Applied Mathematical Sciences, Vol. 1, 2007, no. 29, 1399 - 1417

Trade-off between Time Complexity and Makespan for Flexible Flow-Shop Group Scheduling Problems with Two Machine Centers

Tzung-Pei Hong

Department of Computer Science and Information Engineering
National University of Kaohsiung
Kaohsiung, 811, Taiwan, R.O.C.
tphong@nuk.edu.tw

Pei-Ying Huang

Department of Computer Science and Information Engineering
National Taiwan University
Taipei, 106, Taiwan, R.O.C.
d93012@csie.ntu.edu.tw

Gwoboa Horng

Department of Computer Science
National Chung-Hsing University
Taichung, 40227, Taiwan, R.O.C.
gbhorng@cs.nchu.edu.tw

Abstract

The flexible flow-shop group scheduling problem is investigated in this paper to minimize the makespan. Two algorithms have been proposed to solve the problem with two machine centers, which have the same number of parallel machines. The first one is a heuristic algorithm. It first determines the sequence of jobs in each group by Sriskandarajah and Sethi's approach of solving the flexible flow-shop problems of two machine centers. It then determines the sequence of groups by the Johnson algorithm. The second one is a nearly optimal algorithm based on the search-and-prune technique, but can get better solutions. It can also be used to measure the performance of the first algorithm. Experimental results show that the

second approach can solve the problem with only a very limited size due to its high time complexity. The heuristic approach can, however, quickly obtain the approximate results, with much less computation time than that by the nearly optimal algorithm and with a similar quality of solutions. The proposed heuristic algorithm thus provides a feasible solution to large group scheduling problems that cannot be solved by the nearly optimal one. A trade-off can be easily achieved between accuracy and time complexity.

Mathematics Subject Classification: 90B35

Keywords: group scheduling, flexible flow shop, Johnson algorithm, machine center.

1 Introduction

Scheduling is an important process widely used in manufacturing, production, management, computer science, and so on. Appropriate scheduling can reduce material handling costs and time. Finding good schedules for given sets of jobs can thus help factory supervisors effectively control job flows and provide solutions for job sequencing.

In the past, Johnson first proposed an efficient algorithm which guaranteed optimality in a two-machine flow-shop problem [7]. Campbell, Dudek and Smith (CDS) then proposed a heuristic algorithm to solve the flow-shop problems of more than two machines [2]. Palmer also proposed an algorithm for achieving the same purpose [14]. Sriskandarajah and Sethi presented a heuristic algorithm based on the Johnson algorithm for solving flexible flow-shop problems of two machine centers with the same number of machines [18].

As to group scheduling, Yang and Chern considered the two-machine flow shop group scheduling problems with group removal time and job transportation time [19]. Dannenbring proposed a heuristic algorithm which combined the advantages of the Palmer and the CDS algorithms [4]. Allison compared the performance of single-pass and multiple-pass heuristics for solving group scheduling problems [1]. Schaller developed a new lower bound in a branch-and-bound procedure to evaluate partial sequences for the flow-shop group scheduling problem [16]. Logendran *et al.* investigated the group flexible flow-shop problems for minimizing makesapans [8]. Logendran *et al.* presented a two-machine group scheduling problem with sequence-dependent set-up time [10]. Yoshida and Hitomi developed an optimal algorithm for minimizing the total completion time in a two machine group scheduling problem with sequence-independent set-up time [20]. Many researches in this field are still in progress.

The problem addressed in the paper is a special case of the flexible flow shop problem in group scheduling. This paper specifically focuses on minimizing the total completion time of flexible flow shop in group scheduling with two machine centers, which have the same number of parallel machines. Two algorithms have been developed to solve it. The first one is a heuristic algorithm. It first determines the sequence of jobs in each group by Sriskandarajah and Sethi's approach of solving the flexible flow-shop problems with two machine centers. It then determines the sequence of groups by the Johnson algorithm. The second one is a nearly optimal algorithm, which uses the search-and-prune technique with an upper bound to determine the job sequence in each group and also uses the Johnson algorithm to determine the group sequence in the final schedule. Experimental results show that the proposed nearly optimal approach can solve the problem with only a very limited size due to its high time complexity. The proposed heuristic approach can, however, quickly obtain the approximate results, with much less computation time than that by the nearly optimal algorithm. The proposed heuristic algorithm can thus provide a feasible solution to large group scheduling problems that cannot be solved by the nearly optimal one. A trade-off can be easily achieved between accuracy and time complexity.

The remainder of this paper is organized as follows. Related scheduling algorithms are reviewed in Section 2. The assumptions and notation used in this paper are described in Section 3. The first algorithm for heuristically scheduling on a group flexible flow shop with two machine centers is proposed in Section 4. An example to illustrate the proposed heuristic scheduling algorithm is given in Section 5. The second algorithm for obtaining a nearly optimal makespan based on the search-and-prune technique is proposed in Section 6. Experiments for comparing the makespans and execution times of the two proposed algorithms are described in Section 7. Finally, conclusions are given in Section 8.

2 Review of Related Scheduling Algorithms

As mentioned above, flexible flow-shop problems are NP-hard. The flexible flow-shop group scheduling problems are also NP-hard since it is even more difficult than the traditional flexible flow-shop problems. No algorithms can find the optimal solutions in polynomial time. In the paper, we propose two algorithms to solve flexible flow-shop group scheduling problems with two machine centers. Some related scheduling algorithms are first introduced below.

2.1 Review of the LPT Scheduling Algorithm

The discovery of scheduling algorithms for a set of independent tasks with

arbitrary execution time and an arbitrary number of processors is a classic sequencing problem of wide interest and applications. Among the proposed scheduling algorithms, the LPT (Longest-Processing-Time-first) scheduling algorithm is the simplest and is widely used in many real-world situations.

The scheduling problem for LPT is stated as follows. Given a set of n independent tasks (T_1 to T_n), each with arbitrary execution time (t_1 to t_n), and a set of m parallel processors or machines (P_1 to P_m), the LPT scheduling algorithm assigns the task with the longest execution time (among those not yet assigned) to a free processor whenever this processor becomes free. For cases when there is a tie, an arbitrary tie-breaking rule can be assumed. The algorithm is described as follows.

The LPT scheduling algorithm:
Input: A set of n tasks, each with arbitrary processing time, and a set of m processors.
Output: A schedule and the final finishing time of all the tasks.
Step 1: Sort the tasks in a descending order according to the processing time.
Step 2: Initialize the current finishing time of each processor to zero.
Step 3: Assign the first task in the task list to the processor with the minimum finishing time.
Step 4: Set the new finishing time of the processor = the old finishing time of the processor + the execution time of the task.
Step 5: Remove the task from the task list.
Step 6: Repeat Steps 3 to 5 until the task list is empty.
Step 7: Among the finishing time of the processors, choose the longest as the final finishing time.

The finishing time by the LPT scheduling algorithm is in general not minimal. The computational time spent by the LPT scheduling algorithm is, however, much less than that by an optimal scheduling algorithm.

2.2 Review of the Johnson Scheduling Algorithm

The Johnson algorithm [7] was proposed to schedule job sequencing for a flow shop with two machines. Given a set of n independent jobs, each having two tasks (T_{11}, T_{21}, T_{12}, T_{22}, …, T_{1n}, T_{2n}) that must be executed in the same sequence on two machines (P_1 and P_2), the Johnson scheduling algorithm seeks a minimum completion time of the last job. The Johnson scheduling algorithm arranges the jobs which take less execution time on machine 1 than on machine 2 to be executed earlier, and the jobs which take less execution time on machine 2 than on machine 1 to be executed later. When a machine is free, the next unexecuted

job is then put on it for execution. Johnson proved that such scheduling achieved a minimum makespan. The detailed algorithm is stated as follows:

The Johnson scheduling algorithm:
Input: A set of n jobs, each having two tasks executed respectively on each of two machines.
Output: A schedule with a minimum completion time of the last job.
Step 1: Form the group of jobs U that takes less time on the first machine than on the second such that, $U = \{j \mid t_{1j} < t_{2j}\}$.
Step 2: Form the group of jobs V that takes less time on the second machine than on the first such that, $V = \{j \mid t_{1j} \geq t_{2j}\}$.
Step 3: Sort the jobs in U in ascending order of t_{1j}'s.
Step 4: Sort the jobs in V in descending order of t_{2j}'s.
Step 5: Schedule the jobs on the machines in the sorted order of U, then in the sorted order of V.

After Step 5, scheduling is finished and a completion time has been found.

2.3 Review of Sriskandarajah and Sethi's Scheduling Algorithm

Sriskandarajah and Sethi proposed a heuristic algorithm [18] for solving the flexible flow-shop problem of two machine centers. They also showed the completion time of the derived schedules was close to the optimum. Sriskandarajah and Sethi decomposed the problem into the following three sub-problems and solved each heuristically.

Part 1: Form the machine groups, each of which contains a machine from each center;
Part 2: Use the LPT method to assign jobs to each machine group (flow shop);
Part 3: Deal with job sequencing and timing using the Johnson algorithm.

In this paper, we will use the above approaches to solve the flexible flow-shop group scheduling problems with two machine centers.

3 Assumptions and Notation

Assumptions and notation used in this paper are described in this section.

Assumptions:

- Jobs are not preemptive.
- Each job has two tasks with processing times, executed respectively at each of two machine centers.
- Both the machine centers have the same number of parallel machines.
- Different groups of jobs cannot simultaneously be operated at the same machine center, but can simultaneously be operated at different machine centers.

Notation:

l: *The number of groups.*
n: *The number of jobs in a certain group.*
m: *The number of tasks in each job.*
mc_i: *The i-th machine center, i = 1 to 2.*
p: *The number of machines in each machine center.*
D_{ji}: *The j-th machine in the i-th machine center, j = 1 to p and i = 1 to 2.*
d_{ji}: *The completion time of the j-th machine in the i-th machine center.*
d_j: *The completion time of the j-th machine in a certain flowshop.*
c_{ji}: *The completion time of the j-th machine center for the i-th job.*
F_i: *The i-th allocated machine group (flow shop), i = 1 to p.*
F_{ji}: *The j-th machine of the flowshop F_i, j = 1 to 2.*
f_i: *The completion time of the i-th flowshop.*
f_{ji}: *The completion time of the j-th machine in the i-th flowshop.*
T_{jik}: *The j-th task of the i-th job in the k-th group, j = 1 to 2, i = 1 to n, and k = 1 to l.*
t_{jik}: *The execution time of T_{jik}.*
tt_{ik}: *The total execution time of the i-th job in the k-th group.*
mc_{ijk}: *The completion time in the i-th flow-shop at the j-th machine center for the k-th group.*
mc_{jk}: *The completion time at the j-th machine center for the k-th group.*
ff: *The final completion time of the whole schedule.*

4 A Heuristic Algorithm for Flexible Flow-shop Group Scheduling with Two Machine Centers

A heuristic algorithm for solving the flexible flow-shop problem with two machine centers is proposed by Sriskandarajah and Sethi in 1989 [18]. In this paper, we extend it to solve flexible flow-shop group scheduling problems with two machine centers. The proposed flexible flow-shop group scheduling algorithm first determines the job sequence in each group by Sriskandarajah and

Sethi's approach. It then determines the group sequence by the Johnson algorithm. The proposed algorithm is stated below.

The heuristic flexible flow-shop group scheduling algorithm:
Input: l groups of jobs, each with two tasks to be executed respectively on each of two machine centers with p parallel machines.
Output: A schedule with a near optimal completion time.

Level 1: Determining the job sequence in each group
Step 1: Set the variable k to one, where k is used to represent the number of the current group to be processed.
Step 2: Repeat Steps 3 to 15 until $k > l$.

Part 1: Forming the machine groups
Step 3: Form p machine groups, F_1, F_2, …, F_p, each of which contains one machine from each machine center. Each machine group can be thought of as a simple flow shop.
Step 4: Initialize the completion time f_1, f_2, …, f_p of each flow shop F_1, F_2, …, F_p to zero.

Part 2: Assigning the jobs in the k-th group to machine groups
Step 5: For each job J_{jk} in the k-th group, find its total execution time $tt_{jk} = t_{1jk} + t_{2jk}$ (j = 1 to n, k = 1 to l).
Step 6: Sort the jobs in descending order of processing time tt_{jk}; if any two jobs have the same tt_{jk} values, sort them in an arbitrary order.
Step 7: Find the flow shop F_i with the minimum processing time f_i among all the flow shops; if two flow-shops have the same minimum f_i value, choose one arbitrarily.
Step 8: Assign the first job J_{jk} in the sorted list to the chosen flow shop F_i which has the minimum completion time f_i among all the p flow shops.
Step 9: Add the total time tt_{jk} of job J_{jk} to the completion time of the chosen flow shop, F_i; that is:

$$f_i = f_i + tt_{jk}.$$

Step 10: Remove job J_{jk} from the job list.
Step 11: Repeat Steps 7 to 10 until the job list is empty.

After Step 11, jobs in each job group are clustered into p groups and are allocated to the p machine flow shops.

Part 3: Dealing with the job sequence in each flow shop
Step 12: For each flow shop F_i, set the initial completion time of the machines f_{ji} (j = 1 to 2, i =1 to p) to zero.

Step 13: Find the completion time of each flow shop f_i by the Johnson algorithm stated in Section 2.
Step 14: Save the corresponding job sequence.
Step 15: Set $k = k + 1$.

After Step 15, the individual job sequence for each group has been found.

Level 2: Determining the group sequence in the whole schedule
Step 16: Set the processing time mc_{jk} needed for the n jobs in group k on machine center j (j = 1 to 2, k = 1 to l) as:

$$mc_{jk} = \max_{i=1}^{p}(f_{ijk}) - \min_{i=1}^{p}(c_{(j-1)ik}),$$

where f_{jik} is the completion time in each flow-shop i at machine center j for group k and $c_{(j-1)ik}$ is the completion time of the first job in each flow-shop i at machine center j-1 for group k.
Step 17: Find the group sequence by the Johnson algorithm stated in Section 2 according to mc_{jk} (j = 1 to 2, k = 1 to l).

After Step 17, the group sequence for the entire schedule has been found.

Step 18: Schedule the groups based on the group sequence and schedule the job sequence in each flow-shop of each group to find the final completion time.

After Step 18, the entire scheduling is finished and the final total completion time has been found.

5 An Example for the Proposed Heuristic Algorithm

Assume there are three groups of jobs to be scheduled. Each group has five jobs, J_{1i} to J_{5i} (i = 1 to 3). Each job has two tasks to be executed by two operations. Each operation is run by a machine at its corresponding machine center. Assume each machine center includes only two parallel machines. Also assume the execution times of these jobs are listed in Table 1.

Table 1. Processing times for the three groups of jobs

	G_1					G_2					G_3				
	J_{11}	J_{21}	J_{31}	J_{41}	J_{51}	J_{12}	J_{22}	J_{32}	J_{42}	J_{52}	J_{13}	J_{23}	J_{33}	J_{43}	J_{53}
Task 1	3	9	6	8	6	9	2	6	4	6	7	3	8	6	3
Task 2	2	4	7	2	5	3	3	5	7	1	1	7	8	4	4

The algorithm first runs the steps on level 1 as follows. It first determines an appropriate job sequence in each of the three groups. Each group of jobs can then be scheduled independently. The processing steps are decomposed into three parts. Part 1 first forms two machine groups, F_1 and F_2, since each machine center has two machines. Each machine group can be thought of as a two-machine flow-shop. Part 2 then assigns the jobs in each group to the machine groups. Results for this example are shown in Table 2.

Table 2. The jobs allocated to each flow shop for each group

$Group_j$	G_1	G_2	G_3
$Flowshop_i$	*Jobs allocated*		
F_1	J_{31}, J_{41}, J_{11}	J_{32}, J_{42}	J_{23}, J_{43}, J_{53}
F_2	J_{21}, J_{51}	J_{12}, J_{52}, J_{22}	J_{33}, J_{13}

Part 3 then deals with the job sequence in each flow shop for each group. The results are shown in Table 3.

Table 3. The job sequence in each flow shop for each group

Group		G_1	G_2	G_3
Job sequence	F_1	J_{31}, J_{41}, J_{11}	J_{42}, J_{32}	J_{23}, J_{53}, J_{43}
	F_2	J_{51}, J_{21}	J_{22}, J_{12}, J_{52}	J_{33}, J_{13}

The steps on level 2 are then executed to determine the group sequence in the whole schedule. The finishing time of each group of jobs at each machine center is first calculated and shown in Table 4.

Table 4. The processing time of each group of jobs at each machine center

Machine Center	G_1	G_2	G_3
	Processing Time		
Machine Center 1	17	17	15
Machine Center 2	13	16	15

In Table 4, the processing time for processing the first tasks of all the jobs in

Group 1 at machine center 1 is 17 and for processing the second tasks at machine center 2 is 13. Similarly, the processing time evaluated for Group 2 is 17 and 16, respectively, and for Group 3 is 15 and 15, respectively. The Johnson procedure is then used to schedule the three groups according to the processing time at each machine center. The obtained group sequence for this example is G_2, G_3, G_1. All the groups of jobs are then scheduled according to the above group sequence together with its job sequence in each flow shop. The final scheduling results are shown in Figure 1. The final completion time is 51.

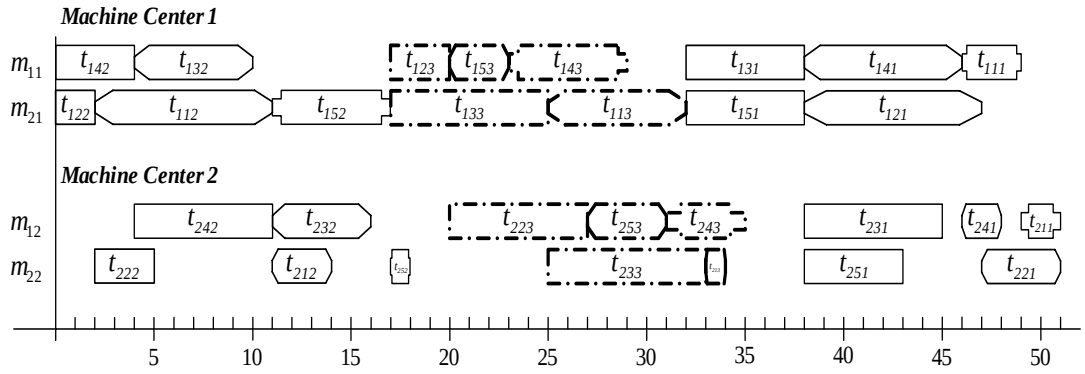

Figure 1: The final scheduling result in the example

6 A Nearly Optimal Scheduling Algorithm Based on the Search-and-Prune Technique

The completion time by the first algorithm is generally not minimal since it is a heuristic algorithm. For getting a more optimal schedule, the tasks in a set of jobs may need to be executed in different machine groups. In this section, we thus propose another scheduling algorithm based on the search-and-prune technique to get nearly optimal solutions, which can also be used to measure the performance of the first algorithm. It is not optimal because the Johnson algorithm used at the second level is not guaranteed to get the best group sequence since the calculated time used for scheduling is not exactly of the same meaning as in the conventional two-machine flow shops. The job sequence in each group from the first level is, however, optimal. The proposed nearly optimal algorithm is stated below.

The proposed nearly optimal group scheduling algorithm for two machine centers:

Input: l groups of jobs, each with two tasks to be executed respectively on each of two machine centers with p parallel machines.

Output: A schedule with a nearly optimal completion time.

Level 1: Determining the job sequence in each group

Step 1: Set the variable k to one, where k is used to represent the number of the current group to be processed.

Step 2: Repeat Steps 3 to 12 until $k > l$.

Step 3: Set the initial upper bound v_{max} of the final completion time in each group as ∞ .

Step 4: For each possible permutation of jobs in each machine center, do the following steps.

Step 5: In each machine center, set the initial completion time of each machine to zero.

Step 6: Schedule the first tasks of all the jobs in the machines of the first machine center according to the permutation generated. That is, for each task T_{1ik} of the i-th job in the k-th group allocated to the j-th machine in the first machine center D_{j1}, do the following substeps:

(a) Add the processing time t_{1ik} to the completion time d_{j1} of the machine D_{j1}; That is:

$$d_{j1} = d_{j1} + t_{1ik}, \text{ and}$$
$$c_{1i} = d_{j1.}$$

(b) If d_{j1} is larger than v_{max}, neglect all the permutations with this sequence in the first machine center and go to Step 4 for trying another permutation.

Step 7: Schedule the second tasks of all the jobs in the machines of the second machine centers according to the permutation generated. That is, for each task T_{2ik} of the i-th job in the k-th group allocated to the j-th machine of the second machine center D_{j2}, do the following substeps:

(a) Find the completion time d_{j2} of the machine D_{j2} as:

$$d_{j2} = max(d_{j2}, c_{1i}) + t_{2ik}, \text{ and}$$
$$c_{2i} = d_{j2.}$$

(b) If d_{j2} is larger than v_{max}, neglect all the permutations with this sequence in these two machine centers and go to Step 4 for trying another permutation.

Step 8: Set the completion time d_2 of the current schedule $= \max_{j=1}^{p}\left(d_{j2}\right)$ among all the p machines in the second machine center.

Step 9: If d_2 is smaller than v_{max}, then set $v_{max} = d_{2.}$

Step 10: Repeat Steps 4 to 9 until all the possible permutations have been tested.

Step 11: Save the corresponding job sequence for the k-th group.

Step 12: Set $k = k + 1$.

After Step 12, the optimal individual job sequence in each group has been found.

Level 2: Determining the group sequence in the whole schedule

Step 13: Set the processing time mc_{jk} needed for the n jobs in group k on machine center j (j = 1 to m, k = 1 to l) as:

$$mc_{jk} = \max_{i=1}^{p}(f_{ijk}) - \min_{i=1}^{p}(c_{(j-1)ik}),$$

where f_{jik} is the completion time in each flow-shop i at machine center j for group k and $c_{(j-1)ik}$ is the completion time of the first job in each flow-shop i at machine center j-1 for group k.

Step 14: Find the group sequence by the Johnson algorithm according to mc_{jk} (j = 1 to 2, k = 1 to l).

Step 15: Schedule the groups based on the group sequence and schedule the job sequence in each flow-shop of each group to find the final completion time.

After Step 15, the entire scheduling is finished and the final total completion time has been found.

7 Experimental Results

This section reports on experiments made to show the performance of the proposed scheduling algorithms. They were implemented by Visual C++ at an Intel Pentium IV with 2.40GHz CPU. Two parameters were considered, the group number l and the job number n of each group. In the first case, the group number l was fixed at 3 and the job number of each group varied from 3 to 8. In the second case, the group number l varied from 3 to 9, with the job number n of each group fixed at 7. Each job had two tasks and each machine center had two parallel machines. The execution time of each task was randomly generated in the range of 5 to 50. Each set of problems was executed for 20 tests and the makespans and computation times were measured. The proposed nearly optimal approach used a pruning technique to increase its efficiency. It could not, however, work for more than three groups with eight jobs for the first case and for more than nine groups with seven jobs for the second case in our environments due to its large amount of computation time.

For the first case in which the group number is 3, the average makespans for problems of three to eight jobs in each group by the two proposed methods are shown in Figure 2.

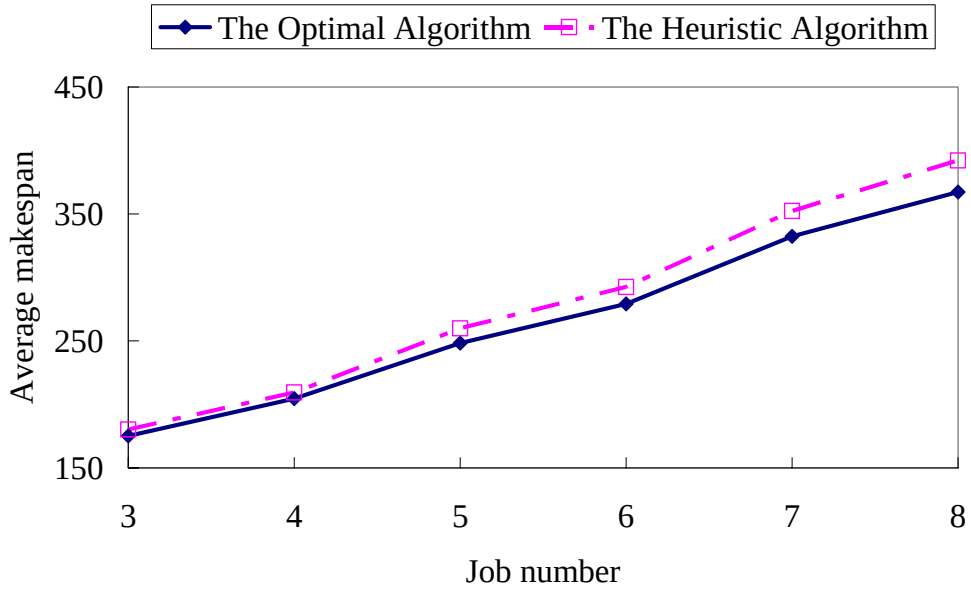

Figure 2: The average makespans for the group number l = 3 with n = 3 to 8

The deviation percentages of the average makespans by the proposed heuristic algorithm from those by the nearly optimal algorithm for different numbers of jobs in each group are shown in Table 5. The average deviation percentage is 4.59%.

When the group number l is 3, the average CPU times for problems of three to eight jobs in each group are shown in Figure 3. The second algorithm proposed for nearly optimal solutions could not run over three groups of eight jobs in the limitation of eighty minutes due to its high time complexity.

Next, in the second set of experiments, the job number n of each group was fixed at 7. The average makespans for problems of three to nine groups by the two proposed methods are shown in Figure 4.

Table 5. The distribution of deviation rates for different numbers of jobs when the group number is 3 and the run number is 20

n	*l*	Run Number	Run number with a deviation range			Average Deviation (%)
			0%	0%< to≦ 5%	>5%	
3	3	20	11	2	7	2.79
4	3	20	9	7	4	2.51
5	3	20	5	5	10	4.59
6	3	20	5	6	9	4.98
7	3	20	1	6	13	5.97
8	3	20	0	6	14	6.71
Total		120	31	32	57	Avg. 4.59

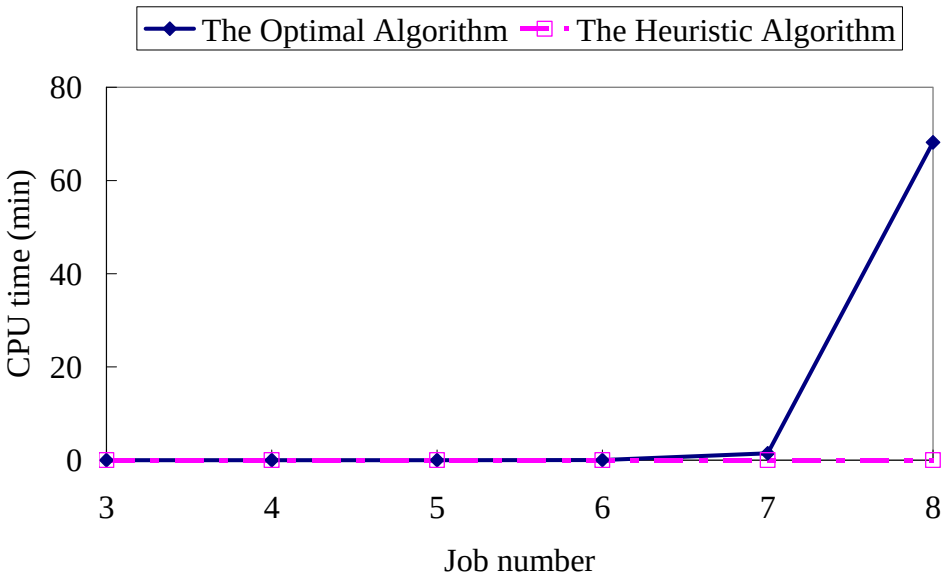

Figure 3: The average CPU times for different numbers of jobs with $l = 3$

Table 6 gives the distribution of the deviation rates of the proposed heuristic algorithm from the nearly optimal one for different number of groups and $n = 7$. The average deviation percentage is 5.76%.

When the job number n of each group is 7, the average CPU times for problems of three to nine groups are shown in Figure 5. The second algorithm proposed for nearly optimal solutions could not run over nine groups in this case

with the limitation of 25 hours due to its high time complexity.

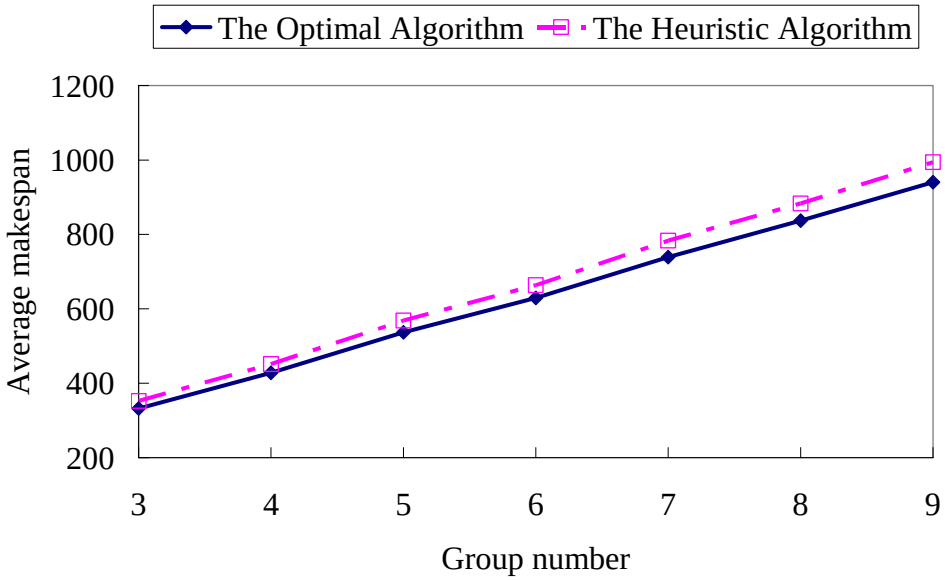

Figure 4: The average makespans for the job number n = 7 with l = 3 to 9

Table 6. The distribution of deviation rates for different numbers of groups when the job number is 7 and the run number is 20

n	*l*	Run Number	Run number with a deviation range			Average Deviation (%)
			0%	0%< to≦ 5%	>5%	
7	3	20	1	6	13	5.97
7	4	20	0	7	13	5.61
7	5	20	1	8	11	5.90
7	6	20	0	9	11	5.42
7	7	20	0	9	11	6.05
7	8	20	1	10	9	5.55
7	9	20	0	6	14	5.84
Total		140	3	55	82	Avg. 5.76

From the above figures and tables, it is easily seen that the first proposed algorithm got only a little larger makespans than the second one did. The computational time needed by the second algorithm was, however, much larger than that needed by the first approach, especially when the job or group number is

large. Actually, since the flexible flow-shop group scheduling problem is NP-hard, the second approach can work only for a small number of groups and jobs. The first proposed heuristic approach can solve this problem and is thus more suitable for real applications than the second proposed nearly optimal one.

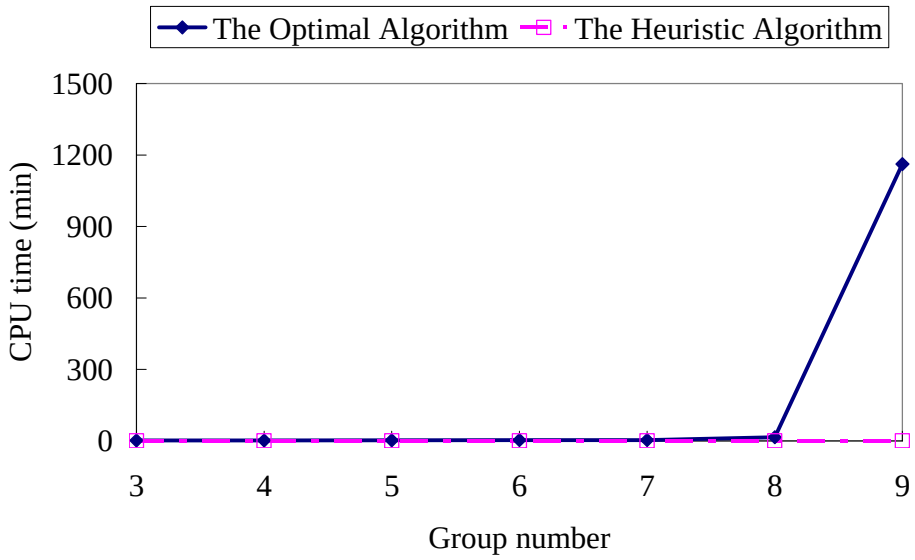

Figure 5: The average CPU times for different numbers of groups with $n = 7$

At last, experiments for large job numbers and group numbers were made to show the performance of the heuristic algorithm. Experiments were made respectively for n from 1000 to 9000 with the group number l being 100 and for n being 100 with l from 1000 to 9000. The average CPU times for the above cases are shown respectively in Figures 6 and 7, both being within 50 seconds.

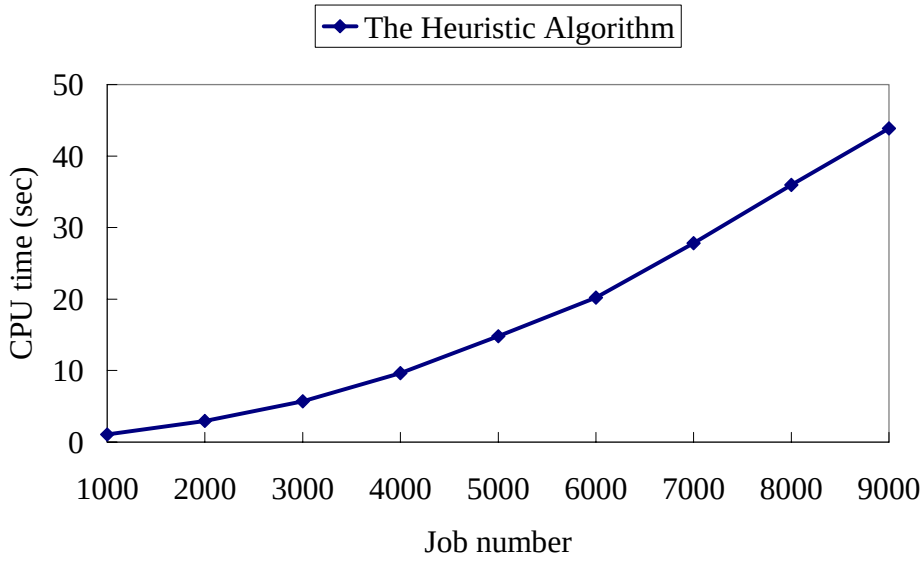

Figure 6: The average CPU times for l = 100 and n = 1000 to 9000

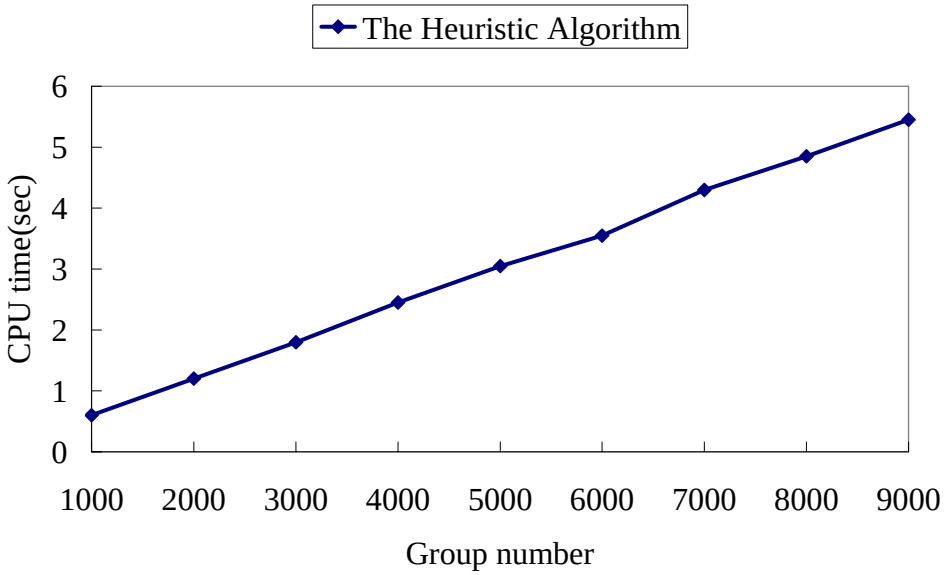

Figure 7: The average CPU times for n = 100 and l = 1000 to 9000

Hence, the first proposed heuristic approach is feasible even for a large number of groups and jobs. It is thus more suitable than the second proposed approach for real applications.

8 Conclusion

Scheduling jobs in group flexible flow shops is an NP-hard problem. In this paper, we have proposed two algorithms to solve this problem with two machine centers, which have the same number of parallel machines. The first proposed heuristic algorithm assigns jobs to flow shops, whereas the second proposed nearly optimal one allows jobs to be executed among flow shops. Since flow line arrangement may reduce the costs of material handling and production time, the heuristic approach may thus be advantageous in practice. Experimental results show that the second approach can solve the problem with only a very limited size. Although the proposed heuristic approach can not guarantee to get optimal solutions, its average deviation rates are quite low. Besides, it can quickly obtain the results, with much less computation time than the nearly optimal algorithm. The proposed heuristic approach can thus provide a feasible solution to flexible flow-shop group scheduling problems with large sizes, which cannot be solved by the optimal approach. A trade-off can thus easily be achieved between accuracy and time complexity. In the future, we will attempt to consider other constraints, such as setup times, due dates, and priorities.

References

[1] J. D. Allison, Combining Petrov's heuristic and the CDS heuristic in group scheduling problems, The 12th Annual Conference on Computers and Industrial Engineering, 1990, 457-461.

[2] H. G. Campbell, R. A. Dudek and M. L. Smith, A heuristic algorithm for the *n* job, *m* machine sequencing problem, Management Science, 16(1970), B630-B637.

[3] S. C. Chung and D. Y. Liao, Scheduling flexible flow shops with no setup effects, The 1992 IEEE International. Conference on Robotics and Automation, 1992, 1179-1184.

[4] D. G. Dannenbring, An evaluation of flowshop sequencing heuristics, Management Science, 23(1977), 1174-1182.

[5] R. A. Dudek, S. S. Panwalkar and M. L. Smith, The lessons of flowshop scheduling research, Operations Research, 40(1992), 7-13.

[6] J. N. D. Gupta, A functional heuristic algorithm for the flowshop scheduling problem, Operations Research, 40(1971), 7-13.

[7] S. M. Johnson, Optimal two- and three-stage production schedules with set-up times included, Naval Research Logistics Quarterly, 1(1954), 61-68.

[8] R. Logendran, S. Carson and E. Hanson, Group scheduling in flexible flow shops, International Journal of Production Economics, 96(2005), 143-155.

[9] R. Logendran and N. Nudtasomboon, Minimizing the makespan of a group scheduling problem: a new heuristic, International Journal of Production Economics, 22(1991), 217-230.
[10] R. Logendran, N. Salmasi and C. Sriskandarajah, Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups, Computers and Operations Research, 33(2006), 158-180.
[11] R. Logendran and C. Sriskandarajah, Two-machine group scheduling problem with blocking and anticipatory setups, European Journal of Operational Research, 69(1993), 467-481.
[12] T. E. Morton and D. W. Pentico, Heuristic Scheduling Systems with Applications to Production Systems and Project Management, John Wiley & Sons Inc., New York, 1993.
[13] M. Nawaz, J. E. E. Enscore and I. Ham, A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem, Omega, Vol. 11(1983), 91-95.
[14] D. S. Palmer, Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum, Operational Research Quarterly, 16(1965), 101-107.
[15] V. A. Petrov, Flow Line Group Production Planning, Business Publications, London, 1966.
[16] J. Schaller, A new lower bound for the flow shop group scheduling problem, Computers and Industrial Engineering, 41(2001), 151-161.
[17] M. Solimanpur, P. Vrat and R. Shankar, A heuristic to minimize makespan of cell scheduling problem, International Journal of Production Economics, 88(2004), 231-241.
[18] C. Sriskandarajah and S. P. Sethi, Scheduling algorithms for flexible flow shops: worst and average case performance, European Journal of Operational Research, 43(1989), 143-160.
[19] D. L. Yang and M. S. Chern, Two-machine flowshop group scheduling problem, Computers & Operations Research, 27(2000), 975-985.
[20] T. Yoshida and K. Hitomi, Optimal two-stage production scheduling with setup times separated, AIIE Transactions, 11(1979), 261-263.

Received: December 31, 2006