Int J Adv Manuf Technol (2002) 20:464–473
Ownership and Copyright

The International Journal of
Advanced Manufacturing Technology

A Memetic Algorithm for the $n/2$/Flowshop/$\alpha F + \beta C_{max}$ Scheduling Problem

W.-C. Yeh

Department of Industrial Engineering, e-Integration & Collaboration Laboratory, Feng Chia University, Taichung, Taiwan

Machine scheduling has been a popular area of research during the past four decades. Its object is to determine the sequence for processing jobs on a given set of machines. The need for scheduling arises from the limited resources available to the decision-maker. In this study, a special situation involving a computationally difficult n/2/Flowshop/αF + βC_{max} *flowshop scheduling problem is discussed. We develop a memetic algorithm (MA, a hybrid genetic algorithm) by combining a genetic algorithm and the greedy heuristic using the pairwise exchange method and the insert method, to solve the* n/2/Flowshop/$\alpha F + \beta C_{max}$ *flowshop scheduling problem. Preliminary computational experiments demonstrate the efficiency and performance of the proposed memetic algorithm. Our results compare favourably with the best-known branch-and-bound algorithm, the traditional genetic algorithm and the best-known heuristic algorithm.*

Keywords: Branch-and-bound algorithm; Flowshop scheduling problem; Genetic algorithm; Heuristic algorithm; Local search method; Memetic algorithm

1. Introduction

Makespan and total flow-time are two commonly used performance measures in flowshop scheduling literature [1–11]. Flow-time is defined as the time spent by each job in the system and makespan is the time at which the last job completes its processing on the last machine. Minimising makespan is important in situations where a simultaneously received batch of jobs is required to be completed as soon as possible, for example, a multi-item order submitted by a single customer that must be delivered in the minimal possible time. The makespan criterion also increases the use of resources. There are other real-life situations in which each completed job is required as soon as it is processed. In such situations, we are interested in minimising the mean or sum of flowtimes of all jobs rather than minimising makespan. This objective is particularly important in real-life situations in which reducing inventory or maintaining cost is of primary concern.

Nagar et al. [4, 12] were the first to address the two-machine flowshop problem using the weighted sum of makespan and flow-time criteria. They presented a branch-and-bound algorithm that works well for special cases. Yeh [13] developed additional branch-and-bound algorithms for the same problem. This scheduling problem can be represented as n/m = 2/Flowshop/$\alpha F + \beta C_{max}$, where $0 \leq \alpha, \beta \leq 1$ and $\alpha + \beta = 1$. The four variables are: the number of jobs n, number of machines m, the shop configuration, and the two criteria to be optimised, namely, flow-time F and makespan C_{max}. "Flowshop" is a shop configuration in which machines are arranged serially with a unidirectional work flow. Yeh [14] improved the branch-and-bound algorithm developed by Yeh [13] even further.

The objective of this work is to minimise a weighted linear combination of job flow-time and schedule makespan. When $\alpha = 0$, this problem is a two-machine problem, in which the objective is to minimise the makespan. It is well known that such situation can be solved using Johnson's algorithm [8, 9]. However, if $0 < \alpha \leq 1$, n/m = 2/Flowshop/$\alpha F + \beta C_{max}$ is NP-Hard [10, 11]. For further details on the complexity of general scheduling problems see Lawler et al. [11].

The problem considered in this study finds applications in both industrial and planning areas where each job must undergo two basic processes in the same sequence. For example, in a flat glass production problem, a cutting stock machine takes glass and cuts it according to patterns to produce shapes that are processed at a second operation. The second machine may perform any activity such as packaging, assembling, and finishing [12–15].

Genetic algorithms (GAs) are stochastic search methods for optimisation problems based on the mechanism of natural selection and genetics, which is based on the survival-of-the-fittest tenet of Darwinian evolution. Recently, GAs have been applied to harder combinatorial optimisation problems because they have better characteristics, e.g. there is less effect on calculation when the system becomes more complex or larger. However, the weakness of a GA for local searches is well acknowledged [16–22].

Correspondence and offprint requests to: Dr W.-C. Yeh, Department of Industrial Engineering, Feng Chia University, PO Box 67-100, Taichung, Taiwan 407. E-mail: yeh@ieee.org

Moscato and Norman [20] introduced the term "memetic algorithm" (MA) to describe genetic algorithms in which a local search plays a significant part. In MAs, a local optimiser is applied to each offspring before it is inserted into the population in order to make it climb the local optimum [21, 23]. With the hybrid method [16–23], GAs are used to perform global exploration within a population, while local optimisers are used to perform local exploitation around chromosomes. Since the properties of GAs and conventional local optimisers are complementary, MAs are often better than either method operating alone [16–23].

The purpose of this paper is to present an efficient memetic algorithm (MA) by combining the GA, hybrid local search method (HLSM), and greedy heuristic method (GHM) to solve the scheduling problem proposed by Nagar et al. [12]. To indicate the performance and efficiency of the proposed MA, a traditional GA (TGA) was also developed, and compared to the proposed MA. Our results compare favourably with the existing best-known branch-and-bound algorithm (BB) [14] and the best-known heuristic algorithm (the two-phase hybrid heuristic algorithm, 2XI) [14], and TGA.

This paper is organised as follows. The terminology and problem formulation are described in Section 2. In Section 3, a simple TGA for the $n/2/\text{Flowshop}/\alpha F + \beta C_{\max}$ scheduling problem is given. In Section 4, an efficient MA is proposed. The 2XI is listed in Section 5. Computational experiments in a series of randomly generated problems are provided in Section 6. Concluding remarks are given in Section 7.

2. The Notation and Assumption

In the remainder of this study, the following notation is used:

J_i, $J_{[i]}$ = job i and the job which occupies the position i in the sequence of jobs, respectively; $i = 1, 2, \ldots, n$

$t_{i,j}$, $t_{[i],j}$ = processing time of the J_i and $J_{[i]}$ on the machine j, respectively; $i = 1, 2, \ldots, n$, $j = 1, 2$

n = job number

α, β = weights associated with flow-time and makespan ($0 \leq \alpha, \beta \leq 1$ and $\alpha + \beta = 1$), respectively

S = any arbitrary sequence of n jobs

S_i = a subschedule of the first i jobs in S, $0 \leq i \leq n$ and $S_0 = \varnothing$

$C(S)$, $F(S)$ = completion time and flow-time of S, respectively

$I_k(S)$ = idle-time induced from the kth job in S, where $1 \leq k \leq n$

C_{ij} = jth chromosome of the ith generation

$Fit(C_{ij})$ = fitness value of the chromosome C_{ij}

pop = population size

gen = maximum number of generations allowed

$cgen$ = generation number at which MA began to converge

Avg. = Time: average running times (of all different test data sets in the group problem)

Avg. = Value: average of the final schedule objective function values of MA (of all different test data sets in the group problem)

Avg. = Error: average error between the corresponding results and optimum

Avg. = Difference: average error between the corresponding results and final schedule objective function values of MA

The problem considered here is to schedule n jobs on two machines and determine the optimal weighted combination of flow-time and schedule makespan, so that the objective function [12–14]

$$Z(s) = \alpha F(S) + \beta C(S) = \alpha \sum_{i=1}^{n} (n-i+1)\{t_{[i],2} + I_i(S)\} + \beta \sum_{i=1}^{n} \{t_{[i],2} + I(S)\} \quad (1)$$

is minimised. The following assumptions are used to characterise a flowshop [12–14]:

1. All jobs are independent and available for processing simultaneously at time zero.
2. Set-up times are known and are included in the processing times.
3. Machines are continuously available but cannot process two or more jobs simultaneously.
4. Job pre-emption and job splitting are not permitted.
5. There is an infinite buffer between the machines.

3. A Traditional Genetic Algorithm

To show the difference between the traditional GA and the proposed MA, this section outlines a simple TGA for solving the $n/2/\text{Flowshop}/\alpha F + \beta C_{\max}$ problem.

3.1 Initialisation

The initial population can be created in either a random way or a well-adapted method. Liepins and Hilliard [22] suggested that the use of a well-adapted population provides few advantages despite fast convergence. Therefore, all of the initial population is generated randomly. A fixed population size is used in this study.

3.2 Chromosome Representation

As the initialisation process of a GA, a solution is encoded as a finite-length string of elements called chromosomes. Here, the permutation encoding is used because the order of items can be most naturally modelled in this way. In permutation encoding, every chromosome is a feasible schedule and is described by a vector: $C_{ij} = [J_{ij1}, \ldots, J_{ijn}]$, where $J_{ijk} \neq J_{ijl}$ for all $k \neq l$, and J_{ijk} is the kth job in C_{ij}. The length of the chromosome is equal to the total number of jobs to be scheduled.

3.3 Fitness Function

Chromosomes are selected to form new solutions (offspring) according to their fitness function value – the more suitable they are, the more chances they have to reproduce. In this study, the fitness function value for each chromosome is equal to the schedule objective function, i.e. $Fit(C_{ij}) = Z(C_{ij})$.

3.4 Selection

The selection process discussed here is a mixed method of elite and rank selection. The elite method selects a few best, say n_{sel} which is 80% of the chromosomes of the parents in this study, for the next generation to prevent losing the best found solution and to rapidly increase the performance of the GA. The rank method is based on all of the chromosomes having a chance to be selected. It is used here to select the other $pop - n_{sel}$ chromosomes.

3.5 Crossover

Crossover and mutation are the most important parts of a GA. The crossover plays an important role in exchanging information among chromosomes. It leads to an effective combination of partial solutions in other chromosomes and speeds up the search procedure early in the generation. The single-point crossover is developed here to produce two offspring for each pair of parents. In this method, one cut point is randomly chosen first. The two offspring are produced in the following way: for every job before (after) this point, copy from the first parent, and then copy the other distinct jobs of the second parent. The crossover is applied with a probability of 0.8 per chromosome in this study.

3.6 Mutation

To prevent all solutions in population from falling into a local optimum; mutation takes place after a crossover is performed. The swapping mutation is applied here by choosing randomly two jobs within the selected chromosome, and exchanging their positions. The mutation is applied with the relatively high probability of 0.2 per chromosome.

4. A Memetic Algorithm

To overcome the weakness of the GA for local searches, an efficient MA is proposed here by combining the GA, HLSM and GHM to solve the scheduling problem proposed by Nagar et al. The GA part in the proposed MA is similar to the TGA proposed in Section 3. Hence, only the differences, in HLSM and GHM are discussed in this section.

4.1 A Well-Adapted Initial Population

To speed up the convergence of the MA process, all of the initial population is generated randomly first, except that the first chromosome is created by a simple greedy method, described in Section 4.2. All of the chromosomes are then improved using an HLSM that is discussed in Section 4.3.

4.2 A Simple Greedy Heuristic Method

The first chromosome is constructed using a GHM in the initial process. The GHM is adapted from the upper-bound procedure of Yeh [13,14]. It is very simple and can be used to find the upper-bound of even large-scale problems. Its underlying principle is to pull back the schedule objective function at each stage of schedule generation. In this greedy procedure, the remaining unscheduled jobs are sequenced with consideration of both the average processing time on machine 1 and the idle-time induced after sequencing.

4.3 A Hybrid Local Search Method (HLSM)

One of the major drawbacks of a GA is that it converges too slowly. HLSM is implemented to prevent this drawback and to guide the search towards unexplored regions in the solution space. The proposed HLSM combines two well-known local improvement methods: the pairwise exchange procedure (XP) and insert procedure (IP). If a new chromosome is produced by the crossover or in the initial population, then it is improved using XP, otherwise it is improved using IP.

In XP, the positions of every pair of jobs are exchanged. In IP, each job is removed from its current position and inserted into the other position. Both XP and IP in the proposed MA must be performed until the fitness function value is unchanged for three consecutive times. To reduce the number of duplications of new chromosomes after implementing HLSM, the starting position must be exchanged in XP or inserted in IP, according to a random number. Hence, two different chromosomes with the same encoding, must have different encoding after HLSM.

4.4 Overall Procedure

The flowchart for the general procedure of the proposed MA is shown in Fig. 1. The blocks with a dashed-line border are implemented and function as in the TGA discussed in Section 3.

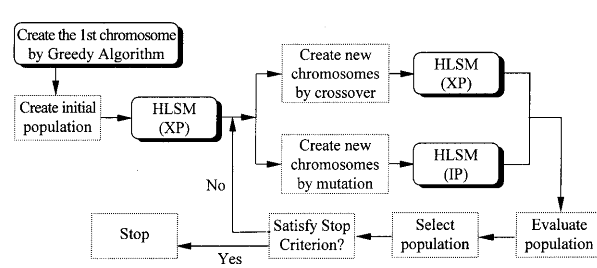

Fig. 1. The flowchart of the memetic algorithm.

5. The Best-Known Heuristic Algorithm

To demonstrate the role of GHM, XP, and IP in the proposed MA, the best-known heuristic algorithm (called 2XI here) proposed in [13,14] is presented to compute an approximate solution for the problem. In the first phase of 2XI, the greedy algorithm proposed in Section 4.2 is employed first to produce a feasible schedule. In the second phase, a hybrid method combining XP and IP, (the two local improvement methods are discussed in Section 4.3), is employed to improve the initial schedule obtained in the first phase.

In the second phase of 2XI, IP is just a subroutine of XP. It is employed only while there is no improvement after exchanging a pair of jobs during the process of XP. After that, XP is restarted. The above procedure is repeated until the objective function value cannot be improved or is unchanged for three consecutive times.

6. Computational Results

The closer the result is to the optimum, the higher the quality of the algorithm. To compare the efficiency (running time) and quality of the proposed MA against TGA, the best-known existing BB, and the best-known existing heuristic algorithm (2XI), all of the above algorithms were implemented in C^{++}, and run on a Pentium III 1033 personal computer. The running time was measured in seconds.

There were four experiments (see Table 1). Each test problem group contained 10 different data sets. The processing time on the 1st and 2nd machine for each job in every data set was randomly generated in a uniform discrete distribution $U(10, 99)$. The number of job for each data set were all equal. Very little time was required to solve the problem when the total processing time was greater on the second machine than on the first [12–14]. Hence, to create a more practical experiment, the processing times on machines 1 and 2 were exchanged and rerun again after each data set. Except for $n > 50$, experiment 3 was run with 4 sets of weights, namely $\alpha = 0.25, 0.50, \ldots, 1.0$, i.e. there were 80 test problems in each group problem. In the other experiments and for $n \leqslant 50$ in experiment 3, each data set was run with 10 sets of weights, namely $\alpha = 0.1, 0.2, \ldots, 1.0$, i.e., there are 200 test problems in each problem group.

Table 1. The four experiments.

Experiment	Remarks
1	Compare MA to TGA for $n = 10$ under different *pop* and *gen*, and $\alpha = 0.1, 0.2, \ldots, 1$;
2	Compare MA to BB and 2XI for $n = 5, 6, \ldots, 15$, $pop = n, 2n, \ldots, 5n$ and $\alpha = 0.1, 0.2, \ldots, 1$;
3	Compare MA to 2XI for $n = 10, 15, \ldots, 50$ with $pop = 2n$ and $\alpha = 0.1, 0.2, \ldots, 1$, and $n = 55, 60, \ldots, 100$ with $pop = n$ and $\alpha = 0.25, 0.5, \ldots, 1$;
4	Compare MA to BB for the 2nd machine is dominant, $n = 10, 15, \ldots, 90$, $pop = 2n$, and $\alpha = 0.1, 0.2, \ldots, 1$;

In the results tables, the notations *n*, *pop*, *gen*, *cgen*, *Avg. Time*, *Avg. Optimum*, *Avg. Value*, *Avg. Error* and *Avg. Difference* represent the job number, population size, generation size, the average generation that began to converge, the average of the running times (for all different test data sets in the problem group), the average optimum, the average of final schedule objective function value (of all different test data sets in the group problem), the average error between the corresponding results and optimum, and the average error between the corresponding results and values. To allow observation of the behaviour and convergence stability of MA and TGA, each new generation was executed again from the initial population process in experiments 1 and 2.

In experiment 1, the stop criterion for TGA and MA were the generation numbers. The stop criterion for the remaining experiments in MA was a little different from experiment 1. In order to take advantage of a better solution quality, a flexible termination criterion was used. In each test problem, the algorithm terminated after a number of generations, carried out with an equal fitness function value for all populations in the same generation.

6.1 MA vs. TGA

In experiment 1, the proposed MA was first tested on 10 group problems (200 test problems), each for 10 jobs to compare with the proposed TGA in Section 3. In this experiment, the MA was run with generations 1, 2, to 20, and the population sizes were 10, 20, to 50, respectively. The TGA was run with generations 500 and 550 to 1000, and the population sizes were 50, 100, 200 and 300, respectively.

Table 2 shows only the running time and average error for the MA before the MA converged to an optimum, i.e. from $gen = 1$ to *cgen*. Therefore, there are only at most 18 generations in Table 2. Table 3 shows the TGA results and Table 4 shows the corresponding results for the TGA and the MA in Tables 2 and 3 for each α, respectively. All of the average errors in Tables 2 to 4 were compared to the optimum obtained from the best-known BB. From Table 3, the TGA not only converged very slowly, even with 1000 generations and a population of 300, but also converged to a solution with a larger average error. In contrast, the MA converged steadily and very fast to the optimum with smaller sized generations, population and running time (see Table 2). In Table 3, the MA ran with $pop = 10$ and $gen = 18$ were even better than the TGA run with $pop = 300$ and $gen = 1000$ at the convergence speed and solution quality to each α. Therefore, the MA is more efficient than the TGA.

6.2 The Comparison Among the MA, BB and 2XI

The focus of the next experiment shifted to a comparison among MA, and the best-known BB and the best-known heuristic algorithm (2XI) for small problems (see Tables 5 and 6).

Because of the characteristics of NP-hard problems and the limitations of personal computers, the BB could solve only medium-sized problems. Therefore, only 5 to 15 jobs were

Table 2. Results of MA in experiment 1 (the average optimum is 2033.818).

gen	pop = 10, cgen = 18		pop = 20, cgen = 9		pop = 30, cgen = 5		pop = 40, cgen = 3		pop = 50, cgen = 2	
	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error
1	0.000	0.399	0.000	0.064	0.001	0.046	0.000	0.011	0.001	0.015
2	0.000	0.191	0.001	0.065	0.002	0.015	0.002	0.002	0.003	0.000
3	0.001	0.219	0.003	0.007	0.005	0.000	0.004	0.000		
4	0.002	0.107	0.004	0.000	0.008	0.002				
5	0.004	0.054	0.006	0.000	0.010	0.000				
6	0.005	0.091	0.007	0.004						
7	0.007	0.063	0.010	0.000						
8	0.009	0.022	0.013	0.020						
9	0.010	0.009	0.016	0.000						
10	0.013	0.057								
11	0.015	0.072								
12	0.017	0.018								
13	0.019	0.044								
14	0.022	0.028								
15	0.025	0.041								
16	0.028	0.013								
17	0.031	0.008								
18	0.035	0.000								

Table 3. Results of TGA in experiment 1 (the average optimum is 2033.818).

gen	pop = 50		pop = 100		pop = 200		pop = 300	
	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error
500	0.056	2.232	0.152	0.645	0.482	0.293	0.988	0.137
550	0.063	2.080	0.170	0.618	0.532	0.202	1.085	0.137
600	0.067	2.048	0.184	0.618	0.580	0.202	1.185	0.137
650	0.072	2.048	0.199	0.618	0.627	0.202	1.282	0.137
700	0.082	2.048	0.215	0.618	0.676	0.145	1.384	0.137
750	0.090	1.945	0.230	0.618	0.725	0.145	1.482	0.137
800	0.095	1.945	0.246	0.616	0.772	0.145	1.580	0.137
850	0.097	1.943	0.260	0.606	0.820	0.145	1.679	0.137
900	0.099	1.908	0.276	0.606	0.868	0.145	1.776	0.137
950	0.104	1.839	0.292	0.606	0.919	0.145	1.877	0.137
1000	0.111	1.834	0.307	0.600	0.965	0.110	1.976	0.137

considered in this experiment, i.e. each method was tested on 11 group problems (2200 test problems), separately. Moreover, the MA ran for five different population sizes from one times the job number to five times the job number to see how the different population sizes affected the speed of convergence.

The results presented in Table 5, indicate that MA and 2XI are both better than BB in running time and the average running times for both were less than 0.3 s. However, the final schedule objective function value for each problem (see Table 5) and for each α (see Table 6), obtained with the MA were much better than 2XI in quality. From a comparison among the different population sizes, the MA converged quickly in fewer generation (*cgen*) when the population size was increased. Thus, the MA is much better than 2XI in both running time and quality even with $pop = 2n$ only. The greater the population, the slower the speed of convergence and the better the MA solution quality.

6.3 The Comparison Between MA and 2XI

Experiment 3 compared the MA to 2XI for a larger problem. The number of jobs for the MA and 2XI were all from 10, with an increment of 5 until the average running time was over 120 s. The MA population for each test group problem was $2n$ and $\alpha = 0.1, 0.2, \ldots, 1$ for $n \leq 50$ (see Tables 7, 9, and 10). To reduce the running time of the MA and 2XI, $pop = n$ and $\alpha = 0.25, 0.5, \ldots, 1$ for MA (see Tables 8, 11, and 12).

The results presented in Tables 7–12, show that the MA is better than 2XI in both quality and efficiency, especially for larger numbers of jobs and different α. In particular, the quality of the schedule obtained with 2XI decreased, and the running time increased, when the number of jobs increased. However, the MA took only 2 min for $n = 100$. Therefore, the MA can still be used to find a better solution than 2XI within a reasonable running time when the number of jobs is larger.

Table 4. Results in experiment 1 for each α.

α	BB	TGA								MA									
		pop = 50		pop = 100		pop = 200		pop = 300		pop = 10 cgen = 18		pop = 20 cgen = 9		pop = 30 cgen = 5		pop = 40 cgen = 3		pop = 50 cgen = 2	
	Avg. Optimum	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error	Avg. Time	Avg. Error
0.1	872.910	0.010	0.628	0.028	0.265	0.088	0.015	0.179	0.000	0.001	0.020	0.002	0.006	0.002	0.004	0.002	0.007	0.000	0.000
0.2	1132.160	0.010	1.286	0.028	0.365	0.087	0.000	0.180	0.000	0.002	0.064	0.001	0.042	0.003	0.000	0.002	0.000	0.004	0.000
0.3	1390.620	0.010	1.136	0.027	0.270	0.088	0.447	0.179	0.000	0.002	0.026	0.002	0.000	0.002	0.000	0.001	0.000	0.000	0.000
0.4	1648.260	0.010	0.560	0.029	0.590	0.088	0.000	0.180	0.000	0.002	0.046	0.002	0.001	0.001	0.038	0.001	0.000	0.004	0.000
0.5	1905.725	0.010	1.052	0.027	0.475	0.088	0.009	0.180	0.000	0.001	0.119	0.002	0.006	0.002	0.000	0.001	0.000	0.000	0.000
0.6	2163.170	0.010	2.300	0.029	1.210	0.088	0.000	0.180	0.000	0.002	0.055	0.001	0.009	0.002	0.000	0.001	0.000	0.003	0.000
0.7	2420.590	0.010	4.284	0.028	0.000	0.089	0.528	0.180	0.560	0.003	0.240	0.002	0.024	0.001	0.000	0.003	0.035	0.001	0.000
0.8	2677.940	0.010	1.148	0.028	0.715	0.087	0.230	0.180	0.000	0.002	0.053	0.002	0.020	0.003	0.024	0.002	0.000	0.000	0.000
0.9	2934.955	0.010	3.391	0.028	1.015	0.088	0.074	0.181	0.810	0.002	0.084	0.002	0.014	0.003	0.000	0.002	0.000	0.001	0.000
1	3191.850	0.010	4.100	0.028	1.250	0.087	0.400	0.179	0.000	0.002	0.089	0.002	0.056	0.002	0.060	0.001	0.000	0.003	0.075
Avg	2033.818	0.010	1.989	0.028	0.615	0.088	0.170	0.180	0.137	0.002	0.080	0.002	0.018	0.002	0.013	0.001	0.004	0.002	0.008

Table 5. Results in experiment 2 for job number from 5 to 15.

n	BB		2XI		MA														
					pop = n			pop = 2n			pop = 3n			pop = 4n			pop = 5n		
	Avg. Time	Avg. Optimum	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error
5	0.000	616.87	0.000	0.000	1.6	0.000	0.177	1.9	0.000	0.000	1.8	0.000	0.000	2.1	0.000	0.000	1.9	0.000	0.000
6	0.000	909.69	0.000	0.026	1.6	0.000	1.583	1.7	0.000	0.008	1.7	0.000	1.011	1.8	0.000	0.000	1.9	0.000	0.000
7	0.000	1192.05	0.000	0.045	1.6	0.000	0.024	1.7	0.000	0.000	1.6	0.000	0.178	1.7	0.001	0.036	1.7	0.001	0.000
8	0.000	1344.61	0.001	0.045	2.5	0.000	1.836	2.7	0.001	1.430	3.2	0.001	0.000	3.2	0.001	0.000	3.0	0.002	0.000
9	0.001	1712.94	0.001	0.128	2.9	0.001	0.008	2.7	0.001	0.602	3.1	0.001	0.000	2.9	0.002	0.210	3.0	0.002	0.000
10	0.004	2033.82	0.002	0.231	4.1	0.001	0.040	4.3	0.001	0.000	4.2	0.003	0.000	4.3	0.003	0.000	4.2	0.003	0.017
11	0.017	2398.82	0.004	0.151	3.1	0.001	0.537	3.3	0.002	0.005	3.3	0.002	0.228	3.4	0.003	0.000	3.6	0.003	0.000
12	0.041	2636.26	0.006	0.218	4.2	0.001	0.123	4.3	0.004	0.036	4.1	0.006	0.154	4.3	0.008	0.260	4.3	0.010	0.000
13	0.316	3074.13	0.007	0.678	5.0	0.003	0.788	5.1	0.005	0.414	5.5	0.009	0.000	5.6	0.012	0.002	5.4	0.016	0.002
14	0.664	3633.75	0.014	1.243	5.0	0.004	0.245	4.9	0.008	0.167	5.4	0.012	0.000	5.2	0.012	0.000	5.1	0.018	0.000
15	7.118	4184.14	0.019	2.247	5.9	0.006	0.408	5.7	0.010	0.185	5.9	0.016	0.000	5.5	0.019	0.000	5.4	0.025	0.000
Avg.	0.742	2157.92	0.005	0.456	3.4	0.001	0.524	3.5	0.003	0.259	3.6	0.005	0.143	3.6	0.006	0.046	3.6	0.007	0.002

Table 6. Results in experiment 2 for job number from 5 to 15 for each α.

α	BB		2XI		MA														
	Avg. Time	Avg. Value			pop = n			pop = 2n			pop = 3n			pop = 4n			pop = 5n		
			Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error	Avg. cgen	Avg. Time	Avg. Error
0.1	0.831	896.1	0.004	0.166	3.58	0.001	0.203	3.95	0.003	0.140	4.06	0.006	0.018	4.28	0.008	0.008	4.06	0.008	0.000
0.2	0.770	1178.1	0.004	0.230	3.75	0.002	0.353	3.76	0.003	0.092	3.67	0.003	0.042	3.90	0.006	0.002	3.84	0.008	0.001
0.3	0.771	1459.1	0.007	0.345	3.39	0.001	0.555	3.53	0.002	0.016	3.79	0.004	0.082	3.66	0.004	0.051	3.62	0.007	0.000
0.4	0.753	1739.5	0.003	0.498	3.40	0.001	0.348	3.32	0.002	0.155	3.73	0.005	0.055	3.57	0.005	0.085	3.62	0.006	0.000
0.5	0.743	2019.7	0.006	0.495	3.21	0.001	0.780	3.40	0.003	0.127	3.57	0.004	0.095	3.56	0.005	0.000	3.45	0.008	0.000
0.6	0.734	2299.3	0.004	0.499	3.49	0.002	0.343	3.31	0.003	0.088	3.48	0.004	0.181	3.65	0.005	0.104	3.42	0.008	0.000
0.7	0.710	2578.7	0.005	0.596	3.29	0.001	0.370	3.55	0.003	0.986	3.75	0.005	0.012	3.34	0.006	0.015	3.55	0.005	0.015
0.8	0.714	2857.7	0.004	0.497	3.29	0.002	0.731	3.44	0.003	0.341	3.46	0.004	0.425	3.41	0.006	0.077	3.52	0.007	0.000
0.9	0.697	3136.3	0.005	0.548	3.21	0.002	0.947	3.14	0.003	0.286	3.50	0.004	0.000	3.36	0.005	0.000	3.45	0.007	0.000
1	0.695	3414.7	0.005	0.682	3.25	0.001	0.614	3.36	0.003	0.355	3.23	0.006	0.518	3.45	0.004	0.118	3.42	0.009	0.000
Avg.	0.742	2157.9	0.005	0.456	3.39	0.001	0.524	3.48	0.003	0.259	3.62	0.005	0.143	3.62	0.006	0.046	3.59	0.007	0.002

Table 7. Results in experiment 3 for n = 10, 15, …, 50.

n	MA(pop = 2n)			2XI	
	Avg. cgen	Avg. Time	Avg. Value	Avg. Time	Avg. Difference
10	4.17	0.001	2033.82	0.002	0.231
15	5.81	0.018	4184.14	0.019	0.011
20	9.47	0.079	6297.75	0.095	0.010
25	11.21	0.240	9479.71	0.298	0.043
30	17.00	0.754	13457.34	0.783	0.038
35	18.87	1.518	17382.87	1.900	0.046
40	22.73	3.156	22548.63	3.765	0.085
45	29.33	6.645	27007.32	6.722	0.099
50	33.17	11.604	33118.94	11.743	0.118

Table 8. Results in experiment 3 for n = 55, 60, …, 100.

n	MA(pop = 2n)			2XI	
	Avg. cgen	Avg. Time	Avg. Value	Avg. Time	Avg. Difference
55	33.14	5.576	47177.81	21.896	12.849
60	34.30	8.245	52618.05	28.916	13.882
65	39.36	13.139	64494.59	33.814	15.541
70	42.49	19.326	73412.88	43.464	21.878
75	46.05	28.226	84917.88	63.206	24.089
80	48.69	38.192	94439.38	87.649	28.908
85	50.53	51.347	108182.96	127.556	38.896
90	53.05	68.641	119325.69		
95	56.53	90.950	134349.06		
100	61.15	120.808	143946.40		

6.4 The Comparison Between the MA and BB when Machine 2 is Dominant

The second machine is said to be dominant when the processing time is greater on the second machine than on the first. The BB solved very large problems in this special case [12]. The last experiment compared MA to BB for this special case problem. There were 17 group problems in which the number of jobs was 10, 15 to 90, i.e. there are 3400 test problems. The population of MA for each test group problem was $2n$. The final schedule objective function value for 3400 test problems obtained in MA were all equal to the optimum. The results presented in Tables 13 and 14 indicated that BB is more efficient for this special case than the MA for each n and α. However, the MA outperformed BB when processing times were randomly generated from experiments 1, 2, and 3.

7. Conclusion

Genetic algorithms have aroused intense interest in the past few years because of their flexibility, versatility, and effectiveness in solving problems in which traditional optimisation methods are insufficient. GAs need no simplifying assumptions of linearity, continuity, etc., and thus can solve highly complex real-world problems. Nevertheless, there are many situations in which simple GAs do not perform particularly well, and so various methods of hybridisation, e.g. the MA, in which a local search plays a significant part, have been proposed. Because of the complementary properties of GAs and conventional heuristics, an MA often outperforms either method operating alone.

In this study, an MA is proposed to solve a special computationally difficult flowshop scheduling case by combining the GA with some local optimisers, such as the greedy algorithm and tabu search, to enhance the performance of the GAs.

The proposed MA is compared with a simple traditional genetic algorithm (TGA), and an efficient heuristic method (2XI), which also combines nearly the same local optimisers employed in the MA, and the pure branch-and-bound (BB) procedure, which was discussed in our previous paper, to verify that our MA method produces a good quality solution within a reasonable running time. A number of numerical tests in a series of randomly generated problems are used and indicate that the performance and efficiency of the proposed MA outperforms pure genetic algorithms, and pure branch-and-bound algorithms and an efficient heuristic. Because of the special structure of the processing time, BB is better than the MA when the second machine is dominant. However, our results compare favourably with the best-known existing branch-and-bound algorithm, and with the traditional genetic algorithm and with the best-known efficient heuristic algorithm for general case.

Acknowledgement

I wish to thank both the editor and referees for their constructive comments and recommendations, which have significantly improved the presentation of this paper. This research was supported in part by the National Science Council of Taiwan, under grant NSC 87-2213-E-025-001.

References

1. W. E. Smith, "Various optimizers for single stage production", Naval Research Logistics Quarterly, 3, pp. 59–66, 1956.
2. S. K. Gupta and J. Kyparisis, "Single machine scheduling research", Omega, 15, pp. 207–227, 1987.
3. T. D. Fry, R. D. Armstrong and H. A. Lewis, "Framework for single machine multiple objective sequencing research, Omega, 17, pp. 595–607, 1979.
4. A. Nagar, J. Haddock and S. S. Heragu, "Multiple and bicriteria scheduling: a literature review", European Journal of Operational Research, 81, pp. 88–104, 1995.
5. R. A. Dudek, S. S. Panwalker and M. L. Smith, "The lessons of flowshop scheduling research", Operations Research, 40, pp. 7–13, 1992.
6. W. J. Selen and D. Hott, "A mixed integer goal programming formulation of the standard flowshop scheduling problem", Journal of the Operational Research Society, 37, pp. 1121–1128, 1986.
7. J. M. Wilson, "Alternative formulations of a flowshop scheduling problem", Journal of the Operational Research Society, 40, pp. 395–399, 1989.
8. T. Gonzalez and T. Sen, "Flowshop and Jobshop Schedules: Complexity and Approximations", Operations Research, 26, pp. 36–52, 1978.
9. M. R. Garey, D. S. Johnson and R. R. Sethi, "The complexity of flowshop and jobshop scheduling", Operations Research, 1, pp. 117–129, 1976.
10. S. French, Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. Ellis Horwood, Chichester, 1982.

Table 9 Results in experiment 3 of MA for α = 0.1,0.2, …,1 and n = 10, 15, …, 50.

α \ n	10			15			20			25			30		
	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value
0.1	4.45	0.000	872.91	6.50	0.022	1532.19	10.70	0.088	2100.12	11.30	2.555	2912.91	16.90	0.764	3884.76
0.2	4.45	0.003	1132.16	5.50	0.019	2122.35	10.75	0.091	3034.42	11.75	1.277	4374.52	16.30	0.731	6013.53
0.3	3.65	0.000	1390.62	5.45	0.019	2711.90	9.30	0.074	3967.49	10.90	0.769	5834.99	17.00	0.761	8140.73
0.4	4.30	0.003	1648.26	5.90	0.011	3301.26	9.55	0.074	4900.02	10.60	0.563	7294.11	16.75	0.745	10268.57
0.5	4.30	0.003	1905.73	5.80	0.016	3890.45	8.05	0.071	5832.70	11.55	0.505	8752.40	16.35	0.728	12395.50
0.6	3.95	0.003	2163.17	6.20	0.019	4479.30	10.75	0.091	6765.09	10.90	0.398	10210.62	17.10	0.761	14522.20
0.7	3.35	0.000	2420.59	5.40	0.022	5068.15	8.70	0.074	7697.31	10.40	0.314	11668.95	18.25	0.797	16648.19
0.8	3.90	0.000	2677.94	5.95	0.008	5656.90	8.20	0.074	8629.08	11.30	0.288	13125.94	17.45	0.777	18773.13
0.9	4.35	0.003	2934.96	5.85	0.022	6245.44	8.00	0.066	9560.21	11.85	0.281	14583.04	17.10	0.747	20900.37
1.0	5.00	0.000	3191.85	5.50	0.016	6833.50	10.65	0.085	10491.10	11.55	0.242	16039.65	16.75	0.731	23026.45
Avg.	4.17	0.001	2033.82	5.81	0.018	4184.14	9.47	0.079	6297.75	11.21	0.719	9479.71	17.00	0.754	13457.34

α \ n	35			40			45			50		
	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value
0.1	18.40	1.503	4801.43	24.20	3.371	5984.46	28.75	6.646	6963.42	36.20	12.810	8307.68
0.2	18.25	1.451	7598.45	23.55	3.297	9666.77	29.45	6.714	11418.66	34.70	12.060	13821.62
0.3	19.70	1.582	10394.30	21.65	3.019	13348.83	32.60	7.321	15872.01	32.45	11.280	19336.08
0.4	18.80	1.503	13190.30	21.50	2.986	17029.04	28.10	6.379	20326.17	34.80	12.250	24849.96
0.5	17.30	1.401	15985.93	22.90	3.203	20709.73	26.35	6.019	24781.25	30.20	10.720	30363.30
0.6	18.60	1.500	18781.00	23.45	3.266	24389.47	30.05	6.835	29234.07	36.25	12.538	35875.81
0.7	19.30	1.536	21577.26	25.20	3.393	28070.72	28.80	6.511	33686.83	30.15	10.566	41389.16
0.8	18.90	1.525	24371.66	20.25	2.885	31750.24	30.80	6.956	38142.49	29.75	10.393	46903.10
0.9	20.80	1.673	27167.44	21.45	3.008	35429.97	30.15	6.761	42596.18	33.15	11.519	52415.62
1.0	18.65	1.511	29960.95	23.10	3.129	39107.10	28.25	6.308	47052.15	34.00	11.898	57927.11
Avg.	18.87	1.518	17382.87	22.73	3.156	22548.63	29.33	6.645	27007.32	33.17	11.604	33118.94

Table 10. Results in experiment 3 of 2XI for α = 0.1,0.2, …,1 and n = 10, 15, …, 50.

α\n	10		15		20		25		30		35		40		45		50	
	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference
0.1	0.000	0.055	0.022	0.215	0.102	0.435	3.379	2.245	0.819	2.755	2.115	1.465	4.077	2.830	7.505	2.755	12.965	4.580
0.2	0.003	0.080	0.014	0.760	0.099	0.520	1.429	4.310	0.805	2.970	2.005	3.810	3.915	3.750	7.121	7.690	12.206	10.950
0.3	0.003	0.165	0.027	1.650	0.082	1.710	0.952	5.350	0.830	4.825	1.783	6.085	4.047	7.480	6.703	11.790	11.416	14.080
0.4	0.003	0.160	0.008	2.140	0.093	2.200	0.749	6.820	0.810	6.320	1.813	6.710	3.788	14.600	6.764	16.390	12.397	17.560
0.5	0.003	0.200	0.027	2.100	0.099	2.200	0.615	7.625	0.761	7.900	1.819	7.800	3.648	16.400	6.448	20.900	10.849	20.100
0.6	0.003	0.290	0.019	2.520	0.096	2.340	0.472	8.920	0.734	7.050	1.964	9.880	3.533	21.460	6.676	22.550	12.689	25.680
0.7	0.003	0.280	0.016	3.500	0.088	2.560	0.424	10.530	0.766	9.845	1.940	11.050	3.396	21.670	6.451	25.690	10.693	30.830
0.8	0.000	0.320	0.019	2.720	0.102	2.060	0.385	12.320	0.755	12.350	1.871	13.870	3.654	24.940	6.459	28.410	10.518	33.417
0.9	0.003	0.360	0.014	3.060	0.096	2.435	0.314	12.925	0.769	11.270	1.835	14.880	3.780	27.960	6.445	30.816	11.658	37.674
1.0	0.000	0.400	0.025	3.800	0.088	2.700	0.291	15.000	0.780	11.100	1.854	16.100	3.810	29.250	6.648	30.849	12.042	41.441
Avg.	0.002	0.23	0.019	2.247	0.095	1.916	0.901	8.605	0.783	7.638	1.900	9.165	3.765	17.034	6.722	19.784	11.743	23.631

Table 11. Results in experiment 3 of MA for α = 0.25,0.5, …,1 and n = 55, 60, …, 100.

α\n	55			60			65			70			75		
	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value
0.25	31.10	5.271	20 787.33	34.95	8.408	23 050.99	39.70	13.423	28 040.15	40.70	19.058	31 762.88	44.75	27.868	36 543.35
0.50	32.40	5.468	38 382.63	33.85	8.156	42 762.18	41.80	13.824	52 346.73	41.55	18.871	59 528.80	46.05	27.930	68 794.08
0.75	33.90	5.677	55 974.38	36.85	8.715	62 471.44	39.95	13.191	76 644.94	44.50	19.847	87 299.65	48.30	29.474	101 038.51
1.0	35.15	5.888	73 566.90	31.55	7.699	82 187.60	36.00	12.120	100 946.55	43.20	19.526	115 060.20	45.10	27.633	133 295.60
Avg.	33.14	5.576	47 177.81	34.30	8.245	52 618.05	39.36	13.139	64 494.59	42.49	19.326	73 412.88	46.05	28.226	84 917.88

Table 11. Results in experiment 3 of MA for α = 0.25,0.5, …,1 and n = 55, 60, …, 100 (continued).

α\n	80			85			90			95			100		
	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value	Avg. cgen	Avg. Time	Avg. Value
0.25	50.55	39.842	40474.74	54.55	55.781	46182.06	50.95	66.974	50773.75	54.35	89.617	56977.11	60.70	119.867	60 920.78
0.50	45.80	35.908	76454.70	46.45	47.846	87520.18	48.75	63.846	96479.25	57.50	91.619	108558.40	65.45	128.293	116 256.78
0.75	49.15	38.556	112424.76	52.15	52.291	128844.25	53.55	69.097	142181.65	60.25	95.001	160137.43	59.95	118.148	171 633.06
1.0	49.25	38.460	148403.30	48.95	49.469	170185.35	58.95	74.645	187868.10	54.00	87.561	211723.30	58.50	116.923	226 975.00
Avg.	48.69	38.192	94439.38	50.53	51.347	108182.96	53.05	68.641	119325.69	56.53	90.950	134349.06	61.15	120.808	143 946.40

Table 12. Results in experiment 3 of 2XI for α = 0.25,0.5, …,1 and n = 55, 60, …, 100.

α\n	55		60		65		70		75		80		85	
	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference	Avg. Time	Avg. Difference
0.1	21.392	5.69	31.672	9.47	28.389	5.55	44.351	9.79	66.093	11.45	89.934	7.80	124.185	15.25
0.2	22.070	13.10	33.709	13.25	27.225	9.13	43.623	17.00	61.772	20.70	85.591	17.98	122.700	27.27
0.3	22.422	12.85	34.942	17.31	30.261	19.43	42.888	24.22	62.548	33.31	86.822	45.46	131.300	51.81
0.4	21.700	19.75	34.935	15.50	29.790	28.05	42.994	36.50	62.411	30.90	88.247	44.40	132.039	61.25
Avg.	21.896	12.849	33.814	13.883	28.916	15.541	43.464	21.878	63.206	24.089	87.649	28.907	127.556	38.896

Table 13. Results in experiment 4 of 2XI for n = 10,15, …, 90.

n		BB		MA(pop = 2n)	
	Avg. Time	Avg. Optimum	Avg. cgen	Avg. Time	Avg. Error
10	0.000	2 295.52	1.16	0.000	0.000
15	0.000	4 764.85	1.43	0.000	0.000
20	0.000	7 410.75	1.44	0.003	0.000
25	0.000	11 432.93	1.33	0.007	0.000
30	0.001	15 786.79	1.32	0.014	0.000
35	0.001	20 914.86	1.55	0.029	0.000
40	0.001	27 976.77	1.53	0.051	0.001
45	0.002	33 937.46	1.46	0.077	0.002
50	0.002	43 643.57	1.65	0.129	0.002
55	0.002	50 943.35	1.37	0.167	0.003
60	0.003	60 240.29	1.32	0.220	0.004
65	0.003	71 314.25	1.47	0.340	0.005
70	0.004	82 835.90	1.61	0.516	0.005
75	0.005	91 638.00	1.57	0.662	0.006
80	0.006	104 991.05	1.33	0.721	0.006
85	0.007	119 088.32	1.64	1.125	0.008
90	0.009	132 949.69	1.97	1.747	0.009
Avg.	0.005	5 189 201.85	1.479	0.342	0.003

Table 14. Results in experiment 4 for each α.

α		BB		MA(pop = 2n)	
	Avg. Time	Avg. Optimum	cgen	Avg. Time	Avg. Error
0.1	0.002	12 286.21	1.41	0.333	0.000
0.2	0.003	21 087.53	1.46	0.330	0.000
0.3	0.003	29 888.82	1.43	0.335	0.000
0.4	0.004	38 690.11	1.50	0.352	0.000
0.5	0.002	47 491.39	1.59	0.378	0.000
0.6	0.002	56 292.67	1.46	0.339	0.002
0.7	0.002	65 093.95	1.48	0.327	0.004
0.8	0.004	73 895.22	1.51	0.358	0.006
0.9	0.002	82 696.5	1.51	0.337	0.008
1.0	0.003	91 497.78	1.44	0.327	0.011
Avg.	0.003	51 892.02	1.48	0.342	0.003

11. E. L. Lawler, L. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, "Sequencing and scheduling: algorithms and complexity", in S.C. Graves, A. H. G. Rinnooy Kan and P. Zipkin (ed.), Handbooks in Operations Research and Management Science, vol 4, Logistics of Production and Inventory, North-Holland, Amsterdam, pp. 455–522, 1993.
12. A. Nagar, S. H. Sunderesh and J. Haddock, "A branch-and-bound approach for a two-machine flowshop scheduling problem", Journal of the Operational Research Society, 46, pp. 721–734, 1995.
13. W. C. Yeh, "A new branch-and-bound approach for the n/2/flowshop/$\alpha F+\beta C_{max}$ flowshop scheduling problem", Computers and Operations Research, 26, pp. 1293–1310, 1999.
14. W. C. Yeh, "An efficient branch-and-bound algorithm for the two-machine bicriteria flowshop scheduling problem", Journal of Manufacturing Systems, 20, pp. 113–123, 2001.
15. N. R. Smith, F. Al-Khayyal and F. Griffin, "A tree-search method for solving a cutting stock assignment problem", International Journal of Production Research, 38, pp. 1557–1578, 2000.
16. C. A Cotta, J. F. Aldana, A. J. Nebro and J. M. Troya, "Hybridizing genetic algorithms with branch-and-bound techniques for the resolution of the TSP", Artificial Neural Nets and Genetic Algorithm, pp. 277–280, 1995.
17. S. Ghoshray, K. K. Yeh and J. Andrian, "Modified genetic algorithms by efficient unification with simulated annealing", Artificial Neural Nets and Genetic Algorithm, pp. 487–490, 1995.
18. E. M. Robert, "Breeding hybrid strategies: optimial behavior for ologopolists", Proceeding of 3th International Conference on Genetic Algorithms, pp. 198–207, 1989.
19. K. Takashi, K. Hiroaki and N. Masakazu, "A hybrid search for genetic algorithms: combining genetic algorithms, tabu search, and simulated annealing", Proceeding of 5th International Conference on Genetic Algorithms, p. 641, 1993.
20. P. Moscato and M. Norman, "A memetic aapproach for the traveling salesman problem: implementation of a computational ecology for combinatorial optimization on message-passing system", Proceeding of 20th International Conference on Parallel Computing and Transportation Applications, 1992.
21. M. Hattori, M. Naruse, J. Shirataki and T. Tomikawa, "A study of parameter optimization in mega-genetic algorithm", Proceeding of 20th International Conference on Computers and Industrial Engineering, pp. 445–448, 1996.
22. G. Leipins and M. Hilliard, "Genetic algorithm: foundation and applications", Annals of Operations Research, 21, pp. 31–58, 1989.
23. W. C. Yeh, "A memetic algorithm for the min k-cut problem", Control and Intelligent Systems, 28, pp. 47–55, 2000.