Genetic Algorithms with a Permutation Approach to the Parallel Machines Scheduling Problem*

Yong Ho Han**

ABSTRACT

This paper considers the parallel machines scheduling problem characterized as a multi-objective combinatorial problem. As this problem belongs to the NP-complete problem, genetic algorithms are applied instead of the traditional analytical approach. The purpose of this study is to show how the problem can be effectively solved by using genetic algorithms with a permutation approach. First, a permutation representation which can effectively represent the chromosome is introduced for this problem. Next, a schedule builder which employs the combination of scheduling theories and a simple heuristic approach is suggested. Finally, through the computer experiments of genetic algorithms to test problems, we show that the niche formation method does not contribute to getting better solutions and that the PMX crossover operator is the best among the selected four recombination operators at least for our problem in terms of both the performance of the solution and the operational convenience.

1. Introduction

Generally, production scheduling problems are considered as one of the hardest combinatorial problems because they belong to the class of NP-complete problems. Therefore, a crucial task of scheduling is the creation of near-optimal schedules without searching through the entire problem space. Meanwhile, genetic algorithms (GAs) have recently gained considerable popularity as general-purpose robust optimization and search techniques[6,7,9,14]. In the field of the scheduling GAs have successfully solved different production scheduling problems over the last ten years[3]. Nevertheless, in many industries, there is frequently no ideal method for the scheduling to the extent that some

* 본 연구는 연암문화재단의 지원을 받아 1995년 8월부터 1996년 8월까지 미국 State University of New York at Buffalo 에서 수행되었음.

** Department of Management Information Systems, Pusan University of Foreign Studies

shop-floors may commonly schedule operations using rules of thumb. The scheduling problem at the steel cutting shop in shipbuilding yard is a typical example of such problems. The motivation of our study originally stems from the need for an efficient algorithm to the scheduling problem in this type of shop.

There are a few previous studies that are closely related to our study. Suresh & Chaudhuri[16] consider the problem of scheduling n jobs, all requiring a single stage of processing, on m unrelated parallel machines. There are two objectives: to minimize the makespan and to minimize the maximum tardiness. A tabu search based algorithm is suggested for this. Fang et al.[5] introduce a GA approach for the job shop scheduling problem. The approach uses a variant of the ordinal representation used for traveling salesman problem. Hou et al.[10] apply GAs to solve the multiprocessor scheduling problem. The representation of the chromosome is based on the order of the operations being executed in each individual processor. Kobayashi et al.[11] present a method based on GA for solving job shop scheduling problems. They suggest a new crossover operator. Bierwirth et al.[2] solves the job shop problem with GA. A new permutation representation is suggested. In all of the four studies given above a single objective function is used. Stanley & Mudge[15] present GAs as an optimization approach to a multi-objective microprocessor design problem. Shaw and Fleming[13] demonstrate the application of a pareto optimal multi-objective GA. Major characteristics of these previous studies are summarized in Table 1.

Unlike these, in this paper a problem characterized as a bicriteria scheduling problem is solved by GAs. It is generally recognized that some of the GA components such as the chromosome representation, the type of crossover operator, and the niche formation method greatly affect the performance of GAs. The purpose of this study is to suggest an effective representation method of chromosome and to identify the combination of crossover operators with/without niche formation method which generates the best performance of solution to the given problem. In section 2, the target problem is described. In section 3, an effective representation of permutation-type is suggested for

Table 1. Summary of Previous Researches

	Scheduling Problem	Multi-objective Problem	GA Applied
Suresh & Chaudhuri [16]	yes	yes	no
Fang et al. [5] Hou et al. [10] Kobayashi et al. [11] Bierwirth et al. [2]	yes	no	yes
Stanley & Mudge [15] Shaw & Fleming [13]	no	yes	yes

encoding the chromosome. In section 4, four types of crossover operators and a niche formation method are described. In section 5, test problems are solved under different combinations of crossover operator with/without niche formation method. Computational results are analyzed. Finally in section 6, some concluding remarks are made.

2. Problem Description

In this paper a bicriteria formulation for the scheduling of unrelated parallel machines is considered. It is assumed that there are n jobs, all available initially, and m machines are available throughout the scheduling period. Each job has a due date. Only one operation is needed to complete processing of a job. Any machine can process any job but with different processing time. No preemption of job is allowed and a machine can process only one job at a time. Under this environment we should determine for each job both the machine it is assigned to and the start time (or completion time) of it which minimize both the makespan and the maximum tardiness. To describe the problem more clearly, we formulate it as an integer linear program with two objectives. The following notation is used in the formulation.

Indices

i machine $i \in I \equiv \{1, 2, \ldots, m\}$

j job $j \in J \equiv \{1, 2, \ldots, n\}$

Parameters

p_{ij} processing time of job j on machine i

d_j due date of job j

Decision Variables

C_j completion time of job j

$C_{\max}$ makespan

T_j tardiness of job j

$T_{\max}$ maximum tardiness

X_{ij} $= 1$ if job j is assigned to machine i, and 0 otherwise.

Three different approaches have been adopted in the scheduling literature for considering multiple criteria[16]: (a) Both of the criteria are considered in the objective function as a weighted sum of the two criteria. This approach considers the trade-off between conflicting objectives. (b) The criteria are considered explicitly as objective functions. This approach generates the so-called efficient solution set[7] and the best solution is derived from this set. (c) One of the two criteria acts as the objective

function and the other as a constraint. In this paper the first approach mentioned earlier is taken. So two objective functions are combined into a single objective function as given below:

$$\text{Minimize } w_1 C_{\max} + w_2 T_{\max} ,$$

where w_1 and w_2 denote weights for the makespan and maximum tardiness, respectively.

Makespan is defined as

$$C_{\max} = \max\{C_i \mid i \in I\}.$$

Tardiness of job j is defined as

$$T_j = \max(0, C_j - d_j)$$

and

$$T_{\max} = \max\{T_j \mid j \in J\}.$$

Therefore, the mathematical model for our problem is given as follows:

$$\text{Minimize } w_1 C_{\max} + w_2 T_{\max} ,$$

$$\text{subject to } \sum_j p_{ij} X_{ij} \le C_{\max} \quad \text{for } i = 1, 2, \ldots, m, \quad (1)$$

$$\sum_i X_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n, \quad (2)$$

and

$$X_{ij} \in \{0, 1\} \quad \text{for all } i, j. \quad (3)$$

In the formulation, constraint set (1) states that the completion time of each machine should be less than or equal to the makespan. Constraint set (2) ensures that each of the jobs should be assigned to one of the machines.

3. Permutation Representation

The selection of an appropriate chromosome representation of candidate solutions to the problem at hand is the foundation for the application of GA to a specific problem. In this section an efficient representation whose characteristics can be defined to be domain-independent, indirect, and of permutation type is suggested for the chromosome of the problem, based on theories of the parallel machines scheduling. The corresponding schedule builder is introduced.

3.1 Problem-Specific Knowledge

It is necessary to introduce the exact meanings of some problem-specific knowledges. Assume that job j arrives in the shop at time r_j and that machine i is firstly available for processing at time a_i. When all r_j are 0, we define the situation as the static job arrival. When all a_i are 0, we define it as the static machine availability. When both r_j and a_i are static, it is defined as the static case.

When both are dynamic, we define it as the dynamic case. In a mixed case, we specify which is static and which is dynamic. When all machines are equal, that is, $p_{ij}=p_j$, for all i, such a situation is defined as the equal parallel machine case. When all machines differ from each other only by constant speed factors, that is, $p_{ij}=p_j/s_i$, where s_i is the speed factor for machine i, it is defined as the proportional parallel machine case. Finally, when the p_{ij} are arbitrary, it is defined as the general parallel machine case. Objective function $Z=f(W_1, W_2, \ldots, W_n)$ is defined to be regular when Z will increase only if at least one W_j increases. Problem specific knowledges for the problem can be described by the follwing three Propositions, as in [12].

Proposition 1 Consider a general parallel machine problem with dynamic arrivals, dynamic availability, and regular objective function. There is an optimal solution of the form: sequence the n jobs in an optimal order (method for obtaining optimum order is not known) in a list, and schedule each in turn on the machine that can finish it first.

Proposition 2 For the static single machine problem with minimizing maximum lateness as the objective, there is an optimal policy satisfying $d_1 \leq d_2 \leq \ldots \leq d_n$ (this is called EDD, the "earliest due date rule").

Proposition 3 The maximum lateness objective and the maximum tardiness objective are both minimized by the same policy. Thus for the static case EDD minimizes maximum tardiness as well.

From proposition 1 the general parallel machine problem is reduced to a one dimensional sequencing problem. Propositions 2 and 3 are exploited when sequencing of jobs are made on each machine to minimize the maximum tardiness (i.e., the second objective function).

3.2 Chromosome Representation

It would be helpful to retrospect the previous representational schemes research on scheduling before we design the chromosome representation for our problem. The representational schemes of previous GA-based research on scheduling can be classified into three types as follows[3]:

a) Domain-independent indirect representation: It contains no auxiliary information regarding the particular domain. Since the representation does not represent a schedule, a transition from chromosome representation to a legal production schedule has to be made by a schedule builder prior to evaluation. Common recombination operators that are also applicable to other domain problems are applied.

b) Problem-specific indirect representation: Knowledge of the problem domain is explicitly represented

in the chromosomes. In order to work on the resultant expanded representation, new domain-dependent genetic operators have to be designed.

c) Direct representation: The production schedule itself is used as a chromosome. No decoding procedure is therefore necessary.

In another respect, the representation can also be classified into two types, binary representation and permutation representation. According to which representation is used, the genetic operator to be used should be different.

To apply GAs ideally, the following criteria should be satisfied[11].

a) Completeness: Any solution should have its encodings.

b) Soundness: Any code produced by genetic operators should have its corresponding solution.

c) Non-redundancy: Codes and solutions should correspond one to one.

d) Characteristics-preservingness: Children should inherit useful characteristics from parents.

If it is impossible to represent a problem with standard coding techniques in a way that infeasible solutions cannot get into the coding scheme, one type of remedy could be used among the three given below[1].

a) A detected infeasible genotype can be penalized with a relatively bad fitness value to drive the individual out from the population.

b) A detected infeasible genotype can be transformed into a similar feasible genotype by a small modification of the string representation.

c) A non-standard representation can be designed, which avoids the coding of infeasible solutions. If ordinary genetic operators destroy the scheme-structure of the representation, new operators (at least a crossover) preserving this structure, have to be developed.

Now we represent a chromosome for the problem as a permutation. A permutation is defined as an ordering of the elements of a finite job set J of size n. A permutation is for example given by

$$(P_1, P_2, \ldots, P_n),$$

where P_j denotes the jth job of a job order which is referred to in Proposition 1. The number of different permutations is equal to $n!$.

This representation is actually the result of applying the problem-specific knowledge (Proposition 1) to the problem. But, the representation literally looks like a domain-independent indirect representation because of the contents of Proposition 1. So we do not need to design a new genetic operator, but use one of the conventional permutation operators.

In view of proposition 1, we see that the representation is complete, sound and non-redundant. Charateristic-preservingness of GA can be obtained later in the process of choosing a crossover operator. The schedule builder in Section 3.3 always transforms each permutation into a corresponding

feasible solution by utilizing propositions 1, 2 and 3. Therefore, any type of remedy to avoid generating infeasible solutions must not be taken any more.

3.3 Schedule Builder

A schedule builder should make a transition from chromosome representation to a legal production schedule and evaluate its performance. It should be chosen with respect to the performance measure of optimization. Based on the three propositions in Section 3.1, the schedule building procedure for a string is described as follows: It is assumed that n jobs are already assumed to be sequenced in an order $(J_1, J_2, \ldots, J_n)$.

Step 1: Assign the job J_j on the machine that can finish it first, for $j=1,2,\ldots,n$.

Step 2: The makespan is derived from the equation $C_{\max} = \max\{C_i \mid i \in I\}$.

Step 3: Sort the jobs assigned to macine i in ascending order of the due date d_j, for $i=1,2,\ldots,m$.

Step 4: The maximum tardiness is derived from equations

$$T_j = \max(0, C_j - d_j) \quad \text{and} \quad T_{\max} = \max\{T_j \mid j \in J\}.$$

Step 5: Obtain the value of the objective function from $w_1 C_{\max} + w_2 T_{\max}$.

4. Crossover Operator and Niche Formation

Several crossover operators for permutation have been suggested. Among them four operators given below have been frequently used: partially matched crossover (PMX), order crossover (OX), cycle crossover (CX), and uniform order-based crossover (UOBX). For a detailed information of these operators, see Davis[4] and Goldberg[7]. Unfortunately it is not known which operator is the best and how much differences in solution performance exist among these four crossover operators for our problem.

Often the crossover operator is too strong and it ends up driving the GA to create a population of individuals that are almost exactly the same. When the population consists of similar individuals, the likelihood of finding new solutions typically decreases. On one hand we want the GA to find good individuals, but on the other hand we want it to maintain diversity. One of the most common methods for maintaining diversity is to form niches through De Jong style crowding. De Jong style crowding is basically the same thing as replace-most-similar replacement scheme. When new offsprings are created, they replace the individuals in the population that are most similar to them. If the niche formation is induced for a multi-modal function in GA, then the domain would be explored more effectively. We want to identify how much difference exists in solution performance between the method with niche

formation and that without it.

Depending upon the combination of which operator is used and whether a niche formation method is used or not, the performance of GA to a given problem could vary significantly. To identify the best crossover operator and to decide if we should apply the niche formation method or not, we need to simulate GA under each combination of the two factors levels for some test problems

5. Computational Results

5.1 Test Problems

The test problems are generated from the combination of three different number of jobs (20, 40 and 80) and three different number of machines (3, 5 and 10). The job processing time is sampled from uniform distribution [1, 100] or [50, 100]. The due dates are computed using two parameters: the tardiness factor f, where $0 \le f \le 1$ and the range factor r, where $0 \le r \le 1$, as is the case in Suresh and Chaudhuri[16]. The due date d_j for a job j is sampled from the uniform distribution in the range [d-w/2, d+w/2], where d and w are computed as given below:

$$d = E(p)(1-f)n/m,$$

where $E(p)$ is the mean processing time.

$$w = (r/m)\sum_{j=1}^{n} p_j^*,$$

where $p_j^* = \min\{p_{ij} \mid 1 \le i \le m\}$.

The values of the two factors f and r are selected such that the due dates are generated with varying degree of tightness. In detail, the values are set to as follows: tardiness factor f = 0.5 or 0.8 and range factor r = 0.5 or 0.8. Specifications of the resulting nine test problems are described in Table 2.

Table 2. Specification of Nine Test Problems

Problem ID	Number of Machines	Number of Jobs	Processing Time	f Value	r Value
1	3	20	U[1, 100]	0.5	0.5
2	3	40	U[50,100]	0.5	0.8
3	3	80	U[1, 100]	0.8	0.5
4	5	20	U[50,100]	0.8	0.5
5	5	40	U[1, 100]	0.5	0.5
6	5	80	U[50,100]	0.5	0.8
7	10	20	U[1, 100]	0.8	0.5
8	10	40	U[50,100]	0.5	0.8
9	10	80	U[1, 100]	0.5	0.5

5.2 Design of Experiment

For each of the nine test problems, GA control parameters are set to be as follows: the crossover rate is set to 0.6 and the mutation rate is set to 0.01. The size of the population is set to be equal to the number of jobs for the problem. The stochastic remainer selection method[7] is adopted as a selection scheme. The swapping mutation operator, which swaps two randomly selected fields from a permutation-type chromosome, is applied here. The elitism is adopted. An execution of GA is terminated when either the convergence ratio of alleles in the population is over 95% or the maximum allowable number of generations (here it is set to the 5000th generation) has been reached.

The purposes of experiment are to identify the best crossover operator among the four different types of crossover operators (i.e., PMX, OX, CX and UOBX) and to make decision about whether we should apply the niche formation method or not (i.e., YES, NO). We experiment with the model of the three-way factorial design. We define the first factor of our experimental design to be the type of crossover operators. The levels of the first factor are composed of PMX, OX, CX and UOBX. The second factor is defined to be the alternative of decision making about niche formation. The levels of the second factor are composed of YES and NO. For the purpose of the experiment, a third factor is defined to be the type of the test problem. The levels of the third factor are composed of nine problems, from problem 1 through problem 9. Both the first and the second factor belong to the fixed factor. But the third factor (problem type) is a random factor. The experiment is repeated three times under each of the total 72 (= 4 x 2 x 9) combinations of the three factors, by changing only the value of the initial random seed. GALOPPS (Genetic ALgorithm Optimized for Portability and Parallelism System)[8] is selected as the software package with which our GA is implemented. The schedule builder and other required modules are coded in C in GALLOPS. Simulations are implemented on a personal computer under a UNIX environment.

5.3 Analysis of Results

Both the obtained solution and the number of created generations for each of 216 implementations are summarized in Table 3.

Notice that without the exception for the OX operator, GA continued until the maximum allowable number of generations was reached. The number of generations created during the execution of GA varies significantly depending on the operator used for a given problem. To evaluate the performance of each combination of the two factors (type of crossover and decision making about speciation), the performance of solution and the number of generations created are selected as the first and the second criterion respectively.

Table 3. Solution and Number of Generations for Each Implementation

			PMX		CX		OX		UOBX	
			Value	Gen. no.	Value	Gen. no.	Value	Gen. no.	Value	Gen. no.
Problem 1	Y	1st	98.5	71	105.5	16	96	nc	96	122
		2nd	97	65	97	11	96	nc	97	203
		3rd	96	49	102.5	12	96	nc	96	179
	N	1st	96	26	96	26	96	nc	96	68
		2nd	97	48	98.5	23	96	nc	97	114
		3rd	96	58	100.5	275	96	nc	96	111
Problem 2	Y	1st	507.5	335	512	18	482.5	nc	481.5	nc
		2nd	495.5	299	505	27	484.5	nc	477.5	nc
		3rd	484.5	919	515.5	27	487.5	nc	479.5	nc
	N	1st	491.5	363	520	31	477.5	nc	476	4122
		2nd	491.5	153	523	22	480	nc	474	516
		3rd	488	147	519.5	29	481	nc	481	479
Problem 3	Y	1st	518.5	nc	539	63	524	nc	524.5	nc
		2nd	516.5	nc	556	46	520.5	nc	526	nc
		3rd	512	nc	552	36	512.5	nc	523.5	nc
	N	1st	515.5	2392	541.5	89	525.5	nc	520.5	nc
		2nd	513.5	2000	539	32	512.5	nc	520	nc
		3rd	519.9	2263	554	47	517	nc	523	nc
Problem 4	Y	1st	210.5	43	211	29	201	nc	208	155
		2nd	206	181	222.5	11	201	nc	208	177
		3rd	201	42	218.5	19	201	nc	201	134
	N	1st	206.5	33	209.5	45	201	nc	201	66
		2nd	207	55	208	21	201	nc	205	90
		3rd	208	49	216	104	201	nc	208	106
Problem 5	Y	1st	69	561	73	33	69	nc	69	4076
		2nd	70	709	72	15	68	nc	69	4521
		3rd	69	177	74.5	23	68.5	nc	69	522
	N	1st	69	154	73	41	69.5	nc	69	541
		2nd	69	132	73	108	68	nc	69	630
		3rd	70	204	73.5	24	68	nc	68	394
Problem 6	Y	1st	558	nc	605.5	33	571	nc	564.5	nc
		2nd	557	nc	603	117	556	nc	566	nc
		3rd	557	nc	599.5	51	569	nc	574.5	nc
	N	1st	554.5	nc	593.5	44	566	nc	571	nc
		2nd	561	nc	584.5	85	569.5	nc	556	nc
		3rd	560.5	nc	590.5	59	551	nc	564	nc
Problem 7	Y	1st	22.5	99	24	17	22.5	nc	22.5	248
		2nd	22.5	67	27.5	94	22.5	nc	22.5	418
		3rd	22.5	86	22.5	90	22.5	nc	22.5	462
	N	1st	22.5	63	24	46	22.5	nc	24	50
		2nd	22.5	37	24	47	22.5	nc	24	99
		3rd	22.5	63	22.5	36	22.5	nc	22.5	194
Problem 8	Y	1st	153	217	160.5	23	147.5	nc	141.5	862
		2nd	148.5	185	170	21	150	nc	150.5	834
		3rd	150	199	157.5	79	150	nc	145	2261
	N	1st	147.5	288	161	24	148.5	nc	148	834
		2nd	152.5	210	153	22	140.5	nc	144	983
		3rd	146	218	163.5	26	144	nc	148	333
Problem 9	Y	1st	48.5	nc	52	55	49.5	nc	46.5	nc
		2nd	47.5	nc	51	46	49.5	nc	46.5	nc
		3rd	47	nc	52.5	41	49.5	nc	47	nc
	N	1st	47.5	1922	55	50	48.5	nc	46.5	nc
		2nd	46.5	2979	50	109	49	nc	47	nc
		3rd	47.5	2140	51.5	82	49.5	nc	47	nc

† nc means the experiment was not converged but continued to the 5000th generation.

Table 4. ANOVA Table

Source	DF	SS	MS	F Value	Pr > F
Model	36	9542094.9	265058.2	17772.88	0.0001
Error	179	2669.5	14.9		
Corrected Total	215	9544764.4			
Source	DF	SS	MS	F Value	Pr > F
OPERATOR	3	8432.2	2810.7	188.47	0.0001
PROBLEM	8	9526323.1	1190790.4	79845.76	0.0
CROWDF	1	109.1	109.1	7.31	0.0075
OPERATOR*PROBLEM	24	7230.5	301.3	20.20	0.0001

As a first step, variance is analyzed for the data in Table 3 to evaluate only with the first criterion (the solution performance). The result of the variance analysis is shown in Table 4.

From Table 4, we see that each of the three factors (the problem type, the crossover type and the crowding factor) and the interaction effect between "the problem type" factor and "the crossover type" factor are highly significant. As expected, "the problem type" factor is highly significant, since the scales of the input data were set to differ significantly among the problems. The high significance of "the crossover type" factor means that there exists a high significant difference in the solution performance between at least a couple of crossover operators. As "the niche formation" factor is significant, the Tukey's Studentized range test is subsequently implemented. The result of the test shows that the niche formation method aggravates somewhat the performance on the average of solutions, contrary to the generally known effect of the speciation. The phenomenon might be due to the difficulty of adjusting control parameters to diversify chromosomes in the population. To get more detailed information about superiority among different operators, multiple comparisons were made by implementing the Tukey's Studentized range test using the SAS package. As the interaction effect between "the problem type" factor and "the crossover type" factor is highly significant, the Tukey's test is implemented for each level of "the problem type" factor. The result of implementation is summarized in Table 5.

From Table 5, we cannot discriminate the best operator among PMX, OX and UOBX operators, because the operator whose performance is superior to others, differs depending upon the problem and the performances of all the three operators are approximately the same. Meanwhile, the operator CX is somehow inferior to these three operators. The inferiority of the CX operator could be explained in part by the fact that the number of generations created under CX operator until convergence is much less than other operators and that accordingly CX operator has less opportunity of searching the domain than the others.

The previous comparisons are based on the solution performance. Now another comparison is made

Table 5. Results of Tukey's Studentized Range Test

problem		levels of operator			
		PMX	CX	OX	UOBX
1	mean	96.75	100	96*	96.33
	SD	0.99	3.58	0.00	0.52
2	mean	493.08	515.83	482.17	478.25*
	SD	7.98	6.55	3.52	2.95
3	mean	515.83*	546.91	518.67	522.92
	SD	2.75	7.91	5.61	2.31
4	mean	206.5	214.25	201.00*	205.1
	SD	3.13	5.68	0.00	3.43
5	mean	69.33	73.17	68.50*	68.83
	SD	0.52	0.82	0.63	0.41
6	mean	558.00*	596.08	563.75	566.00
	SD	2.43	8.00	8.26	6.38
7	mean	22.5 *	24.08	22.5 *	23.00
	SD	0.00	1.83	0.00	0.77
8	mean	149.58	160.91	146.75	146.17*
	SD	2.78	5.72	3.78	3.27
9	mean	47.4	52.00	49.25	46.75*
	SD	0.66	1.70	0.42	0.27

Asterisk (*) is marked in the cell which corressponds to the operator under which the best solution was generated.

based on the second criterion (the number of generations created) for the three operators (OX, UOBX and PMX), showing nearly the same performances. Among the three operators, there is a large difference in the number of created generations to obtain the solution in Table 3 for a problem. From Table 3, we can see that less generations were created under PMX operator until convergence than under UOBX operator for almost all the problems (except one problem), while the two corresponding solutions from both PMX and UOBX are nearly the same. Therefore we can conclude that the PMX operator is definitely superior to the UOBX operator.

Finally, it remains undetermined as to which operator is superior betweeen PMX and OX. The sequence of solutions under OX operator never converges due to the property of the operator itself. That is, GA is executed until the maximum allowable number of generation (5000th generation) has been reached. This property of OX operator lets the direct comparison difficult to be made. So we make an indirect comparison as follows:

A couple of solutions from under both PMX and OX at a designated generation are derived for each problem and are summarized in Table 6. The generation at which the sequence of solutions begin to converge above 95% under PMX operator is designated.

Table 6. Solutions under PMX and OX at a Given Number of Population

		Solution under PMX			Gen. no.			Solution under OX		
		1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
Problem 1	Y	98.5	97	96	71	65	49	97	96	97
	N	96	97	96	26	48	58	97	96	96
Problem 2	Y	507.5	495.5	484.5	335	299	919	500	484.5	491.5
	N	491.5	491.5	488	363	153	147	492	501.5	502
Problem 3	Y	518.5	516.5	512	nc	nc	nc	524	520.5	512.5
	N	515.5	513.5	519.9	2392	2000	2263	525.5	512.5	517
Problem 4	Y	210.5	206	201	43	181	42	208	204	208
	N	206.5	207	208	33	55	49	207.5	208	210
Problem 5	Y	69	70	69	561	709	177	70	69	70.5
	N	69	69	70	154	132	204	70	71.5	70
Problem 6	Y	558	557	557	nc	nc	nc	571	556	569
	N	554.5	561	560.5	nc	nc	nc	566	569.5	551
Problem 7	Y	22.5	22.5	22.5	99	67	86	22.5	22.5	22.5
	N	22.5	22.5	22.5	63	37	63	22.5	22.5	22.5
Problem 8	Y	153	148.5	150	217	185	199	153	150	150
	N	147.5	152.5	146	288	210	218	148.5	153	154
Problem 9	Y	48.5	47.5	47	nc	nc	nc	49.5	49.5	49.5
	N	47.5	46.5	47.5	1922	2979	2140	48.5	49	49.5
		240.9			mean			242.9		

† nc means that the sequence of solutions did not converged but continued to the maximum allowable 5000th generation in the experiment.

From Table 6, we see that means of solutions under both operators are nearly the same. (240.9 for the PMX and 242.9 for the OX) But the OX operator has a serious shortcoming that we cannot know in advance when to stop the execution of GA, because OX operator itself does not permit the convergence of solutions. Therefore we conclude that PMX crossover operator is the best for our problem with respect to both the performance of solution and the operational convenience.

6. Conclusion

In this paper the parallel machines shop scheduling problem was considered as a representative of the multi-objective combinatorial problems. A permutation representation which can effectively represent the chromosome was introduced. An efficient schedule builder was made by employing both the theory of the exact analysis and a simple heuristic. Next we showed through simulations that the niche formation method through the crowding factor does not contribute to obtaining better solutions and that the PMX operator is the best crossover operator for our problem in view of the performance of

solution and the operational convenience.

Traditionally scheduling problems were investigated in the area of operations research, from which comprehensive research results are available. Therefore as a topic for further research, it is worthwhile to study a number of useful ways by which the conventional optimization theory can be exploited in the process of applying GAs such as chromosome representation or searching the domain space of GAs. On the other hand, more efforts are required to reveal the characteristics of the various types of crossover operators and thus to make possible the choice of more adequate operator for a given problem.

References

[1] Bierwirth, C., "A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms", OR Spectrum, 17, (1995), pp. 87-92.

[2] Bierwirth, C., Mattfeld D. C. and Kopfer, H., "On Permutation Representations for Scheduling Problems", Research paper, Dept. of Economics, University of Bremen, Germany, 1996.

[3] Bruns, R., "Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling", Proceedings of International Conference on Genetic Algorithms, (1993), pp.352-359.

[4] Davis, L., "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York, 1991.

[5] Fang, H., Ross, P. and Corne, D., "A promising Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems", Proceedings of International Conference on Genetic Algorithms, (1993), pp.375-382.

[6] Filho, J. L. R., Treleaven, P. C. and Alippi, C., "Genetic-Algorithm Programming Environments", Computer, 27, (1994), pp.28-43.

[7] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[8] Goodman, E. D., "An Introduction to GALOPPS (The "Genetic ALgorithm OPtimized for Portability and Parallelism" System) Release 3.01, Technical Report #95-06-01, Michigan State University, East Lansing, 1995.

[9] Han, Y. H. and Ryu, K. R., "Genetic Algorithms for Optimization: A Case Study of Machine-Part Group Formation Problems", Korean Management Science Review, 12(2), (1995), pp.105-127.

[10] Hou, E. S. H., Ansari, N. and Ren, H., "A Genetic Algorithm for Multiprocessor Scheduling", IEEE Transactions on Parallel and Distributed Systems, 5(2), (1994), pp.113-120.

[11] Kobayashi, S., Ono, I. and Yamamura, M., An Efficient Algorithm for Job Shop Schedulings Problems, Proceedings of International Conference on Genetic Algorithms, (1995), pp.506-511.

[12] Morton, T. E. and Pentico, D. W., Heuristic Scheduling Systems, John Wiley & Sons, 1993.

[13] Shaw, K. J. and Fleming, P. J., "Initial Study of Multi-Objective Genetic Algorithms for Scheduling the Production of Chilled Ready Meals", Research Paper, Automatic Control & Systems Engineering Department, University of Sheffield, Sheffield, UK, 1996.

[14] Srinivas, M. and Patnaik, L. M., "Genetic Algorithms: A Survey", Computer, 27 (1994), pp.17-26.

[15] Stanley, T. J. and Mudge, T., A Parallel Algorithm for Multiobjective Microprocessor Design Problems, Proceedings of International Conference on Genetic Algorithms, (1995), pp.597-604.

[16] Suresh, V. and Chaudhuri, D., "Bicriteria Scheduling Problem for Unrelated Parallel Machines", Computers and Industrial Engineering, 30, 1, (1996), pp.77-82.