Genetic Algorithm on Job Shop Scheduling Problem under the Batch Splitting Environment

Masatsugu SHIIHARA a*

aFaculty of Information Science and Technology, Osaka Institute of Technology, 1-79-1 Kitayama, Hirakata-city, Osaka, 573-0196 Japan

Abstract

This paper suggested improvement of the searching ability in genetic algorithm on job-shop scheduling with batch splitting. Batch splitting reduces make-span because similar batches which occur by splitting can be in processes simultaneously. However, the number of batches increases as a result of splitting the original batches. It may complicate scheduling problems.

So, this paper proposed genetic algorithm on job shop scheduling considered influence by batch splitting. The algorithm generates either of an active schedule or a non-delay schedule according to a gene. It has an influence good for the searching ability because the generating ratio changes dynamically by selection. Numerical experiments show the characteristics of the proposed algorithm.

Key words: Job Shop Scheduling, Batch Splitting, Genetic Algorithm, Non-delay schedule, Numerical experiment.

1. Introduction

This paper developed a GA (Genetic Algorithm) with the high searching ability on job shop scheduling. The algorithm is considering the influence of batch splitting and minimizes the make-span with aim. Many scholars have been investigating the effectiveness of batch splitting for improvement of production schedules.

Kropp and Smunt [1] analyzed the effect of mean flow-time reduction under the flow shop environment. Kanezashi and Mitani [2] suggested a partition method of batches and an application way of GA under the flow shop environment. Shiihara et al. [3] clarified the relation between the number of split and the make-span in dispatching method by approximate expression under the job shop environment. Wagner and Ragatz [4] were also recognized as an effective measure on due date performance in dispatching method under the dynamic job shop scheduling environment.

Batch splitting reduces idle time because similar batches which occur by splitting can be included in processes simultaneously. Reduction in idle time leads to reduction in make-span. Its effect is monotonically increasing as a batch will be small by division. It has an influence also better for other criteria of production schedule. However, the number of batches increases as a result of splitting the original batches. So, setup time may increase even if idle time is reduced by batch splitting. It will be harder to solve a scheduling problems and it will be also complicated shop floor control.

On the other hand, it is not necessary to minimize the make-span in actual manufacturing systems. An adjustment with Master Production Schedule and productive capacity is needed in order to make Material Requirements Planning system. We have to discuss appropriate the number of split in each job. Jeong et al. [5] tried to improve a schedule by batch splitting and setting up the machine before the actual arrival of jobs to that machine. They also proposed repetitive selecting and splitting a batch until make-span was below the required time. The production schedule is improved using left shift method after splitting a batch. However, rescheduling is able to acquire better make-span than improvement by only left shift method though it is necessary longer computing time to generate new schedule. Therefore, this research adopted production scheduling with a GA.

There are many studies aiming at improvement of the searching ability of GA on scheduling problem to minimize make-span. Hirano [6] introduced the way to express genes in a chromosome by the job number. A

*Corresponding author: shiihara@is.oit.ac.jp

Received: April 30, 2012
Accepted: February 15, 2013
A semi-active schedule is obtained certainly from this chromosome. The schedule is converted to an active schedule by left shift method. Ida and Osawa [7] suggested the algorithm for shortening idle time in the schedule instead of left shift method. The algorithm which is called Eshift tries to generate an active schedule by arranging all operations in turn in idle time. They put the shortening method into GA and evaluated the performance of their GA by numerical experiments using benchmark tests. This shortening method confirms whether every operation is able to shift without making make-span be aggravated. If an operation which is possible to shift is found, a new schedule is calculated each time. So, the shortening method needs much calculating time as increasing the number of batch.

In this paper, an efficient GA which takes increase in the number of batch by splitting into consideration was developed and its numerical experiments were put into effect. A gene which is not the job number was appended to the end of chromosome in the proposed GA. The gene decides whether it generates an active schedule by Eshift or it generates a non-delay schedule. The non-delay schedules have a possibility not including the optimum solution, but those can be generated easily and obtained a better make-span on the average. It may be more profitable as jobs are divided into smaller batch. It is expected that search efficiency improves because the respective numbers of individuals fluctuate dynamically as result of selection. Numerical experiments showed the characteristics of the proposed GA.

2. The model description

This paper deals with the static job shop scheduling problem permitted batch splitting. The precondition of the model in detail is as follows.

(1) There are different \( m \) machines in the shop.
(2) There are \( k \) jobs which consist of the different \( k \) kinds and each job has more than one patterns of process order.
(3) Each job has \( p \) operations which are processed different \( p \) machines.
(4) The processing time by each machine for each job is based on uniform distribution.
(5) Setup time isn't considered to estimate decrease of idle time by batch splitting.
(6) Every job is divided into \( n \) batches equally like Figure 1. Therefore it's necessary to deal with \( k \) jobs problem as \( nk \) jobs problem.

3. Proposal of Genetic Algorithm

3.1 Schedule classification of the JSP

Schedule type on a job shop scheduling problem is classified into the following four kinds [8]. This paper focuses on both of the active schedules and the non-delay schedules.

(1) The feasible schedules
These are meaning that the schedules satisfy all constraints.
(2) The semi-active schedules
In these schedules, any operations on any machines can not start earlier without passing other operations.
(3) The active schedules
The active schedules are included in semi-active schedules. In this schedule, any jobs on any machines can not start earlier without shifting other operations to the right.
(4) The non-delay schedules
These are a part of the active schedules. No operations which can be processed exist while idle time on any machines.

Figure 2 is an example of active schedule. But this isn't non-delay schedule because job2 can start to process at machine 2 in the idle time A. The idling time A is shorter than processing time of Job2, so the schedule can't be improved by left shift method.

Figure 3 indicates an example of non-delay schedule. That's different from a schedule on figure 2, and there is no operation which can start to process in every idle time. The non-delay schedules have a
possibility not including the optimal solution. But the non-delay schedules are obtained a better make-span on the average than the other active schedules and they can be generated easily.

3.2 Methods of shortening idle time

There are some coding methods of chromosome for job shop scheduling problem. In the proposed GA, a chromosome is expressed by the batch number [6]. A Gantt chart is drawn according to the batch number written in a chromosome. It is necessary to shorten idle time on the schedule for improving the searching ability of GA. The proposed GA has two types of shortening method as follows.

(1) Type-A

This method is same as Eshift which expands left shift. The shift of the operation occurs by the left shift method when the processing time of the operation is shorter than the idle time. It occurs by the Eshift method even if the processing time of the operation is not shorter. The Eshift is applied after generating a Gantt chart according to a chromosome. The general procedure of Eshift is as follows [7].

(A1) One operation is chosen from a chromosome in turn. If reaching at the end of a chromosome, this method is finished.

(A2) If an idle time doesn't exist in front of the operation, return to step 1.

(A3) The influence of delay on the operation passed by the shift is calculated. Figure 4 is the conceptual diagram which indicated the influence of delay. The operation p0 is passed operation by shifting the operation s0. The operation p1 is the same job number as the operation p0 and is the next process of the operation p0. The machine which processed the operation p1 should process the operation p2 in the next. The operation p3 is the same job number as the operation p1 and is the next process of the operation p1. It's predicted whether starting times are not delayed on both of the operation p2 and p3 by shifting of the operation s0.

(A4) If starting times are not delayed, the operation s0 is rearranged and a Gantt chart and a new make-span are calculated again.

(A5) Return to A1.

This method has much reduction effect of make-span than left shift method, but calculating time which redraws a Gantt chart repeatedly is needed as the number of operations increases.

(2) Type-N

In this type, a non-delay schedule is generated from a chromosome without rearranging. The dispatching method is applied to generating a schedule. The locus of a chromosome is interpreted as the priority of respective operation. This priority ranks operations in the queue instead of SPT or MWKR. A non-delay schedule is obtained easily by repeating the following rules.

(N1) When a machine has finished processing an operation, the operation with the highest priority is chosen in the queue at that machine. If the queue is empty, the machine becomes idle.

(N2) If an operation arrives at a machine during idle time, the operation is run immediately.

There is a possibility that the optimal solution isn't obtained by using Type-N. But this method has the higher effect on the average than Type-A. Moreover, calculating time for generating a schedule is shorter because Type-N isn't necessary to draw a Gantt chart repeatedly. This method becomes so profitable that batch splitting is developed.

3.3 Reorganization of the chromosome

The proposed GA shortens idle time from the original schedule. The generated schedule is different from the schedule generated according to chromosome. The reorganized chromosome based on the improved schedule is taken over in the next generation. When type-A is used, g2 algorithm which is suggested by Ida and Osawa [7] is applied. If type-N is selected, the chromosome is reorganized by sorting batch number with starting time of each operation.

3.4 Design of chromosome

It's originality of the proposed GA to generate two kinds of the schedules which are an active schedule and a non-delay schedule. Their ratio in the population
fluctuates according to the searching situation. Therefore, one gene which doesn’t show job number is appended to the chromosome in the proposed GA. The gene decides whether the proposed GA generates an active schedule or it generates a non-delay schedule. The chromosome is designed like figure 5.

The respective genes in main part are expressed by the batch number [6]. The batch numbers of the \( kn \) kinds have to be written in the \( p \) time respectively in the chromosome. The added gene is “A” or “N”. If it is “A”, a schedule is calculated by Type-A method. When it is “N”, a non-delay schedule is calculated according to Type-N method. After the schedule is estimated, a chromosome is reorganized from the schedule.

In the proposed GA, population consists of two kinds of individuals classified by the last gene on a chromosome. The algorithm has a good influence for the search efficiency because their population ratio changes dynamically by selection.

3.5 Genetic operations

Individuals with active schedules and individuals with non-delay schedules are created into just the same number in initial time. The outlines of genetic operations are as follows.

(1) Crossover
Two individuals chosen from the population at random intersect and order crossover is used. The last gene is disposed of independently with other genes. When the crossover occurs, the genes of two parents are exchanged by a probability of one in two.

(2) Mutation
Two genes chosen at random are swapped by a fixed mutation probability. The last gene isn’t the target.

(3) Selection
When child’s make-span isn’t worse than parent’s, the parent will change with the child whose chromosome is reorganized with the schedule.

4. Evaluation the performance of the proposed Genetic Algorithm

4.1 The preconditions for experiments

The performance and the characteristic of the proposed GA were evaluated by some numerical experiments. The CPU used for some experiments was Intel Core 2 Duo E8500 (3.16GHz). The preconditions for experiments are as follows.

(1) Ten production facilities exist in the shop.
(2) Every job has 10 operations.
(3) The number of jobs is 10.
(4) The processing times are composed by multiplier of 16 and it ranges from 32 to 320.
(5) The numbers of split are 1, 2, 4, 8 and 16.
(6) The population size is 100.
(7) Genetic operations are continued until 100,000 new individuals are generated.
(8) The crossover rate and the mutation rate are made 0.8 and 0.2 respectively.
(9) One thousand tests are prepared at random.

4.2 Comparison of the shortening methods

The performance of each shortening method is evaluated by using one in the 1,000 tests. The make-spans with each shortening method are calculated after 5,000,000 chromosomes were prepared at random. The statistics of type-A are shown in Figure 6. This figure indicates the impact of batch splitting on make-span performance. The average and standard deviation of the make-span will be small as the large number of split.

Table 2 is comparing the make-spans obtained by the respective methods. The other methods are also calculated for reference. Figure 7 indicates the necessary CPU time when applying each method. The Eshift spends much time on recalculation of a schedule.
It increases in the number of batches by splitting, so computing time is needed much. Table 2 and figure 7 show that Type-N is also superior in the CPU time as well as the make-span to Type-A. As it increases in the split number, Type-N becomes profitable. The searching ability of GA which applied only Eshift isn't enough on job shop scheduling under the batch splitting environment.

But the non-delay schedules have a possibility not including the optimum solution and Type-A is profitable when a batch is few. It's effective to search around the approximation solution using Type-A. The development of switching method which fits in with the scheduling environment and the searching situation is important. The added gene plays its role on the proposed GA.

<table>
<thead>
<tr>
<th>Table 2 Comparison of methods by the make-span</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>Non shortening method</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Left shift method</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Type-A</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Type-N</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Figure 7 CPU time for generating 5,000,000 schedules.

4.3 Performance of the proposed GA

The objective of this section is to evaluate the searching ability of the proposed GA. Two kinds of GA are prepared for the numerical experiment. One is the conventional GA which applies only type-A as the method of shortening idle time. This is the same as one of Ida and Osawa [7]. Another is the proposed GA.

Table 3 indicates mean of the solutions in each split number n by GAs. Both GAs could obtain almost the same result at the small number of split. But as the number of split is large, the proposed GA is more profitable about 1% than the conventional GA. The proposed GA acquired better results under the batch splitting environment as it uses type-A and type-N appropriately by the added gene.

Table 4 indicates mean numbers of new individuals which were generated by the genetic operations until respective GA gets the minimum value. The numbers of individuals on both of GAs are the largest at the split number 2. The proposed GA finds the minimum value under fewer individuals than the conventional GA. To search the solution in only the non-delay schedules is effective under the batch splitting environment.

Table 5 expresses the CPU times, not the number of individuals. As the number of batches is larger, more CPU time is needed for recalculating the schedule repeatedly by Eshift on the conventional GA. The total number of the recalculation in the conventional GA is about 150 times against the proposed GA's as the number of split is 16.

4.4 The flexibility of the proposed GA

The flexibility against scheduling environment of the proposed GA was inspected by recording changes in

Vol. 5 No. 1 (2013)
the ratio of type-N. The population on the proposed GA consists of two kinds of individual which are classified by the last gene. They are individuals of type-A and individuals of type-N.

Figure 8 indicates changes in the ratio of type-A to population in time series. The population size is 100 and every graph has started from 50%. The figure is expressing time by the number of generated individuals and the ratios are the averages of 1,000 tests.

Increase of the split number makes the ratio of type-A decrease. The non-delay schedules become more profitable than the other active schedules as the number of split is large. The proposed GA fits the scheduling environment since it adjusts the ratio automatically. We are freed from the parameter setting which considered the influence of batch splitting because of using the proposed GA.

The graph $n = 1$, $2$ and $4$ start to gain after getting small once. The graph $n = 8$ and $16$ are monotonous decrease. This phenomenon means that the algorithm is also adjusting the ratio according to the searching situation. The figure shows that the proposed GA maintains flexibly by adjusting the ratio of type-A and type-N to the scheduling environment and the searching situation.

5. Further experiments

The searching ability of the proposed GA has been already clear by numerical experiments under the batch splitting environment. However, the searching ability is also important when batch splitting is not considered. It is also necessary to evaluate a performance of the algorithm by a benchmark test. The algorithm examined the benchmark test using $ft10$ by Fisher and Thompson [9]. It is one of the most famous benchmark test. The minimum make-span is 930 in the benchmark test.

The preconditions for further experiments are as follows.

(1) The population size is 600.
(2) Genetic operations are continued until 4,000,000 new individuals are generated.
(3) The crossover rate and the mutation rate are made 0.8 and 0.2 respectively.
(4) One thousand kinds of random number seeds are implemented.

This section compared the proposed GA with the conventional GA in two kinds of criteria. The first criterion is how many times of optimal solution could be found in 1,000 tests. The second is the average of CPU time for finding the optimum solution. The outlines of experimental results are as follows.

(1) The conventional GA gained the optimum solution in tests of 99.1%. On the other hand, the proposed GA gained the optimum solution in every experiment.
(2) The CPU times in the proposed GA are about 70.1% of the conventional GA's.

These outlines show the improvement of searching ability was achieved by the proposed GA. Figure 9 expresses the changes in make-spans and the ratio of individuals with type-A. The ratio of individuals means the percentage of individuals with gene of type-A to 600 individuals in the proposed GA. It’s possible to consider that every individual has the gene of type-A in the conventional GA.

The average of make-spans in the population of the proposed GA is less than the conventional GA's because the proposed GA generates non-delay schedules in startup time. The difference of average make-span on initial populations is about 1.1%. The proposed GA improves the searching ability because the number of individuals increases after getting small once. The minimum value in the graph of ratio is about 47% and the maximum value is 100%.
6. Conclusion

This paper developed a genetic algorithm on Job Shop Scheduling under the batch splitting environment. The proposed GA has a chromosome described job numbers and two methods of shortening idle time. One locus is added at the end of a chromosome. The last gene decides either generating an active schedule by Eshift or generating a non-delay schedule like a dispatching method. We evaluated searching ability of the algorithm by some numerical experiments.

The proposed GA becomes so profitable that the number of split becomes large. It is effective not only under the batch splitting environment but also under the non-splitting environment. The algorithm maintains flexibly by dynamically adjusting the ratio of shortening methods as result of selection to the scheduling environment and the searching situation. We achieved improvement of the searching ability by the proposed GA.

However, the algorithm seems to have scope for improvement. It's necessary to investigate a method of genetic operation and to optimize parameters in the proposed GA. It's a problem that non-delay schedules have a possibility not including the optimal solution. When the method of converting from a non-delay schedule to the other active schedule has been developed, the proposed GA will obtain a better performance.

References