Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-3 (2015) 893–897

# Integrating Genetic Algorithm with Time Control for Just-In-Time Scheduling Problems

**Junru Chen[a], Wei Weng[a], Gang Rong[b], Shigeru Fujimura[a]**

[a] *Graduate School of Information, Production and Systems, Waseda University, 2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan (e-mail: chen.junru@toki.waseda.jp).*

[b] *State Key Laboratory of Industrial Control Technology, Institute of Cyber-systems and Control, Zhejiang University, Hangzhou, China, 310027*

**Abstract:** To reduce inventory costs and increase customer satisfaction in a real production environment, Just-in-Time (JIT) manufacturing should be combined with more flexibility. Flexibility in manufacturing system ensures improved product quality and mass customization. In this paper, we consider flexible job-shop problem (FJSP) to include several types of flexibility. Prior research on JIT scheduling problems paid little attention to flexible job shop systems. In this paper, an optimization method for FJSP for JIT manufacturing was proposed, genetic algorithm and time control are integrated to improve the JIT performance. Computational simulations and comparisons demonstrate that the proposed method shows more competitive performance than other evolutionary algorithms.



*Keywords:* Production planning and scheduling, just-in-time, flexible job-shop, genetic algorithm, arrival time control.

## 1. INTRODUCTION

Just-In-Time (JIT) manufacturing originated in Japan in the 1950s to reduce inventory costs and satisfy customer demands with on-time deliveries (Józefowska (2007)). Nowadays, intensified competition has driven organizations to offer increased levels of customer service in order to survive. Flexibility is an effective tool for surviving in the new manufacturing environment with extreme uncertainties. Flexibility in manufacturing system ensures improved product quality and mass customization (Singh and Singh (2013)). JIT should be combined with more flexibility to meet the precise requirements of customers. System flexibility could be found in system components, organisation, and control methods (Joseph and Sridharan (2011)). In this paper, the flexible job-shop problem (FJSP) is considered to include several types of flexibility, such as machine, operation, routing and product.

The FJSP is an extension of the classical job shop scheduling problem (JSP), where each operation is allowed to be executed on any machine from a given set, rather than one specified machine. Compared with the classical JSP, the FJSP is much closer to a real production environment and has more practical applicability (K. (2009)). FJSP is proved to be strongly NP-hard (Garey et al. (1976)). Exact algorithms are not effective for the FJSP due to its complexity, so the meta-heuristic methods have become the main solutions to FJSP, such as tabu search (TS) (Mastrolilli and Gambardella (2000)), evolutionary-based approaches (Pezzella et al. (2008), Geem et al. (2001)), and constraint-based approaches (Shaw (1998)). In recent year, evolutionary algorithms have been applied successfully to the FJSP. The basic mechanism of evolutionary algorithms for solving FJSP is to encode the schedule to some form of codes, and then decode the codes and evaluate them during the optimization process. Up to now, many evolutionary algorithms have been studied for FJSP, such as genetic algorithm (GA) (Pezzella et al. (2008)), harmony search (HS) (Geem et al. (2001)), and artificial bee colony (ABC) algorithm (Wang et al. (2012)). In this paper, GA is chosen to solve FJSP for the following reasons: GA can generate high quality solutions in a reasonable time; there are a variety of encoding ways in GA; GA offers various operators which can avoid local minima; GA is naturally parallel, exhibits implicit parallelism (Portmann (1997)), because it evaluates and improves a set of solutions simultaneously rather than a single solution; GA can be combined with other optimization methods to improve the solution.

Among the existing works, little research on FJSP targeted JIT objectives. Most of the studies dealing with the FJSP have been limited to the static problem with constraint relaxation and/or hard system assumptions (Demir and İşleyen (2013)). A sequential approach coupling genetic algorithms and distributed arrival-time control dealing with the FJSP with JIT objectives has been proposed by Rey et al. (2014). The sequential approach firstly gets a solution by GA as the initial solution for time control, and then do the time control part. In the sequential approach, the initial solution got by GA is the best solution without considering time control. There is a big probability that applying time control to this initial solution can make the result worse. In such a way, the time control can


10.1016/j.ifacol.2015.06.196

hardly improve the initial solution. In this paper, a genetic strategy for FJSP for the JIT objective was proposed. It controls schedules dynamically by integrating genetic algorithm and time control. The integrated method considers the time control part from the very beginning with the machine routing and operation sequence. During the whole optimization process, these three parts will effect each other to get the final solution. In this integrated method, the machine routing part and operation sequence part benefit from the time control part due to the integration of the arrival time of each job, the time control part benefits from the operation sequence and machine routing process executed by the GA. The integrated method takes full advantage of the capability of GA to get optimal results.

The remainder of the paper is organized as follows. Section 2 describes the problem and the objective function. Section 3 introduces the proposed GA. Afterwards, experimental studies are reported in Section 4. Finally, the last section concludes the paper.

## 2. PROBLEM DESCRIPTION

### *2.1 The Flexible Job-Shop Problem*

The flexible job-shop problem (FJSP) can be defined as follows. There are a set of m unrelated machines and a set of n independent jobs. Each job consists of a sequence of ordered operations. Each operation can be processed on any machine selected among a given subset from the set of machines. The processing time of each operation is machine dependent.

The following assumptions are made in this paper: all machines are available at time 0; each machine can not execute more than one operation at a time; each operation must be executed without interruption; the operations of each job are precedence constrained; the setting up time of each machine and the transportation time of each operation can be negligible.

A sample instance of FJSP is shown in table 1. $O_{ij}$ is the $j$th operation of job $J_i$, $M_i$ is the $i$th machine, and the symbol "-" means that the corresponding operation can not be executed on the machine.

Table 1. An example of FJSP

| Job | Operation | $M_1$ | $M_2$ | $M_3$ |
|---|---|---|---|---|
| $J_1$ | $O_{11}$ | 2 | 3 | - |
| | $O_{12}$ | 4 | - | 2 |
| $J_2$ | $O_{21}$ | 1 | 3 | 2 |
| | $O_{22}$ | - | 2 | 3 |
| $J_3$ | $O_{31}$ | 4 | 1 | 3 |
| | $O_{32}$ | 3 | - | 2 |

### *2.2 The Objective Function*

In this paper, the average earliness and tardiness is considered as the optimality criterion for JIT production, denoted as ET

$$ET = \frac{1}{n}\sum_{i=1}^{n}|C_i - D_i| \quad (1)$$

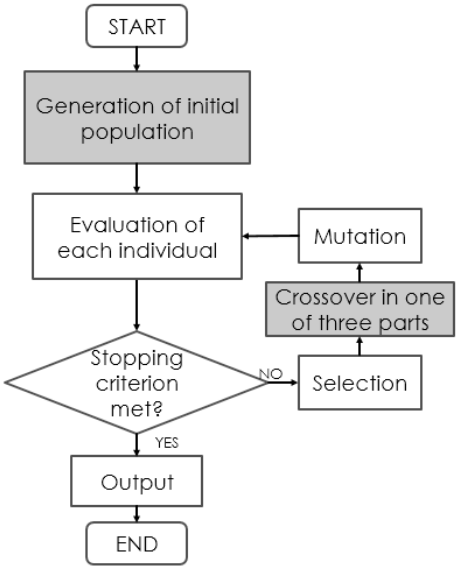


Fig. 1. Algorithmic flow of the proposed GA

where $n$ is the total number of jobs, and $C_i$ is the completion time of job i, and $D_i$ is the due date of job i.

## 3. PROPOSED METHOD

GA is an effective meta-heuristic for solving combinatorial optimization problems. It has been successfully adopted to solve the FJSP. GA processes populations of individuals to get the final solution. Each individual represents a candidate solution for the problem. All these individuals are evolved towards better solutions. After an initial population is generated randomly, each individual is evaluated by the objective function. The algorithm improves the solutions through repetitive application of selection, crossover, and mutation operators. With these operators, individual chromosomes are copied and modified to form the new generation.

The algorithmic flow of the proposed GA is depicted in Fig. 1. The chromosome representation and the crossover part are different from traditional GA. Among current research on FJSP, most chromosome representations of GA are divided into two parts: machine assignment part and operation sequence part, corresponding to two sub problems of FJSP. In this paper, the chromosome representation is divided into three parts to integrate the time control. The chromosome representation will be explained in detail in section 3.1. In the crossover part, the crossover operator is applied in three parts in turn. The crossover operator will be introduced in section 3.2.

### *3.1 Representation of Chromosome*

The chromosome representation is very important for successful implementation of GA. In this paper, the chromosome representation is divided into three parts as shown in Fig. 2: machine assignment part, operation sequence part, and time control part.

In machine assignment (MA) part, sequence of integer values are used. The integer value represents the index of the

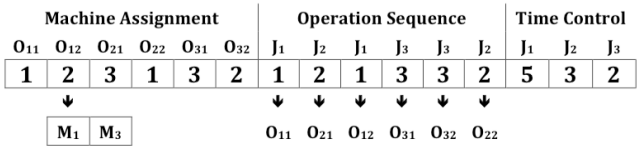


Fig. 2. Chromosome representation

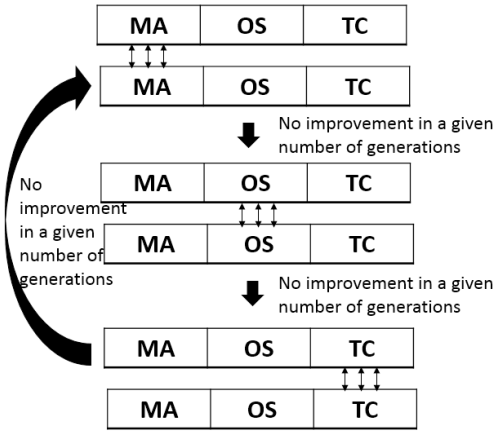


Fig. 3. Crossover in one of the three parts in turn

selected machine for the operation. In operation sequence (OS) part, the operations in one job are represented with the same job index. In time control (TC) part, each value represents the release time of the job. For the chromosome in Fig. 2, $O_{12}$ is executed on $M_3$. According to the OS part, the operation sequence is $O_{11} \rightarrow O_{21} \rightarrow O_{12} \rightarrow O_{31} \rightarrow O_{32} \rightarrow O_{22}$. And the arrival time of $J_1$ is 5.

### *3.2 Genetic Operators*

*Selection* chooses solutions from the population for later recombination. Usually, this decision is based on the fitness of the solutions. Fitter solutions are more likely to be selected to breed the new generation. In this paper, three-sized tournament selection operator was used. In detail, three individuals are selected at random from the population, and the one with the highest fitness is chosen for crossover.

*Crossover* replaces some of the genes in one parent by the corresponding genes of the other to produce a child. The operator should be carefully designed to ensure that the child is better than its parents. In this paper, the crossover operator only modifies one of the three parts of the chromosome in each generation as shown in Fig. 3. Crossover in turn helps keep the good parts of the parents. If the crossover is applied on all the three part at a time, there is a big probability that the child will not inherit the good parts of its parents. In detail, crossover is first applied on MA part. If the current best solution has not been improved for a given number of generations, the crossover operator will be applied on OS part in the next generation. For the MA part and TC part, the two-point crossover proposed by Varela et al. (2003) is applied. In detail, two random genes are selected in the part, and the genes between the selected two genes are exchanged. In the OS part, Precedence Preserving Order-based crossover (POX) of Lee et al. (1998) is used. With POX, we do not

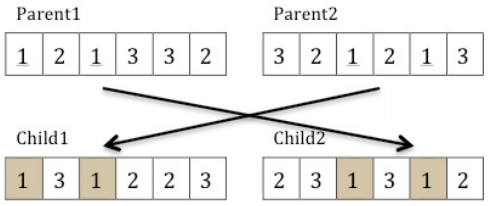


Fig. 4. POX crossover

need to do repairing after crossover. POX generates two children starting from two parents. Firstly, randomly pick several job indexes into a sub job set $S$. Secondly, copy job indexes of OS part which are in the sub job set $S$ from parent 1 to child 1 and from parent 2 to child 2 and keep the locations of these job indexes. Thirdly, copy job indexes of OS part which are not in the sub job set $S$ from parent 1 to child 2 and from parent 2 to child 1 and keep the order of theses job indexes. Fig. 4 gives an example of POX crossover for the problem in Table 1. Job 1 is chosen into the sub job set $S$. It can be found that child1 preserves the locations of job 1 in parent1 and the order of job 2 and job 3 in parent2, and child2 preserves the locations of job1 in parent2 and the order of job2 and job3 in parent1.

*Mutation* maintains genetic diversity of a population. It is usually applied on each chromosome with small probability. In this paper, mutation is applied to three parts of the chromosome separately to enhance the diversity of each part. In the MA part, first an operation is chosen randomly, and the selected machine of the operation is replaced by a random machine from the alternative machine set. In the OS part, a shift mutation is performed. Randomly select a gene and insert it to some other position of the OS part. In the TC part, an exchange mutation is applied. Two randomly selected genes on TC part interchange with each other.

## 4. EXPERIMENTAL STUDY

### *4.1 Experimental Setup*

To test the performance of the proposed genetic algorithm with time control (GATC), the whole procedure is implemented in Java and run on an Intel 2.70GHz Core processor with 4.0GB of RAM.

Table 2. Problem instances

| | njob | nmac | nop | meq | proc |
|---|---|---|---|---|---|
| mk01 | 10 | 6 | 5-7 | 3 | 1-7 |
| mk02 | 10 | 6 | 5-7 | 6 | 1-7 |
| mk03 | 15 | 8 | 10-10 | 5 | 1-20 |
| mk04 | 15 | 8 | 3-10 | 3 | 1-10 |
| mk05 | 15 | 4 | 5-10 | 2 | 5-10 |
| mk06 | 10 | 15 | 15-15 | 5 | 1-10 |
| mk07 | 20 | 5 | 5-5 | 5 | 1-20 |
| mk08 | 20 | 10 | 10-15 | 2 | 5-20 |
| mk09 | 20 | 10 | 10-15 | 5 | 5-20 |
| mk10 | 20 | 15 | 10-15 | 5 | 5-20 |

njob : number of jobs;
nmac : number of machines;
nop : minimum and maximum number of operations per job;
meq : maximum number of equivalent machines per operation;
proc : minimum and maximum processing time per operation;

In the experiment, a well known benchmark set named BRdata set is used. BRdata consists of 10 instances from Brandimarte (1993). The details of each instance are shown in Table 2. The due-date of each job is generated accroding to SL rule from Sridharan and Li (2008).

Due to the natural uncertainty of the algorithms, each instance is run for fifty times to get the average results. Three metrics including the average result of average earliness and tardiness (ET), the average computational time in seconds (CT), and the Just-In-Time rate (JITR) which is defined as

$$JITR = \frac{J_o}{n} \tag{2}$$

where $J_o$ is the number of jobs that are completed on their due dates and $n$ is the total number of jobs.

To evaluate the performance of our GATC algorithm, we compare our computational results with GA (Demira and İşleyen (2014)) and HS (Yuan and Xu (2013)) for their good performance on BRdata. ET and JITR are used as the main comparison metrics.

### *4.2 Performance analysis*

Table 3. Parameters setting

| Parameter | Value |
|---|---|
| Population Size | Number of jobs * 10 |
| Crossover Probability | 0.7 |
| Mutation Probability | 0.3 |
| Termination Criterion | No improvement in 50 iterations |

In this section, the performance of GATC is analyzed and the parameters are given in Table 3.

In Table 4, the GATC is compared with two evolutionary algorithms, including GA and HS. As we can see from Table 4, the GATC compares favorably with the other algorithms on the BRdata. In particular, the GATC outperforms GA in all the instances and outperforms HS in 7 out of 10 instances, in terms of ET. HS gets the best results on mk01 and mk02 in terms of ET, but it performs worse along with the problem size getting larger. GATC outperforms GA in 9 out of 10 instances and outperforms HS in 8 out of 10 instances for the JIT obtained. As for the efficiency, the overall average computational time of GATC is less than GA for half of all the instances and less than HS for the most instances. We can conclude that the GATC is enough for solving some medium to large FJSP instances effectively.

## 5. CONCLUSION AND FUTURE WORK

In this paper, genetic algorithm is integrated with time control for the FJSP with earliness and tardiness criterion. The integrated method offers a better performance than pure version of evolutionary algorithm. This integrated method can be regarded as a kind of framework rather than a single algorithm. The genetic algorithm can be replaced by other evolutionary algorithms. In future research, we will focus on improving the efficiency and stability of the method.

Table 4. Comparison of GATC with existing algorithms on BRdata

| Instance | GATC | | | GA | | | HS | | |
|---|---|---|---|---|---|---|---|---|---|
| | ET | JITR | CT | ET | JITR | CT | ET | JITR | CT |
| mk01 | 2.08 | 0.38 | 2.4 | 2.70 | 0.36 | 3.0 | **1.30** | **0.60** | 3.0 |
| mk02 | 2.34 | **0.38** | 2.4 | 3.12 | 0.26 | 2.4 | **1.80** | 0.35 | 2.0 |
| mk03 | **4.72** | 0.29 | 6.8 | 5.77 | 0.30 | 6.8 | 6.01 | **0.40** | 6.9 |
| mk04 | **2.77** | 0.36 | 3.2 | 3.01 | **0.37** | 0.16 | 2.93 | 0.31 | 3.3 |
| mk05 | **13.65** | **0.13** | 5.0 | 15.49 | 0.12 | 3.0 | 14.2 | 0.06 | 4.8 |
| mk06 | **1.86** | **0.36** | 9.0 | 3.52 | 0.30 | 6.6 | 5.80 | 0.10 | 10.4 |
| mk07 | 12.30 | **0.17** | 6.8 | 12.64 | 0.10 | 15.2 | **7.70** | 0.07 | 10.0 |
| mk08 | **16.13** | **0.13** | 19.2 | 19.58 | 0.11 | 21.2 | 33.00 | 0.02 | 30.4 |
| mk09 | **13.70** | **0.16** | 33.4 | 17.52 | 0.12 | 31.3 | 57.78 | 0.02 | 54.2 |
| mk10 | **12.49** | **0.14** | 52 | 17.09 | 0.12 | 43.2 | 90.79 | 0.00 | 78.3 |

## ACKNOWLEDGEMENTS

The authors acknowledge the support of the Grant-in-Aid for International Collaboration Research (2014) from Kitakyushu Foundation for the Advancement of Industry, Science and Technology (FAIS), Japan, and the National High Technology R&D Program of China (2013AA040701).

## REFERENCES

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3), 157–183.

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

Demira, Y. and İşleyen, S.K. (2014). An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations. *International Journal of Production Research*.

Garey, M.R., Johnson, D.S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2).

Geem, Z.W., Kim, J.H., and Loganathan, G. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2), 60–68.

Joseph, O.A. and Sridharan, R. (2011). Effects of routing flexibility, sequencing flexibility and scheduling decision rules on the performance of a flexible manufacturing system. *The International Journal of Advanced Manufacturing Technology*, 56(1-4), 291–306.

Józefowska, J. (2007). *Just-in-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems*. Springer US.

K., U. (ed.) (2009). *Computational intelligence in flow shop and job shop scheduling*, volume 230 of *Studies in computational intelligence*. Springer.

Lee, K.M., Yamakawa, T., and Lee, K.M. (1998). A genetic algorithm for general machine scheduling problems. *Knowledge-Based Intelligent Electronic Systems*, 2, 60–66.

Mastrolilli, M. and Gambardella, L.M. (2000). Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3(1), 3–20.

Pezzella, F., Morganti, G., and Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers Operations Research*, 35(10), 3202–3212.

Portmann, M.C. (1997). Scheduling methodologies: optimization and compusearch approaches. In *The Planning and Scheduling of Production Systems*. Springer US.

Rey, G.Z., Bekrar, A., Prabhu, V., and Trentesaux, D. (2014). Coupling a genetic algorithm with the distributed arrival-time control for the jit dynamic scheduling of flexible job-shops. *International Journal of Production Research*, 52(12).

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *CP-98, Fourth international conference on principles and practice of constraint programming*, 1520.

Singh, D.K. and Singh, S. (2013). Working with jit requires a flexible approach. *IOSR Journal of Mechanical and Civil Engineering*, 5(6), 24–28.

Sridharan, V. and Li, X. (2008). Improving delivery reliability by a new due-date setting rule. *European Journal of Operational Research*, 186(3), 1201–1211.

Varela, R., Vela, C.R., Puente, J., and Gomez, A. (2003). A knowledge-based evolutionary strategy for scheduling problems with bottlenecks. *European Journal of Operational Research*, 145(1), 57–71.

Wang, L., Zhou, G., Xu, Y., Wang, S., and Liu, M. (2012). An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 60, 303–315.

Yuan, Y. and Xu, H. (2013). An integrated search heuristic for large-scale flexible job shop scheduling problems. *Computers Operations Research*.