International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **3**, March **2013**

ICIC International ©2013 ISSN 1349-4198
pp. **1045–1063**

HYBRID TAGUCHI-BASED GENETIC ALGORITHM FOR FLOWSHOP SCHEDULING PROBLEM

Ching-I Yang[1], Jyh-Horng Chou[1,2,*] and Ching-Kao Chang[1]

[1]Institute of Engineering Science and Technology
National Kaohsiung First University of Science and Technology
No. 2, Jhuoyue Road, Nanzih District, Kaohsiung 811, Taiwan
{ u9315908; cyrus }@nkfust.edu.tw; *Corresponding author: choujh@nkfust.edu.tw

[2]Department of Electrical Engineering
National Kaohsiung University of Applied Sciences
No. 415, Chien Kung Road, Kaohsiung 807, Taiwan
choujh@cc.kuas.edu.tw

Received January 2012; revised May 2012

Abstract. *A hybrid Taguchi-based genetic algorithm (HTGA) is developed for solving multi-objective flowshop scheduling problems (FSPs). Search performance is improved by using Taguchi-based crossover to avoid scheduling conflicts, and dynamic weights are randomly selected by a fuzzy inference system. The conventional approach to selecting dynamic weights randomly ignores small value for the objective when the weight value is very small. A numerical example is given to demonstrate the application of the proposed hybrid method and its good performance. The numerical results show that the hybrid method effectively enhances the genetic algorithm. The improvement achieved by the HTGA also exceeds that obtained by existing methods reported in the literature for finding Pareto optimum solutions for FSPs. Therefore, the HTGA effectively solves multi-objective flowshop scheduling problems.*

Keywords: Flowshop scheduling problem, Dynamic weight, Taguchi-based crossover, Fuzzy inference system

1. **Introduction.** Although genetic algorithms have proven effective for solving single-objective optimization problems [1-3], obtaining effective solutions for real world problems often requires simultaneous consideration of multiple objective functions. Another practical problem is that conflicts in the considered objectives often make achieving a perfect multi-objective solution that simultaneously optimizes each objective function almost impossible. Therefore, when solving multi-objective problems, the ultimate goal is finding the best solution set, i.e., the Pareto optimum solutions. After considering tradeoffs, the decision maker can then choose the preferred solution.

In the industrialized world, resource scarcity is becoming a critical problem. Scheduling optimizes the use of available resources and satisfies performance measurement criteria. Multi-objective flowshop scheduling is one of the most well known flowshop scheduling problems (FSPs). Finding the optimal solutions to scheduling problems is usually very difficult for production plant engineers. Therefore, the motivation of this paper is to construct a novel algorithm for finding Pareto optimal solutions, which are better than existing solutions for the above reasons. To increase machine availability, completion time (makespan) must be minimized. The widespread adoption of just in time (JIT) manufacturing, in which jobs are processed only as needed, has expanded the role of tardy production in process planning. To improve completion time and to minimize the

tardiness problem, effective solutions are needed for the FSP with the two objectives of minimizing makespan and maximum tardiness.

The GA optimization is applied to solve the FSP, which can be defined as the problem of dealing with n jobs on m machines or work centers in a facility in which all jobs are processed on all machines in the same sequence. The scheduling procedure known as the Johnson rule is used to solve the two-machine problem [4]. Problems involving more than two machines or jobs are called NP-complete or NP-hard problems [5]. Some of the studies investigating the use of genetic algorithms (GAs) to solve this problem [6-23] have combined the objective functions into a scalar fitness function so that the multi-objective optimization problem can be solved by a single objective genetic algorithm. Assigning a constant to each weight of the scalar fitness function obtains a constant search direction. Ishibuchi et al. [10,13] first proposed the use of dynamic weights to find the best feasible solutions by increasing the number of search directions. Nevertheless, the conventional approach of randomly selecting dynamic weights may ignore a small value objective with a very small dynamic weight value. To solve this problem, we propose the use of dynamic weights controlled by a fuzzy inference system (FIS) to enhance search ability instead of using randomly selected dynamic weights. Additionally, instead of using a two-point cut for the crossover operation based on a random process, e.g., Ishibuchi et al. [10,13], this study proposes a more systematic reasoning way of Taguchi-based crossover operation to generate improved offspring. Moreover, the Pareto optimum solutions found so far are generated by using a new population consisting of the Pareto optimum solutions for previous generations and for the current population after local search. Thus, in addition to weight selection and crossover operations, updating the Pareto optimum solutions found so far is also essential for solving multi-objective optimization problems. Therefore, the objective of this study was to develop an improved GA approach with three features: (I) enhancement of search capability by the use of dynamic weights selected by FIS instead of dynamic weights selected randomly [10,13]; (II) a systematic reasoning approach to performing a Taguchi crossover operation that avoids the scheduling conflicting problem and finds an optimal solution instead of performing crossover operation based on a random process [10,13]. The two major tools in the Taguchi crossover operation are (1) the signal-to-noise ratio (SNR) used to measure quality and (2) the orthogonal arrays used for simultaneously studying multiple parameters; (III) an easy way to generate the Pareto optimum solutions found so far so that the best offspring (solutions) can be retained. This proposed hybrid Taguchi-based genetic algorithm (HTGA) provides dynamic weight selection by FIS and a new way of using systematic reasoning in the Taguchi-based crossover operation to generate an optimal individual. The best Pareto optimum solutions found so far are always kept in the next generation. Because of the combined use of Taguchi-based crossover operation and GA, the proposed HTGA is robust and achieves quick convergence.

A review of the GA literature [6-23] shows that the algorithms [10,13] proposed by Ishibuchi et al. for solving the FSP are structurally complete. Although the multi-objective genetic local search (MOGLS) algorithm [10] and its modification [13] have shown good potential, further improvement in solution performance is needed. The proposed HTGA was compared not only with the MOGLS algorithm as reported by Ishibuchi and Murata [10], but also with the modified MOGLS algorithm developed by Ishibuchi et al. [13]. The comparison results show that HTGA is better than both the original MOGLS algorithm and the modified MOGLS algorithm.

This paper is organized as follows. Section 2 introduces the FSP with two objective functions. Section 3 describes the HTGA. A numerical example is discussed to illustrate the proposed method in Section 4. Section 5 presents the conclusions.

2. **Flowshop Scheduling Problem.** The problem considered here was finding the job schedule with the minimum weighted sum of completion time and maximum tardiness. Because the feasible solutions are widely spread in the solution space, Ishibuchi et al. [10,13] argued that unit searching directions can overlook some better solutions when using fixed weight method. They proposed the use of dynamic weights to find better feasible solutions by increasing the number of searching directions. Therefore, this study evaluated the use of dynamic weights controlled by an FIS to improve search ability. The proposed HTGA for solving the FSP has the following properties:

(1) All jobs are available at time zero.
(2) Physical buffer space between two successive machines is sufficient.
(3) Setup times for the operations are sequence independent and are included in the processing times.
(4) All machines are continuously available.
(5) Individual operations are not preemptive.

The sequence of n jobs is denoted by an n dimensional vector $(J_1, J_2, \ldots, J_n)$. The n jobs are processed on a series of machines $(M_1, M_2, \ldots, M_m)$ in the same sequence. Where J_i denotes the i-th processing job and M_j denotes the j-th machine, the processing time of job i on machine j is $P_{i,j}$. The completion time of job i is defined as $C_{i,m}$, that is, the completion time of job i on the last machine m, where

$$\begin{cases} C_{1,1} = P_{1,1}, \\ C_{1,j} = C_{1,j-1} + P_{1,j}, \quad \text{for } j = 2, \ldots, m, \\ C_{i,1} = C_{i-1,1} + P_{i,1}, \quad \text{for } i = 2, \ldots, n, \\ C_{i,j} = \max\{C_{i,j-1}, C_{i-1,j}\} + P_{i,j} \quad \text{for } i = 2, \ldots, n, \text{ for } j = 2, \ldots, m. \end{cases} \tag{1}$$

The makespan of the sequence of n jobs is defined as $C_{\max} = C_{n,m}$, that is, the maximum completion time of the last job n on the last machine m. The tardiness T_i of job i is defined as

$$\begin{cases} T_i = \max\{(C_{i,m} - D_i), 0\}, \\ \text{for } i = 1, 2, \ldots, n, \text{ and } D_i \text{ is the due date of job } i. \end{cases} \tag{2}$$

The maximum tardiness $T_{\max}$ of the scheduling is defined as

$$T_{\max} = \max\{T_1, T_2, \ldots, T_n\}. \tag{3}$$

3. **Hybrid Taguchi-Based Genetic Algorithm for FSP.** This section introduces the use of the proposed HTGA for solving FSPs. Its objectives are to minimize makespan and to minimize maximum tardiness. The HTGA combines GA with local search and an elitist preservation strategy. The steps of the HTGA approach are depicted in Figure 1 and are described as follows. The parameters used in the algorithm are set as shown in Table 6.

(I) Encoding a schedule

A sequence of jobs $\boldsymbol{S} = (x_1, x_2, \ldots, x_n)$ is a chromosome representing the job processing sequence. That is, the processing of job x_1 is followed by the processing of job x_2, and so on.

(II) Initialization

Randomly generate an initial population in which population size (q) is equal to $2n$, where n is the number of jobs.

(III) Evaluation

For a solution x, a fitness function for the HTGA algorithm is defined by the weighted sum of n objectives:

$$f(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x), \tag{4}$$

where $f_1(x), f_2(x), \ldots, f_n(x)$ are n objectives to be maximized and $w_1 + w_2 + \ldots + w_n = 1$. The objective is to find all non-dominated solutions for the multi-objective optimization problem.

The objective of this study is to minimize makespan and maximum tardiness. The feasible solutions are widely spread in the solution space. Ishibuchi et al. [10,13] proposed the use of dynamic weights to find better feasible solutions by increasing the number of search directions. Since the conventional approach of using randomly selected dynamic weights may ignore the small value objective when the dynamic weight value is very small, this study proposes the use of an FIS performed in the following steps.

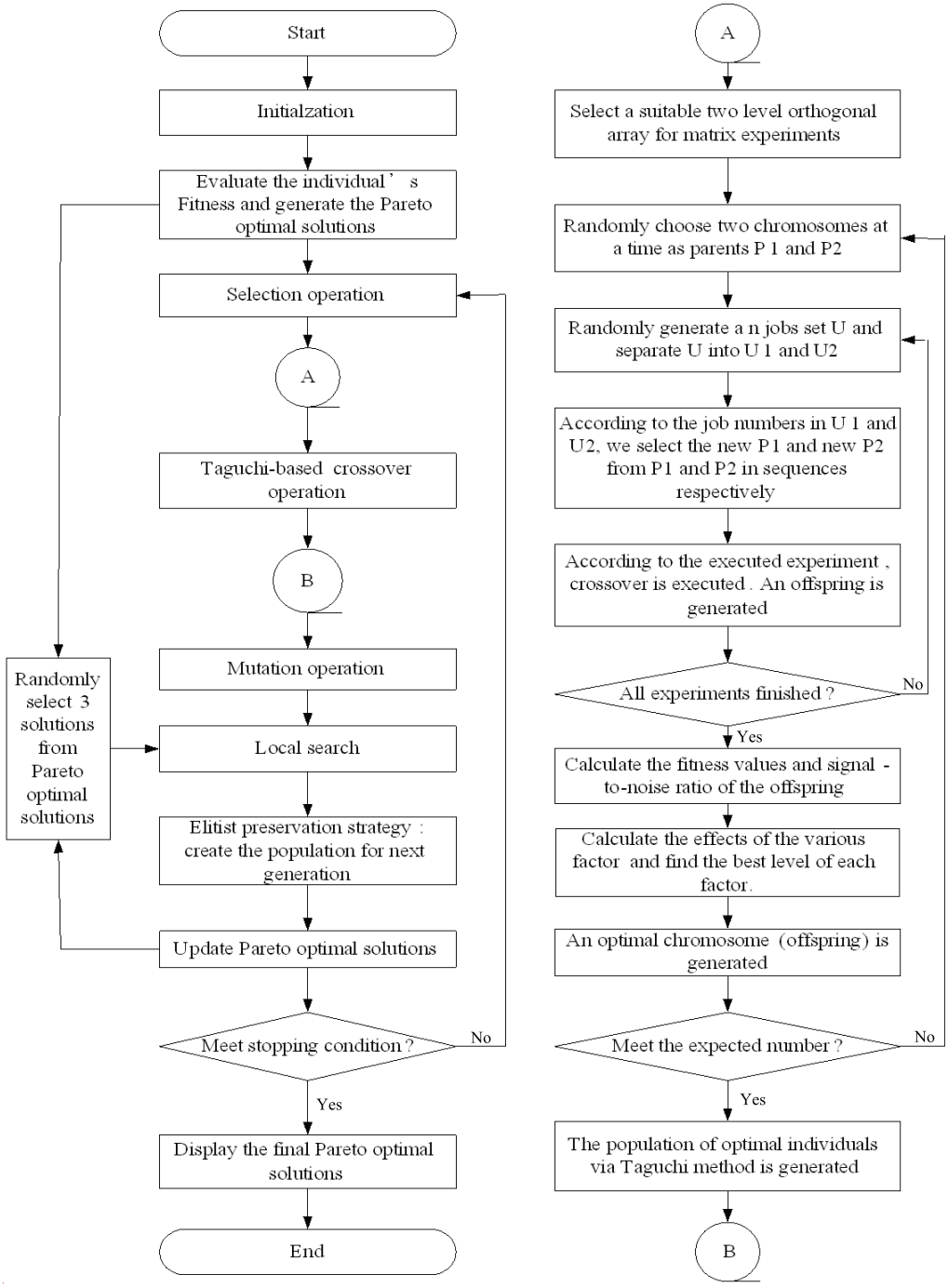

FIGURE 1. Hybrid Taguchi-based genetic algorithm for solving flowshop scheduling problem

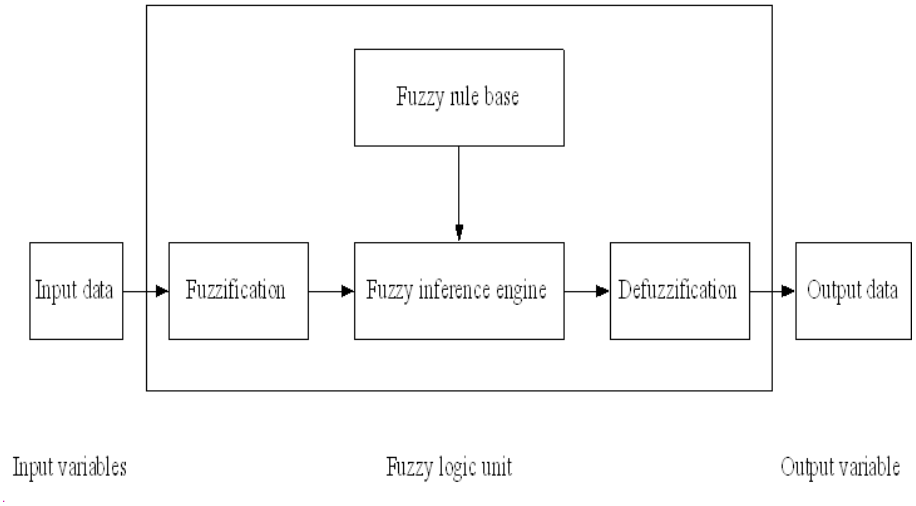

FIGURE 2. Fuzzy inference system

(1) Calculate the objective values of the makespan and the maximum tardiness of initial population.

Let $f_{ob1}(x)$ denote the makespan and $f_{ob2}(x)$ denote the maximum tardiness.

(2) Generate a set of Pareto optimal solutions for the initial population.

(3) Calculate the fitness function $f(x)$ by the weighted sum approach.

Because these two objectives should be minimized, a fitness function for solution x is defined by the weighted sum of two objectives.

$$f(x) = -w_1 f_{ob1}(x) - w_2 f_{ob2}(x), \tag{5}$$

where w_1 and w_2 are the dynamic weights of the objective function $f_{ob1}(x)$ and $f_{ob2}(x)$, respectively, and $w_1 + w_2 = 1$.

The conventional approach is to select the dynamic weights randomly. However, since $w_1 f_{ob_1}(x)$ decreases to a small value when weight w_1 is very small, only $f_{ob2}(x)$ is considered here. Therefore, the search direction is improved by using the fuzzy inference system to adjust w_1 and w_2.

(4) Fuzzy inference system

Figure 2 shows that the fuzzy inference system consists of input variables, a fuzzy logic unit and an output variable. The four main components of the fuzzy logic unit are the fuzzification component, the fuzzy rule base, the fuzzy inference engine and the defuzzification component.

Two objectives are used as the input variables, and the weight of $f_{ob1}(x)$ is used as the output variable.

(4-1) Normalizing input variables

Since the two input variables have different ranges, the values for the two objectives are normalized to within the same range, from zero to one, before fuzzification. The transformation used in this study [24] is

$$x_{new} = \frac{x_{old} - x_{\min}}{x_{\max} - x_{\min}}(D_{\max} - D_{\min}) + D_{\min}, \tag{6}$$

where

$x_{\min}$: minimum value of input variables,

$x_{\max}$: maximum value of input variables,

$D_{\min} = 0$ and $D_{\max} = 1$.

For example: $n = 10$ (see Table 1).

TABLE 1. Normalization of input variables

No.	Chromosomes	Before normalization		After normalization	
		$C_{\max}$	$T_{\max}$	$C_{\max}$	$T_{\max}$
1	8 9 2 10 1 4 3 7 5 6	916	319	0.8315	0.3559
2	9 8 3 1 7 10 2 5 4 6	907	453	0.7809	0.7500
3	3 5 4 6 2 7 8 9 1 10	860	326	0.5169	0.3765
4	10 5 7 1 3 8 6 9 2 4	819	450	0.2865	0.7412
5	8 2 1 5 10 9 3 7 6 4	905	536	0.7697	0.9941
6	2 10 7 4 5 3 9 6 8 1	781	311	0.0730	0.3324
7	4 1 7 9 5 3 6 2 8 10	920	435	0.8539	0.6971
8	5 3 1 4 9 8 6 10 7 2	844	448	0.4270	0.7353
9	8 5 7 10 9 3 1 2 6 4	907	538	0.7809	1.000
10	9 4 6 1 10 2 5 3 8 7	818	352	0.2865	0.4529
11	6 4 5 9 2 1 10 8 3 7	768	307	0	0.3206
12	10 8 9 7 5 3 1 2 4 6	924	470	0.8764	0.8000
13	10 9 3 1 2 5 4 6 7 8	791	354	0.1292	0.4588
14	2 7 6 10 3 4 8 1 9 5	798	198	0.1685	0
15	5 2 7 4 1 10 8 3 9 6	851	291	0.4663	0.2735
16	6 10 9 7 2 8 5 3 4 1	855	453	0.4888	0.7500
17	10 9 4 6 2 8 5 7 1 3	833	402	0.3652	0.6000
18	8 3 5 4 7 1 6 9 2 10	946	499	1	0.8853
19	4 5 10 6 1 3 7 9 8 2	793	397	0.1404	0.5853
20	10 2 3 4 9 7 8 6 1 5	892	266	0.6966	0.2000

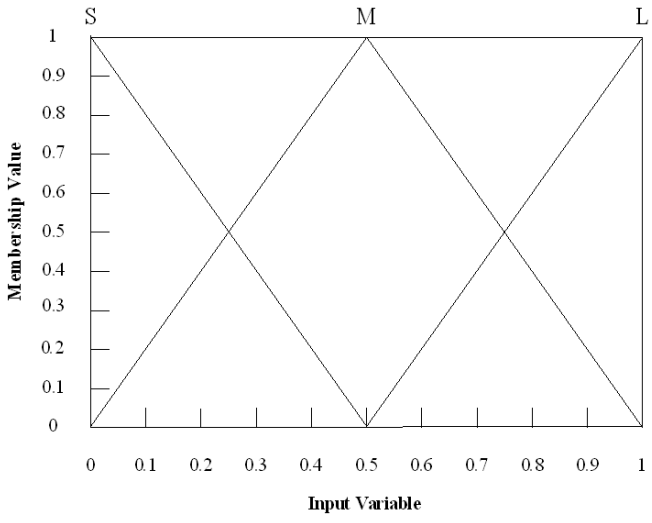

FIGURE 3. Membership function for input variables

(4-2) Fuzzification

Input variables are partitioned into three fuzzy membership functions, S, M and L (Figure 3), and output variable is partitioned into five fuzzy membership functions, S, SM, M, ML and L (Figure 4).

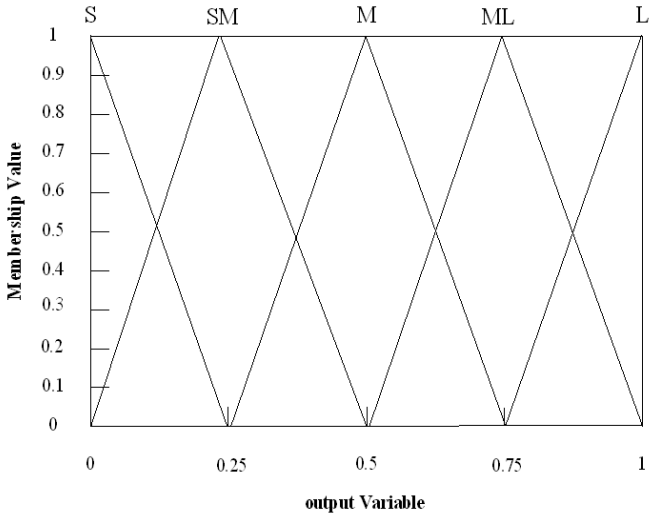

FIGURE 4. Membership function for output variable

TABLE 2. Fuzzy rule base

Output variable weight 1 y		Input variable $T_{\max}$ $x2$		
		S(B1)	M(B2)	L(B3)
Input variable $C_{\max}$ $x1$	S(A1)	M(C1)	ML(C2)	L(C3)
	M(A2)	SM(C4)	M(C5)	ML(C6)
	L(A3)	S(C7)	SM(C8)	M(C9)

(4-3) Fuzzy rule base

The fuzzy rule base consists of a group of fuzzy IF-THEN rules that include all possible fuzzy relations between inputs and outputs. Table 2 shows the nine fuzzy rules used in this study. For example,

R1: if $x1$ is A1 and $x2$ is B1, then y is C1.

R2: if $x1$ is A1 and $x2$ is B2, then y is C2.

.....

Subsets Ai, Bi and Ci are fuzzy subsets defined by their corresponding membership functions, i.e., $u_{\mathrm{A}i}$, $u_{\mathrm{B}i}$ and $u_{\mathrm{C}i}$, respectively (Figure 5).

(4-4) Fuzzy inference engine

The fuzzy inference engine performs the fuzzy inference operation by applying the Mamdani max-min inference method. The fuzzy output represents the weight of $f_{ob1}(x)$ and is determined by the following equation.

$$\mu_{C0}(y) = \left(\mu_{\mathrm{A1}}\left(x1\right) \wedge \mu_{\mathrm{B1}}\left(x2\right) \wedge \mu_{\mathrm{C1}}(y)\right) \vee \left(\mu_{\mathrm{A1}}(x1) \wedge \mu_{\mathrm{B2}}(x2) \wedge \mu_{\mathrm{C2}}(y)\right), \tag{7}$$

where $\wedge$ denotes the minimum operation and $\vee$ denotes the maximum operation.

(4-5) Defuzzification

Since the fuzzy output from the fuzzy inference engine of a fuzzy system must be a crisp number, a defuzzification method is needed to convert the fuzzy result into a crisp one. After considering various defuzzification methods, including centroid, weighted average and height method, the height method was selected because of its simplicity. First, the consequent membership function Ci was converted into a crisp consequent $y = Z_i$, where

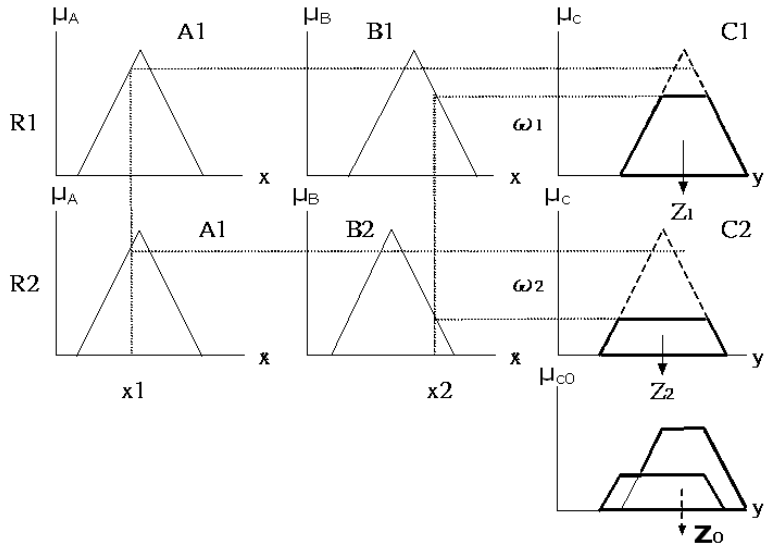

FIGURE 5. Graphic depiction of fuzzy inference computation

Z_i is the center of gravity of Ci. The crisp value of output variable Z_0 is then represented by (8).

$$Z_0 = \frac{\sum \omega_i Z_i}{\sum \omega_i}, \tag{8}$$

where ω_i is the membership function value of the ith activated rule [28,29].

(IV) Selection

The main purpose of the selection procedure is to retain the better solutions, i.e., the solutions with the best fitness values, in the solution set. From the current population, chromosomes are selected according to the selection probability $P(x)$ of chromosome x. The selection procedure is as follows.

(1) Calculate the fitness value of each chromosome $f(x)$ in the population Ψ.
(2) Calculate the Roulette wheel ratio $W_i(x)$.
Let $Wheel\_\min = f_{\min}(\Psi) - 1$

$$W_i(x) = \frac{f_i(x) - Wheel\_\min}{\sum_{i=1}^{N} (f_i(x) - Wheel\_\min)} \tag{9}$$

where $f_{\min}(\Psi)$ is the fitness value of the worst solution in the current population Ψ and N is the number of chromosomes of the current population Ψ.
(3) Sort the chromosomes of the current population according to the Roulette wheel ratio in ascending order, and calculate the selection probability $P_i(x) = \sum_{l=1}^{i} W_l(x)$.
(4) New chromosome selection
Select a random number r_i, $0 < r_i < 1$, $i = 1, 2, \ldots, N_{sel}$ where N_{sel} is the number of selected chromosomes.
If $r_i < P_1(x)$, select N_1.
If $P_i(x) \leqq r_i < P_{i+1}(x)$, select N_{i+1}.
Repeat the procedure until N_{sel} chromosomes are formed.

Various crossover and mutation methods have been developed for the many different encodings. In FSP problems, Ishibuchi et al. [10,13] used two genetic operators, two-point crossover and shift mutation, which are good operators for a two-objective GA for solving the FSP. However, the GA in this study is improved by adding two genetic operators,

Taguchi based-crossover and shift mutation.

(V) Taguchi-based crossover operation

The GA includes a crossover operator to find a better solution. From the selected population, two individuals are randomly selected as parents that can generate better offspring (better solutions) by exchanging information. Here, two-point crossover method is typically used [10,13]. Using Taguchi method [27] to find a better sequence generates non-feasible solutions that must be repaired. Taguchi-based crossover method has proven effective for optimizing particles in job shop scheduling problems without scheduling conflicts [28]. To solve the FSP, this study used a Taguchi-based crossover method without scheduling conflicts to find the optimal or near-optimal solutions for the particle after performing the experiments. Because it avoids the scheduling conflict problem in flow-shop scheduling, Taguchi-based crossover is applied.

(1) Taguchi method

The fundamental concept of the Taguchi method is to perform the minimum number of experiments needed to minimize the causes of variation and improve product quality. Two major tools in this method are orthogonal array (OA) and SNR . In an experiment involving k factors (variables) and q levels for each factor, q^k combinations generally cannot be tested if the k factors have many values. Therefore, the Taguchi method uses OA to minimize the number of experiments needed for comprehensive testing. The example considered in this study included 10 factors and two levels, and 2^{10} combinations of experiments were performed. The OA $L_{12}(2^{11})$ requires only twelve experiments to solve the problem. In each experiment, SNR is calculated, and the following steps are performed to calculate the effects of various factors. The optimal level is the one with highest effect E_{fl} for each factor. Therefore, an optimal solution with an optimal level usually approaches the objective value that generates the smallest variance. The details regarding the Taguchi method can be found in [29,30].

(2) The detailed steps of the Taguchi-based crossover algorithm are as follows:

(2-1) Perform $k+1$ experiments to select a suitable two-level OA $L_{k+1}(2^k)$. The n jobs are allocated in the first n columns of the OA.

(2-2) From the selected population, randomly select two individuals as parents P1 and P2.

(2-3) Randomly generate a set of n jobs $U = \{1, 2, 3, \ldots, n\}$, and separate U into U_1 and U_2. The job number in U_1 and U_2 correspond to factor level 1 or 2 in the executed experiment, respectively, where $U_1 \cup U_2 = U$ and $U_1 \cap U_2 = \varphi$.

(2-4) According to the job numbers in U_1 and U_2, sequentially select the new P1 and the new P2 from P1 and P2, respectively.

(2-5) Generate an offspring based on the experimental results.

(2-6) After all experiments are completed, calculate the fitness value and the SNR η for each experiment.
A fuzzy inference system is used to adjust dynamic weights w_1 and w_2 for the fitness function. In the case of a smaller-the-better characteristic, the SNR η is defined as $\eta = -10 \log \left(\frac{1}{k} \sum_{i=1}^{k} y_i^2 \right)$, where y_i is the fitness value of the experiment. In this case, k equals 1.

(2-7) Calculate the effects E_{f1} and E_{f2} for various factors. If $E_{f1} \geqq E_{f2}$, level 1 is optimal for factor f. Generate the best level for each factor where $E_{fl} =$ sum of η_l for factor f at level l.

(2-8) Generate an optimal individual (offspring).
Repeat steps (2-3) to (2-5) to generate the optimal individual.

TABLE 3. Orthogonal array $L_{12}(2^{11})$

Experiment No (i)	Factors										
	A	B	C	D	E	F	G	H	I	J	K
	Column number										
	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	2	2	2	2	2	2
3	1	1	2	2	2	1	1	1	2	2	2
4	1	2	1	2	2	1	2	2	1	1	2
5	1	2	2	1	2	2	2	2	1	2	1
6	1	2	2	2	1	2	1	1	2	1	1
7	2	1	2	2	1	1	2	2	1	2	1
8	2	1	2	1	2	2	1	1	1	1	2
9	2	1	1	2	2	2	1	2	2	1	1
10	2	2	2	1	1	1	1	2	2	1	2
11	2	2	1	2	1	2	2	1	1	2	2
12	2	2	1	1	2	1	2	1	2	2	1

TABLE 4. Selected parents

Factors	A	B	C	D	E	F	G	H	I	J
P1 (L1)	1	10	4	8	7	3	6	9	2	5
P2 (L2)	8	7	4	9	2	10	5	6	3	1

(2-9) Repeat steps (2-2) to (2-8) until the expected number of the population have been met. Finally, generate the population of optimal individuals.
For example, for a job number $n = 10$.

(1) Select a suitable two-level OA $L_{12}(2^{11})$ with 12 experiments. Use the first ten columns for the ten jobs (Table 3).
(2) Select two parents (P1 and P2) as shown in Table 4.
(3) Randomly generate a set of ten jobs
$U = \{4, 2, 1, 7, 9, 6, 5, 8, 3, 10\}$.
If experiment no. 3 is executed, the factor levels are (1 1 2 2 2 1 1 1 2 2).
For level 1, $U_1 = \{4, 2, 6, 5, 8\}$.
For level 2, $U_2 = \{1, 7, 9, 3, 10\}$.
(4) Select a new P1 and a new P2.
P1 $= (1, 10, 4, 8, 7, 3, 6, 9, 2, 5) \rightarrow$ new P1 $= (4, 8, 6, 2, 5)$
P2 $= (8, 7, 4, 9, 2, 10, 5, 6, 3, 1) \rightarrow$ new P2 $= (7, 9, 10, 3, 1)$
(5) According to the factor levels (1 1 2 2 2 1 1 1 2 2), execute crossover and generate offspring S as shown in Table 5.
S $= (4, 8, 7, 9, 10, 6, 2, 5, 3, 1)$
(6) After completing all experiments, calculate the fitness values and SNR η for each experiment as shown in Table 5.
(7) Calculate the effects E_{f1} and E_{f2} for the various factors, and find the best level of each factor as shown in Table 5.
(8) Generate an optimal chromosome (offspring) as shown in Table 5.
Repeat steps (3), (4) and (5) to find an optimal chromosome as follows.

(3) Randomly generate a set of ten jobs
$U = \{4, 2, 1, 7, 9, 6, 5, 8, 3, 10\}$.
Execute the optimal level (2 2 1 2 1 1 1 2 2 1).
For level 1, $U_1 = \{1, 9, 6, 5, 10\}$.
For level 2, $U_2 = \{4, 2, 7, 8, 3\}$.
(4) P1 = $(1, 10, 4, 8, 7, 3, 6, 9, 2, 5) \rightarrow$ new P1 = $(1, 10, 6, 9, 5)$
P2 = $(8, 7, 4, 9, 2, 10, 5, 6, 3, 1) \rightarrow$ new P2 = $(8, 7, 4, 2, 3)$
(5) According to the optimal level (2 2 1 2 1 1 1 2 2 1), execute crossover and generate offspring S (see Table 5).
S = $(8, 7, 1, 4, 10, 6, 9, 2, 3, 5)$

(VI) Mutation operation

After a few generations, the individuals in the population are very close. Local optimization can be avoided by performing a mutation operation to change only a few genes. The procedure used in this study was shift change operator (Figure 6) [10], which is performed as follows.

(1) Randomly select one parent from an optimal chromosome matrix.
(2) Randomly select two points along the job sequence.
(3) Move a job from one position to another position.

(VII) Local search

The population is diversified by using the new LS population, which consists of the population after mutation N_{mu} and three non-dominated solutions N_{elite} randomly selected from the previous set of Pareto optimal solutions. The purpose of LS method is to

TABLE 5. Generating improved offspring from two parents by using Taguchi-based method

Exper-iment No (i)	Factors										Fitness values y_i	S/N ratio η
	A	B	C	D	E	F	G	H	I	J		
	Job number											
	1	2	3	4	5	6	7	8	9	10		
1	1	10	4	8	7	3	6	9	2	5	–565.2968	–55.0455
2	1	4	7	9	2	8	10	5	6	3	–624.0732	–55.9047
3	4	8	7	9	10	6	2	5	3	1	–517.1973	–54.2731
4	1	8	10	7	9	4	2	5	3	6	–454.0288	–53.1417
5	4	8	9	7	2	10	5	6	3	1	–674.2050	–56.5758
6	10	7	2	6	4	3	8	9	1	5	–523.9099	–54.3851
7	8	3	7	4	6	9	10	5	2	1	–724.4954	–57.2007
8	4	10	9	8	6	1	7	3	2	5	–708.6616	–57.0088
9	8	1	10	7	4	9	2	6	3	5	–408.5810	–52.2256
10	8	10	5	1	4	7	9	6	3	2	–435.3097	–52.7760
11	7	4	1	2	8	10	5	3	9	6	–499.7075	–53.9743
12	7	4	1	8	2	3	10	9	5	6	–637.6585	–56.0918
E_{f1}	–329.3260	–331.6584	–326.3836	–329.9684	–329.2864	–328.5288	–325.7141	–330.7786	–332.9468	–324.5826		
E_{f2}	–329.2771	–326.9447	–332.2196	–325.2005	–329.3167	–330.0743	–332.8890	–327.8245	–325.6563	–334.0205		
Optimal level	2	2	1	2	1	1	1	2	2	1		
Optimal chromosome	8	7	1	4	10	6	9	2	3	5		

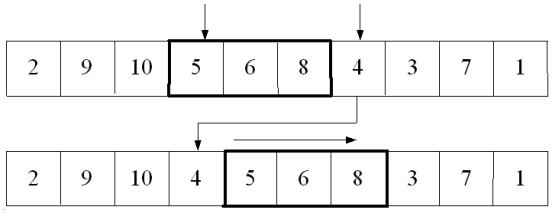

FIGURE 6. Shift change operator

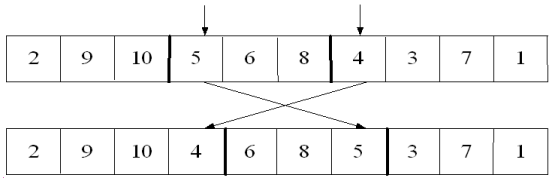

FIGURE 7. Swap change operator

improve the solutions obtained by the crossover and mutation operation. Two operators used for the LS are the shift change operator and the swap change operator.

The algorithm uses the swap change operator (Figure 7) to change the jobs at positions i and j.

For the swap change operator [31] of a chromosome consisting of n jobs, the number of full neighborhood solutions is $n(n-1)/2$. For example, when $n = 10$, the number of neighborhood solutions that must be examined for a chromosome is 45, which is a small problem size. Although the full neighborhood solutions are used in this study, computing time may be overly long when the job number is high. Therefore, computing speed is increased by using a single adjacent pairwise interchange neighborhood composed of the following $n-1$ schedules for $(J_i, J_2, J_3, J_4, \ldots, J_{n-1}, J_n)$:

$$(J_2, J_1, J_3, J_4, \ldots, J_{n-1}, J_n),$$
$$(J_i, J_3, J_2, J_4, \ldots, J_{n-1}, J_n),$$
$$(J_i, J_2, J_4, J_3, \ldots, J_{n-1}, J_n),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$(J_i, J_2, J_3, J_4, \ldots, J_n, J_{n-1}).$$

For example, when $n = 10$, only nine adjacent pairwise interchange neighborhood solutions are considered.

(VIII) Elitist preservation strategy

A new population with N_{ls} chromosomes generates after local search. The elitist preservation strategy always keeps the best chromosomes of a population in the next generation. Therefore, the following procedure is used to select the best N_{ini} solutions after local search for next generation:

(1) Calculate the values of the objective function for N_{ls} chromosomes.
(2) Calculate the fitness function for N_{ls} chromosomes.
To enlarge the searching space, randomly selected dynamic weights are used for the fitness function.
(3) Sort all chromosomes in the population by fitness function values, and delete those with the lowest fitness function values; obtain a new population with N_{ini} chromosomes for the next generation.

(IX) Update the set of Pareto optimal solutions

(1) Generate a population consisting of the new population for the next generation and the set of Pareto optimal solutions for the last generation.
(2) Use the population to update the set of Pareto optimal solutions.

(X) Termination

Use the number of generations as the termination condition. If the current generation is equal to the number of maximum generations, then stop; otherwise, go to step IV.

(XI) Display the final Pareto optimal front

The HTGA gives decision makers the best solutions for the final Pareto optimal front. The decision makers can then select any one of the solutions based on their preferences.

TABLE 6. Hybrid Taguchi-based genetic algorithm parameter settings

Parameter	Description	Setting	
		Test problem 1	Test problem 2
n	Job number	10	40
m	Machine number	5	20
q	Population size	20	20
	Maximum generation	3000, 6000, 9000, 12000.	500
ω	Dynamic weight selection by FIS		
c_r	Crossover rate	0.9	0.9
m_r	Mutation rate	0.3	0.3
N_{sel}	Number of selected chromosomes	17	17
N_{elite}	Number of elite solutions	3	3
Stopping condition	Maximum generation	Maximum generation	Maximum generation

TABLE 7. Processing time for each job on each machine

Job No.	Machine number					Due date
	1	2	3	4	5	
1	32	21	10	51	33	674
2	1	27	42	19	45	396
3	61	87	66	23	58	431
4	42	45	75	85	97	369
5	62	59	41	86	91	626
6	61	24	24	81	85	597
7	3	71	3	93	30	790
8	97	34	36	31	38	437
9	26	20	85	75	17	656
10	9	28	74	23	51	780

TABLE 8. Global optimal solutions obtained by hybrid Taguchi-based genetic algorithm

No. of jobs	1	2	3	4	5	6	7	8	9	10	11
$C_{\max}$	681	691	702	710	713	716	733	749	752	753	782
$T_{\max}$	197	136	123	120	106	90	75	70	53	21	4

4. **Numerical Example and Simulation Results.** This section evaluates the performance of the proposed HTGA approach by comparing its optimization results with those obtained for the same cases using the MOGLS in [10] and the modified MOGLS in [13]. Table 6 shows the HTGA parameter settings. The same crossover rates and mutation rates are used in test problems 1 and 2 for comparisons between the MOGLS in [10] and the modified MOGLS in [13].

(I) Test problem 1

The same example given in [10] (see Table 7) is used to test the proposed HTGA approach and to compare its performance with the GA approach in [10]. Table 8 shows the global optimal solutions (GOS) obtained by the HTGA. Figure 8 shows the Pareto front of the HTGA. Figures 9 and 10 show the number of global optimal solutions (GOS) for four generations (3000, 6000, 9000, 12000) in 30 runs of the HTGA with weights selected randomly and with weights selected by FIS method, respectively.

The simulation results reveal two findings.

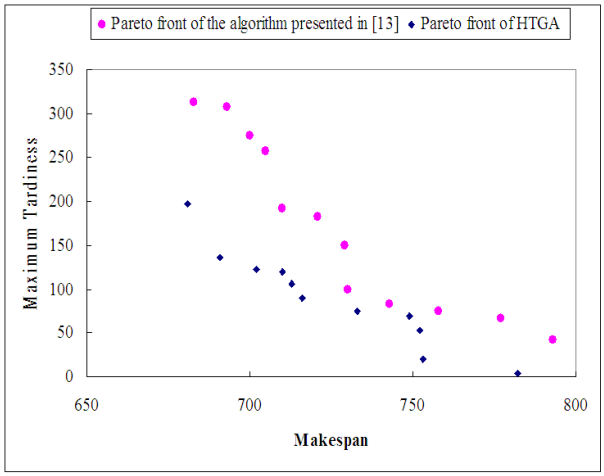

FIGURE 8. Pareto fronts of hybrid Taguchi-based genetic algorithm (HTGA) and the algorithm presented in [10] for ten-job and five-machine flowshop scheduling problem

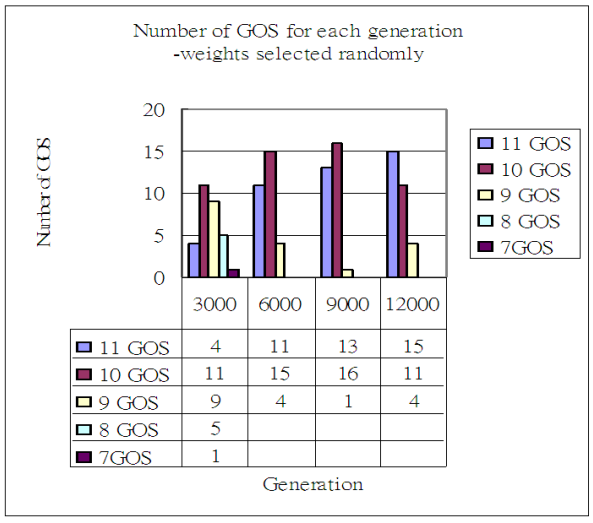

FIGURE 9. Number of globally optimal solutions (GOS) generated by weights selected randomly

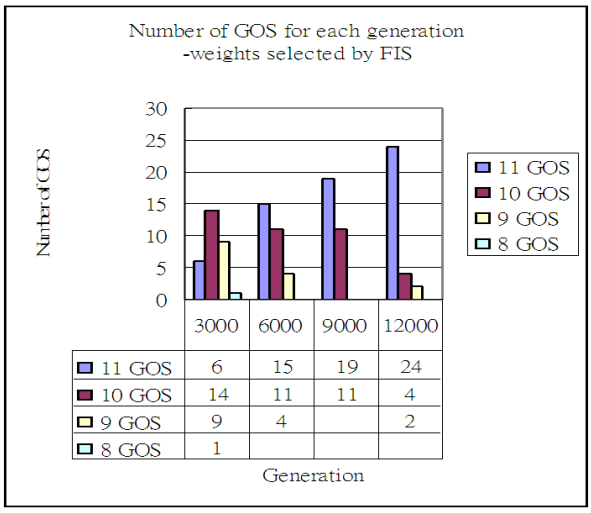

FIGURE 10. Number of globally optimal solutions (GOS) generated by weights selected by fuzzy inference system (FIS) method

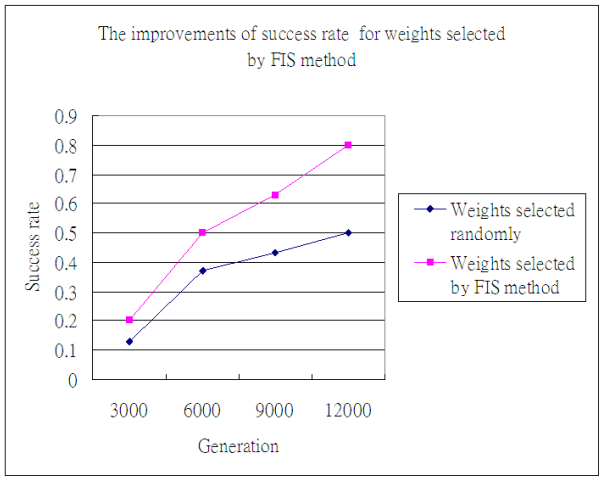

FIGURE 11. Improvement in the success rate of the hybrid Taguchi-based genetic algorithm

(1) Comparison of HTGA with the algorithm developed by Ishibuchi and Murata [10].

The algorithm developed by Ishibuchi and Murata [10] used randomly selected dynamic weights and conventional two-point crossover. Figure 8 shows that the Pareto front obtained by the HTGA is better than that obtained by the algorithm presented in [10].

(2) Selection of improved dynamic weights using FIS method.

Figures 9 and 10 show that the numbers of runs with 11 GOS increase when the maximum number of generations increases. Figure 11 shows the success rates (number of runs with 11 GOS/total number of runs) for four generations of HTGA with weights selected randomly and weights selected by FIS method. Figure 11 confirms that the success rates of the dynamic weights selected by FIS method are better than those of the dynamic weights selected randomly. Therefore, we can conclude that the robustness of the HTGA approach with weights selected by FIS increases as the maximum number of generations increases. Therefore, the fuzzy inference system substantially improves the HTGA solutions.

(II) Test problem 2

The same example considered in [13] is used for random generation of a 40-job, 20-machine flowshop scheduling problem as described in section IIIA of [10]. The population size (q) is equal to 20. Taguchi based crossover is performed by using orthogonal array $L_{64}(2^{63})$. The first 40 columns are used for 40 jobs. The full neighborhood solutions and 500 generations are used.

Compared with the approach used by Ishibuchi et al. [13], (see Figure 4 in [13]), the HTGA algorithm achieves better solutions for the same problem (Figure 12).

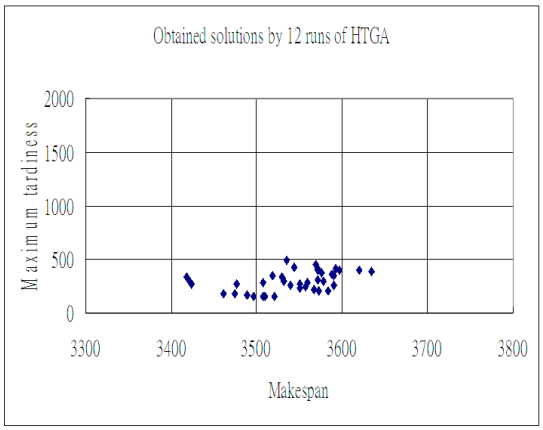

FIGURE 12. Non-dominated solutions obtained in 12 runs of hybrid Taguchi-based genetic algorithm (HTGA) for 40-job and 20-machine flowshop scheduling problem

TABLE 9. Simulation results for ten-job and five-machine flowshop scheduling problem in test problem 1 with numbers of globally optimum solutions (GOS) and numbers of runs for crossover rates (Cr) of 0.7, 0.8 and 0.9

Maximum generation	Cr = 0.7		Cr = 0.8		Cr = 0.9	
	No. of GOS	No. of runs	No. of GOS	No. of runs	No. of GOS	No. of runs
3000	11 10 9 8	4 12 13 1	11 10 9 8	6 14 7 3	11 10 9 8	6 14 9 1
6000	11 10 9 8	12 12 4 2	11 10 9	14 14 2	11 10 9	15 11 4
9000	11 10 9 8	14 9 4 3	11 10 9	18 10 2	11 10	19 11
12000	11 10 9	16 11 3	11 10	20 10	11 10 9	24 4 2

TABLE 10. Computation results using different crossover rates (0.7, 0.8, 0.9) for ten-job and five-machine flowshop scheduling problem in test problem 1

Maximum generation	Cr = 0.7			Cr = 0.8			Cr = 0.9		
	Best	Avg.	Std. dev.	Best	Avg.	Std. dev.	Best	Avg.	Std. dev.
3000	11	9.633	0.752	11	9.766	0.882	11	9.833	0.778
6000	11	10.133	0.884	11	10.4	0.611	11	10.366	0.706
9000	11	10.133	0.991	11	10.533	0.618	11	10.633	0.481
12000	11	10.433	0.667	11	10.666	0.471	11	10.733	0.573

The proposed HTGA consistently obtained robust solutions under varying parameters. For example, we use different crossover rates (0.7, 0.8, 0.9) in test problem 1, the result is still robust. Tables 9 and 10 show the simulation results and the computation results. Table 10 shows that the HTGA obtains much smaller standard deviations for different crossover rate in each maximum generation. Hence, the proposed HTGA obtains better quality solutions in terms of robustness and stability.

5. **Conclusions.** This study presented a hybrid genetic algorithm that uses Taguchi-based crossover to solve global optimal solutions for the FSP. Weighted sum method is used to find the fitness of each chromosome. This HTGA combines GA with a local search and elitist preservation strategy. The numerical examples in this study confirm that the HTGA has the following performance characteristics:

(I) Whereas randomly selected weights are generally used for the fitness function in each calculation, this study improved search performance by using FIS instead. Figure 11 shows that weight selection by FIS method is more effective than random weight selection.

(II) The proposed Taguchi-based crossover avoids the scheduling conflict problems that occur in conventional Taguchi crossover and easily finds global optimal solutions by using two-level orthogonal arrays and SNR. Therefore, the Taguchi-based crossover enhances the GA. The simulation results in Figure 9 and Figure 10 confirm that the proposed HTGA obtains better and more robust solutions compared with the conventional approach. Figure 8 shows that the Pareto front obtained by the HTGA is better than that obtained by the algorithm presented in [10].

(III) In this study, the set of Pareto optimum solutions is updated by using the population for the current generation and the Pareto optimum solutions for the last generation. The Pareto optimum solutions are then inherited by the next generation, and the global optimal solutions are obtained.

Computer simulations showed that the merits of the proposed HTGA include its simplicity, its global exploration capability, its fast convergence and its robustness. The illustrative examples showed that the proposed HTGA approach effectively solves the FSP and outperforms the MOGLS algorithm by Ishibuchi and Murata [10] and the modified MOGLS algorithm by Ishibuchi et al. [13].

Acknowledgment. This work was in part supported by the National Science Council, Taiwan, under grant number NSC 99-2221-E-151-071-MY3.

REFERENCES

[1] C. L. Chen, V. S. Vempait and N. Aljaber, An application of genetic algorithms for flow shop problems, *European Journal of Operational Research*, vol.80, pp.389-396, 1995.

[2] T. Murata, H. Ishibuchi and H. Tanaka, Genetic algorithms for flowshop scheduling problems, *Computers Ind. Engng.*, vol.30, no.4, pp.1061-1071, 1996.
[3] B. W. Cheng and C. L. Chang, A study on flowshop scheduling problem combing Taguchi experimental design and genetic algorithm, *Expert Systems with Application*, vol.32, pp.415-421, 2007.
[4] D. R. Sule, *Industrial Scheduling*, PWS Publishing Company, Boston, 1996.
[5] C. Wu and X. Gu, A genetic algorithm for flowshop scheduling with fuzzy processing time and due date, *Proc. of the 5th World Congress on Intelligent Control and Automation*, Hangzhou, China, pp.2938-2942, 2004.
[6] W. H. Ho, J. X. Chen, I. N. Lee and H. C. Su, An ANFIS-based model for predicting adequacy of vancomycin regimen using improved genetic algorithm, *Expert Systems with Applications*, vol.38, pp.13050-13056, 2011.
[7] W. H. Ho, Takagi-Sugeno fuzzy model of nonlinear HIV dynamics: Chebyshev-series approach integrated with genetic algorithm, *International Journal of Innovative Computing, Information and Control*, vol.8, no.2, pp.1439-1451, 2012.
[8] F. S. Ismail, R. Yusof, M. Khalid, Z. Ibrahim and H. Selamat, Performance evaluation of self organizing genetic algorithm for multi-objective optimization problem, *ICIC Express Letters*, vol.6, no.1, pp.1-7, 2012.
[9] F. N. Ferdinand, K. H. Chung, H. J. Ko and C. S. Ko, Genetic algorithm-based approach to multi-objective decision making model for strategic alliances in express courier services, *ICIC Express Letters*, vol.6, no.4, pp.929-934, 2012.
[10] H. Ishibuchi and T. Murata, A multi-objective genetic local search algorithm and its application to flowshop scheduling, *IEEE Trans. on System, Man, and Cybernetics – Part C: Applications and Reviews*, vol.28, pp.392-403, 1998.
[11] H. Ishibuchi and T. Murata, Multi-objective genetic local search algorithm, *Proc. of the 3rd IEEE Int. Conf. Evolutionary Computation*, Nagova, Japan, pp.119-124, 1996.
[12] H. Ishibuchi, T. Yoshida and T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, *IEEE Trans. Evol. Comput.*, vol.7, pp.204-223, 2003.
[13] H. Ishibuchi, T. Yoshida and T. Murata, Selection of initial solutions for local search in multiobjective genetic local search, *Proc. of Congr. Evol. Comput.*, Honolulu, HI, USA, pp.950-955, 2002.
[14] T. Murata, H. Ishibuchi and H. Tanaka, Multi-objective genetic algorithm and its application to flowshop scheduling, *Computers Ind. Engng.*, vol.30, pp.957-968, 1996.
[15] T. Murata and H. Ishibuchi, MOGA: Multi-objective genetic algorithms, *Proc. of IEEE International Conference on Evolutionary Computation*, Perth, Australia, pp.289-294, 1995.
[16] V. R. Neppalli, C. L. Chen and J. N. D. Gupa, Genetic algorithm for two-stage bi-criteria flowshop problem, *European Journal of Operational Research*, vol.95, pp.356-373, 1996.
[17] P. C. Chang, J. C. Hsieh and S. G. Lin, The development of gradual-priority weighting approach for multiobjective flowshop scheduling problem, *Int. J. Prod. Econ.*, vol.79, pp.171-183, 2002.
[18] F. Dugardin, F. Yalaoui and L. Amodeo, New multi-objective method to solve reentrant hybrid flow shop scheduling problem, *European Journal of Operational Research*, vol.203, pp.22-31, 2010.
[19] N. Karimi, M. Zandieh and H. R. Karamooz, Bi-objective group scheduling in hybrid flexible flowshop: A multi-phase approach, *Expert Syst. Appl.*, vol.37, pp.4024-4032, 2010.
[20] J. E. C. Arroyo and V. A. Armentano, Genetic local search for multiobjective flowshop scheduling problem, *European Journal of Operational Research*, vol.167, pp.717-738, 2005.
[21] N. Melab, M. Mezmaz and E. G. Talbi, Parallel cooperative meta-heuristics on the computational grid, A case study: The bi-objective flow-shop problem, *Parallel Computing*, vol.32, pp.643-659, 2006.
[22] B. B. Li and L. Wang, A hybrid quantum-inspired genetic algorithm for multi-objective flow shop scheduling, *IEEE Trans Syst Man Cybern B*, vol.37, pp.576-591, 2007.
[23] P. C. Chang, S. H. Chen and C. H. Liu, Sub-population genetic algorithm with mining gene structures for multi-objective flowshop scheduling problems, *Expert Syst. Appl.*, vol.33, pp.762-771, 2007.
[24] Y. F. Tseng, H. M. Huang and Y. H. Zeng, CNC milling process optimization using fuzzy-Taguchi with principle component analysis approach, *Proc. of the 13th National Conference on Fuzzy Theory and Its Applications*, Taiwan, pp.899-906, 2005.
[25] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control, and Information*, Prentice Hall, New Jersey, 1998.
[26] D. L. Hung and W. F. Zajak, Design and implementation of a hardware fuzzy inference system, *Information Science*, vol.3, pp.193-207, 1995.

[27] L. Y. Tseng and Y. T. Lin, A hybrid genetic local search algorithm for the permutation flowshop scheduling problem, *European Journal of Operational Research*, vol.198, pp.84-92, 2009.
[28] J. T. Tsai, T. K. Liu, W. H. Ho and J. H. Chou, An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover, *Int. J. Adv. Manuf. Technol.*, vol.38, pp.987-994, 2008.
[29] D. C. Montgomery, *Design and Analysis of Experiments*, Wiley, New York, 1991.
[30] M. S. Phadke, *Quality Engineering Using Robust Design*, McGraw-Hill, New York, 1989.
[31] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood Limited, John Wiley&Son, New York, 1981.