A Genetic Algorithm with Algorithm Shortening Idle Time for Job Shop Scheduling Problem

Kenichi Ida† and Akira Osawa†
†Department of Information Engineering
Maebashi Institute of Technology
Maebashi 371-0816, Japan
Email: ida@maebashi-it.ac.jp

***Abstract*— The purpose of JSP is to find a schedule which is the minimum makespan. We suppose it effective to reduce idle time of a machine, in order to improve the makespan effectively. The existing method cannot arrange work which is processing time longer than idle time to idle time. We propose the method of arranging work which is longer than the idle time to idle time. We suppose that makespan can be improved efficiently by performing the operation.**

I. Introduction

Scheduling problems exists almost everywhere in real-world situations, especially, industrial engineering world [1]. Job-shop scheduling problem (JSP) is one of most difficult combinatorial optimization problems. Genetic algorithm (GA) has been applied to many difficult combination optimization problems. The purpose of JSP is to find a schedule, that is, an allocation of the minimum duration required to complete all jobs.

We suppose that the most effective method of shortening makespan is to cut down on idle time. Shift left is most known method in existing method for shortening idle time. Shift left is method that arranges work to idle time. Shift left arranges work to idle time, without shifting another work to the right. For that reason shift left can not arrange work to idle time that is shorter than processing time of work. We focused on such idle time. Recently, Ida et al. proposed the algorithm for shortening such idle time [2],[3]. The algorithm arranges work to idle time shorter than processing of work. GA with the algorithm was able to obtain good results in experiment. This shows that the theory is valid. But, the algorithm also obtains bad solution. That is, the algorithm obtains unstable solution. It is one serious drawback of the algorithm. The purpose of this study is to conquer drawback of the algorithm.

To begin with, we propose two kinds of algorithm. The first algorithm arranges work to idle time that is shorter than processing time of work. The second algorithm alleviates the start time of work restricted by job. Next, we combine two kinds of algorithm and we incorporate the algorithm into GA. By applying the proposed algorithms to benchmark problem, we show its effectiveness.

II. Job-shop Scheduling Problem

The problem is to determine the operation sequences on the machines in order to minimize the makespan — that is, the time required to complete all jobs.

In general, JSP is described as follows. There are n different jobs and m different machines to be scheduled. Each job is composed of a set of operation and the operation order on machines is prespecified. The required machine and the fixed processing time characterize each operation. There are several constraints on jobs and machines [4].:

1) A job does not visit the same machines twice.
2) There are no precedence constraints among operations of different jobs.
3) Operation can not be interrupted.
4) Each machine can process only one job at a time.
5) Neither release times nor due dates are specified.

III. Idle Time

A. Restricted Start Time

Schedule of JSP has time zone that is not processing work. The time zone is called idle time. We start processing of work from start time restricted, when scheduling. Start time restricted produces idle time. JSP has two kinds of start time restricted. The first restriction is start time restricted by job. The second restriction is start time restricted by machine. Start time of work is the latest time in two kinds of start time restricted.

1) Start Time Restricted by Job:

Start time restricted by job is end time of work that is work of same job number. A work can process in idle time, when idle time existed between this restriction and work.

2) Start Time Restricted by Machine:

Start time restricted by machine is end time of a work that is processed by the same machine.

B. Idle Time Shortening Method and Drawback of the Method

Shift left is most famous method in existing method for shortening idle time. Shift left is method that arranges work to idle time, without shifting another work to the right. For that reason, shift left can not arrange work to idle time that is shorter than processing time of work. It is one serious drawback of shift left. Recently, Ida et al. proposed the algorithm for beating the drawback. The algorithm proposed by Ida et al. arranges work to idle time, and its algorithm also can arrange work to idle time shorter than processing time of work. That idea was in the

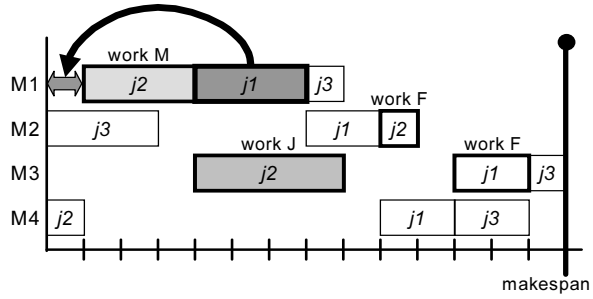

Fig. 1. Expand the coverage of shift left

right, and GA with the algorithm proposed by Ida et al. was able to obtain many superior solutions in experimentation. But, the algorithm proposed by Ida et al. needs to shift other work to the right for arranging work to idle time. For that reason, there is possibility of deterioration. The operation which arranges work to idle time is performed probable. For that reason, GA with the algorithm obtains unstable solution. The algorithm has many such drawbacks. Shift left does not deteriorate makespan. It is serious advantage of shift left. We suppose that we need to incorporate shift left's advantage in the algorithm. We suppose that will go a long way toward beating the drawback of the algorithm.

IV. Expanded Coverage of Shift Left

We propose the algorithm that expands the coverage of shift left. We called the algorithm Eshift. Figure 1 shows Eshift. We explain Eshift by the figure. Since idle time is between start time restricted by job and work, j_1 of $M1$ can process to idle time. But, since idle time is shorter than processing time of work, shift left can not arrange work to idle time. As Ida et al. point out, shift left is not enough to improvement of makespan. The purpose of Eshift is to arrange work to idle time which is shorter than processing time of work. Makespna does not deteriorate when Eshift arranges work to idle time which is shorter than processing of work. Eshift shifts other works to the right for the purpose of arranging a work to idle time. But, Eshift does not shift work to the right, when makespan gets worse. For that reason, makespna does not deteriorate when Eshift arranges a work to idle time.

First, Eshift produces a Gannt chart. Works are arranged to idle time in its Gannt chart. Second, Eshift makes a chromosome from a Gannt chart. We use chromosome of expression method proposed by Hirano [5]. We apply Eshift to all chromosomes that were generated by GA's each operation.

A. Generation of Gannt Chart (Eshift)

step 1 All works are arranged to a Gannt chart by a chromosome. Set initial generation $i = 1$.

step 2 If i is smaller than Q (Q is the total number of work.), the i-th gene is read.

step 3 Work W is a work matched to the i-th gene. A work W is processed by machine M. K is the processing turn in machine M. Set $y = 1$.

step 4 If y is bigger than K, set $i = i + 1$ and go back to $step2$.

step 5 A work Y is processed to the y-th by machine M. A work Y-1 is processed to the $y - 1$-th by machine M. If work W can process in idle time that is between work Y and work Y-1 (If work Y-1 does not exist, set end time of work Y-1=0.), set u=y and go to next step. If work W can not process in its idle time, set $y = y + 1$ and go back to $step4$.

step 6 Imagine, work W is arranged to its idle time, and other works are rearranged for satisfying restricted conditions. A work U is processed to the u-th by machine M. If work U does not shift to the right, go to $step10$.

step 7 Work J is processed next to the work U and work J is the same job number as work U. If work J does not shift to the right and go to $step10$.

step 8 Work E and work F are processed work next to the work J. Work E is the same job number as work J. Work J is processed work by machine J. Work F is processed work by machine J. If work E and work F does not shift to the right, go to $step10$. If work E or work F shifts to the right, set $y = y + 1$ and go back to $step4$. If work E and work F does not exists, go to next step.

step 9 If end time of work J is later than makespan, set $y = y + 1$ and go back to $step4$.

step 10 Set $u = u + 1$. If u is smaller than Y, go to $step6$.

step 11 Work W is arranged to idle time and other works are rearranged for satisfying restricted conditions. Set $i = i + 1$. Go to $step4$.

B. Generation of Chromosome

1) Generation of Chromosome 1 (g1):

step 1 Set initial generation $i = 1$.

step 2 A work which has the earliest end time is chosen in a Gannt chart. A job number of the work is stored in the i-th gene of a chromosome. The work is removed from a Gannt chart.

step 3 Set $i = i + 1$. If i is smaller than Q (Q is the total number of work.), go back to $step2$.

2) Generation of Chromosome 2 (g2):

step 1 Set initial generation $i = 1$.

step 2 A work which has the earliest start time is chosen in each machine on Gannt chart. A work is chosen at random in works. A job number of the work is stored in the i-th gene of a chromosome. The work is removed from a Gannt chart.

step 3 Set $i = i + 1$. If i is smaller than Q (Q is the total number of work.), go back to $step2$.

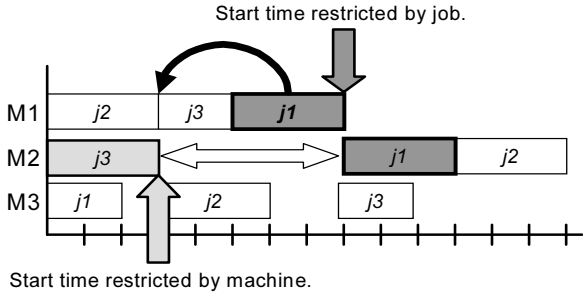

Fig. 2. Relaxation of Restriction by Job

V. Relaxation of Restriction by Job

If restriction of start time by job is later than restriction of start time by machine, idle time is generated in before a work. We propose algorithm that shortens such idle time. We called the algorithm Derestrict-Job. Derestrict-Job relaxes restriction by job for the purpose of shortening a idle time. Derestrict-Job does not make makespan worse for the purpose of obtaining a stable solution.

Figure 2 shows Derestrict-Job. We explain Derestrict-Job by J_1 of $M2$ in the figure. Start time restriction of J_1 by job is later than restriction by machine. That is, J_1 of $M1$ is restricting J_1 of $M2$. Because of it, idle time is generated in before J_1 of $M2$. If J_1 of $M1$ shifts to the left, J_1 of $M2$ can early start. Derestrict-Job shifts work that is restricting work of same job number to the left, for shortening idle time. The conditions which shift work to the left are the same as Eshift. Derestrict-Job shortens idle time that exists in front of a work. Its work exists next to the same job number as work that is shifted to the left. Therefore, Idle time is shortened when Derestrict-Job arranges to idle time, furthermore, Idle time is shortened when Derestrict-Job arranges to the time zone that has not idle time. Derestrict-Job arranges work to the time zone that are not idle time and idle time.

First, Derestrict-Job produces a Gannt chart. Start time restriction by Job on its Gannt chart is relaxed. Second, Derestrict-Job makes a chromosome from a Gannt chart. Generation methods of chromosome are the same as Eshift. We apply Derestrict-Job to all chromosomes that were generated by GA's each operation.

• Generation of Gannt Chart (Derestrict-Job)

Derestrict-Job is the algorithm which changed two places of Eshift. First, the condition is excepted from step5 of Eshift. Its condition is that idle time must exist to arrange, and it is the conditions for going back to step4. Second, step4.5 is added to Eshift. Step4.5 is appeared in below.

step 4.5 Work V is processed next to the work W, and work V is the same job number as work W. If start time restriction by job of work V is later than start time restriction by machine of work V, set $i = i + 1$ and go to $step3$.

VI. Combination of Two Kinds of Idle Time Shortening Algorithm

The purpose of Eshift is to shorten idle time that is smaller than processing time of work. The purpose of Derestrict-Job is to shorten idle time that was generated by start time restriction. In that respect Eshift and Derestrict-Job differs. We propose the algorithm which combined Eshift and Derestrict-Job, and we called the algorithm Two-short. We suppose that Two-short can shorten more idle time.

First, Two-short produces Gannt chart which shortened idle time. Second, Two-short makes a chromosome from a Gannt chart. Generation methods of chromosome are the same as Eshift. We apply Two-short to all chromosomes that were generated by GA's each operation.

• Generation Method of Gannt Chart (Two-short)

Two-short is the algorithm which added step4.5 to Eshift. Step4.5 is appeared in below.

step 4.5 Work V is processed next to the work W, and work V is the same job number as work W. If start time restriction by job of work V is later than start time restriction by machine of work V, the condition is excepted from $step5$ of Eshift. Its condition is that idle time must exist to arrange, and it is the conditions for returning to $step4$.

VII. Numerical Experiment

A. Check on Validity of the Proposed Algorithm

We perform some experiments to examine the performance of the proposed algorithms. We compare pGA1 with pGA1+2, GA, Left. pGA1 is GA with Eshift. pGA1+2 is GA with Eshift and Derestrict-Job. GA is GA without the proposed algorithm. Left is GA with the proposed method by Hirano [5]. When Hirano's method applied shift left to chromosome, it does not reflect the result to a chromosome. The benchmarks used in the experiments are Fisher's and Thompson's 10×10 and 20×5 problems known as hard problems [6]. For this numerical experiment, we follow the environment given in Watanabe's paper. In our experiment, to be fair, each method is allowed to generate the new solutions of about 3,000,000 in each trial. We set the population size to be 600, crossover ratio on GA to be 0.8 and the number of trials to be 100 in each experiment. We do not use mutation.

The experimental results are shown in table I. Here, the terms "Best","Worst","Ave" and "Time" mean the best solution (A optimal solution was obtained this number.), the worst solution, the average solution and average time when the best solution was obtained in the numerical experiment respectively. If generation method (g1) of chromosome generated a chromosome, the result is shown in the first line. If generation method (g2) of chromosome generated a chromosome, the result is shown in the second line. In ft10×10, the makespan of optimal solution is 930, and in ft20×5, it is 1,165. The programming language is C++, and it runs on Windows XP with Pentium 4.

TABLE I

EXPERIMENTAL RESULTS1

		Best	Worst	Ave	Time(s)
ft10	GA	930(1)	973	952.2	**17.2**
	Left	930(1)	956	941.3	49.4
	pGA 1	930(10)	945	937.7	111.0
		930(100)	**930**	**930.0**	73.3
	pGA 1+2	930(56)	939	933.0	78.5
		930(100)	**930**	**930.0**	53.8
ft20	GA	1183(0)	1235	1209.9	23.4
	Left	1173(0)	1190	1180.2	70.0
	pGA 1	1165(98)	1173	1165.2	44.6
		1165(38)	1178	1172.5	181.8
	pGA 1+2	**1165(100)**	**1165**	**1165.0**	**18.7**
		1165(71)	1178	1168.7	63.6

pGA1 was able to obtain solutions superior to Left. Therefore, Eshift is better performance than shift left. pGA1 which added Derestrict-Job was able to obtain solutions superior to pGA1. Hence one can say that, the algorithm which combined two kinds of the algorithm is the most effective to shortening of idle time. From the standpoint of time, pGA1 and pGA1+2 are inferior to other methods, but pGA1 and pGA1+2 obtained many optimal solution, from the comparison with other methods. Hence one can say that, the proposed algorithm is effective in improvement of makespan. pGA1+2 generated chromosome from Gannt chart by two kinds of algorithm. When pGA1+2 used g2, pGA1+2 obtained many optimal solution in ft10×10. When pGA1+2 used g1, pGA1+2 obtained many optimal solution in ft20×5.

B. Validity to Initial Population

We perform experiments to examine the performance of Two-short to initial population. We compare Two-short with shift left and without shift left. We applied these algorithms to ft10×10. We generate the chromosomes of 5,000,000 at random, and We applied these algorithms to all chromosomes. Without shift left performs nothing to chromosome.

Figure 3 shows the frequency distribution of makespan. It is shown here that the distribution of Two-short has width narrower than shift left and without shift left, and is located in a left. Therefore, it is confirmed that Two-short shows better performance than shift left and without shift left. The standard deviation and average of Two-short are 1188.9 and 56.2 respectively. The standard deviation and average of shift left are 1263.8 and 63.6 respectively. The standard deviation and average of without shift left are 1723.0 and 152.9 respectively. The result of our experiment clearly shows that the proposed algorithms are effective in improvement of makespan.

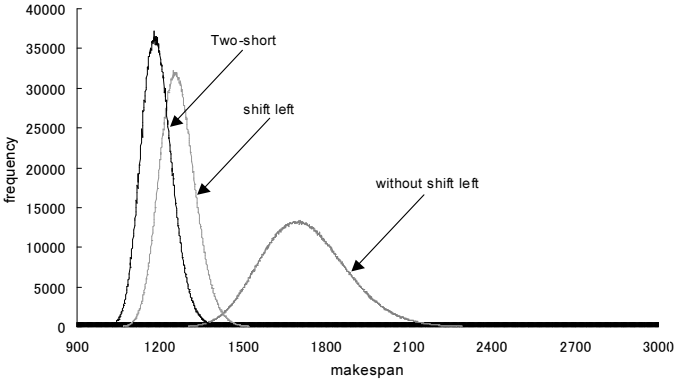

Fig. 3. Frequency distribution of makespan

VIII. CHANGE OF METHOD WHICH GENERATE CHROMOSOME

ft10×10 and ft20×5 has the structure of solution space which differs clearly. pGA obtained superior solution by useing g1 and g2 to such problems. Consequently, We propose the algorithm which changes the generation method of chromosome. If best solution was not updated, the proposed algorithm changes the generation method of chromosome, and the proposed algorithm changes parent individual into child individual, in order to slip out of local minimum. The algorithm is appeared in below. We judge whether a population lapsed into local minimum in step5. Y is the number of generations for judging. K is the number of generations for changing parent individual and child individual. We called Two-short with this algorithm Ishort.

- Proposed Algorithm (Ishort)

step 1 Set initial generation $k = 2$, $i = 0$, $g = 0$ and $j = 0$.
step 2 Two-short produces a Gannt chart from a chromosome which is generated by GA's each operation.
step 3 If $k = 1$, generate a chromosome from a Gannt chart using generation method (g1) of chromosome. If $k = 2$, generate a chromosome from a Gannt chart using generation method (g2) of chromosome.
step 4 If the best individual is not better than the best individual of the former generation in the population, set $i = i + 1$. If the best individual is better than the best individual of the former generation in the population, set $i = 0$.
step 5 If $i = Y$ and $g = 0$, set $g = 1$ and $j = K$, and go to next step. If $i \neq Y$ or g $\neq 0$, go to $step7$.
step 6 If $k = 1$, set $k = 2$. If $k = 2$, set $k = 1$.
step 7 If $g = 1$, parent individual is replaced with child individual and set $j = j - 1$. If $g = 0$ and a child individual is better than a parent individual, parent individual is replaced with child individual.
step 8 If $j = 0$, set $g = 0$.
step 9 Go back to $step2$.

TABLE II
EXPERIMENTAL RESULTS2

		Best	Worst	Ave	Time(s)
ft10	GTX	936(0)	974	967.7	–
	SXX	930(52)	960	934.3	–
	JOX	930(84)	938	931.1	–
	asol	930(20)	956	938.3	**38.6**
	IGA	930(22)	955	938.0	314.3
	pGA	**930(100)**	**930**	**930.0**	51.6
ft20	GTX	1207(0)	1290	1250.9	–
	SXX	1180(0)	1256	1217.4	–
	JOX	1165(95)	1173	1165.4	–
	asol	1165(4)	1178	1173.8	**46.7**
	IGA	1165(7)	1178	1173.1	356.1
	pGA	**1165(100)**	**1165**	**1165.0**	90.9

IX. COMPARISON EXPERIMENT

We perform some experiments to examine the performance of Ishort. We called GA proposed by Watanabe et al. with Ishort pGA. We compare pGA with GA proposed by Ida et al. (IGA) [2],[3], GA proposed by Ono et al. (JOX) [7], GA proposed by Watanabe et al. (asol) [4], GA proposed by Yamada et al. (GTX), and GA proposed by Ono et al. [8]. The results of experiment other are compared with those given in each paper. Those results are other than asol. For this numerical experiment, we follow the same environment given in Watanabe's paper. The condition is shown in section VII of this paper. IGA can not obtain sufficient result on this conditions. For that reason, IGA is allowed to generate the new solutions of about 10,000,000 in each trial.

The experimental results are shown in table II. pGA obtained the optimal solution 100 times in 100 trials. It is clear that pGA obtained more optimal solution than other methods. We improved IGA for obtaining stable solution, and pGA obtained superior solutions and time, from the comparison with IGA. Hence one can say that, its improvement was effective. From the standpoint of time, pGA is inferior to asol, but it is a little difference. The result of our experiment clearly shows that pGA is superior to other methods. Hence one can say that, Ishort is effective to improvement of solution.

X. CONCLUSION

In this paper, we proposed the algorithm which shortens idle time for job-shop scheduling problem. First, we proposed two kinds of algorithm. We were able to obtain superior solution to such problems by combining two kinds of the proposed algorithm. ft10$\times$10 and ft20$\times$5 has the structure of solution space which differs clearly. We were able to obtain superior solution to such problems by combining two kinds of the proposed algorithm which generate chromosome.

The proposed algorithm shifts other works to the right for arranging to the idle time which is shorter than processing time of work. For that reason, the proposed algorithm can not always generate chromosome which can produce active schedule. But, it is unusual for the proposed algorithm to generate chromosome which produces semi-active schedule. Such chromosome is generated in the ratio of about three to three million individual. In order for us to change semi-active schedule into active schedule, it is necessary to search for semi-active schedule. If we searched in a million individual, only one individual is found and it is improved. We suppose that the rate of an improvement is very low compared with time cost for search. For this reason, we did not change semi-active schedule into active schedule. But, we suppose that it leaves some room for consideration. We would like to improve Two-short so that Two-short can always generate chromosome which can produce active schedule. By expanding the coverage, shift left was able to obtain superior solution from the comparison with ordinary shift left. We would like to propose the algorithm that more expands the coverage of shift left. We suppose that its algorithm obtains more superior solution. In this paper, the proposed algorithm was applied to two kinds of problem. We would like to apply the proposed algorithm to other benchmarks. We expect the proposed algorithm to robustly work on various problems.

REFERENCES

[1] M. Gen. and R. Cheng. *Genetic Algorithms and Engineering Optimization*, John Wiley & Sons, New York, 2000.

[2] K. Ida. and A. Oosawa. Job-shop Scheduling Problem Using Genetic Algorithm, *Proc. of the 33rd International Conference on Computers and Industrial Engineering*, no.CIE732, CD-ROM, pp.1–6, 2004.

[3] K. Ida. and A. Osawa. A Solution Method of JSP by GA with Idle Time Shortening Algorithm, *2004 Spring Meeting of Japan Industrial Management Association* pp.130–131, 2004 (in Japanese).

[4] M. Watanabe., K. Ida., T .Horita. and M. Gen. Active Solution and Active Solution Space on Job-shop Scheduling Problem, *Proc. of 8th international symposium on AROB*, vol.2, pp.447–450, 2003.

[5] H. Hirano. Genetic Algorithm with Cluster Averaging Method for Solving Job-shop Scheduling Problem, *Journal of Japanese Society for Artificial Intelligence*, vol.33, no.12, pp.1523–1533, 1995 (in Japanese).

[6] D.C. Mattfeld. and R.J.M. Vaessens. OR-Library http://mscmga.ms.ic.ac.uk/jeb/orlib/

[7] I. Ono., Y. Nagata. and S. Kobayashi. A Genetic Algorithm Taking Account of Characteristics Preservation for Job Shop Scheduling Problems *Proc. of the International Conference on Intelligent System*, vol.5, pp.711–718, 1998.

[8] I. Ono., H. Satoh. and S. Kobayashi. A Genetic Algorithm based on Subsequence Exchange Crossover and GT method for Job-shop Scheduling *T.IEE Japan*, vol.117-C, no.7, pp.888–895, 1997 (in Japanese).