Simulation in Production
and Logistics 2015
Markus Rabe & Uwe Clausen (eds.)
Fraunhofer IRB Verlag, Stuttgart 2015

Flexible Job-shop Scheduling with Dynamic Stochastic Machine Sets

Flexible Job-Shop-Planerstellung mit dynamischen stochastischen Maschinen-Mengen

Tao Zhang, Shufang Xie, Oliver Rose, Universität der Bundeswehr München, Neubiberg (Germany), tao.zhang@unibw.de, shufang.xie@unibw.de, oliver.rose@unibw.de

Abstract: In the general flexible job-shop scheduling problem, each type of operation has a given static and deterministic machine set. However, in practice the machine set is mostly neither static nor deterministic. Approaches to the general FJSP will fail to solve the practical case. Our study focuses on this practical FJSP with both dynamic and stochastic machine sets. Because the problem involves stochastic factors, a simulation-based meta-heuristic procedure is introduced. To speed up the solution procedure the stochastic simulation is replaced with a deterministic simulation model and a stochastic process generator. The experimental results show that the computing time is dramatically reduced while the performance of the schedule decreases only a little.

1 Introduction

In the flexible job-shop scheduling problem (FJSP), an operation may be processed on several machines which form a machine set. Comparing to the job shop scheduling problem, the raised issue is the machine allocation, i.e., assigning each operation to a particular machine from the machine set. Usually, for the general FJSP the assumption is made that each type of operation has a given static and deterministic machine set. However, in practice the machine set is mostly neither static nor deterministic. As a consequence, approaches to the general FJSP will fail to solve the practical case.

The FJSP is one of the most important combinatorial optimisation problems. It has been proven to be a NP-hard problem. Usually FJSP is formularised as a mixed binary integer linear programming problem (MBILP) (Demir and Kürşat İşleyen, 2013) in which the machine allocation variables are binary and the starting time of the operations are continuous. There are some exact methods to solve the MBILP problem, such as branch and bound (B&B) (Brucker et al, 1994; Pan and Chen 2005), cutting-plane, branch and cut (Karimi-Nasab and Seyedhoseini 2013), and so

on. Because the size of the practical FJSP can become very large, these methods are not able to find an optimal solution in a reasonably short time. In order to shorten the computing time, researchers are trying to find an approximate optimal solution instead of the exact solution. Hence, a variety of heuristic procedures and meta-heuristic procedures have been used. Decision rules are typical heuristic procedures (Holthaus and Rajendran 1997). The meta-heuristics procedures include local search (Murovec and Šuhel 2004), tabu search (Bożejko and Makuchowski 2009), simulated annealing (Satake et al. 1999), genetic algorithms (Qing-dao-er-ji and Wang 2012), ant colony algorithms (Rossi and Dini 2007), particle swarm algorithms (Lin et al. 2010), and so on. In addition, considering the stochastic factors, simulation is often combined with these procedures to solve the problem. There are two ways to combine the optimisation approaches with the simulation method. One way is to embed the optimisation approaches in the simulation. The optimisation approaches make decisions during the simulation. An example can be found in Balin's work (Balin 2011). Another way is to embed the simulation in the optimisation approaches (Gholami and Zandieh 2009). Running the stochastic simulation many times can help to evaluate candidate solutions in the stochastic environment. The solutions will be improved based on the evaluation (Zhang and Rose 2013). However, in order to obtain good evaluation results, the simulation has to run lots of times for each candidate solution. The simulation runs are very time-consuming and the procedure becomes very slow.

Our study will focus on this practical FJSP with both dynamic and stochastic machine sets. The problem is denoted by FJSP* in this paper. Because the problem involves stochastic factors, we adopt a simulation-based meta-heuristic procedure to solve the problem. To speed up the solution procedure the stochastic simulation is replaced with a deterministic simulation model and a stochastic process generator. The paper is structured as follows. In Section 2, we address the dynamic and stochastic features of the machine sets in details. Section 3 introduces the improved simulation-based meta-heuristic procedures to solve the special FJSP. Experiments are carried out in Section 4 and the paper is concluded in Section 5.

2 Description of FJSP*

2.1 Dynamic Machine Sets

We use an example to illustrate the dynamic machine sets. Given that two machines A and B are able to process one operation and the location of the concerned semiproduct is close to machine A but far away from B, it is inappropriate to carry the semiproduct to machine B. In this case, the machine set for the operation at the decision point actually includes machine A only which is determined by the location of the semiproduct. The current location of the semiproduct depends on the machine allocation decisions made for previous operations. In other words, the previous decisions we made decide upon the machine set of the current operation. Thus, the machine set is changing dynamically.

A machine network is used to map the dynamic relationship between the operations and the machines for each type of product, as shown in figure 1. The nodes denote the machines. The machines are arranged in the order of the operations. The machines that can process the same operation are put in one column. The arrows

originating from one machine point to the alternative machines which can process the next operation if the current operation is processed on the machine.

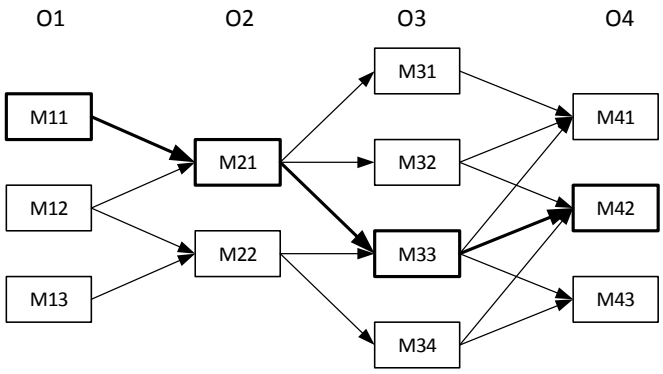

Figure 1: *Machine network for one type of product*

2.2 Stochastic Machine Sets

Many reasons lead to the stochastic machine sets, such as breakdowns, maintenance, utilisation balancing strategies, and so on. We consider two stochastic events, breakdown and maintenance, for each machine. Because the maintenance activities are usually carried out according to the maintenance plan, we assume that the time between two successive maintenance activities follows the normal distribution; the time to maintain follows the normal distribution, too. The maintenance activities are carried out only if the machines are idle. For the machine down times, we assume their probability density functions are piecewise functions,

$$f_B(b;\xi,\lambda,k) = \begin{cases} (1-\xi)\dfrac{k}{\lambda}\left(\dfrac{b}{\lambda}\right)^{k-1} e^{-(b/\lambda)^k}, b > 0 \\ \xi \quad , b = 0 \\ 0 \quad , b < 0 \end{cases} \tag{1}$$

where ξ is the probability that the machine does not break down; k is the shape parameter; λ is the scale parameter. The function is depicted in Figure 2. This function integrates the breakdown time with the breakdown probability. The breakdown time follows the Weibull distribution while the time is greater than 0. The breakdown time "0" means that the machine won't break down at sample points. If the machine breaks down while it is processing one job, we assume that the job will be blocked there and resumed after the machine is fixed.

Note that the probability that the machine breaks down is related to the time difference between current time and the last breakdown time. Usually the probability will go up as the time difference increases because the newly repaired machine has a

higher reliability than the machines which have been running for a long time. Thus ξ can be computed as,

$$\xi = (1-\upsilon)(1-(\tau-\tau_l)/\delta) \tag{2}$$

where υ is the average probability of machine breakdowns, τ is current time, τ_l is the last break time, δ is the remaining life of the machine. Note that υ is not the times of breakdowns during certain period. It is computed from data which is sampled in the fixed time interval. If during the interval, the machine does not break down, the sample data is 0. If the machine is broken, the sample data is the repair time. The histogram of this data set can be found in figure 2.

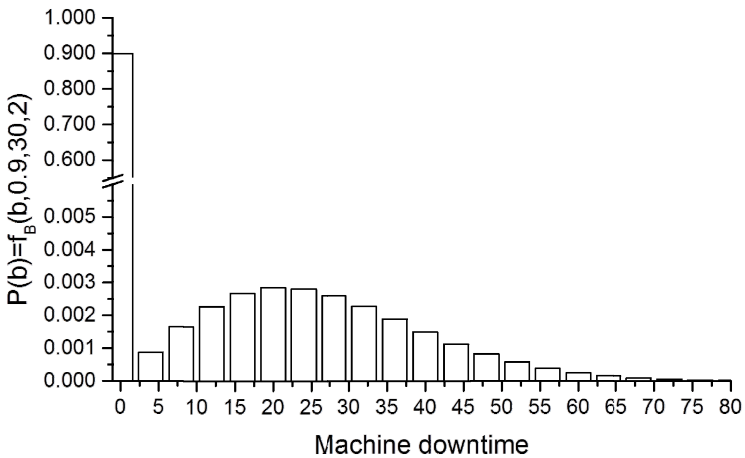

Figure 2: *Probability density function of the machine down time*

3 Simulation-based Meta-heuristic for FJSP*

3.1 Meta-heuristic for FJSP*

A basic metaheuristic procedure contains one loop and three components. The components are initial solution generation, solution evaluation, and solution improvement, as shown in figure 3. The solution evaluation and the solution improvement are carried out repeatedly in the loop. Different meta-heuristics have different solution improvement components. In our study, we will adopt genetic algorithms and ant colony algorithms to solve the problem, respectively. Both the genetic algorithms and ant colony algorithms have population-based improvement components. Because the improvement components in these two algorithms (Qing-dao-er-ji and Wang 2012; Rossi and Dini 2007) are already very mature, we won't discuss them here. The solution evaluation will be addressed in the next section. So the only component left is the generation of the initial solution. The most important issue is to design the appropriate structure of the solution.

From figure 1, we know that each job has only one operation flow (i.e., a series of operations), but possibly has several optional processing paths (i.e., a series of machines). The task of assigning jobs to machines can be transformed into selecting a processing path from the optional paths for each job. Thus, the problem is equivalent to finding an optimal combination of all jobs' processing paths. The solution is a set of selected paths. The solution to the sequencing task can be easily depicted as a set of position indices of operations. Then, the structure of the solution to FJSP* is a combination of the selected path set and the operation position set.

3.2 Improved Simulation-based Solution Evaluation

As we mentioned before, because the problem involves stochastic factors, we adopt a simulation-based metaheuristic procedure to solve the problem. To speed up the procedure the stochastic simulation is replaced with a deterministic simulation model and a stochastic process generator, as shown in figure 3. In the deterministic simulation, the parameters, such as processing times, release times, setup time, and so on, are average values. Thus they are deterministic. The breakdown and the maintenance won't be considered. The deterministic simulation runs only once and the simulation results will be adjusted according to the outputs of the stochastic process generator. The adjusted results are returned to the evaluation module. The number of simulation runs is always one, thus the procedure is speeded up considerably.

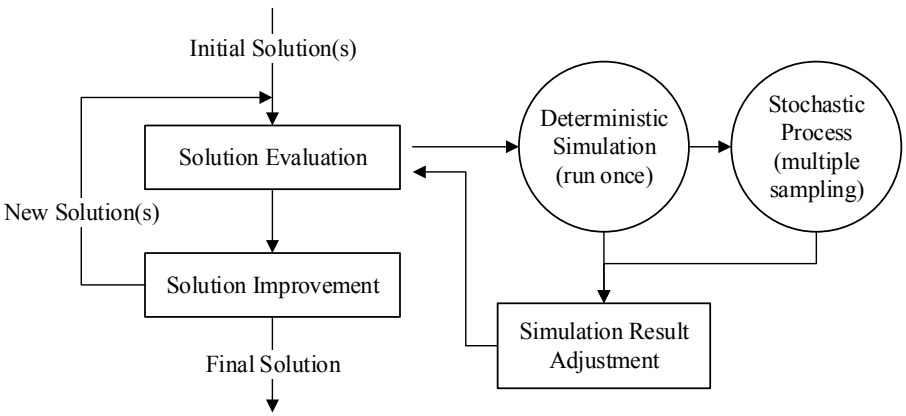

Figure 3: *Improved simulation-based meta-heuristic procedure*

The stochastic process generator and the result adjustment are the kernel of our study. The deterministic simulation records each job's start/end times at each machine. The stochastic process generator generates and lists all stochastic events in the period of the simulation for all machines. We adjust the simulation results, i.e., start times of the jobs, to remove the conflicts caused by the stochastic events. For the same reason, the generator has to generate the event list many times and the adjustment procedure also has to be carried out the same number of times. Finally, we obtain several groups of adjusted simulation results. In our study, the objective is to minimise the makespan. The average makespan is returned to the metaheuristic.

Because the event sampling is very simple and the adjustment procedure is relatively fast, the whole metaheuristic procedure is speeded up considerably compared to a traditional multi-run simulation-based meta-heuristic.

3.3 Stochastic Process Generator and Simulation Result Adjustment

The stochastic process generator is responsible for creating the stochastic events and activities. The maintenance activities are generated in a sequential manner. First, we sample a time period from the stochastic variable of time to next maintenance (TTNM). Current time plus the time period will be the starting time of the first maintenance activity. Then, we sample another time period from the stochastic variable of time to maintenance. This time period is the duration of the first maintenance activity. Next, we sample another time period from the TTNM. The end time plus the time period will be the starting time of the second maintenance activity. The procedure will stop as soon as the starting time of a new maintenance activity is out of scope. The breakdown activities are generated in a fixed-interval manner. The sampling is carried out at some fixed time points. The interval between two successive sample points is fixed and equal to the sample interval used when obtaining the breakdown probability. The sample point is the starting time of the breakdown activity. The sampled time period is the duration of the breakdown activity. If the sampled time period is 0, the activity won't happen.

If we put these generated stochastic activities into the schedule built by the simulation, and replace the other parameters, such as processing times, release times, setup time, and so on, with randomly generated values, there may be some conflicts. For example, a broken machine is processing an operation. Thus we have to remove these conflicts. In our study, the conflict operations will be put off and the related successors are also adjusted. Then a non-conflict schedule is obtained. However, the schedule may not be optimal anymore. Thus a critical-operation adjustment (COA) procedure is introduced. The procedure (algorithm 1) will adjust the non-conflict schedule further and try to achieve a better performance. The COA always focuses on the critical operations, for instance in our study they are the latest finished operations, and try to adjust these operations. After adjustment, the critical operations may not be critical and other operations may become critical. The COA works in a recursive way. The procedure is as follows. If one critical operation cannot be adjusted, we will move to its' predecessor.

4 Experiments

We create a flexible job shop model to evaluate the proposed approach. The objective is to minimise the makespan. In the model, there are 8 machines and 2 types of jobs, type A and B. Type A contains 3 operations while type B contains 4 operations. Each operation has 1 to 4 alternative machines. The operations need different amounts of time if they are processed on different machines. The machine sets for the operations are heavily overlapping. The shop plans to produce 13 jobs of type A and 15 jobs of type B. So there are 99 operations in total. We assume that the processing times follow the normal distribution. The release times and the setup times are constant. Four experimental cases are designed according to the metaheuristics and the evaluation components. Two metaheuristics are involved

including genetic algorithms (GA) and ant colony algorithm (Ant). SS denotes the classical stochastic simulation. DS+SP denotes our improved evaluation. The results are shown in Table 1. From Table 1 we can see that no matter which metaheuristic we adopt the improved approach is always faster than the SS approach while the objective values we achieve are very close.

Algorithm 1:

```
void adjust() {
      while (true) {
            if (!adjust(getCriticalOperation())) {
                  break;
            }
      }
}
boolean adjust(Operation operation) {
      if (isAdjustable(operation)) {
            adjustOperation(operation);
            return true;
      } else if (!operation.isFirstinJob()) {
            return adjust(operation.getPredecessor());
      } else {
            return false;
      }
}
```

Table 1: *Comparisons among different meta-heuristics and evaluations*

Meta-heuristic	Evaluation	Mean Objective (hours)	Mean Run Time (min)
GA	SS	21.9	17.2
	DS+SP	23.5	6.7
Ant	SS	26.1	21.1
	DS+SP	28.7	8.3

5 Conclusion

The difficulties raised by the dynamic machine sets are relieved by converting the machine allocation problem to a path selection problem. The stochastic machine sets may cause unstable schedule performance. This problem is reformulated as a new optimisation problem which tries to minimise the expected value of the objective (makespan) in stochastic situations. Simulation is always a good way to analyse stochastic systems. But when using the simulation in meta-heuristic search algorithms to optimise the performance of stochastic systems, the simulation shows its weakness. The simulation in the meta-heuristic search usually plays the role of an evaluator in stochastic situations. Our study implements this function by means of a stochastic process generator and an adjustment procedure. From the experiment results, the computing time is dramatically reduced while the performance of the schedule decreases only a little.

References

Balin, S.: Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Information Sciences 181 (2011) 17, pp. 3551–3569.

Bożejko, W.; Makuchowski, M.: A fast hybrid tabu search algorithm for the no-wait job shop problem. Computers & Industrial Engineering 56 (2009) 4, pp. 1502–1509.

Brucker, P.; Jurisch, B.; Sievers, B.: A branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics 49 (1994) 1–3, pp. 107–127.

Demir, Y.; Kürşat İşleyen, S.: Evaluation of mathematical models for flexible job-shop scheduling problems. Applied Mathematical Modelling 37 (2013) 3, pp. 977–988.

Gholami, M.; Zandieh, M.: Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. Journal of Intelligent Manufacturing 20 (2009) 4, pp. 481–498.

Holthaus, O.; Rajendran, C.: Efficient dispatching rules for scheduling in a job shop. International Journal of Production Economics 48 (1997) 1, pp. 87–105.

Karimi-Nasab, M.; Seyedhoseini, S.M.: Multi-level lot sizing and job shop scheduling with compressible process times. European Journal of Operational Research 231 (2013) 3, pp. 598–616.

Lin, T.-L.; Horng, S.-J.; Kao, T.-W.; Chen, Y.-H.; Run, R.-S.; Chen, R.-J.; Lai, J.-L.; Kuo, I.H.: An efficient job-shop scheduling algorithm based on particle swarm optimization. Expert Systems with Applications 37 (2010) 3, pp. 2629–2636.

Murovec, B.; Šuhel, P.: A repairing technique for the local search of the job-shop problem. European Journal of Operational Research 153 (2004) 1, pp. 220–238.

Pan, J.C.-H.; Chen, J.-S.: Mixed binary integer programming formulations for the reentrant job shop scheduling problem. Computers & Operations Research 32 (2005) 5, pp. 1197–1212.

Qing-dao-er-ji, R.; Wang, Y.: A new hybrid genetic algorithm for job shop scheduling problem. Computers & Operations Research 39 (2012) 10, pp. 2291–2299.

Rossi, A.; Dini, G.: Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. Robotics and Computer-Integrated Manufacturing 23 (2007) 5, pp. 503–516.

Satake, T.; Morikawa, K.; Takahashi, K.; Nakamura, N.: Simulated annealing approach for minimizing the makespan of the general job-shop. International Journal of Production Economics 60–61 (1999), pp. 515–522.

Zhang, T.; Rose, O.: Simulation-based overhead-crane scheduling for a manufacturing plant. In: Pasupathy, R.; Kim, S.H. (Eds.): Proceedings of the 2013 Winter Simulation Conference. Washington D.C.: 2013, pp.2633-2642.