Parallel Problem Solving from Nature, 2 PP. 281 - 290
R.Männer and B. Manderick (Editors)

A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems

Takeshi Yamada and Ryohei Nakano
NTT Communication Science Laboratories, 2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan

Abstract

This paper proposes a novel method for solving job-shop problems based on Genetic Algorithms (GAs). In this method, each individual represents a solution which is derived directly from operation completion times rather than being coded in a binary string. Applying crossover based on Giffler & Thompson's algorithm for generating active schedules to any two individuals (parents) which are active schedules results in new active schedules (offspring) which inherit the parents' characteristics.

Experiments showed that the proposed method can produce optimal solutions for the difficult Muth & Thompson's 10×10 benchmark and good solutions for randomly generated 20×20 problems.

1. INTRODUCTION

Job-shop problems are among the hardest combinatorial optimization problems and they have been the subject of a significant amount of literature in the operations research area [11, 2, 9, 3, 1, 5]. The algorithms primarily used to solve job-shop problems are the branch and bound methods and the performance of existing algorithms has been evaluated using widely known benchmarks [11]. The main historical progress in solution quality will be shown later with the experimental results.

When solving job-shop problems using GAs, it is difficult to code solutions in binary and define a crossover operator because the resulting binary strings after crossover are not always feasible solutions to the target problems. This situation is similar to that encountered by Whitley *et al.*[14] in the Traveling Salesmen Problems, but more complicated.

In a previous paper by the authors, a binary-coding GA method was proposed to solve job-shop problems by focusing attention on each job pair and its processing order. Good solutions were obtained comparable to those obtained using the branch and bound methods [12].

In this paper, a new method, called GA/GT, has been developed which uses more direct codings and a new crossover operator based on the classic Giffler & Thompson's algorithm for generating active schedules.

Applying this method to Muth & Thompson's difficult 10×10 benchmark, optimal solutions have been obtained. Moreover, applying the method to several larger problems which were randomly generated with no known optimal solutions, very good solutions,

if not optimal, when compared with randomly generated active schedules, have been obtained.

2. JOB-SHOP PROBLEMS

The $n \times m$ (minimum-makespan) job-shop problem is as follows [13]. A set of n jobs $\{J_j\}_{1 \le j \le n}$ is to be processed on a set of m machines $\{M_r\}_{1 \le r \le m}$. The processing of job J_j on machine M_r is the operation $O_{j,i,r}$ where $i \in \{1, \ldots, m\}$ indicates the position of the operation in the technological sequence of the job. Operation $O_{j,i,r}$ requires the exclusive use of M_r for an uninterrupted duration $p_{j,i,r}$, its processing time. The time required to complete all the jobs is the makespan S_{max}. The objective is to determine the set of completion times for each operation $\{c_{j,i,r}\}_{1 \le j \le n, 1 \le i,r \le m}$, which minimizes S_{max}.

Table 1
A 6×6 job-shop problem

Job	Operation routing (processing time)					
1	3(1)	1(3)	2(6)	4(7)	6(3)	5(6)
2	2(8)	3(5)	5(10)	6(10)	1(10)	4(4)
3	3(5)	4(4)	6(8)	1(9)	2(1)	5(7)
4	2(5)	1(5)	3(5)	4(3)	5(8)	6(9)
5	3(9)	2(3)	5(5)	6(4)	1(3)	4(1)
6	2(3)	4(3)	6(9)	1(10)	5(4)	3(1)

Reprinted from reference [11], pp. 226.

Data for a 6×6 job-shop problem taken from reference [11] is given in Table 1. The data includes the routing of each job through each machine and the processing time for each operation (in the parentheses). Note that, given the routing matrix, an operation is uniquely defined by its job number j and either its sequence index i or its machine index r, so either i or r can be omitted like $O_{j,i}$ or $O_{j,,r}$ [1].

There are several ways to represent solutions to job-shop problems. The following are the three main expressions:

A. Specify the set of completion times for each operation. (Gantt chart)
B. Specify the order of the operations on each machine.
C. Specify the order of the operations for each pair of operations on each machine. (disjunctive graph)

For expression **A**, one of the optimal solutions to the data in Table 1 is given in Table 2 (matrix of $\{c_{j,i}\}$). The visual version of Table 2 shown in Figure 1 is called the Gantt chart. In Figure 1, the vertical axis indicates jobs although it usually indicates machines.

Each schedule is represented uniquely by **A**, but not by **B**, because each operation can be postponed arbitrarily.

[1] This index convention is used for all variables.

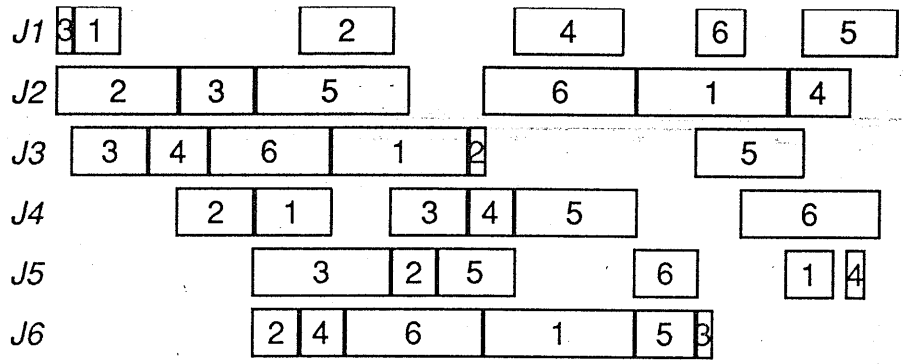

Figure 1. Gantt chart of a solution to the 6×6 job-shop problem

Table 2
Optimal solution to the 6×6 job-shop problem

Job	Completion times					
1	1	4	22	37	45	55
2	8	13	23	38	48	52
3	6	10	18	27	28	49
4	13	18	27	30	38	54
5	22	25	30	42	51	53
6	16	19	28	38	42	43

The schedule which is represented uniquely by **B** is called a *semiactive schedule*, i.e., one in which no operation can be started earlier without altering the machining sequences. With regards to optimization, the acceptable solution must be semiactive.

Consider a semiactive schedule and two operations $O_{j,,r}$ and $O_{k,,r}$ in it which share the same machine M_r. If $O_{k,,r}$ is processed prior to $O_{j,,r}$ and the machine M_r has an idle period longer than $p_{j,,r}$ before processing $O_{k,,r}$, reassigning is possible so that operation $O_{j,,r}$ is processed prior to $O_{k,,r}$. Such reassigning is called *permissible left shift*. A schedule having the property that no operation can be processed earlier by permissible left shift is called an *active schedule* [6]. Optimal solutions are active schedules.

3. GIFFLER & THOMPSON'S ALGORITHM

Giffler & Thompson developed an algorithm to generate any one, or all, active schedule(s) (GT algorithm) [6].

The GT algorithm is briefly reviewed in this section. Notations are the same as in Section 1.

Consider scheduling each operation in a temporal order. A set C of all the earliest operations in technological sequence among the operations which are not yet scheduled is defined (which is called *cut*). The earliest (possible) completion time $EC_{j,i}$ is calculated

for each operation $O_{j,i} \in C$.

(A1) Find O_{j^*,i^*,r^*} which has a minimum[2] EC in C: $EC_{j^*,i^*} = min\{EC_{j,i} \mid O_{j,i} \in C\}$

Specify G: a set of operations which consists of $O_{j,i,r^*} \in C$ sharing the same machine M_{r^*} with O_{j^*,i^*,r^*} and the processing of O_{j,i,r^*} and O_{j^*,i^*,r^*} overlaps. G is called a *conflict set*.

(A2) Choose one of the operations O_{j_s,i_s} from G, and schedule O_{j_s,i_s} according to EC_{j_s,i_s}.

(A3) Update C and ECs.

Repeat Step (A1) $\sim$ Step (A3), until all operations are scheduled, and then an active schedule is obtained.

In Step (A2), if all possible choices are considered, active schedules are generated for all[3] .

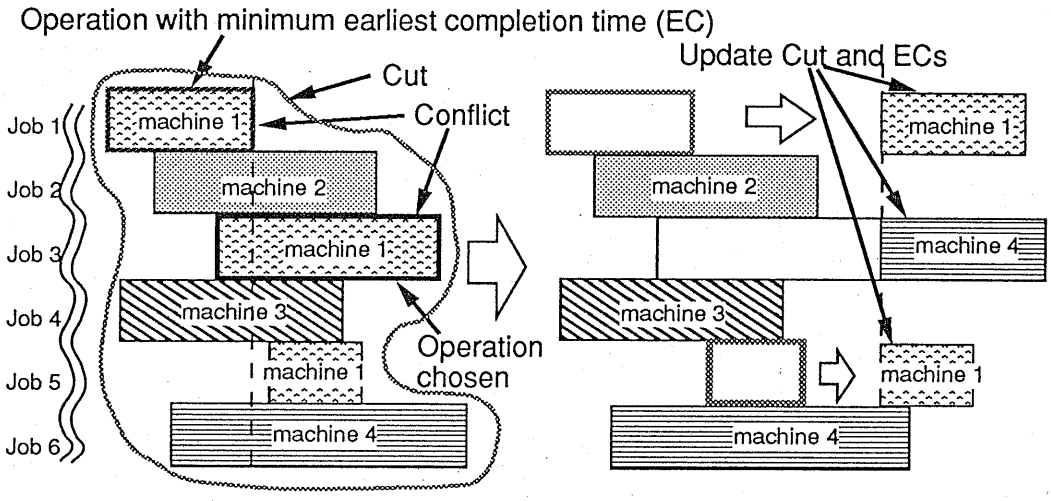

Figure 2. Giffler & Thompson's algorithm

4. THE NEW METHOD: GA/GT

4.1. Representation

In most GAs, each individual is a binary string in which a solution of the target optimization problem is coded. In our method, each individual psn represents an active schedule directly using elements $\{psn_{j,i,r}\}$ of operation completion times according to expression **A** (i.e., $psn_{j,i,r} = c_{j,i,r}$).

[2]Ties are broken randomly.

[3]But the amount is still very large.

The makespan S_{psn} of the schedule represented by the individual psn is calculated as follows:

$$S_{psn} = max\{psn_{j,i} \mid 1 \leq j \leq n,\ i = m\} \tag{1}$$

4.2. GA/GT crossover

Crossover is an operation which generates new individuals from two old individuals. The algorithm describes how to generate a new individual, say a kid, from two old individuals, say mom and dad.

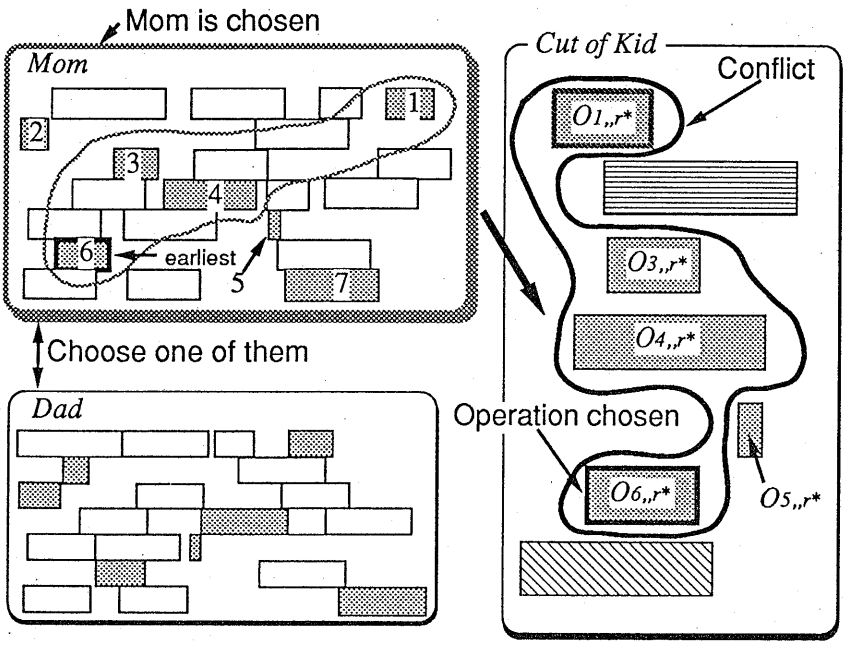

Figure 3. GA/GT crossover

(C1) Do Step (A1) of the GT algorithm, obtain C, ECs and G.

(C2) Choose one of the operations to be scheduled next from G as follows:

(a) generate a random number $\epsilon \in [0,1)$ and compare it with $R_\mu \in [0,1)$ which is a predefined constant called the *mutation rate*.
if $(\epsilon < R_\mu)$ ***then*** choose any operation O_{j_s,i_s} from G (mutation occurs).

(b) ***otherwise*** select either mom or dad with an equal probability $1/2$.
Mom is assumed to be selected.
Find an operation O_{j_s,i_s} which was scheduled earliest in mom among all the operations in G:

$$mom_{j_s,i_s} = min\{mom_{j,i} \mid O_{j,i} \in G\} \tag{2}$$

(c) schedule O_{j_s,i_s} according to EC_{j_s,i_s}, then set $kid_{j_s,i_s} = EC_{j_s,i_s}$.

(C3) Update C and ECs.

By repeating Step (C1) ~ Step (C3) until all operations are scheduled, the new individual *kid* is obtained.

In actual GA runs, these steps were applied twice to a pair of *mom* and *dad*, to obtain two new individuals, say *kid*1 and *kid*2.

In order to keep the fittest individual so far existing in the current population *(elitism)*, *mom'* and *dad'*: individuals for the next generation, are selected from *mom*, *dad*, *kid*1 and *kid*2 as follows:

Let *mom'* be the best one among *mom*, *dad*, *kid*1 and *kid*2.

Let *dad'* be the kid which was not selected as *mom'* or the better kid if neither was selected previously.

4.3. Other operators

Evaluation is straightforward because of the way each individual is represented in Section 4.1 (Equation (1)).

The selection scheme and scaling method[4] used are standard as used in reference [8], because this study focuses on the crossover effect.

Note that mutation is built into Step (C2-a) in the crossover .

4.4. Inheritance of characteristics

Let us examine the GA/GT crossover in more detail. For ease of explanation, only the case $R_\mu = 0$ (no mutation) is considered. According to expression **B** in Section 2, a schedule is the specification of the order of all operations on each machine. Because G consists of the operations on a machine M_{r*}, Step (C2-b) in Section 4.2 defines how to specify the order of operations on machine M_{r*}. If *mom* is chosen with a probability $1/2$, Step (C2-b) makes the scheduling order of operations in G (on M_r^*) in *kid* as close to that in *mom* as possible, because among all operations in G, the operation which was scheduled earliest in *mom* is also scheduled earliest in *kid* (Equation (2)).

In special cases, in Step (C2-b), if *mom* is chosen all the time, *kid* becomes equal to *mom*.

5. EXPERIMENTAL RESULTS

5.1. Muth & Thompson's benchmarks

Muth & Thompson's benchmarks consist of three problems: 6×6, 10×10 and 20×5.

For more than 25 years, researchers in operations research have tested their algorithms on these problems, solutions have gradually improved, and now optimal solutions have been obtained. The historical progress in branch and bound methods[5] together with the

[4]We have to convert minimization problem to maximization problem.

[5]However Adams' method is an approximation method.

Table 3
Muth & Thompson's benchmark

Papers	6×6	10×10	20×5
Balas (1969)	55	1177	1231
McMahon (1975)	55	972	1165
Barker (1985)	55	960	1303
Adams (1988)	55	930	1178
Carlier (1989)	55	930	1165
Nakano (1991)	55	965	1215
Yamada (1992)	55	930	1184
Optimal	55	930	1165
Lower Bound	52	880	1164

authors' best previous and current results are shown in Table 3. The lower bounds are taken from Carlier's paper [5].

The 6×6 problem is relatively easy: solutions with a makespan of 55 are always obtained quickly even with a small population.

For the 10×10 problem, Carlier *et al.* [5] first found a schedule with a makespan of 930 and proved its optimality. According to them, the practical complexity of this problem is due to a gap of 15 percent between the initial lower bound and the optimal schedule value. Optimal schedules with a makespan of 930 were found four times among 600 trials using the proposed method which does not use lower bound information (Figure 4). One trial requires about 10 minutes on a Sparc Station 2 and the programs are written in C language.

To put it simply, the branch and bound method (BAB) consists of two operations: a *branch* operation and a *bound* operation. The classic GT algorithm is often used as the *branch* operation to solve job-shop problems using BAB [4].

The *bound* operation is the pruning and selection of one node according to the lower bound after node generation by the *branch* operation. But the larger the problem, the larger the gap between the initial lower bound and optimal solutions, which means bounding is less effective. In such cases, an approximation method like the GA/GT method in this paper can be an effective alternative.

5.2. Larger problems

Four 20×20 problems were generated with processing times randomly drawn from a uniform distribution on the interval [10, 50], and the GA/GT method was applied to these problems. This resulted in very good solutions if not optimal, when compared with the randomly generated 400,000 active schedules for each problem. The results are shown in Table 4. A histogram of the results of problem No.1 in Table 4 is shown in Figure 5.

5.3. GA/GT crossover effect

In the GA/GT crossover, if the algorithm is modified so that a new individual is created by only using the mutation from one individual, an asexual recombination operator is defined. To clarify the efficiency of the GA/GT crossover, the results of the (sexual) GA/GT method were compared with those of the asexual method.

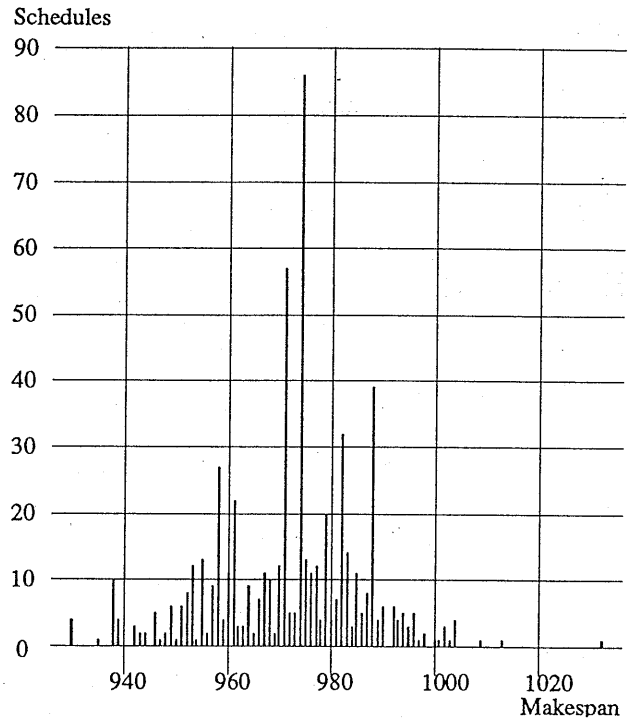

Figure 4. Distribution of solutions (10×10 problem)

Table 4
GA/GT v.s Random Samplings

20×20 problems	Random		GA/GT	
No.	Average	Best	Average	Best
1	1356.8	1126	979	967
2	1318.6	1104	953	945
3	1313.5	1107	957	951
4	1414.3	1202	1060	1052

The changes in the best values in the population for each generation during the experiments for problem No.1 are shown in Figure 6. Solid lines indicate 10 runs with crossover and mutation ($R_\mu = 0.01$) and dotted lines indicate 10 runs with mutation only ($R_\mu = 0.1$). Notice that without crossover, it takes more time to get a good solution than with crossover, and the best solutions obtained vary widely for each run.

6. CONCLUSION

A method solving minimum-makespan job-shop problems using a Genetic Algorithm based on Giffler & Thompson's algorithm (GA/GT method) was investigated. Experiments showed that the proposed approach generates good solutions for problems whose size the

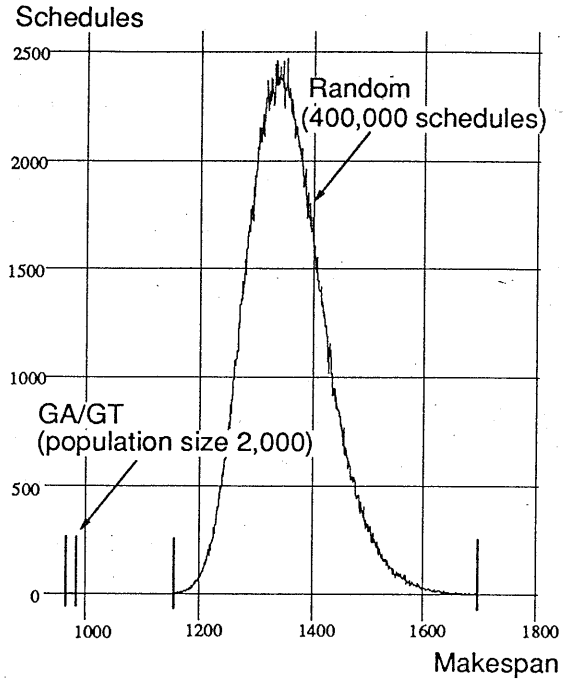

Figure 5. GA/GT v.s. Random Samplings

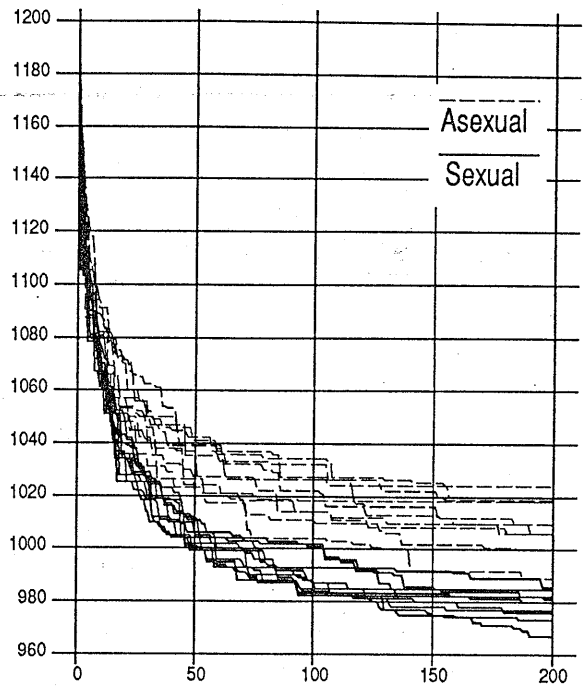

Figure 6. Sexual v.s. Asexual

branch and bound approach can treat, and also larger problems.

Because the proposed method can be interpreted as a combination of GAs and the *branch* operation of BAB, this approach suggests new potential GA applications to combinatorial optimization problems where BAB is the algorithm primarily used for solutions.

Currently, the rate of obtaining optimal solutions is still small, so further improvements to the algorithm are required, for example, integrating an alternative selection scheme [7], or combining it with local search heuristics [10].

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Seishi Nishikawa, the director of NTT communication science laboratories, for his encouragement. Thanks are also due to Profs. Toshihide Ibaraki, Yoshikazu Nishikawa and Dr. Hisashi Tamaki for their helpful comments and discussions.

REFERENCES

1 J. Adams, E. Balas, and D. Zawack, The shifting bottleneck procedure for job shop scheduling, Manage. Sci. 34 (1988), No. 3, 391–401.

2 E. Balas, Machine sequencing via disjunctive graphs: an implicit enumeration algorithm, Oper. Res. 17 (1969), 941–957.
3 J.R. Barker and G.B. McMahon, Scheduling the general job-shop, Manage. Sci. 31 (1985), No. 5, 594–598.
4 G.H. Brooks and C.R. White, An algorithm for finding optimal or near optimal solutions to the production scheduling problem, The Journal of Industrial Engineering 16 (1969), No. 1, 34–40.
5 J. Carlier and E. Pinson, An algorithm for solving the job-shop problem, Manage. Sci. 35 (1989), No. 2, 164–176.
6 B. Giffler and G.L. Thompson, Algorithms for solving production scheduling problems, Oper. Res. 8 (1969), 487–503.
7 D.E. Goldberg and K. Deb, A comparative analysis of selection schemes used in genetic algorithms, Foundations of Genetic Algorithms, Morgan Kaufmann, 1991, pp. 69–93.
8 J.J. Grefenstette and N.N. Schraudolph, A user's guide to genesis 1.2ucsd, Navy Center for Applied Research in Artificial Intelligence, Washington, D.C., 1990.
9 G. McMahon and M. Florian, On scheduling with ready times and due dates to minimize maximum lateness, Oper. Res. 23 (1975), No. 3, 475–482.
10 H. Mühlenbein, Parallel genetic algorithms, population genetics and combinatorial optimization, Proc. 3rd Int. Conf. on Genetic Algorithms and their Applications (Arlington,Va.), 1989, pp. 416–421.
11 J.F. Muth and G.L. Thompson, Industrial scheduling, Prentice-Hall, Englewood Cliffs, N.J., 1963.
12 R. Nakano and T. Yamada, Conventional genetic algorithm for job shop problems, Proc. 4th Int. Conf. on Genetic Algorithms and their Applications (San Diego,CA.), 1991, pp. 474–479.
13 R.V. Rogers and Jr. K.P. White, Algebraic, mathematical programming, and network models of the deterministic job-shop scheduling problem, IEEE Trans. Syst. Man Cybern. 21 (1991), No. 3, 693–697.
14 D. Whitley, T. Starkweather, and D. Fuquay, Scheduling problems and traveling salesman: The genetic edge recombination operator, Proc. 3rd Int. Conf. on Genetic Algorithms and their Applications (Arlington,Va.), 1989, pp. 133–140.