Simulation in Produktion und Logistik
Entscheidungsunterstützung von der Planung bis zur Steuerung
Wilhelm Dangelmaier, Christoph Laroque & Alexander Klaas (Hrsg.)
Paderborn, HNI-Verlagsschriftenreihe 2013

Scheduling in a Flexible Job Shop with Continuous Operations at the Last Stage

Ressourceneinsatzplanung in einer flexiblen Werkstattfertigung mit Stetigbetrieb im letzten Arbeitsschritt

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

Abstract: This paper presents a special flexible job shop scheduling problem which has continuous-operation constraints at the last stages. In other words, jobs at the last stage which need the same machine must be processed one by one without intervals. An integrated approach combining the simulation of production processes and the genetic algorithm is used to solve the problem. The processing paths of all jobs are decision variables and designed to be one individual in the genetic algorithm. The simulation with the reverse process flows evaluates the feasibility and fitness values of jobs. A satisfactory solution can be got after evolution. Finally, we apply the proposed approach to one steel-making plant.

1 Introduction

The Flexible Job-shop Scheduling Problem (FJSP) is an extension of the classical job shop scheduling problem which allows a job to be processed by any machine from a given set. The tasks are (1) to assign each job to a machine and (2) to sequence the jobs on the machines, so that certain objectives are achieved. Flexible job-shop scheduling problem is one of the most important combinatorial optimization problems, and is kind of NP-hard problem. Hence, a variety of heuristic procedures and meta-heuristic procedures have been applied to solve these problems and a near optimal schedule can be found in a reasonable time. Dispatching rules(Tay and Ho 2008) and beam search(Wang and Yu 2010) are two typical heuristic procedures. The meta-heuristics procedures includes local search(Yazdani, Amiri et al. 2010), tabu search(Li, Pan et al. 2010), simulated annealing(Xia and Wu 2005), genetic algorithm(De Giovanni and Pezzella 2010; Zhang, Gao et al. 2011; Teekeng and Thammano 2012), ant colony algorithm(Rossi and Dini 2007) and particle swarm algorithm(Moslehi and Mahnam 2011).

In this paper, we handle a special FJSP with continuous operations at the last stages. The problem is abstracted from the steel-making industry. When using the meta-heuristic procedures to solve the general FJSP, initial feasible solutions and new

feasible solutions are easy to be obtained. In the most case, arbitrary allocation of machines and sequencing can be a feasible solution. However, if using the meta-heuristic procedures to solve this problem, it is difficult to meet the continuous constraints while we try to obtain initial feasible solutions and generate new feasible solutions. For this special problem, Tang, Liu et al. (2000) generate a rough schedule firstly, which meets the continuous constraint but may contain machine conflicts. Then they use a mathematical programming model to remove the conflicts. The allocation of machines is not counted as decision variable in the scheduling problem. Atighehchian, Bijari et al. (2009) combine ant colony algorithm with non-linear optimization to solve the problem. The ant colony algorithm is used to assign machines to jobs and get the sequence of jobs on each machine. Then the non-linear programming model is used to arrange the starting time of each job on every machine. The continuous constraint is met in the no-linear programming model. Both of them take hierarchical ways (two steps) to solve the problem.

We present an integrated approach in this paper. The simulation and the genetic algorithm combine to solve the problem. The processing paths of all jobs are decision variables and designed to be one individual in the genetic algorithm. The simulation with the reverse process flows evaluates the feasibility and fitness values of jobs. A satisfactory solution can be got after evolution. Finally, we apply the proposed approach to one steel-making plant.

2 Problem Description

In this problem, the objective is to minimize the makespan of all jobs. All of jobs have the continuous operation at the last stages. Jobs can only be processed on one specified machine at the last stages. Machines at the last stages process jobs continuously, i.e. jobs at the last stage which need the same machine must be processed one by one without intervals. In other words, there is no time interval between two successive jobs processed on the same machine at the last stage.

The jobs belong to different batches. Jobs in the same batch are processed on the same machine at the last stages and there is no constraint on the sequence of jobs in the same batch. Jobs in the same batch have the same process flow too. The last machine, the number of jobs in each batch and the starting time of each batch at the last stages are given in a batching plan in this problem. The batches must start to be processed exactly at the starting time given in the batching plan. Consequently, each job has to finish at a given time because of the continuous operation at the last stages. That is to say that a strict due date is specified for each job. Moreover, the machines at the last stages cannot stop running and cannot process the job which belongs to other batches after starting to process a batch.

Except that the last stages of all jobs have only one machine, the other stages have several identical machines in parallel. Jobs at these stages may be processed on any one of the machines. Therefore, each job has only one process flow (i.e. a series of operations), but possibly has more optional processing paths (i.e. a series of machines). The task of assigning jobs to machines can be transformed into selecting processing path from the optional paths for each job. The problem turns to find an optimal combination of jobs' processing paths.

In addition, there are no buffers in front of machines and at most one job is allowed to be waiting in the transportation line before machines. A job which has completed on a given machine cannot leave the machine (is blocked) if the preceding job has not yet completed on the next machine and one job is waiting in front of the next machine.

3 Problem Solution

We combine the genetic algorithm with the simulation of production processes to solve the problem. Processing paths of all jobs make up one individual (solution) in the genetic algorithm and evolve from generation to generation in order to achieve the scheduling objective. The simulation with the reverse process flow is embedded in the evolution and controlled by the genetic algorithm. When evaluating an individual, the genetic algorithm starts the simulation in which jobs are processed following the processing paths represented by the evaluated individual. The function of the simulation is to provide the fitness value of the current individual for the genetic algorithm. Figure 1 shows the mechanism of the approach.

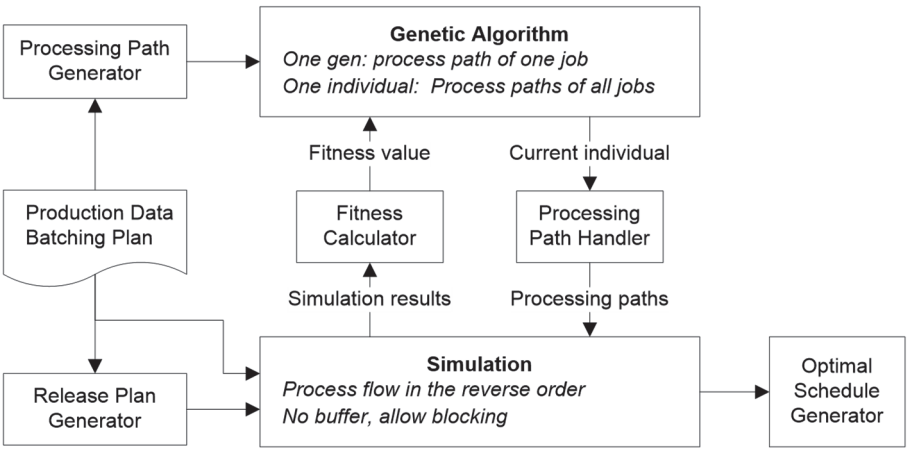

Figure 1: *Mechanism of the combination of simulation and genetic algorithm*

3.1 Genetic Algorithm

In the general FJSS problem, the decision variables are starting time of each job at each stage and binary variables which represent if machines are assigned to jobs. In this problem, the starting time of jobs at the last stages is given. The starting time at other stages can be calculated according to the processing paths under constraints. Besides, we use the selections of processing paths to replace the selections of machines. Thus, the processing paths are the only decision variables, i.e. gene. We map the problem onto the genetic algorithm and the relationship is shown in Table 1. According to the production layout and the batching plan, the processing path generator creates all possible processing paths for each job and assigns a unique ID for each path. The possible processing paths are the foundations of reproduction of individuals. A gene is one decimal number, i.e. path ID, represented one selected

processing path for one job. The processing paths of jobs in the same batch make up one chromosome and these chromosomes compose one individual. Thus, the number of chromosomes in one individual equals to the number of batches. The amount of gene in one chromosome equals to the amount of jobs in the batch.

Table 1: *Relationship among the optimization, the genetic algorithm and the problem*

Optimization	GA	The Problem
Decision variable	Gen	Processing path of each job
Variable group	Chromosome	Processing paths of jobs in the same batch
Feasible solution	Individual	Processing paths of all jobs
Objective	Fitness value	*M–makespan*, M is a big enough number
Constraint	Constraint on generating and reproducing individuals	Continuous constraint, due date, and other FJSS constraints
Process of Solving the problem	Evolution of the population	Looking for the best combination of all jobs' processing paths

The fitness proportionate selection, also known as roulette wheel selection, is used to select potentially useful solutions for reproduction. The probability of individual selection is proportionate to the fitness values. A random number $\varepsilon \in [0,1]$ is chosen. If the *k*-th individual meets the following condition (Eq. 1), the individual will be selected.

$$\sum_{i=1}^{k-1} f_i / \sum_{i=1}^{N} f_i < \varepsilon \le \sum_{i=1}^{k} f_i / \sum_{i=1}^{N} f_i \tag{1}$$

where f_i is the fitness value of the *i*-th individual and *N* is the population size. The selection operation selects pop-sized individuals into the next generation. The individual with higher fitness may be selected more times. The rest will drop out. The dropping rate varies from generation to generation due to the roulette wheel selection. The crossover and mutation will carry out among these selected individuals. The number of individuals taking part in the crossover and mutation is specified by the crossover probability and mutation probability.

The crossover takes place between two matching chromosomes in two different individuals, shown in figure 2. The number and position of genes for crossing over on the chromosomes are not limited, because one chromosome indicates the processing paths of jobs in one batch and one job's processing path is suitable for other jobs in the same batch. The mutation is subject to the restriction of possible

processing paths of each job and alters the gene value (processing path) to one of other possible gene values (other possible processing paths) , so as to meet the constraint on process flows. Figure 3 shows the mutation.

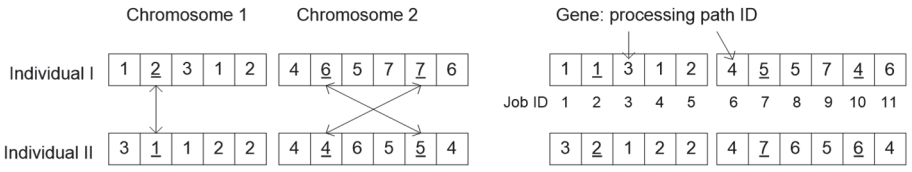

Figure 2: *Crossover operations (a) before crossover (b) after crossover*

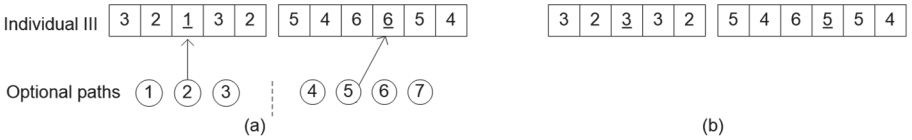

Figure 3: *Mutation operations (a) before mutation (b) after mutation*

In practice, the main issue of the genetic algorithm is to make sure that individuals meet the constraints. The general way is to modify the unfeasible individuals which do not meet the constraints after generated. In other words, the constraints are considered in the process of generating and reproducing individuals.

Algorithm 1: *Genetic algorithm*

```
Step 1: Generate initial population
Step 2: Evaluate each individual
     while the current individual (CI) is unfeasible
          Discard CI
          if CI is generated randomly
               Generate a new individual (NI) randomly
          else if CI is generated by crossover (cro.)
               Generate NI by cro. between the same parents
          else if CI is generated by mutation
               Generate NI by mutation on the same parent
          end if
          Replace CI with NI
     end while
Step 3: if the end condition is met, goto Step 5
Step 4: Reproduce the offspring, goto Step 2
Step 5: Stop and return the best individual
```

In this problem, it is difficult to modify the unfeasible individuals (processing paths) to meet the constraints, especially the continuous constraints. So we put the constraints into the process of calculating the fitness value. The feasibility of individuals is determined over the period of calculating the fitness value. The unfeasible individual will be discarded directly and not participate into reproduction.

A new individual is generated to replace the unfeasible individual. The number of unfeasible individual varies from generation to generation due to the randomization of the crossover and mutation. If the individual is feasible, we get the fitness value after calculating. In this paper, the simulation with reverse process flow is introduced to evaluate the feasibility and fitness value of individuals (see details in section 3.2). The genetic algorithm is outlined in Algorithm 1.

3.2 Simulation of production with reverse process flows

The simulation with reverse process flow is introduced to evaluate the feasibility and fitness values of individuals. The feasible individual is the individual which meets all of constraints in the problem. For the general FJSS problem, there are following constraints: each job at one stage is assigned to exactly one machine; the starting time of a job at one stage must not be earlier than the point at which the job is completed at the previous stage; a machine can process at most one job at a time. These constraints can be satisfied by using some simple rules in the simulation.

In order to meet the continuous constraints at the last stages, we build the production model with the reverse process flows, shown in Figure 4 (a) to (b). The last stages become the first and jobs are processed from the last stages to the first stages. The continuous operations convert to the job releasing with fixed time intervals which equal to the processing time at the last stages. A release plan is generated from the batching plan by the release plan generator before the simulation starts.

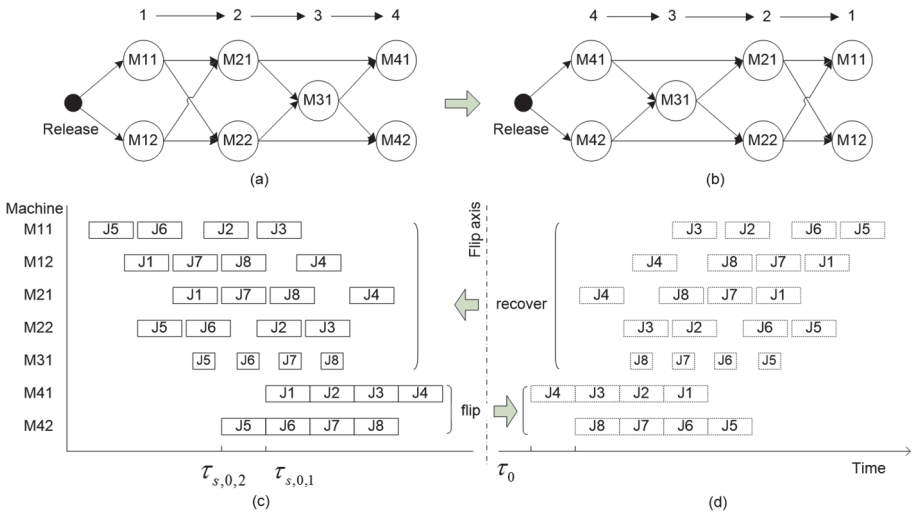

Figure 4: *Production model in the reverse order: (a) normal process flow; (b) reverse process flow; (c) normal schedule; (d) reverse schedule*

The release plan is input into the simulation. Then the simulation runs following the processing paths (reverse) represented by the evaluated individual. Once one job is blocked at the first stage (reverse), which means that the continuous constraint is not satisfied, the individual is unfeasible. Otherwise, the individual is feasible. After simulation, the makespan can be got easily as well as the fitness value.

In the coming paragraph the way how to generate the release plan is presented. The batching plan consists of the following fields: batch ID, product family, job number, last machine, starting time at the last machine. Firstly, the generator gets the completion time of all jobs at the last stages. And then it flips all of the completion time about an arbitrary time axis ($\tau = \tau_0$) to get symmetrical mirror time, i.e. starting time at the first stages in the reverse process flow, shown in Figure 4 (c) to (d). Equation 2 and 3 are for calculating the starting time (reverse),

$$\tau_{c,i} = \tau_{s,0} + (i-1)p \tag{2}$$

$$\tau'_{s,i} = 2\tau_0 - \tau_{c,i} \tag{3}$$

where $\tau_{c,i}$ is the completion time of the *i*-th job in a batch at the last stage; $\tau_{s,0}$ is the starting time of the batch at the last stage; *p* is the processing time of the job at the last stage; $\tau'_{s,i}$ the starting time (reverse) of the *i*-th job. The starting time (reverse) can be treated as the releasing plan directly which will be input into the simulation while the simulation is running.

The completion time (reverse) τ'_c for each job on each machine are recorded while the simulation is running. After the simulation, we flip all of the completion time about the specified time axis ($\tau = \tau_0$) to get symmetrical mirror time, i.e. normal starting time τ_s at each machine, shown in Figure 4 (d) to (c). Then the normal schedule is obtained. Equation 4 is for calculating the starting time.

$$\tau_s = 2\tau_0 - \tau'_c \tag{4}$$

4 Application

The proposed approach is applied to a steel plant in China. In the steel plant the billets or slabs are produced from the hot metal (molten metal) through three operations successively, including primary refining (BOF), secondary refining and continuous casting (CC). At the first two stages, the hot metal is contained in ladles and processed one ladle by one ladle. One ladle of hot metal is one job in this scheduling problem. There are two sub operations in the secondary refining stage, LF and RH. According to the steel grade of the product, some jobs only need to be processed on LF and some need both of them. Continuous casting is the last stage where the hot metal is poured into a mould and solidified into a billet or slab. The hot metal in the same batch must be poured out continuously, i.e. when a ladle turns empty the successive ladle must already arrive and start to pour the hot metal in relays. If a ladle arrives late, the continuous casting operation will be interrupted and the related equipments need to setup again. The setup takes much time. The interruption also causes unnecessary waste. Therefore, the interruption must be avoided. The layout of the plant is shown in figure 5.

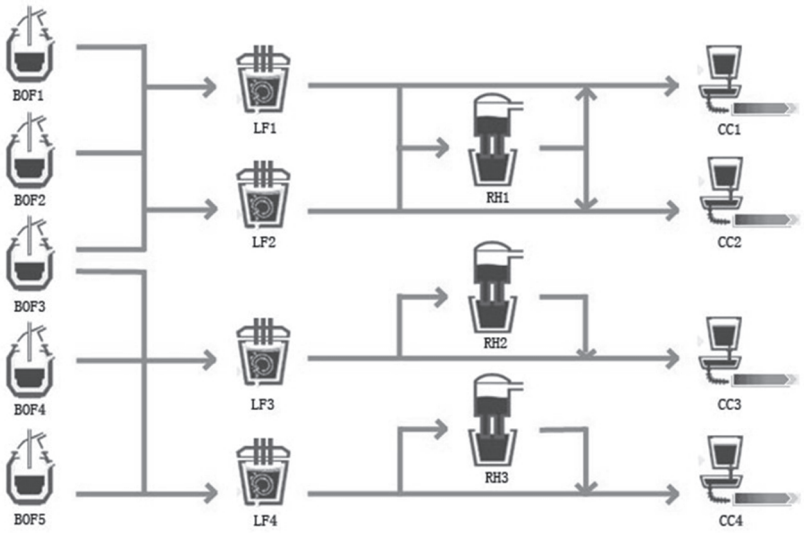

Figure 5: *Layout of the steel plant*

We accept one batching plan of one shift, shown in Table 2, and use the proposed approach to generate one near optimal static schedule. The parameters of the genetic algorithm are set as follows – generation number: 60, population size: 20, crossover probability: 0.8, mutation probability: 0.1.

Table 2: *Batching plan of one shift*

Batch ID	Steel grade	Number of jobs	Caster	Starting time of the batch	Secondary refining
Lot1	LSt12	8	CC1	19:12:00	LF+RH
Lot2	St12	9	CC2	16:56:00	LF
Lot3	DX52D	8	CC3	19:00:00	LF
Lot4	SPHD	9	CC4	17:18:00	LF+RH

The objective (makespan) is tending towards stability at 9.18 hours (shown in Figure 6). It takes 40 seconds to finish the evolution of 60 generations (CPU: 2.5GHz, Memory: 2.0GB). The final schedule is shown in Figure 7 in Gantt chart form. From Figure 7, we can see that (1) the batches start at the given time at the last stages; (2) the continuous constraint is satisfied; (3) there are no conflicts over machines.

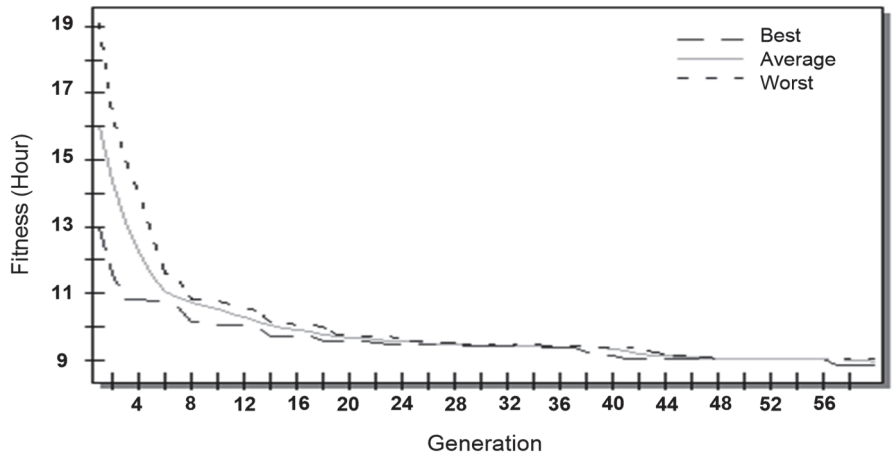

Figure 6: *Convergence during the evolution*

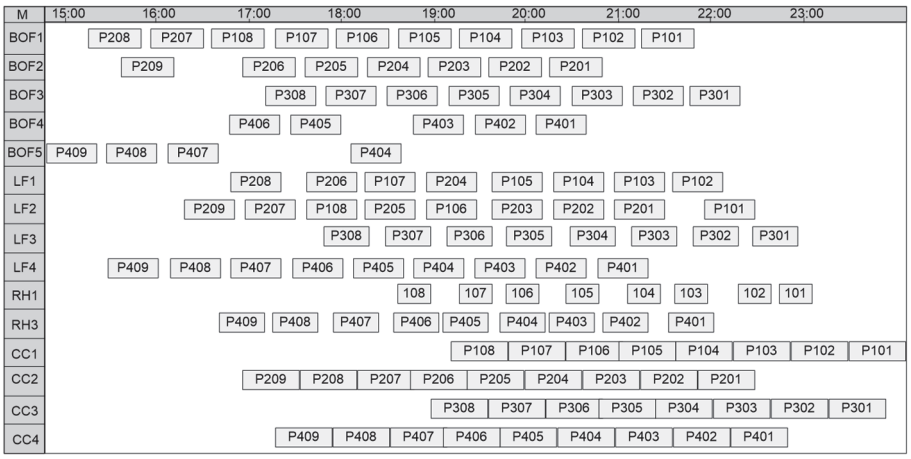

Figure 7: *Gantt chart of the final schedule*

5 Conclusion

We combine the genetic algorithm with the simulation of production processes to solve the FJSS problem with the continuous operation at the last stages. The gene indicates the processing path of the job. The selection of machines for each job at each stage in the problem is replaced by the selection of the processing path, so the crossover and mutation are simplified. The constraints are considered in the process of calculating the fitness value by the simulation. All of constraints in the general FJSS problem can easily covert to the simple rules in the simulation model. The continuous constraint is satisfied by reversing the process flows in the simulation model. The feasibility of individuals is determined by the simulation. Unfeasible individuals will be discarded directly so as to avoid complex modification to the unfeasible individuals. The fitness value of the feasible individual is easy to get after

simulation. The approach is applied to a steel plant. A static schedule made in an acceptable time meets the continuous constraints and due date. However, the proposed approach has an obvious drawback. The robustness of the schedule is poor. For example, when a disturbance takes place, e.g. breakdown, the schedule may not meet the continuous constraints any more. In this case, emergency measures in a dynamic scheduling system will adjust the in-process jobs to ensure that the last stages can still operate continuously later. Then the remaining unfinished jobs will be rescheduled. These contents will be included in our future works.

References

Atighehchian, A.; Bijari, M.; Tarkesh, H.: A novel hybrid algorithm for scheduling steel-making continuous casting production. Computers & Operations Research 36 (2009) 8, pp. 2450-2461.

De Giovanni, L.; Pezzella, F.: An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem. European Journal of Operational Research 200 (2010) 2, pp. 395-408.

Li, J.-q.; Pan, Q.-k.; Liang, Y.-C.: An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. Computers & Industrial Engineering 59 (2010) 4, pp. 647-662.

Moslehi, G.; Mahnam, M.: A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. International Journal of Production Economics 129 (2011) 1, pp. 14-22.

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.

Tang, L.; Liu, J.; Rong, A.; Yang, Z.: A mathematical programming model for scheduling steelmaking-continuous casting production. European Journal of Operational Research 120 (2000) 2, pp. 423-435.

Tay, J. C.; Ho, N. B.: Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers & Industrial Engineering 54 (2008) 3, pp. 453-473.

Teekeng, W.; Thammano, A.: Modified Genetic Algorithm for Flexible Job-Shop Scheduling Problems. Procedia Computer Science 12 (2012) 0, pp. 122-128.

Wang, S.; Yu, J.: An effective heuristic for flexible job-shop scheduling problem with maintenance activities. Computers & Industrial Engineering 59 (2010) 3, pp. 436-447.

Xia, W.; Wu, Z.: An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computers & Industrial Engineering 48 (2005) 2, pp. 409-425.

Yazdani, M.; Amiri, M.; Zandieh, M.: Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Systems with Applications 37 (2010) 1, pp. 678-687.

Zhang, G.; Gao, L.; Shi, Y.: An effective genetic algorithm for the flexible job-shop scheduling problem. Expert Systems with Applications 38 (2011) 4, pp. 3563-3573.