phase, Maccarthy and Liu [8], addressing the gap in scheduling research, made a review of optimization of heuristic methods in production scheduling. Chen [9] describes analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. Wang and Zheng [10] present an effective hybrid heuristics for flow shop scheduling. Laha and Chakraborty [11] analyze the optimal methods based on the construction search. According to the real life, Rossi et al. [12] apply heuristics for scheduling a two-stage hybrid flowshop with parallel batching machines at a hospital sterilisation plant.

Two scheduling situations are named no-wait and blocking scheduling when there are no buffer storages between stages. In no-wait scheduling (NWS), jobs must be processed from start to finish without any interruption either on or between machines. In blocking scheduling (BS), a job completed at a machine must wait there until the next machine gets ready. These problems are usual in various industries such as smart factory, high technology industry, and chemical processing industry and have taken substantial research attention. Cheng et al. [13] present a genetic algorithm for the multistage and parallel-machine scheduling problem with job splitting-A case study for the solar cell industry. Sirskandarajah [14] describes the performance of scheduling algorithms for no-wait flowshop with parallel machines. Caraffa et al. [15] use the genetic algorithm to search the minimum makespan in a blocking flowshop. Ruiz and Allahverdi [16] study no-wait flowshop with separate setup times to minimize maximum lateness. Liu et al. [17] propose an effective hybrid particle swarm optimization for no-wait flowshop scheduling; this hybrid particle swarm optimization has been verified, that is, a high-efficiency method. Combining with the blocking and no-wait scheduling together, Mascis and Pacciarelli [18] discuss the job-shop scheduling with blocking and no-wait constraints; survey indicates that the blocking scheduling and no-wait are paradoxical things. Luo et al. [19] study a hybrid flowshop scheduling with batch-discrete processors and machine maintenance in time windows.

Although significant progress has been made in above researches, it is difficult to directly apply the relevant model and method to this case because of the characteristics described in Section 1. Considering the problems faced in this case, the clustering model and genetic algorithm will be combined to solve this problem.

3. Mathematical Model

According to the two-stage (HTS and PFS) hybrid flowshop scheduling problem described in Section 1, a mathematical model should be constructed in order to conduct further studies.

Notations used through this paper are listed in notations section.

Based on the notations, the mathematical scheduling model is formulated as follows:

objective function

$$E = \operatorname{Max}\left(\beta_1 \times \eta_1 + \beta_2 \times \eta_2\right), \quad \beta_2 > \beta_1, \quad \beta_1 + \beta_2 = 1, \tag{1}$$

where,

$$\eta_1 = \frac{\sum_1^m \sum_1^i P_i^m}{\sum_1^m \sum_1^i P_i^m + \sum_1^m \sum_1^i B_i^m}, \qquad \eta_2 = \frac{\sum_k \sum_i \left(\overset{k}{\omega}_{i,\alpha} + \overset{k}{\theta}_{i,\alpha}\right)}{\sum_k \sum_i \left(\overset{k}{\omega}_{i,\alpha} + \overset{k}{\theta}_{i,\alpha}\right) + \sum_n R^j}. \tag{2}$$

As seen in (1), E is the efficiency of the two-stage hybrid flowshop scheduling problem. η_1 and η_2 represent the efficiency of HTS and PFS, respectively. β_1 and β_2 are the weighting factors of HTS and FPS; we set $\beta_1 = 0.1$ and $\beta_2 = 0.9$ because the FPS is the bottleneck. In η_1, $\sum_1^m \sum_1^i P_i^m$ is the total processing time of GJ and $\sum_1^m \sum_1^i B_i^m$ is the total blocking of the jobs in the HTS. In η_2, $\sum_k \sum_i (\overset{k}{\omega}_{i,\alpha} + \overset{k}{\theta}_{i,\alpha})$ is the total processing time of the two machines, $\sum_n R^j$ is total waiting time in the PFS.

Constraints of this problem are

$$P^m = \max_i \left\{P_i^m\right\}, \qquad B^m = \min_i \left\{B_i^m\right\}, \tag{3}$$

$$F_i^{gj} - S_i^{gj} \geq P^m, \quad \text{if } B^m = 0, \text{ inequality is } " = ", \tag{4}$$

$$S_{i+1,\alpha}^{\lambda} - F_{i,\alpha}^{\lambda} \geq^k \theta_{i,\alpha}, \tag{5}$$

$$\sum_{\lambda} V_{i,\alpha}^{\lambda} \leq \overset{k}{\omega}_{i,\alpha}, \tag{6}$$

where functions (3) and (4) are the constraints of HTS, where P^m is the processing time constraint of each gj^m, B^m is the blocking time constraint of each gj^m. Functions (5) and (6) are the constraints of FPS; they are used to guarantee that each module has enough setup time and that each module does not work exceeding its nominal working time at each time of use.

4. Clustering and GA-Based Scheduling Method

4.1. Clustering Model. The clustering model is used to classify all jobs into multiple groups of size "c" according to the similarity of job parameters. In order to make the grouped jobs more significant for the next step—GA design, it is essential to analyze the importance of the parameters of the jobs.

Two important parameters should be considered. The first is the size of the workpieces. The second is the processing time of the job in HTS.

Let set $V_s = \{v_1, v_2, \ldots, v_m\}$ be the different values of size range according to the modules processing range of PFS.

Let set $V_{\mathrm{PT}} = \{v_1, v_2, \ldots, v_n\}$ be the different values of processing time of all the jobs.

For each size of V_s, each different value is a split node. The jobs firstly are classed into several groups at each value of v_i. Then, these grouped jobs are further grouped according to the value of V_{PT}.

For the different processing time of each job, a cluster model is designed as follows.

Step 1. Let $G = \{g_1, g_2, \ldots, g_g\}$ records each group (g_i) after the jobs are classed by all the values $V_s = \{v_1, v_2, \ldots, v_m\}$, respectively. For each g_i, $g_i \in G$, it means there are "g" (the size of g_i) jobs in this group with the different processing time parameters.

Step 2. Use a dynamic set GJ^i to record the result of the subgroups of g_i; the size of GJ^i is zero at the beginning.

Step 2.1. Reclass the jobs of g_i using processing time parameters $V_{\mathrm{PT}} = \{v_1, v_2, \ldots, v_n\}$ by means of the same method of size parameter in V_s.

Step 2.2. Let $TG^i = \{tg_1^i, tg_2^i, \ldots, tg_t^i\}$ be a temp array to record each new group (tg_t^i) of group g_i after being reclassed by different values of processing time $\{gp_1 \ldots, gp_g\}$. For each tg_t^i, rules should be obeyed to regroup the size of tg_t^i. The rules can be seen in Table 1. Here, s is the size of tg_t^i and c is a constant value of jobs that need to be simultaneously processed in the same machine.

After the above steps, GJ^i consisted of many small groups and each group consists of "c" jobs according to the size and processing time parameters. Then, the formed GJ^i is ready for GA-based scheduling.

4.2. GA-Based Scheduling Method. The flows of designed GA-based scheduling method follow the standard flows of genetic algorithm. It includes the following main parts.

Table 1: The rules of regroup.

Condition	Regroup rules
$s \le c$	The tg_t^i is dynamically added to GJ^i
$s > c$	Use the operator of division to divide the size of tg_t^i by c, and let a be the aliquot part and b the remainder. If b is equal to zero, the tg_t^i will be decomposed into "a" groups and each group has "c" jobs. If b is not equal to zero, the tg_t^i will be decomposed into "a" groups and another group consists of "b" jobs. The groups created during this process are also dynamically added to GJ^i.

Table 2: Parameters of the modules.

Module type	Processing range (cm)	Notational processing time (min)	Setup time (min)
Module 1	(30, 40)	300	20
Module 2	(40, 50)	280	20
Module 3	(50, 60)	270	40
Module 4	(60, 70)	260	40

4.2.1. Gene and Chromosome. As described in Section 4.1, the "n" jobs have been grouped with the same size $c$according to the size and processing time parameters. Therefore, the matrix $GJ = \{gj^1, gj^i, \ldots, gj^m\}$ will be the important information to design genes.

An integer-based method is used to form the genes and chromosomes. The genes consist of a pair of integers like (i-j), $1 \le i \le m$, $1 \le j \le 4$; here m is the length of the set GJ and j is the corresponding module of PFS. Obviously, each gene includes a group with "c" jobs formed in clustering stage and the corresponding module used in PFS.

Based on genes, the chromosome can be defined as a queue that consists of different genes and char ":". The length of the chromosome is the sum of the different genes. And it is decided by the length of set GJ. For example, chromosome (1-2:3-3:2-1:5-4:6-1) consists of five genes and its length is five.

4.2.2. Decoding Scheme. The decoding scheme is also designed according to the gene and chromosome. And the purpose is to interpret the meaning of the chromosome. The char ":" which appeared in chromosome is to connect the genes, and the integer (i-j) which appeared in chromosome presents a group (gj^i) of jobs which is defined in GJ. The appearing order of i of (i-j) presents the processing order of each group (gj^i) and will use the module "j" in the next stage. For example, the chromosome (1-2:3-3:2-1:5-4:6-1) means that there are five groups ($\{gj^1, gj^3, gj^2, gj^5, gj^6\}$) of jobs which will be processed by $\{m_2, m_3, m_1, m_4, m_1\}$.

4.2.3. Fitness Function. Fitness function is used to calculate the value of each individual in order to determine the performance of chromosomes of the entire population. In this case, the objective function (1) is used to be the fitness function to evaluate the chromosome.

4.2.4. Design of Selection, Crossover, and Mutation Operators. Selection operator is used to select the optimal parents according to the fitness for generating new offspring.

Crossover operator is used as main genetic operator and the performance depends on it. In this case, a multipoint crossover operator is designed to permute the job order as well as keep the legality of the generated chromosome.

The proposed crossover operator takes two parents and creates two offsprings. It propagates the structure and subschedule into offspring from one parent. And then it completes the offspring with remaining jobs derived from another parent. The detailed steps can be seen in Algorithm 1.

Mutation operator just works on a single chromosome and generates offspring by altering one or more genes.

In this research, mutation uses the replaced genes derived from the same chromosome. It is widely applied to general genetic algorithms. Compared to crossover operator, it may be simple. Firstly, choose one chromosome for mutation. Then, two random positions of the chosen chromosome are selected and then replace their genes. For example, assume the parent is (1-2:3-3:2-1:5-4:6-1:8-2:7-3:10-4); two mutation positions (i.e., 2, 6) are chosen randomly. Then replace the genes (i.e., 3-3, 8-2) in selected positions and get the offspring, namely, (1-2:8-2:2-1:5-4:6-1:3-3:7-3:10-4).

5. Case Study

Two hundred jobs are selected to simulate the problems of the collaborative company and verify the effectiveness of the designed clustering and GA method. Table 2 shows the parameters of the four models. Table 3 shows the information of the jobs (j^n), where the columns size and processing time are the initial data of the jobs, and the blocking time and waiting time are the calculated data according to the optimal solution gotten by the designed algorithm.

Table 4 shows the information of grouped jobs in HTS, where size range and processing time are the values of each group jobs (gj^m), blocking time is the minimum blocking time of all the jobs in this group, and η_1 means the efficiency of HTS; it is calculated by

$$\eta_1 = \frac{\sum_1^m \sum_1^i P_i^m}{\sum_1^m \sum_1^i P_i^m + \sum_1^m \sum_1^i B_i^m}. \quad (7)$$

Table 5 shows the information of grouped jobs in PFS, where size range and processing time are the values of each group jobs (gj^m); waiting time is actual waiting time of all the jobs in this group. Table 6 shows the scheduling result of the four modules in PFS, where $\overset{k}{\omega}_{i,\alpha}$ is working time of Model α in machine (k), $\overset{k}{\theta}_{i,\alpha}$ is the setup time of Model α in machine (k), i means the ith loading, and R^j is wait time of the machine j, which is equal the sum of waiting time of jobs in this machine

Table 3: Data of jobs.

j^n	Size (cm)	Processing time (min)	Blocking time (min)	Waiting time (min)
j^1	31	39	0	0
j^2	63	21	9	7
j^3	69	15	7	6
⋮	⋮	⋮	⋮	⋮
j^{50}	40	35	3	1
⋮	⋮	⋮	⋮	⋮
j^{100}	51	29	5	7
⋮	⋮	⋮	⋮	⋮
j^{150}	67	17	4	3
⋮	⋮	⋮	⋮	⋮
j^{199}	37	33	4	2
j^{200}	42	37	4	1

Step 1. Create two crossover points to form crossover-section.
Randomly select two parents ***i*** and ***j***.
Randomly create two integrate ***a*** and ***b***, $a < b; a, b \in [1, m]$
The genes from ***a*** to ***b*** of parents ***i*** and ***j*** are the crossover-sections.
String [] ***tempI*** = The genes from the ***a***th to ***b***th of the parent "***i***"
String [] ***tempJ*** = The genes from the ***a***th to ***b***th of the parent "***j***"
Step 2. Use symbol "0" to replace the genes.
Find the genes of ***tempI*** in parent ***"j"***
Use "***0***" to replace the genes of the parent ***"i***"
String [] ***tempI0*** = The new parent "***i***" with "***0***"
Find the genes of ***tempJ*** in parent "***i***"
Use "***0***" to replace the genes of the parent ***"j***"
String [] ***tempJ0*** = The new parent ***"j***" with "***0***"
Step 3. Make "0" move to the crossover-section
FOR each new parent (*tempI0*; *tempJ0*) DO
Make the symbols "***0***" move from the both extremities to centre until they reach the cross-section, namely from ***a*** to ***b***
END FOR
String [] ***tempI1*** = The new parent "***i***" with centre "***0***" from ***a*** to ***b***
String [] ***tempJ1*** = The new parent "***j***" with centre "***0***" from ***a*** to ***b***
Step 4. Create new parents
Use tempJ to replace of the "0" from *a* to *b* of temI1
String [] ***parentI*** = new formed ***temI1***
Use tempI to replace of the "0" from *a* to *b* of temJ1
String [] ***parentJ*** = new formed ***temJ1***
Outputs: String [] ***parentI; parentJ***

Algorithm 1: Steps of crossover operator with multi-point.

minus the setup time of this machine. The efficiency of PFS is calculated by

$$\eta_2 = \frac{\sum_k \sum_i \left(\overset{k}{\omega}_{i,\alpha} + \overset{k}{\theta}_{i,\alpha} \right)}{\sum_k \sum_i \left(\overset{k}{\omega}_{i,\alpha} + \overset{k}{\theta}_{i,\alpha} \right) + \sum_n R^j}. \tag{8}$$

The result shows that the two stages have good efficiency. The efficiency of HTS is 88.31% and the efficiency of PFS is 95.34%. The efficiency of PFS is high, and each module can be used near to its notational processing time. Therefore, it is a conclusion that the computational solution is a suitable solution for this scheduling problem.

6. Conclusion

This paper introduces a clustering and genetic algorithm based method to solve the scheduling problem of a two-stage, HTS and PFS, hybrid flowshop problem. This problem

Table 4: Data of grouped jobs in HTS.

GJ	Range of sizes (cm)	Processing time (min)	Blocking time (min)	η_1
gj^1	30-31	40	0	
gj^2	31-32	39	0	
gj^3	32-33	38	2	
gj^4	33-34	37	4	
gj^5	34-35	36	2	
gj^6	35-36	35	3	
gj^7	36-37	34	4	
gj^8	37-38	33	4	
gj^9	38-39	32	5	
gj^{10}	39-40	31	6	
gj^{11}	40-41	35	3	
gj^{12}	41-42	34	5	
gj^{13}	42-43	33	4	
gj^{14}	43-44	32	5	
gj^{15}	44-45	31	1	
gj^{16}	45-46	30	2	
gj^{17}	46-47	29	3	
gj^{18}	47-48	28	6	
gj^{19}	48-49	27	7	
gj^{20}	49-50	26	8	88.31%
gj^{21}	50-51	30	5	
gj^{22}	51-52	29	6	
gj^{23}	52-53	28	3	
gj^{24}	53-54	27	4	
gj^{25}	54-55	26	2	
gj^{26}	55-56	25	3	
gj^{27}	56-57	24	4	
gj^{28}	57-58	23	0	
gj^{29}	58-59	22	0	
gj^{30}	59-60	21	0	
gj^{31}	60-61	24	3	
gj^{32}	61-62	23	4	
gj^{33}	62-63	22	8	
gj^{34}	63-64	21	9	
gj^{35}	64-65	20	7	
gj^{36}	65-66	19	2	
gj^{37}	66-67	18	3	
gj^{38}	67-68	17	4	
gj^{39}	68-69	16	5	
gj^{40}	69-70	15	7	

Table 5: Data of grouped jobs in PFS.

GJ	Range of sizes (cm)	Processing time (min)	Waiting time (min)
gj^1	30-31	30	0
gj^2	31-32	27	0
gj^3	32-33	25	3
gj^4	33-34	23	4
gj^5	34-35	21	1
gj^6	35-36	19	2
gj^7	36-37	17	2
gj^8	37-38	16	2
gj^9	38-39	15	3
gj^{10}	39-40	13	5
gj^{11}	40-41	29	1
gj^{12}	41-42	26	3
gj^{13}	42-43	24	1
gj^{14}	43-44	22	3
gj^{15}	44-45	20	0
gj^{16}	45-46	18	0
gj^{17}	46-47	16	0
gj^{18}	47-48	15	2
gj^{19}	48-49	14	4
gj^{20}	49-50	12	5
gj^{21}	50-51	28	6
gj^{22}	51-52	26	7
gj^{23}	52-53	23	5
gj^{24}	53-54	21	6
gj^{25}	54-55	19	2
gj^{26}	55-56	17	3
gj^{27}	56-57	16	4
gj^{28}	57-58	15	0
gj^{29}	58-59	13	0
gj^{30}	59-60	11	0
gj^{31}	60-61	27	4
gj^{32}	61-62	25	6
gj^{33}	62-63	23	5
gj^{34}	63-64	21	7
gj^{35}	64-65	19	0
gj^{36}	65-66	17	0
gj^{37}	66-67	16	0
gj^{38}	67-68	14	3
gj^{39}	68-69	12	4
gj^{40}	69-70	11	6

is characterized by many constraints, such as batching operation, blocking environment, and setup time and working time limitations of modules. The clustering is used to group the jobs according to the processing ranges of the different modules of PFS. The genetic algorithm is used to schedule the optimal sequence of the grouped jobs for HTS and PFS. The designed algorithm is proven useful through a case study. Several advantages such as faster processing time, better utilization, and higher efficiency of HTS and PFS come out with this method.

Currently, we only use the data simulation to verify the designed clustering and genetic algorithm. According to

Table 6: Scheduling result of modules in PFS.

	$\omega_{1,\alpha}^{1}$	$\omega_{2,\alpha}^{1}$	$\omega_{1,\alpha}^{2}$	$\omega_{2,\alpha}^{2}$	$\theta_{1,\alpha}^{1}$	$\theta_{2,\alpha}^{1}$	$\theta_{1,\alpha}^{2}$	$\theta_{2,\alpha}^{2}$	R^1	R^2	η_2
1	285	240	285	220	20	20	20	0	95	115	95.34%
2	275	275	230	205	20	20	20	20			
3	190	270	260	220	40	0	40	40			
4	185	220	260	260	40	40	40	40			

the further requirements of the collaborative company, this method should be improved for application. Furthermore, more practical experiences such as line balance of PFS will be considered and added to mathematical model. This will help to optimize the GA model and help to achieve better scheduling results for real-life manufacturing companies.

Notations

n: The total number of jobs

$J = \{j^1, j^2, \ldots, j^n\}$: The set of n jobs

$\alpha = \{1, 2, 3, 4\}$: α represents the type of the modules in PFS; there are four types of setup time of module α

W_N^{α}: Nominal working time of module α at each time of use

M: The number of grouped jobs after clustering

$GJ = \{gj^1, gj^2, \ldots, gj^m\}$: The set of m grouped jobs; each gj^m has five jobs

ω_{ij}: The jth in group $gj(gj^m)$ of the HTS

$\delta = \{\delta^1, \delta^2\}$: Represents the number of machines of the two stages

P_i^m: The processing time of each job (j^n) in group $gj(gj^m)$ in HTS; it consists of three processing times

B_i^m: The blocking time of job (j^n) in group $gj(gj^m)$ in HTS

B^m: The minimum blocking time of each job (j^n) in group $gj(gj^m)$ in HTS; it consists of three processing times

P^m: Processing time of the gj in HTS; in this research, it is equal to the max processing time of P_i^m

R^j: The wait time of the j in FPS

$\overset{k}{\omega}_{i,\alpha}$: Working time of Model α in machine (k) of FPS; i means the ith loading

$\overset{k}{\theta}_{i,\alpha}$: The setup time of Model α in machine (k) of FPS; i means the ith loading

S_i^{gj}: Start time of the $gj(gj^m)$ in HTS

F_i^{gj}: Finish time of the $gj(gj^m)$ in HTS

$\overset{k}{V}_{i,\alpha}^{\lambda}$: The λth processing time in the model α of FPS; i means the ith loading

$\beta_1 = 0.1, \beta_2 = 0.9$: Start processing time of the λth processing time in the model α in the machine (k) of FPS; i means the ith loading

$\overset{k}{F}_{i,\alpha}^{\lambda}$: Finish processing time of model α in the machine (k) of FPS; i means the ith loading.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to acknowledge financial supports of the National Science Foundation of China (51175435), the Program for New Century Excellent Talents in University (NCET-12-0463), and the Doctoral Fund of Ministry of Education of China (20136102110022).

References

[1] C. Rahendran and D. Chaudhuri, "A multi-stage parallel-processor flowshop problem with minimum flowtime," *European Journal of Operational Research*, vol. 57, no. 1, pp. 111–122, 1992.

[2] F. Riane, A. Artiba, and S. E. Elmaghraby, "A hybrid three-stage flowshop problem: efficient heuristics to minimize makespan," *European Journal of Operational Research*, vol. 109, no. 2, pp. 321–329, 1998.

[3] R. Linn and W. Zhang, "Hybrid flow shop scheduling: a survey," *Computers and Industrial Engineering*, vol. 37, no. 1-2, pp. 57–61, 1999.

[4] C. Oğuz, Y. Zinder, V. H. Do, A. Janiak, and M. Lichtenstein, "Hybrid flow-shop scheduling problems with multiprocessor task systems," *European Journal of Operational Research*, vol. 152, no. 1, pp. 115–131, 2004.

[5] K.-C. Ying and S.-W. Lin, "Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach," *International Journal of Production Research*, vol. 44, no. 16, pp. 3161–3177, 2006.

[6] K. Alaykýran, O. Engin, and A. Döyen, "Using ant colony optimization to solve hybrid flow shop scheduling problems,"

International Journal of Advanced Manufacturing Technology, vol. 35, no. 5-6, pp. 541–550, 2007.

[7] H. Luo, G. Q. Huang, Y. Zhang, Q. Dai, and X. Chen, "Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 6, pp. 962–971, 2009.

[8] B. L. Maccarthy and J. Liu, "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling," *International Journal of Production Research*, vol. 31, no. 1, pp. 59–79, 1993.

[9] B. Chen, "Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage," *Journal of the Operational Research Society*, vol. 46, no. 2, pp. 234–244, 1995.

[10] L. Wang and D.-Z. Zheng, "An effective hybrid heuristic for flow shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 21, no. 1, pp. 38–44, 2003.

[11] D. Laha and U. K. Chakraborty, "A constructive heuristic for minimizing makespan in no-wait flow shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 41, no. 1-2, pp. 97–109, 2009.

[12] A. Rossi, A. Puppato, and M. Lanzetta, "Heuristics for scheduling a two-stage hybrid flowshop with parallel batching machines: application at a hospital sterilisation plant," *International Journal of Production Research*, vol. 51, no. 8, pp. 2363–2376, 2013.

[13] C. Y. Cheng, T. L. Chen, L. C. Wang, and Y. Y. Chen, "A genetic algorithm for the multi-stage and parallel-machine scheduling problem with job splitting-A case study for the solar cell industry," *International Journal of Production Research*, vol. 51, no. 16, pp. 4755–4777, 2013.

[14] C. Sriskandarajah, "Performance of scheduling algorithms for no-wait flowshops with parallel machines," *European Journal of Operational Research*, vol. 70, no. 3, pp. 365–378, 1993.

[15] V. Caraffa, S. Ianes, T. P. Bagchi, and C. Sriskandarajah, "Minimizing makespan in a blocking flowshop using genetic algorithms," *International Journal of Production Economics*, vol. 70, no. 2, pp. 101–115, 2001.

[16] R. Ruiz and A. Allahverdi, "No-wait flowshop with separate setup times to minimize maximum lateness," *International Journal of Advanced Manufacturing Technology*, vol. 35, no. 5-6, pp. 551–565, 2007.

[17] B. Liu, L. Wang, and Y.-H. Jin, "An effective hybrid particle swarm optimization for no-wait flow shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 9-10, pp. 1001–1011, 2007.

[18] A. Mascis and D. Pacciarelli, "Job-shop scheduling with blocking and no-wait constraints," *European Journal of Operational Research*, vol. 143, no. 3, pp. 498–517, 2002.

[19] H. Luo, G. Q. Huang, Y. Feng Zhang, and Q. Yun Dai, "Hybrid flowshop scheduling with batch-discrete processors and machine maintenance in time windows," *International Journal of Production Research*, vol. 49, no. 6, pp. 1575–1603, 2011.