A Fast and Efficient Fluid Relaxation Algorithm for Large-Scale Re-entrant Flexible Job Shop Scheduling

Linshan Ding, Zailin Guan, Zhengmin Zhang

A Fast and Efficient Fluid Relaxation Algorithm for Large-Scale Re-entrant Flexible Job Shop Scheduling

Linshan Ding[1], Zailin Guan[1] and Zhengmin Zhang[1]

[1] Huazhong University of Science and Technology, Wuhan, China
d201980276@hust.edu.cn

Abstract. In this paper, we study a large-scale re-entrant flexible job shop scheduling problem (FJSP) with the objective of makespan minimization. In the proposed problem, machine quantities, job types, and processes of jobs are known in advance. At least one machine is available for each process. The large production demand for each type of jobs leads to a large-scale manufacture feature in this problem. To address the problem, we first establish a fluid model for the large-scale re-entrant FJSP. Then, we design a priority update rule to improve the assignment of jobs and machines. We finally propose a fast and efficient fluid relaxation algorithm (FRA) to solve the large-scale re-entrant FJSP through the relaxation fluid optimal solution. Numerical results show that the FRA is asymptotically optimal with the increase of the problem scale. The scale of problems has little effect on the FRA's solving speed. Therefore, we conclude the FRA is suitable for solving the large-scale re-entrant FJSP.

Keywords: Flexible Job Shop Scheduling, Fluid Model, Fluid Relaxation, Re-entrant flows, Large scale optimization.

1 Introduction

The flexible job shop scheduling problem (FJSP) is an extension of the job shop scheduling problem [1]. It breaks through the constraint of resource uniqueness. FJSPs have a set of available machines for each process and the solution space of FJSP is further expanded. Thus, we consider it as a more complex NP-hard problem [2]. We study a large-scale re-entrant flexible job shop scheduling problem (FJSP). In the large-scale re-entrant FJSP, each process has an available machine set, each process routing may have re-entrant flows, different processes of the same job can be processed on the same machine, and the demand for each type of job is large. The large production demand for each type of jobs leads to a large-scale manufacture feature in this problem. Compared with FJSPs, the large-scale re-entrant FJSP has many identical copies of the fixed set of jobs, which increases the number of jobs to be scheduled. The increasing number of jobs results in the huge exponentially solution space which increases the complexity of scheduling. Hence, the large-scale re-entrant FJSP is more complex than FJSP with same job types. There are the large-scale re-entrant FJSP in actual production, such as semiconductor manufacturing lines [3]. Therefore, it is important to study an efficient solution algorithm for actual production.

In recent years, researchers have done a lot of research for large-scale job shop scheduling by the fluid approach. For instance, Bertsimas and Gamarnik (1999) propose a fluid relaxation algorithm that replaces discrete jobs with continuous fluid flow to solve the large-scale job shop scheduling and packet routing problem [4]. Boudoukh et al. (2001) propose a simulated fluid solution heuristic algorithm that approximated the job shop scheduling problem to a continuous deterministic scheduling problem [5]. Dai and Weiss (2002) propose a fluid heuristic online scheduling algorithm based on bottleneck machine safety inventory. However, the safety inventory calculation is cumbersome, which is not conducive to practical application. The inventory formation and inventory clearance take a long time, which increases the makespan [6]. Nazarathy and Weiss (2010) propose a simple heuristic method to solve a large-scale job shop scheduling problem with the processing time randomly generated. Numerical results show that the proposed heuristic method is asymptotic optimality with the increase of the number of jobs [7].

Some scholars propose some fluid approaches by constructing the fluid solution tracking formula to solve the large-scale job shop scheduling problems. Bertsimas and Sethuraman (2002) construct the fluid solution tracking formula by combining the fluid relaxation approach with fair queuing in communication networks. They propose a fluid synchronization algorithm based on the fluid solution tracking formula [8]. Gu et al. (2017) construct a virtual scheduling formula which as the benchmark to determine the processing priority of each job on each machine. Based on the virtual scheduling formula, they propose a tracking virtual scheduling algorithm to solve the general job shop scheduling problem [9]. Gu et al. (2018) study the large-scale random job shop scheduling problem with a generally similar number of jobs. The objective of this problem is to minimize the maximum completion time. They propose a strategy to solve the random job shop scheduling problem by tracking the fluid schedule [10].

Up to now, scholars have done a lot of research on large-scale re-entrant job shop scheduling. However, no research has been found on large-scale re-entrant FJSPs. Thus, we establish a fluid model and propose an efficient fluid relaxation algorithm for the large-scale re-entrant FJSP.

The remaining parts of this paper are as follows: In section 2, we establish a fluid model to describe our problem. In section 3, we design the priority updating rule and propose a fluid relaxation algorithm to solve the large-scale re-entrant FJSP. In section 4, we verify the performance of the fluid relaxation algorithm on several artificial benchmark problems. The conclusion is presented in section 5.

2 The Proposed Fluid Model

We establish a fluid model for solving the large-scale re-entrant flexible job shop scheduling problem. In our model, processes of different types of jobs are divided into different classes $k = \{1, 2, 3 \ldots K\}$. We use Z_{krj} to indicate whether the j^{th} process of the r^{th} type of job is represented as class k (value 1 for yes, 0 for no). We assume the job is composed of fluid. We use continuous fluid flow instead of discrete jobs. Thus, the number of jobs is not required to be an integer in the fluid model. Within unit time, the

machine can allocate its processing time (the real number between 0 and 1) to different classes. Each machine has a fixed proportion of processing time to deal with each class. The output of our model is the proportion of time allocated to each class by each machine. The notations we used are introduced follows:

Indexes:

i : Machines, ($i = 1,2,3 \dots I$).

r(r′): Job types, ($r = 1,2,3 \dots R$).

j(j′): Processes, ($j = 1, 2, 3 \dots J_r, \forall r \in \{1,2,3 \dots R\}$)

k(k′): Classes,$k = 1,2,3 \dots K, K = \sum_{r=1}^{R} J_r$.

Input data:

N_r: The total number of r^{th} type of job.

M_I: Set of machines, $M_I = \{1,2 \dots I\}$.

t_{ik}: Processing time of class k on machine i.

L_K: Set of classes, $L_K = \{1,2 \dots K\}$.

M_k: Set of available machines for class k.

K_i: Set of available classes for machine i.

Decision variables:

u_{ik}: The proportion of processing time allocated to class k by machine i.

Notations:

Z_{krj}: Binary variable, $Z_{krj} = 1$ if the j^{th} process of the r^{th} type of job is represented as class k, $\sum_r^R \sum_j^{J_r} Z_{krj} = 1 \quad \forall k \in L_K$.

t_{ik}: Processing time of class k on machine i.

e_{ik}: Processing rate of class k on machine i, $e_{ik} = 1/t_{ik}$.

$p_{kk\prime}$: Binary variable, take value 1 if class k′ is the tight post-process of class k.

T_k: Completion time of class k.

c_{ik}: Binary variable, $c_{ik} = 1$ if class k is available for machine i.

E_k: Total processing rate of class k, $E_k = \sum_{i=1}^{I} e_{ik} u_{ik}$.

$Q_k(t)$: The actual number of class k at time t.

$Q_k^+(t)$: The actual total number of class k which at time t has not yet completed processing.

$Q_{ik}^+(t)$: The actual total number of class k which at time t has not yet completed the process by machine i.

$q_k(t)$: The fluid number of class k at time t.

$q_k^+(t)$: The total fluid number of class k which at time t has not yet completed processing.

$q_{ik}^+(t)$: The total fluid number of class k which at time t has not yet completed the process by machine i.

The fluid model satisfies the following equations:

$$Q_k(0) = \sum_{r=1}^{R} N_r Z_{kr1} \quad \forall k \in L_K \tag{1}$$

$$q_k(0) = Q_k(0) \quad \forall k \in L_K \tag{2}$$

$$q_k(t) = q_k(0) - tE_k + t \sum_{k\prime \in L_k} E_{k\prime} p_{k\prime k} \quad \forall k, k\prime \in L_K \tag{3}$$

Equation (1) represents the actual number of each class in time 0. Equation (2) means the fluid number of each class in time 0. Equation (3) represents the fluid number of each class in time t.

$$Q_k^+(0) = \sum_{r=1}^{R} \sum_{j=1}^{J_r} Z_{krj} N_r \quad \forall k \in L_K \tag{4}$$

$$q_k^+(0) = Q_k^+(0) \quad \forall k \in L_K \tag{5}$$

$$q_k^+(t) = q_k^+(0) - tE_k \quad \forall k \in L_K \tag{6}$$

Equation (4) means the actual total number of each class which has not yet completed processing in time 0. Equation (5) represents the total fluid number of each class that has not yet completed processing in time 0. Equation (6) represents the total fluid number of each class in time t.

$$q_{ik}^+(0) = q_k^+(0) \frac{e_{ik} u_{ik}}{\sum_{i=1}^{I} e_{ik} u_{ik}} \quad \forall i \in M_I, k \in L_K \tag{7}$$

$$q_{ik}^+(t) = q_{ik}^+(0) - te_{ik} u_{ik} \quad \forall i \in M_I, k \in L_K \tag{8}$$

$$T_k = \frac{q_k^+(0)}{E_k} \quad \forall k \in L_K \tag{9}$$

In Equation (7), the total fluid number of class k which has not yet completed the process by machine i at the initial time is given. Equation (8) represents the total fluid number of class k which has not yet completed the process by machine i at time t. Equation (9) means the completion time of each class in the fluid model.

$$\sum_{r=1}^{R} \sum_{j=1}^{J_r} z_{krj} = 1 \quad \forall k \in L_K \tag{10}$$

$$\sum_{k=1}^{K} u_{ik} \leq 1 \quad \forall i \in M_I \tag{11}$$

$$0 \leq u_{ik} \leq c_{ik} \quad \forall i \in M_I, k \in L_K \tag{12}$$

$$E_k \leq \sum_{k'=1}^{K} E_{k'} p_{k'k} \quad \forall k, k' \in L_K : q_k(0) = 0 \tag{13}$$

Equation (10) means that each process of each type of job can only be represented by one class. Constraints (11) show that the machine utilization is less than 100%. Constraint (12) represents the range of decision variables. Constraint (13) indicates that the processing rate is less than or equal to the arrival rate if the initial fluid number of class k equal to zero, which guarantees the solution is feasible.

The objective is to minimize the maximum completion time of all classes, as indicated in Equation (14).

$$C_{max} = min(max\{T_1, \dots T_K\}) \tag{14}$$

3 The Priority Update Rule and The Fluid Relaxation Algorithm

We can obtain the proportion of time allocated to each class by each machine using the fluid model. However, we do not specify the machine assignment for each job and the processing sequence of jobs on each machine in detail. Therefore, we propose a priority updating rule based on the fluid model to guide the machine selection and job processing sequence.

3.1 The Priority Update Rule

We define the priority of class k at time t as Equation (15).

$$F_k = \frac{Q_k^+(t) - q_k^+(t)}{Q_k^+(0)} \quad \forall \mathrm{k} \in L_K \tag{15}$$

We use F_k value to represent the priority of class k. We select the class with the maximum F_k value at each decision point. It means we select the job to process which has the maximum F_k value at the decision time. We have to allocate an available machine for each chosen class. Thus, we define Equation (16) to represent the priority of each available machine to each class at time t.

$$B_{ik} = \frac{Q_{ik}^+(t) - q_{ik}^+(t)}{q_{ik}^+(0)} \quad \forall \mathrm{k} \in L_K, i \in M_I \tag{16}$$

We choose an available machine which has the highest B_{ik} to process the job according to F_k.

In each decision time, we select a class k that has the maximum F_k value first (if different classes have the same F_k value, we select a class from them randomly). Then we select an available idle machine that has the maximum B_{ik} value to process the class k (if different machines have the same B_{ik} value, we select a machine from them randomly). Finally, we complete the job selection and machine assignment in manufacturing.

3.2 The Fluid Relaxation Algorithm

We propose a fluid relaxation algorithm (FRA) to solve the large-scale re-entrant flexible job shop scheduling problem. We first solve the fluid model with the CPLEX to obtain the optimal maximum completion time C_{max} (the lower bound value) and the proportion of time allocated to each class by each machine in the fluid model. Then we propose a fluid relaxation algorithm based on the priority update rule.

The detailed steps of the algorithm are as follows:

Step1: Solve the fluid model with the CPLEX to obtain all u_{ik}.

Step2: We assume that the current time is represented by TIME_NOW, the next idle time of machine i represented by TIME_NEXT (i). Then all the machines are idling in the initial time (TIME_NOW = 0).

Step3: Generate the set of M_{now} which includes all the machines which are idling and have available jobs at time TIME_NOW.
Step4: Generate the available class set K_{now},
$\forall i \in M_{now}, if\ k \in K_i\ and\ Q_k(now - time) > 0, then\ k \in K_{now}$. If $K_{now} = \emptyset$, end the algorithm and output the solution.
Step5: Select the class k from a set K_{now} which has the maximum F_k value, then select the machine i from set $M_{now} \cap M_k$ which has the maximum B_{ik} value to process class k .
Step6: Update the value of TIME_NEXT (i), Q_k, Q_k^+, F_k and B_{ik}. Update the set M_{now}. If $M_{now} \neq \emptyset$, go to step 4.
Step7: Update the value of TIME_NOW. TIME_NOW $= min_{i \in M_I} TIME\_NEXT(i)$, go to step3.
The asymptotic optimality of FRA:
Theorem 1. Consider a flexible job shop scheduling problem with R job types and I machines. The FRA produces a schedule with makespan time C_{FRA} such that:

$$C_{max} \leq C_{FRA} \leq C_{max} + (R+2)t_{max}J_{max} \tag{17}$$

In Equation (17), C_{max} is the lower bound provided by the fluid model, t_{max} is the maximum processing time over all processes in all machines. J_{max} is the maximum number of processes of any job type. The proof of Theorem 1 is similar as the Theorem 4 which proposed by Dai and Weiss [6].
In our problem, $(R+2)t_{max}J_{max}$ is a constant in a finite range. The FRA satisfies the following Equation (18) as the total number of jobs goes to infinity.

$$\frac{C_{FRA}}{C_{max}} \approx 1 \tag{18}$$

From Equation (18), we see that the FRA is asymptotically optimal as the number of jobs increases.

4 Computational experiments and results

To verify the superiority of the proposed fluid relaxation algorithm, we develop a numerical experiment for the large-scale re-entrant FJSP. The algorithm is implemented on a PC with Inter (R) Core (TM) I5-9300 CPU @ 2.40GHz 2.40 GHz and 8GB RAM. We construct an artificial example with 3 job types and 10 machines. We initialize the set of available machines for each process randomly. The processing time of all processes obeys the uniform random distribution U (1, 5). In **Table 1**, we provide the available machines and the processing time of each process. "-"indicates the machine is not available for this process.

Table 1. Processing times.

Job	Process	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
	O11	3.3	-	4.1	-	2.2	4	1.5	1.2	-	3.9
	O12	-	-	2.5	-	3.7	-	1.5	3	4.3	1.5
R1	O13	-	-	4.2	-	-	4.7	1.4	-	-	4.5
	O14	-	-	-	2.1	-	5	-	3.9	2.9	-
	O15	-	3.4	-	-	-	1.6	-	-	3.9	-
	O21	4.9	-	3.5	-	-	-	-	4	4.1	3.7
	O22	3.8	-	-	4.6	-	2.8	-	1.9	4.7	-
R2	O23	-	3.6	3.7	-	3.7	-	-	2.2	2.1	-
	O24	1.1	-	1.4	-	1.5	-	-		1.7	1.3
	O25	4.9	2.3	-	-	1.1	1.5	1.5	1.4	-	-
	O31	-	2.2	1.4	-	3.8	1.4	4.9	3.2	2.6	-
	O32	3.1	2.8	1.3	-	-	-	-	2.7	-	3.4
R3	O33	1.9	-	2.8	-	-	2.8	-	1.1	-	3.7
	O34	-	2	-		4	-	3.2	-	-	2.6
	O35	-	1.1	-	4.5	-	-	4	-	2.1	3.7

In **Table 2**, we calculate the results of FRA solving the re-entrant FJSP with different problem scales. The problem scale is denoted by the total number of jobs multiply the total number of machines (n×m). C_{max} is the lower bound value of the problem which we obtain with CPLEX. C_{FCFS} is the results which we obtain with the First Come First Serve policy (FCFS). GAP represents the proximity of the FRA solution to C_{max}. We get the GAP value through the following Equation (19).

$$\text{GAP} = \frac{C_{FRA}-C_{max}}{C_{max}} \times 100\% \tag{19}$$

Table 2. Results obtained by FRA.

Instance	R1	R2	R3	n×m	C_{max}	C_{FRA}	C_{FCFS}	GAP	Computing Time (s)
1	6	8	10	24×10	21.2	24.6	37.8	16.03%	0.81
2	12	16	20	48×10	42.4	47.1	74.8	11.08%	0.96
3	24	32	40	96×10	84.9	89.6	144	5.53%	1.05
4	48	64	80	192×10	169.9	175.6	285.2	3.35%	0.73
5	96	128	160	384×10	339.9	348.1	545.8	2.41%	1.24
6	192	256	320	768×10	679.9	691.7	1147.3	1.73%	1.74
7	384	512	640	1536×10	1359.8	1378.3	2290.1	1.36%	2.36
8	768	1024	1280	3072×10	2719.7	2755.9	4487.5	1.33%	3.44

The GAP gradually decreases with the increase of problem scale, indicating that FRA is asymptotically optimal with the increase of the number for various jobs. The CPU

time of FRA within 4 seconds for all instances and the increments in CPU time is small as problem scale increases. The results show that the fluid relaxation algorithm (FRA) can solve the problem efficiently and quickly, the problem scale has little effect on the solving speed. The FRA is suitable for solving the large-scale re-entrant FJSP.
The FCFS policy is a traditional dispatching rule in semiconductor manufacturing [11]. The following **Fig. 1** compares the FRA and the FCFS policy in eight instances. Note that the FRA outperforms the FCFS policy in all instances. The results indicate that the FRA has the potential to improve the productivity in semiconductor manufacturing.

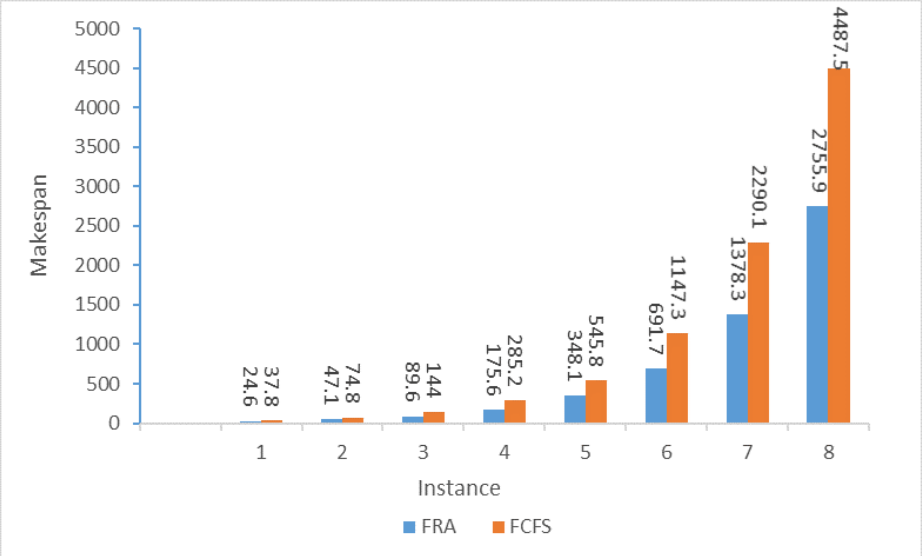

Fig. 1. The makespan of FRA and FCFS

5 Conclusion

In this paper, we study a large-scale re-entrant FJSP with machine numbers, job types, and processes of job are known and fixed. Moreover, the demand quantity for each type of jobs is a large value. We present a fluid relaxation algorithm (FRA) and a priority update rule based on the fluid model to solve the large-scale re-entrant FJSP. We construct a large-scale re-entrant FJSP with 3 job types and 10 machines to evaluate the proposed algorithm. Numerical results show that the fluid relaxation algorithm is asymptotically optimal with the increase of the number of various jobs. Moreover, the problem scale has little effect on the solving speed of the FRA. The FRA outperforms the FCFS policy in all instances which indicates the FRA has the potential to improve the productivity in semiconductor manufacturing. Furthermore, the low computation times make the FRA practical for implementation.
However, we do not consider the set-up time and machine failure in the fluid model which effects the productivity in semiconductor manufacturing. In future research, we

can incorporate set-up time and machine failure in the fluid model or as an additional step in the FRA.

References

1. Pezzella, F., Morganti, G., and Ciaschetti, G.: 'A genetic algorithm for the Flexible Job-shop Scheduling Problem', Computers & Operations Research, 2008, 35, (10), pp. 3202-3212
2. Chaudhry, I.A., and Khan, A.A.: 'A research survey: review of flexible job shop scheduling techniques', Int T Oper Res, 2016, 23, (3), pp. 551-591
3. Kim, S.H., and Lee, Y.H.: 'Synchronized production planning and scheduling in semiconductor fabrication', Comput Ind Eng, 2016, 96, pp. 72-85
4. Bertsimas, D., and Gamarnik, D.: 'Asymptotically optimal algorithms for job shop scheduling and packet routing', J Algorithms, 1999, 33, (2), pp. 296-318
5. Boudoukh, T., Penn, M., and Weiss, G.: 'Scheduling jobshops with some identical or similar jobs', J Scheduling, 2001, 4, (4), pp. 177-199
6. Dai, J.G., and Weiss, G.: 'A fluid heuristic for minimizing makespan in job shops', Oper Res, 2002, 50, (4), pp. 692-707
7. Nazarathy, Y., and Weiss, G.: 'A fluid approach to large volume job shop scheduling', J Scheduling, 2010, 13, (5), pp. 509-529
8. Bertsimas, D., and Sethuraman, J.: 'From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective', Mathematical Programming, 2002, 92, (1), pp. 61-102
9. Gu, M.Z., Lu, X.W., and Gu, J.W.: 'An asymptotically optimal algorithm for large-scale mixed job shop scheduling to minimize the makespan', J Comb Optim, 2017, 33, (2), pp. 473-495
10. Gu, J.W., Gu, M.Z., Lu, X.W., and Zhang, Y.: 'Asymptotically optimal policy for stochastic job shop scheduling problem to minimize makespan', J Comb Optim, 2018, 36, (1), pp. 142-161
11. Siebert, M., Bartlett, K., Kim, H., Ahmed, S., Lee, J., Nazzal, D., Nemhauser, G., and Sokol, J.: 'Lot targeting and lot dispatching decision policies for semiconductor manufacturing: optimisation under uncertainty with simulation validation', Int J Prod Res, 2018, 56, (1-2), pp. 629-641