한국지능정보시스템학회논문지 제13권 제3호 2007년 9월(pp.63~82)

Adaptive Hybrid Genetic Algorithm Approach to Multistage-based Scheduling Problem in FMS Environment

YoungSu Yun
Schoo l of Business Adminstration,
Chosun University
(ysyun@chosun.ac.kr)

KwanWoo Kim
Department of Industrial Engineering,
Hanyang University
(kimkwanwoo@hanyang..ac.kr)

In this paper, we propose an adaptive hybrid genetic algorithm (ahGA) approach for effectively solving multistage-based scheduling problems in flexible manufacturing system (FMS) environment. The proposed ahGA uses a neighborhood search technique for local search and an adaptive scheme for regulation of GA parameters in order to improve the solution of FMS scheduling problem and to enhance the performance of genetic search process, respectively.

In numerical experiment, we present two types of multistage-based scheduling problems to compare the performances of the proposed ahGA with conventional competing algorithms. Experimental results show that the proposed ahGA outperforms the conventional algorithms.

Key Words : Adaptive hybrid Genetic Algorithm, Local search, Adaptive scheme, Multistage-based scheduling problem, Flexible manufacturing systems

Received : February 2007 Accepted : July 2007 Corresponding author : KwanWoo Kim

1. Introduction

In modern manufacturing systems such as flexible manufacturing system (FMS), multi-stage-based scheduling problem composed of multi-state components is common. It provides a detailed description of manufacturing capabilities and requirements for transforming raw materials into end products through a multistage process.

In multistage-based scheduling, the task of creating any feasible schedule, let alone the one with the optimal solution, is usually quite compli-

* This study was supported by research funds from Chosun University, 2006.

cated because of the complex interrelationships between the units of the different stages. The search space of feasible schedules, therefore, grows exponentially as there are certain increases in the number of different jobs that must be processed, the number of the operations required by each job, and the number of facilities that can perform the process of each job. This exponential growth makes it very difficult or even impossible to use conventional mathematical programming or exhaustive search approaches for finding global optimal schedule in terms of any performance measure for the problems with practical complexity (Belton and Elder, 1996 ; Blazewicz et al., 1994 ; De and Lee, 1990 ; French, 1982).

Recently a glowing interest in applying genetic algorithm (GA) approaches to effectively deal with multistage-based scheduling in FMS environment has been shown (Jawahar et al., 1998; Gen and Cheng, 2000 ; Kim et al, 2004 ; Yang, 2001). The important advantage with GA approaches, when compared with conventional approaches, is that they are simple in principle and abstract the specific problem characteristics, which makes it possible to apply them to wide range of scheduling problems. However, their applications to more complex scheduling environments such as multistage-based scheduling problems have been restricted since creating the complete schedules for each job highly requires computational effort (Charalambous, 1997). There are also other problems in representing GA population, designing GA operators, and controlling GA parameters, which are liable to generate illegal schedules.

To overcome these weaknesses of GA in application to multistage-based scheduling problem, hybridized algorithms using GA and conventional heuristics have been developed (Gen and Cheng, 1997 ; Yun, 2006). With hybrid approaches, GA can guarantee a global optimal solution in global search space using the increase information from conventional heuristics.

Holsapple et al. (1993) investigated a hybrid scheduler combining GA with conventional heuristics. They tried to generate job sequence and feasible schedule simultaneously and also developed a new sub-tour chunking crossover and single-swap mutation for representing job sequence. Yang (2001) developed a GA-based discrete dynamic programming (GA-DDP) approach to generate static schedules in FMS environment. This approach generates job sequence in the GA approach and the local optimization of partial schedules in the DDP approach.

However, the above-stated two hybrid approaches may not guarantee to generate global optimal schedule for large-scaled and multistage-based scheduling problem in the FMS environment with multi-objective functions, since the first approach only concentrates on two-stage scheduling problem and the second approach on single objective scenario. They do not also consider alternative schedules that may be considered in FMS environment when optimal schedule is not available.

More recently Kim et al. (2004) proposed a network-based hybrid GA (nhGA) approach to generate static scheduling in FMS environment

with multi-objective functions. They used an adaptive scheme using a fuzzy logic controller (FLC) to adaptively regulate the rate of GA parameters. However, only mutation operator was used for the adaptive scheme, crossover operator was not applied in the nhGA procedure. In general, the use of crossover operator in GA implementation has been known as a key factor for improving the performance of GA. Therefore, both the use of crossover and mutation operators usually produces better performance than that of one operation alone. Also, there are some weaknesses in applying the FLC to the nhGA, that is, the computational scheme for controlling the given maximum and minimum values of fuzzy membership function highly depend on the problems under consideration.

To overcome the above-stated weaknesses, we develop a new adaptive hybrid GA (ahGA). The proposed ahGA uses i) GA approach with adaptive scheme and ii) neighborhood search approach with local search, in order to effectively locate the optimal schedule of multistage-based scheduling problems in FMS environment with multi-objective functions. The adaptive scheme used regulates the rates of crossover and mutation operators adaptively, and the local search performs a precision search around the convergence area after the run of GA.

The rests of the paper are organized as follows. In Section 2, we describe a hypothetical concept and description of multistage-based scheduling problem in FMS environment and the mathematical model for multistage-based scheduling problem is then given in Section 3. In Section 4, we propose the logics and overall implementation procedure of the proposed ahGA including chromosome representation of GA and local search. In Section 5, numerical experiments are presented to demonstrate the efficiency of the proposed ahGA. Finally, some concluding remarks are given in Section 6.

2. Multistage-based Scheduling Problem in FMS Environment

FMS is an enhancement of the cellular manufacturing paradigm and represents the state-of-art in the design of manufacturing systems. Typically, FMS is composed of multiple workstations (or machine centers), material handling systems, and loading-unloading stations, which may be regarded as a multistage-based scheduling form. That is, it is a set of the stages which should perform specified tasks and also consist of different several equipments that must process a given task at each stage, therefore, it can be decomposed into several stages and the stages are connected each other. The continuous stages are also considered simultaneously in order to coordinate between stages and ensure that the obtained schedule is feasible.

Some assumptions for implementing the multistage-based scheduling problem are defined as follow.

There are several tasks to be processed at each stage, each of which should be treated in

more than one workstation and also requires the processing time for the completion of its treatment at each stage. The environment of task, including processing and transportation times, is already known. It is assumed that all tasks are started at time zero, and are not interrupted by any other tasks from its start time to its end time. We also assume that the sequence of each task is not changed.

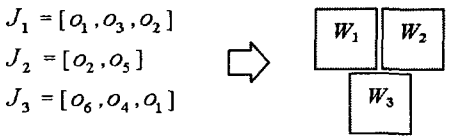
Under this type of multistage-based environment, locating any feasible route, let alone the one with the optimal makespan, is usually quite a complicated task due to the complex interrelationships among different stages and the expensive computational effort that may be involved in solving complex multistage-based scheduling problems.

The objective is to locate the optimal makespan under certain criteria such as minimum time and minimum cost. Even using notable shortest path algorithms such as Dijkstra algorithm (Jensen and Barnes, 1980) and Floyd algorithm (Jensen and Barnes, 1980), or out-of-kilter (Whatley, 1985), it is still not easy to deal with a complex multistage-based scheduling problem. Therefore, for the multistage-based scheduling problem with multi-objective functions, these conventional techniques may not be used. Although several techniques based on goal programming for multi-objective functions have been developed, they can only illustrate for small multistage-based scheduling problem (Sancho, 1986 ; Shiedovich, 1988).

Using the above-stated basic concept on multistage-based scheduling problem in FMS environment, we suggest a new description of the problems in the following Section.

2.1 Description of Multistage-based Scheduling Problem

[Figure 1] shows a simple scheduling problem with three workstations (W_1, W_2, W_3), three jobs (J_1, J_2, J_3), and six operations (o_1, o_2, o_3, o_4, o_5, o_6) in FMS environment. This problem was already examined in the conventional works (Holsapple, 1993 ; Yang, 2001). Each job requires two or three kinds of operations among the six operations and is run at each workstation. Note that in this example an operation may be transferred from one job to any of others.



[Figure 1] An example of simple scheduling problem

Detailed processing information for the operations on the three workstations is shown in *<Table 1> This problem consists of three flexible* workstations. Collectively, each workstation can perform various operations. No single workstation can perform all jobs, but more than one workstation may be able to perform a given job. For instance, workstation W_1 can perform only operations o_1, o_2, o_4, o_5, and o_6. On the other hand, operation o_1 can be performed on W_1 and W_2.

This is reflected in the different processing times for the same operation across different workstations.

<Table 2> depicts the relevant data of the jobs performed along with the precedence requirements of the operations that must be scheduled in this system.

In <Table 2> t_k^{ATP} is the average value of total processing time for each job J_k ($k=$, *1, 2, ···, K*), t_k^D is the due date for each job J_k, and t_k^{TP} is the sum of the total penalty cost for each job J_k. For instance, the three quantities ($t_1^{ATP}=$ 86.5, $t_1^D=90.0$, and $C_1^{TP}=10$) of job J_1 in Table 2 are explained as follow.

We first assume that the batch size of each job is 1 unit. J_1 requires the three operations o_1, o_3 and o_2 to be performed on that sequence. The t_1^{ATP} is the sum of the average processing time for the o_1, o_3 and o_2 in this system. From the information of <Table 1>, we compute this quantity as $23.0+40.0+23.5=86.5$ time units.

<Table 1> Processing times of operations

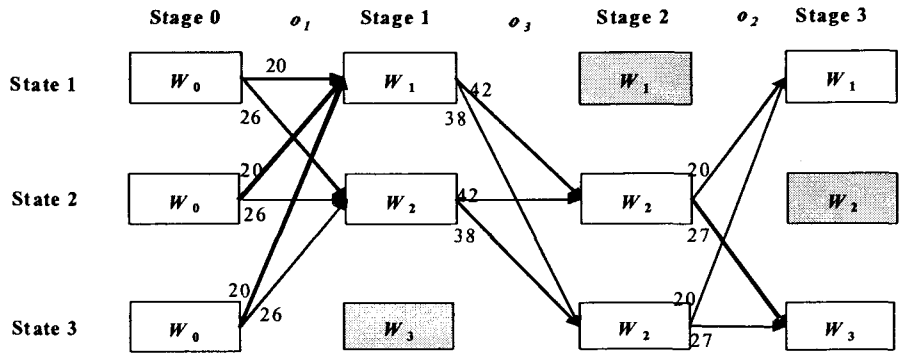
Workstation	o_1	o_2	o_3	o_4	o_5	o_6
W_1	20	20	-	39	26	37
W_2	26	-	42	41	29	-
W_3	-	27	38	34	-	29
Average	23.0	23.5	40.0	38.0	27.5	33.0

<Table 2> Job-related data for simple scheduling in each workstation problem

Job No.	Operation sequence	t_k^{ATP}	t_k^D	C_k^{TP}
J_1	$o_1 \rightarrow o_3 \rightarrow o_2$	86.5	90.0	10
J_2	$o_2 \rightarrow o_5$	51.0	90.0	11
J_3	$o_6 \rightarrow o_4 \rightarrow o_1$	94.0	95.0	13

The t_k^D is the time limit that a job should be completely finished within the time from its starting time to its end time. For the J_1, it is 90. Finally, the last column in <Table 2> contains the information on the penalty cost per unit time associated with each job. We can thus assess a C_k^{TP} on the total number of time units by which an operation is tardy. For the J_1 (and all other jobs), this quantity is 10/unit time. In reality, more complex penalty cost structures could prevail. The remainder of the <Table 2> is interpreted as a same way.

Using [Figure 1], <Tables 1> and <Tables 2>, we can represent legal schedules as a type of multistage. For instance, the J_1 with the fixed operation sequence ($o_1 \rightarrow o_3 \rightarrow o_2$ can be expressed as shown in [Figure 2]. In [Figure 2], each state is defined as workstations, and each stage represents the process of transferring an operation from a workstation to the next one. For the J_1, stage 0 stands for the initial conditions of the multistage-based scheduling problem in [Figure 1]. Stages 1, 2 and 3 represent the processing status of the o_1, o_3 and o_2, respectively. A solid line connecting two rectangles of two adjacent stages denotes an action, which indicates that the J_1 is transferred from the left workstation to the right one and the current operation is processed on the right workstation. The numbers associated with the lines mean the processing times when an operation is processed from a workstation to another. If an operation cannot be processed on a workstation, there is no line between the workstations associated with the operation.



$J_1 = [O_1, O_3, O_2]$

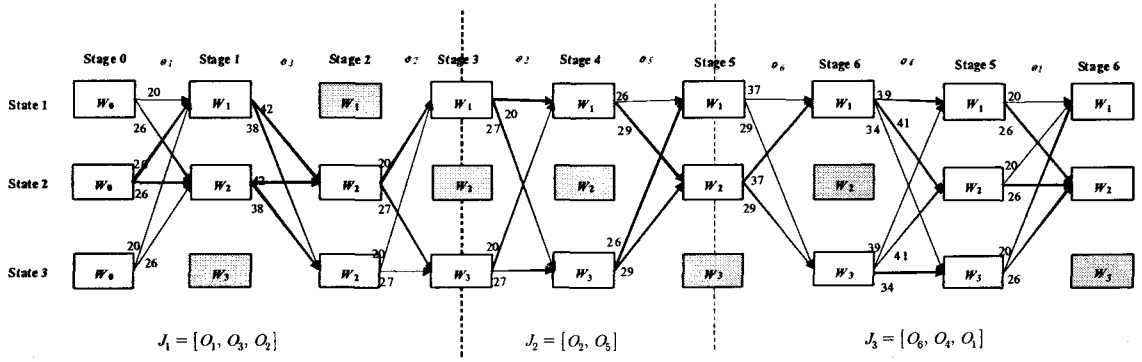
[Figure 2] A multistage form for J_1

Like the multistage form of the J_1, the remainder jobs (J_2, J_3) are also represented as multistage forms. [Figure 3] shows the whole process with the job sequence $J_1 \rightarrow J_2 \rightarrow J_3$. There are the processing times of two or three operations at each job such as $J_1(o_1, o_3, o_2) \rightarrow J_2(o_2, o_5) \rightarrow J_3(o_6, o_4, o_1)$. For [Figure 3], <Tables 1> and <Tables 2> are used to demonstrate the realization. The whole process can be divided into several stages and states.

3. Mathematical Formulation

In general, scheduling problem in FMS environment is how to assign the operations of jobs onto workstations so that certain objectives can be optimized. Optimizing objectives is to locate an optimal schedule among all feasible schedules given objectives such as minimum cost, minimum time, or maximum quality.

For multistage-based scheduling problem in



[Figure 3] Multi-stage form for simple scheduling problem

FMS environment, many stages and states by increased problem size must be considered though this problem can be approached by conventional shortest-path method (Jensen and Barnes, 1980) or dynamic programming, which will highly affect the efficiency of the procedures implementing shortest path method or dynamic programming in getting the optimal solution. This is usually known as the dimension explosion.

If the multistage-based scheduling problem has multi-objective functions, it will be more difficult to solve this problem using the conventional algorithms mentioned above. Moreover, if the problem size increase, it will become much more difficult to deal with even on the case of a single-objective function.

In this paper, we develop an efficient approach to solve the multistage-based scheduling problem in FMS environment with three objective functions. Our objective searches the optimal total penalty out of minimized total flow time in minimized makespan. First, we minimize the makespan T_M, and its mathematical formulation is as follow :

$$minimize \quad T_M = \max_{i,j,k}\{t^e_{ijk}\} \tag{1}$$

where t^e_{ijk} is the finish time of operation o_i on workstation W_j for each job J_k. Equation (1) defines the makespan that the processing time of last operation o_i on workstation W_j is finished.

Secondly, we minimize the total flow time T_F that means the finish time of last operation o_i on workstation W_j at each job J_k as follow :

$$minimize \quad T_F = \sum_{k=1}^{K} \max_{i,j} \{t^e_{ijk}\} \tag{2}$$

Lastly, we minimize the total tardiness penalty P_T which defines the sum of penalty costs for all orders, where the penalty cost for a job is the multiplication of its unit penalty cost and the absolute difference between its completion time and its processing time, given that the former is larger than the latter.

$$minimize \; P_T = \sum_{k=1}^{K} \{c_k^{TP} \times \max\{0, \max_{i,j}\{t^e_{ijk}\} - t_k^D\}\} \tag{3}$$

The objective functions (1), (2) and (3) are subjected to the following constraints. Constraint (4) ensures that none of the precedence constraints are violated. Constraint (5) means non-negative integer value.

$$t^e_{hjk} + p_{hjk} \le t^s_{ijk}, \quad \forall_i \in succ_h \tag{4}$$

$$t^s_{ijk}, t^e_{hjk} \ge 0, \quad \forall i, j, k \tag{5}$$

where t^e_{hjk} is the end time of operation o_h on workstation W_j for each job J_k, P_{hjk} is the processing time of operation o_h on workstation W_j for each job J_k, $succ_h$ is a set of successors of operation h($i = 1, 2, \cdots, h, \cdots, I$), and t^s_{ijk} is the start time of operation o_i on workstation W_j for each job J_k,

4. Adaptive Hybrid Genetic Algorithm Approach

Locating an optimal schedule under the

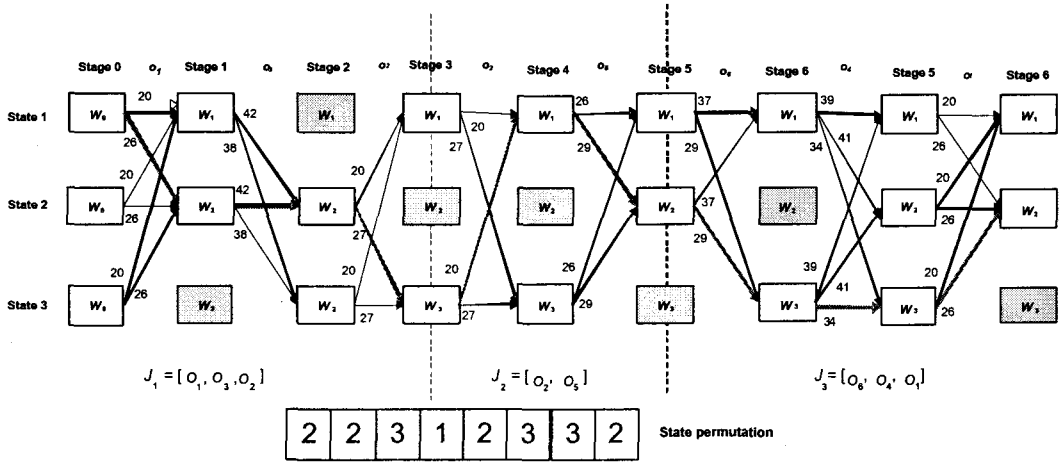
complete enumeration of all possible job sequences is normally impractical for scheduling purpose unless the number of jobs is very small, This is mainly because that the number of sequences of a job k is $k!$.

Suppose $k = 10$, for example, we have 10! = 3,628,800, which means that a schedule generation algorithm needs to be invoked over three and a half million times if all possible job sequences are examined. If a much larger number of jobs is considered like $k = 40$, much greater computation effort is required ($40! = 8.16 \times 10^{47}$ possible sequences) (Yang, 2001). Therefore, it may be impractical for the conventional methods such as shortest-path method ([Jensen and Barnes, 1980]) or dynamic programming to effectively solve these types of complicated job sequences. Therefore, other techniques for job sequence generation are required, such as GA-based approach.

With GA-based approach, we can generate all possible schedules with complete job sequence, and thus avoid unnecessary computational efforts for the incomplete job sequences generated. Secondly, the evaluation of a job sequence using GA approach is based on the actual performance of the sequence instead of any estimate as with tree search methods (Gen and Cheng, 2000), which makes the quality of evaluation using GA approach being better.

For much more improvement of the search ability of GA approach to the optimal solution, an adaptive scheme for GA parameters and a local search technique can be incorporated into GA loop, respectively.

With these schemes, we develop the ahGA approach to effectively solve the multistage-based scheduling problem using the mathematical formulation shown in Section 3. First, a new gene representation method and genetic operators of GA to represent the multistage-based scheduling problem are used. Secondly a local search and an adaptive scheme are proposed, respectively.



[Figure 4] An example of state permutation encoding

4.1 Gene Representation

For the multistage-based scheduling problem shown in [Figure 3], the alternative states at each stage can be expressed by a series of integers to indicate the state. If a state for an operation is chosen at one stage, its corresponding integer can be assigned where the integer is within the number of possible states at that stage. Therefore, the multistage-based scheduling problem can be concisely encoded into a state permutation format by concatenating all the set states of stages.

This state permutation encoding has 1-to-1 mapping for the problem and easy to encode and evaluate. Several advantages using state permutation encoding in multistage-based problems were mentioned in the ref (Gen and Cheng, 2000).

As to the initial population of GA for multistage-based scheduling problem, each individual is represented as a permutation with n-1 integers where the integers are generated randomly with the number of all possible states in the corresponding stage. [Figure 4] shows an example of state permutation encoding using the simple scheduling problem in [Figure 3].

In [Figure 4], the bold lines mean that the operations on the corresponding workstations are processed. This can be presented as a series of state permutation $J_1(2, 2, 3) \rightarrow J_2(1, 2) \rightarrow J_3(3, 3, 2)$ as shown in [Figure 4]. Therefore, we can adapt this encoding scheme for the gene representation of GA. Using the state permutation encoding, a feasible schedule can be located. Let $(J_k, o_i)/(W_j : t^s_{ijk}, t^e_{ijk})$ denotes that operation i of job k is scheduled on workstation j from its start time t^s_{ijk} to its end time t^e_{ijk}. Finally the following feasible schedule can be produced.

$$\begin{aligned} S_3 &= S_{31} \rightarrow S_{32} \rightarrow S_{33} = < J_1 \rightarrow J_2 \rightarrow J_3 > \\ &= < (J_1, o_1)/(W_2 : 0,26), (J_1, o_3)/ \\ &\quad (W_2 : 26,28), (J_1, o_2)/(W_3 : 68,95) \\ &\rightarrow (J_2, o_2)/(W_1 : 0,20), (J_2, o_5)/(W_2 : 68,97) \\ &\rightarrow (J_3, o_6)/(W_3 : 0,29), (J_3, o_4)/ \\ &\quad (W_3 : 29,63), (J_3, o_1)/(W_2 : 97,123) > \end{aligned} \quad (6)$$

The schedule S of the equation (6) means that the operation 1 of job 1 can be processed on workstation 2 from its starting time 0 to its ending time 26, and then the operation 3 of job 1 is processed on workstation 2 from 26 to 68. The remainders for job 2 and job 3 are interpreted as a same way. Using equation (5), we can confirm that the T_M, T_F, and P_T are 123, 315(= 95+97+123), and 491(= (95−90) × 10+(97−90) × 11+(123−95) × 13), respectively.

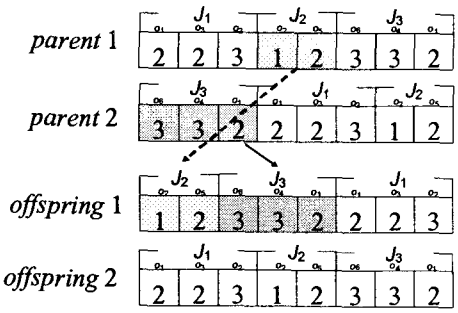
4.2 Combination Crossover

The combinat]ion crossover (CC) operator is used here, which simply selects one position (job) at random for a pair of parents and then combines their contents (jobs). The CC procedure is shown as follow :

procedure 1 : CC operator
begin
$i \leftarrow 0$;
while (i = *pop_size* × rates of crossover (P_c)) do
select two parents (individuals) randomly in current population ;

randomly pick up a job (j_2) in parent 1 and
  a different job (j_3) in parent 2 ;
place the selected two jobs and the first re-
  mainder job (j_1) of parent 1 in offspring 1 ;
place the unselected jobs (j_1 and j_2) in parent
  2 and the job (j_3) of parent 1 in offspring 2 ;
  $i \leftarrow i+1$;
  end
end

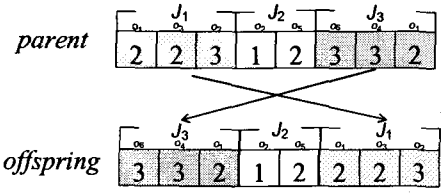
[Figure 5] shows an example of the CC operator using the individuals resulting from the state permutation encoding of [Figure 4]. For the selected two parents from population in [Figure 5], the job j_2 of parent 1 and job j_3 of parent 2 are randomly selected, and then two new offspring are generated, after performing the CC operator.



[Figure 5] An example of CC operator

4.3 Swap Mutation

The swap mutation (SM) operator is used here, which simply selects two positions (jobs) at random and swaps their contents as shown in [Figure 6]



[Figure 6] An example of SM operator

The SM procedure is shown as follow :

procedure 2 : SM operator
begin
  i ← 0;
  while ($i = pop\_size \times$ rate of crossover (P_M)) do
    select a individual randomly ;
    pick up two jobs randomly ;
    exchange their positions ;
    $i \leftarrow i+1$;
  end
end

4.4 Combining Technique by Local Search

Local search technique seeks improved solutions by searching in the neighborhood of an incumbent solution (Ishibuch and Murata, 1998). The implementation of local search technique requires an initial incumbent solution, the definition of a neighborhood for an incumbent solution, and a method for choosing the next incumbent solution.

The idea locating an improved solution by a small change can be used in mutation operator. Observing the SM operator, the individual generated by pair-wise interchange can be viewed as a neighbor of the original individual. The neigh-

borhood of an individual is then defined as a set of individuals generated by such pair-wise interchange. For a pair of genes, one is called as the pivot which is fixed for a given neighborhood and the others are selected randomly as shown in [Figure 7].

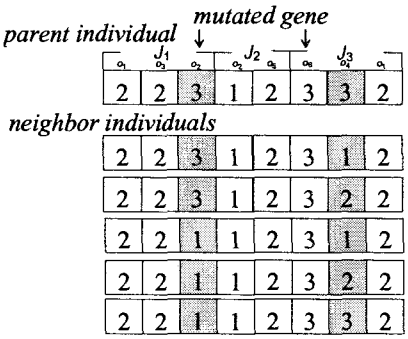
For a given neighborhood, an individual is called a local optimum if it is better than any other individuals with respect to their fitness values. The size of a neighborhood affects the quality of the local optimum.

There is a clear trade-off between small and large neighborhoods: if the number of neighbors is larger, the probability of locating a good neighbor may be higher, but its search takes more time. The procedure of local search technique is as follow :

procedure 3 : local search technique
begin
$i \leftarrow 1$;
let P(parent) be current incumbent individual ;
select a pivot gene randomly in the neighborhood of P ;
while (i < *specified_number*) do
pick up the gene ;
evaluate the neighbors toward both sides until the left and right limits of the gene ;
let a neighbor (offspring) be current incumbent if it is better than the previous one ;
$i \leftarrow i+1$;
end
save the current incumbent ;
end

4.5 Adaptive Scheme for GA Parameters

Basic logic of adaptive scheme is to enhance the performance of the search by adaptively regulating GA parameters during genetic search process, and its implementation is as follow :



[Figure 7] Incumbent individuals and its neighborhood

Whenever a new offspring is added to the population, a pointer is established for the genetic operator that generates the offspring. A check is then made to determine if the fitness of the offspring is better or worse than its parents. The percentage of improvement or degradation is recorded, and this record is reserved for later adjustments of the occurrence rates of GA operators.

In this paper, we use two operators of GA, CC and SM. That is, the occurrence rates of the two operators are adaptively regulated by adaptive scheme during the genetic search process. For this logic, we use the concept of Mak et al. (2000). They employed the fitness values of parent and offspring at each generation in order to construct

adaptive crossover and mutation operators. By this concept, we adapt the two operators, the CC and SM proposed in Sections 4.2 and 4.3. The adapting strategy is as follow :

This scheme increases the occurrence rates of the two operators, if it consistently produces a better offspring during genetic search process; however, it also reduces the occurrence rates of the operators, if it produces a poorer offspring. This is based on the fact that it encourages the well-performing operators to produce more offspring, while also reducing the chance for the poorly performing operators to destroy the potential individuals during genetic search process. The detailed procedure for minimization problem is as follows :

procedure 4 : Regulation of the rates of CC and SM operators

begin

if $\left(\overline{f_{par\_size}}(t)/\overline{f_{off\_size}}(t)\right)-1 \geq 0.1$ **then**

$p_C(t+1)=p_C(t)+0.01$,

$p_M(t+1)=p_M(t)+0.005$;

if $\left(\overline{f_{par\_size}}(t)/\overline{f_{off\_size}}(t)\right)-1 \leq 0.1$ **then**

$p_C(t+1)=p_C(t)-0.01$,

$p_M(t+1)=p_M(t)-0.005$;

if $-0.1<\left(\overline{f_{par\_size}}(t)/\overline{f_{off\_size}}(t)\right)-1<0.1$ **then**

$p_C(t+1)=p_C(t)$, $p_M(t+1)=p_M(t)$;

end

end

where par_size and off_size are the parent size and offspring size satisfying constraints, respectively. $\overline{f_{par\_size}}(t)$ and $\overline{f_{off\_size}}(t)$ are respectively the average fitness values of parents and offspring at generation t. $p_c(t)$and $p_M(t)$ are the rates of CC and SM operators at generation t respectively. In the cases of $\left(\overline{f_{par_size}}(t)/\overline{f_{off_size}}(t)\right)-1 \geq 0.1$and $\left(\overline{f_{par\_size}}(t)/\overline{f_{off\_size}}(t)\right)-1 \leq -0.1$, the adjusted rates should not exceed the range from 0.0 to 1.0 for $p_c(t+1)$ and $p_M(t+1)$. The above-stated procedure is evaluated in all generations during genetic search process, and the occurrence rates of CC and SM operators are adaptively regulated according to the result of the procedure.

4.6 Overall Procedure of the ahGA

By the logics and schemes suggested from Section 4.1 to 4.4, the overall procedure of the ahGA is proposed in the following steps.

Step 1 : Initialization

Randomly generate an initial set of m individuals by state permutation encoding.

Step 2 : Initial evaluation

Evaluate all individuals and calculate their fitness.

Step 3 : Selection

Elitist selection strategy (Gen and Cheng, 1997) is adapted.

Step 4 : CC operator (procedure 1)

Apply CC operator to the population generated by Steps 1 and 2.

Step 5 : SM operator (procedure 2)

Apply SM operator to the population result-

ing from the CC operator.

Step 6 : Local search technique (procedure 3)
Apply local search technique to the population resulting from the SM operator.

Step 7 : Evaluation
The fitness evaluation of offspring is implemented by the three objective functions suggested in equations (1), (2), and (3). Select the optimal total penalty out of minimized total flow time in minimized makespan. The optimal schedules may be revised during genetic search process.

Step 8 : Termination condition
If one of the individuals achieves the pre-defined maximum fitness, then stop and return the best individual. Otherwise, generate a new selection set.

Step 9 : Adaptive regulation of GA parameters (procedure 4). Therefore, the rates in CC and SM operations are regulated by the adapting strategy. Go to Step 3.

5. Numerical Experiment

In numerical experiment, we use a simple size of scheduling problem and much more large-scaled scheduling problem in FMS environment. These two problems are expressed by multi-stage-based forms and are then solved by the procedure of the ahGA in Section 4. For experimental comparison under a same condition, we set the parameters of the ahGA as follows: maximum generation number is 1,000, Population size is 20, Initial p_c is 0.3, Initial p_M is 0.3.

The value of p_c and p_M are adaptively regulated during genetic search process by the adaptive scheme used in the ahGA. Altogether, 30 iterations are executed to eliminate the randomness of the search.

The procedure of the ahGA is programmed and implemented in JAVA language under IBM-PC Pentium 1.4 GHz computer with 512MB RAM.

5.1 Example 1

<Table 3> shows the processing times of each operation on workstations as a simple size of scheduling problem in FMS environment. This problem consists of five workstations denoted as $W_1,\cdots, W_5$. Collectively, these workstations can perform five operations $o_1,\cdots,o_5$. The required operations for each job with the precedence requirements are listed in <Table 4>.

Using <Tables 3> and <Tables 4>, the ahGA was implemented and its optimal schedule is shown in <Table 5>, and the optimal schedule can be also represented as a Gantt chart form in [Figure 8].

To prove the efficiency of the ahGA, we also implemented the three conventional algorithms (shortest average processing time t_{SAP}, discrete dynamic programming (DDP), GA-based discrete dynamic programming (GA-DDP)) (Yang, 2001) and the GA without the local search and adaptive scheme in the ahGA into the same scheduling problem shown in <Tables 3> and <Tables

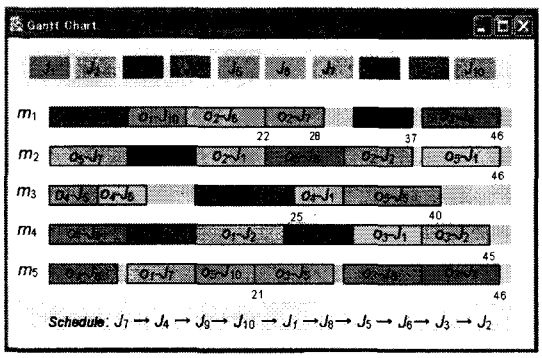
4>. The computational results using the ahGA, GA and the three conventional algorithms are listed in <Table 5>.

<Table 3> Processing times of operations in each workstation for Example 1

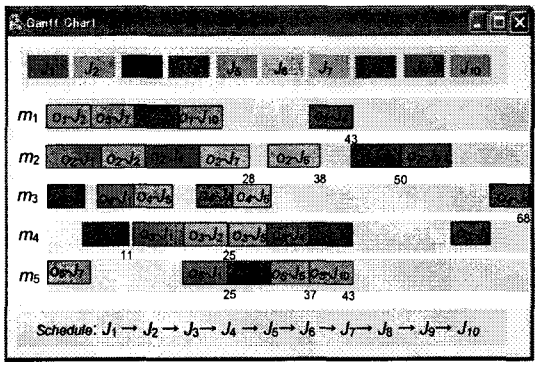
Workstations	Operations				
	o_1	o_2	o_3	o_4	o_5
W_1	6	8	-	10	-
W_2	-	7	12	-	8
W_3	9	-	-	5	10
W_4	-	9	7	8	-
W_5	7	-	8	-	6
Average	7.33	8.00	9.00	7.67	8.00

<Table 4> Job information for Example 1

Job No.	Required operations	t_k^{ATP}	t_k^D	c_k^{TP}
J_1	$o_2 \rightarrow o_4 \rightarrow o_3 \rightarrow o_5$	32.67	50	10
J_2	$o_1 \rightarrow o_2 \rightarrow o_3$	24.33	50	10
J_3	$o_4 \rightarrow o_3$	16.67	50	10
J_4	$o_2 \rightarrow o_4 \rightarrow o_3 \rightarrow o_1$	32.00	50	10
J_5	$o_4 \rightarrow o_3 \rightarrow o_5$	24.67	50	10
J_6	$o_4 \rightarrow o_2$	15.67	55	10
J_7	$o_5 \rightarrow o_1 \rightarrow o_2$	23.33	55	10
J_8	$o_1 \rightarrow o_5 \rightarrow o_3 \rightarrow o_2$	32.33	55	10
J_9	$o_2 \rightarrow o_3 \rightarrow o_4$	24.67	55	10
J_{10}	$o_1 \rightarrow o_5$	15.33	55	10



[Figure 8] Gantt chart using ahGA



[Figure 9] Gantt chart using t_{SAP} algorithm

<Table 5> The results using ahGA, GA and three conventional algorithms

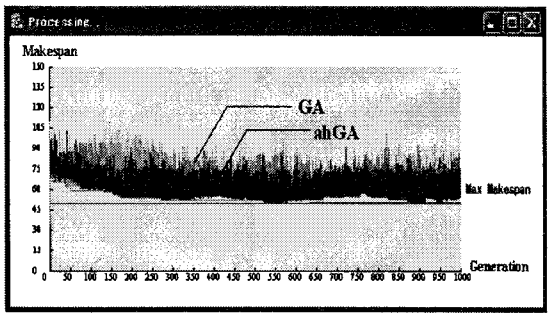
	t_{SAP}	DDP	GA-DDP	GA	Proposed ahGA
T_M	68	59	51	50	46
T_F	368	365	359	357	356

Using the ahGA, [Figure 8] shows the Gantt chart which is scheduled a profile $<J_7 \rightarrow J_4 \rightarrow J_9 \rightarrow J_{10} \rightarrow J_1 \rightarrow J_8 \rightarrow J_5 \rightarrow J_6 \rightarrow J_3 \rightarrow J_2>$ with the best makespan ($T_M = 46$) and $T_F = 356$ ($= 28+37+25+21+46+46+40+22+46+45$). Using t_{SAP} algorithm, [Figure 9] shows Gantt chart with $T_M = 68$ and $T_F = 368$ ($= 25+25+11+43+37+38+28+50+68+43$).

In <Table 5>, the $T_M$and T_F of the GA and ahGA outperforms those of t_{SAP}, DDP and GA-DDP.

In comparison between the GA and ahGA, the latter both with the local search and adaptive scheme is slightly better than the former without any local search and adaptive scheme, which mean that the local search and adaptive scheme

used in the ahGA are well controlling the search to the optimal solution. This also implies that the ahGA is well regulating the trade-off between exploration and exploitation during genetic search process rather than the GA does. For proving the search ability of the adaptive scheme used in the ahGA, we compared the behaviors of the convergence process of the makespan in the GA and ahGA during their genetic search processes. [Figure 10] shows the behaviors in the GA and ahGA. The behaviors of the ahGA have lower makespan than those of the GA, which means that the adaptive scheme and the local search technique used in the ahGA are well controlling the search rather than the GA does.



[Figure 10] Behaviors of the makespans in GA and ahGA

5.2 Example 2

For more large-scaled scheduling problem, we consider various environments. <Table 6> shows the processing times for various operations on 15 workstations as well as the relevant operation-related data. Detailed job information (required operation sequence, t_k^{ATP}, t_k^D and C_k^{TP}) for each job on 5, 10, and 15 workstations are appeared in Appendix A. The ahGA was implemented using the information of <Table 6> and Appendix A, and its optimal schedule for the 10 jobs and 5 workstations is shown in [Figure 11].

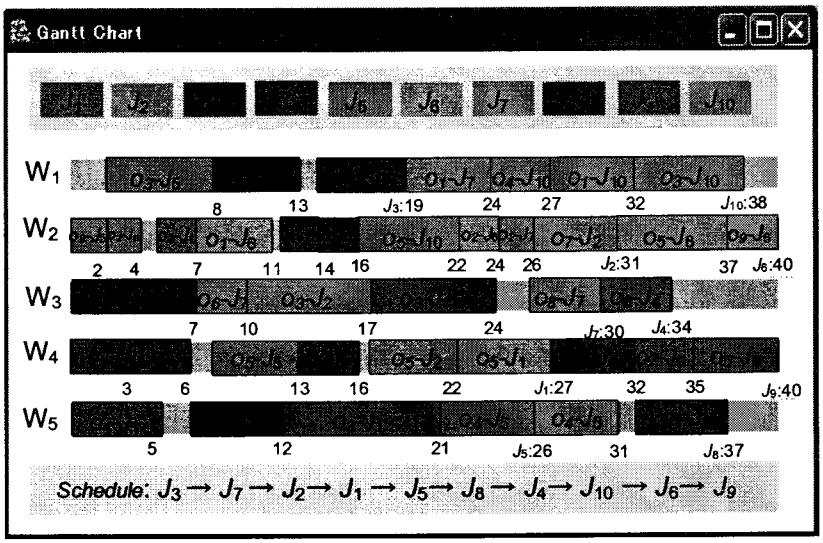
In [Figure 11], the job sequence for optimal schedule is as follow :

$$J_3 \rightarrow J_7 \rightarrow J_2 \rightarrow J_1 \rightarrow J_5 \rightarrow J_8 \rightarrow J_4 \rightarrow J_{10} \rightarrow J_6 \rightarrow J_9 \quad (7)$$

The values of multi-objectives are $T_M =$ 40, $T_F = 322$ $(19+30+31+27+26+37+34+38+40+40=322)$ and $P_T = 0$. To compare the performance between the GA and ahGA, we tested the two algorithms under various environments of GA parameters. <Table 7> shows the optimal results of T_M, T_F, and P_T, after 30 times running

<Table 6> Processing times of operation in each workstation for Example 2

Workstation	Operations								
	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9
W_1	5	-	6	3	-	4	-	3	-
W_2	4	2	-	-	6	-	5	-	3
W_3	-	-	7	6	-	3	-	4	-
W_4	3	3	-	-	5	-	3	-	4
W_5	-	-	9	5	-	5	-	3	-
W_6	4	3	-	2	4	-	6	-	5
W_7	-	-	6	-	-	4	-	5	-
W_8	3	4	-	4	-	2	-	-	5
W_9	-	-	8	5	3	-	4	6	-
W_{10}	5	5	-	-	2	5	-	-	6
W_{11}	-	5	-	4	-	5	-	3	-
W_{12}	3	-	9	-	6	-	5	-	4
W_{13}	-	4	-	5	-	3	-	4	-
W_{14}	4	-	6	-	5	-	3	-	4
W_{15}	3	2	-	6	3	4	-	5	3
Average	3.78	3.50	7.29	4.44	4.25	3.89	4.33	4.13	4.25



[Figure 11] Gantt chart for 10 jobs and 5 workstations in ahGA

for each GA parameter setting in the GA and ahGA.

In <Table 7>, the performance of the ahGA is better than that of the GA in all the comparisons, which implies that using the local search technique and adaptive schemes in the ahGA is

<Table 7> Result of Example 2

J_k	W_j	Test No.	Max_ gen.	Pop_ size	GA			Proposed ahGA		
					T_M	T_F	P_T	T_M	T_F	P_T
10	5	1	500	20	44	613	230	42	597	150
		2	1000	25	44	611	210	41	607	180
		3	1500	30	43	609	190	40	603	160
20	10	1	1000	40	48	679	320	45	968	220
		2	1500	45	48	669	280	44	945	240
		3	2000	50	47	667	240	43	936	200
30	15	1	1500	45	62	1620	380	58	1559	320
		2	2000	50	61	1579	350	57	1538	310
		3	2500	55	60	1554	340	56	1487	290
40	15	1	2000	55	44	613	230	42	597	150
		2	3000	60	44	611	210	41	607	180
		3	4000	65	43	609	190	40	603	160

a better choice when we want to improve the performance of GA under the complicated scheduling problems of FMS environment.

6. Conclusion

In this paper, we have developed a new adaptive hybrid genetic algorithm (ahGA) for effectively solving the multistage-based scheduling problems in FMS environment. The proposed ahGA consists of the procedures of a state permutation encoding, two GA operatiors (combination crossover and swap mutation operators), local search technique and adaptive scheme for GA parameters. For various comparisons with the proposed ahGA, three conventional algorithms and the GA without the local search technique and adaptive scheme in the ahGA have been tested in numerical experiments.

Based on the various analyses, we can conclude that our proposed ahGA is more efficient and more flexible in locating optimal solution than the other competing algorithms.

References

[1] Basu, A., "Perspectives on operations research in data and knowledge management", *European Journal of Operational Research*, Vol.111(1998), 1~14.

[2] Blake, C. L. and C. J. Merz, UCI Repository of Machine Learning Databases [http://www.ici.uci.edu/mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA, (1998).

[3] Bradley, P., U. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases", *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining,* (1998), 9~15.

[4] Bradley, P., J. Gehrke, R. Ramakrishnan, and R. Srikant, "Scaling mining algorithms to large databases", *Communications of the ACM*, Vol.45, No.8(2002), 38~43.

[5] Bradley, P. S., Mangasarian, O. L., and Street, W. N., "Feature selection via mathematical programming", *INFORMS Journal on Com- puting*, Vol.10(1998), 209~217.

[6] Chauchat, J- H. and R. Rakotomalala, "Sampling strategies for building decision trees from very large databases comprising many continuous attributes", In Liu and Motada (eds.) Instance Selection and Construction for Data Mining, Kluwer (2001).

[7] Domingo, C. Gavalda R., and Watanabe, R., "Adaptive Sampling Methods for Scaling Up Knowledge discovery", *Data Mining and Knowledge Discovery*, Vol.6, No.2(2002), 131~152.

[8] Estevill-Castro, V., "Why so many clustering algorithms", *SIGKDD Explorations*, Vol.4, No.1(2002), 65~75.

[9] Ester, M., H.-P. Kriegel, and X. Xu, "Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification", *Proceedings of the 4th International Symposium of Large Spatial Databases,* (1995), 67~82.

[10] Farnstrom, F., J. Lewis, and C. Elkan, "Scalability for clustering algorithms revisited", *SIGKDD Explorations*, Vol.2, No.1 (2000), 51~57.

[11] Forman, G. and B. Zhang, "Distributed data clustering can be efficient and exact", *SIG-*

KDD Explorations, Vol.2, No.2(2000), 34~38.

[12] Guha, S. R. Rastogi, and K Shim, "CURE : An efficient clustering algorithm for large databases", *Proceedings of the ACM SIGMOD International Conference on Management of Data,* (1998), 73~84.

[13] Grabmeier, J. and A. Rudolph, "Techniques of cluster algorithms in data mining", *Data Mining and Knowledge Discovery*, Vol.6 (2002), 303~360.

[14] Jain, A. K., Murty, M. N., and Flynn, P. J., "Data clustering: a review", *ACM Computing Surveys*, Vol.31(1999), 264~323.

[15] John, G. and P. Langley, "Static versus dynamic sampling for data mining", *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* (1996), 367~370.

[16] Kaufman, L. and P. J. Rousseeuw, *Finding Groups in Data : An Introduction to Cluster Analysis*. John Wiley & Sons, New York. (1990).

[17] Kim, J. M., "Optimization under uncertainty with application to data clustering", Ph. D. Diss., Dept. of IMSE, Iowa State University (2002).

[18] Kiven, J. and H. Mannila, "The power of sampling in knowledge discovery", *ACM Symposium on Principles of Database Theory,* (1994), 77~85.

[19] Liu, H. and H. Motoda, *Instance Selection and Construction for Data Mining*, Kluwer, (2001).

[20] Ng, R. T. and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining", *Proceedings of 20th International Conference on Very Large Data Bases,* (1994).

[21] Ólafsson, S., "Iterative ranking-and-selection for large-scale optimization", *Proceedings of the Winter Simulation Conference,* (1999), 479~485.

[22] Ólafsson, S., "*Improving scalability of e-commerce systems with knowledge discovery*", In Prabu, Kumara and Kamath (eds.) Scalable Enterprise System-An Introduction to Recent Advances, Kluwer, (2003).

[23] Provost, F., D. Jenson, and T. Oates, "Efficient progressive sampling", *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining* (1999), 23~32.

[24] Provost, F. and V. Kolluri, "A survey of methods for scaling up inductive algorithms", *Data Mining and Knowledge Discovery*, Vol.3(1999), 131~169.

[25] Shi, L. and S. Ólafsson, "Nested partitions method for global optimization", *Operations Research*, Vol.48(2000), 390~407.

[26] Toivonen, H., "Sampling large databases for association rules", *Proceedings of the 22nd International Conference on Very Large Databases,* (1996). 134~145.

[27] Yim D. S. and H. S. Oh, "Application of Genetic and Local Optimization Algorithms for Object Clustering Problem with Similarity Coefficients", *Journal of the Korean Institute of Industrial Engineers*, Vol.29, No.1(2003), 90~99.

[28] Zhang, T., R. Ramakrishnan, and M. Livny, "BIRCH : An efficient data clustering method for very large databases", *Proceedings of the ACM SIGMOD International Conference on Management of Data,* (1996), 103~114.

<Appendix A. Job information for Example 2>

No. of Jobs	Required operations	t_k^{ATP}			t_k^D	t_k^{IP}
		$5W_S$	$10W_S$	$15W_S$		
J_1	$O_4 \rightarrow O_2 \rightarrow O_3 \rightarrow O_5$	20.00	18.77	19.48	35	20
J_2	$O_3 \rightarrow O_5 \rightarrow O_7$	16.83	15.70	15.87	35	20
J_3	$O_3 \rightarrow O_4 \rightarrow O_2 \rightarrow O_1$	18.50	18.77	19.01	35	20
J_4	$O_1 \rightarrow O_3 \rightarrow O_8$	14.67	15.40	15.20	35	20
J_5	$O_2 \rightarrow O_3 \rightarrow O_5 \rightarrow O_4$	20.00	18.77	19.48	35	20
J_6	$O_1 \rightarrow O_2 \rightarrow O_4 \rightarrow O_5 \rightarrow O_9$	20.17	20.17	20.22	40	20
J_7	$O_6 \rightarrow O_1 \rightarrow O_2 \rightarrow O_8$	13.83	15.43	15.30	40	20
J_8	$O_1 \rightarrow O_2 \rightarrow O_5 \rightarrow O_4$	16.67	15.57	15.97	40	20
J_9	$O_2 \rightarrow O_1 \rightarrow O_7 \rightarrow O_5$	16.00	15.90	15.86	40	20
J_{10}	$O_2 \rightarrow O_5 \rightarrow O_4 \rightarrow O_1 \rightarrow O_3$	24.00	22.77	23.26	40	20
J_{11}	$O_2 \rightarrow O_4 \rightarrow O_8 \rightarrow O_6$	14.50	15.60	15.96	40	15
J_{12}	$O_1 \rightarrow O_2 \rightarrow O_5 \rightarrow O_3$	19.33	18.60	18.82	40	15
J_{13}	$O_4 \rightarrow O_9 \rightarrow O_6 \rightarrow O_3 \rightarrow O_5$	25.00	23.80	24.12	40	15
J_{14}	$O_2 \rightarrow O_4 \rightarrow O_3 \rightarrow O_1$	18.50	18.77	19.01	40	15
J_{15}	$O_7 \rightarrow O_3 \rightarrow O_8 \rightarrow O_5$	20.17	19.90	20.00	40	15
J_{16}	$O_4 \rightarrow O_3 \rightarrow O_2 \rightarrow O_1$	18.50	18.77	19.01	45	15
J_{17}	$O_5 \rightarrow O_7 \rightarrow O_9$	13.00	13.10	12.83	45	15
J_{18}	$O_1 \rightarrow O_5 \rightarrow O_3 \rightarrow O_2$	19.33	18.60	18.82	45	15
J_{19}	$O_2 \rightarrow O_3 \rightarrow O_4$	14.50	14.77	15.23	45	15
J_{20}	$O_1 \rightarrow O_5 \rightarrow O_8$	12.83	12.20	12.16	45	15
J_{21}	$O_5 \rightarrow O_2 \rightarrow O_3 \rightarrow O_4$	20.00	18.77	19.48	50	10
J_{22}	$O_1 \rightarrow O_3 \rightarrow O_2$	13.83	14.60	14.57	50	10
J_{23}	$O_5 \rightarrow O_9 \rightarrow O_1$	13.00	12.60	12.28	50	10
J_{24}	$O_1 \rightarrow O_5 \rightarrow O_2 \rightarrow O_3$	19.33	18.60	18.82	50	10
J_{25}	$O_2 \rightarrow O_7 \rightarrow O_3$	13.83	15.10	15.12	50	10
J_{26}	$O_5 \rightarrow O_1 \rightarrow O_2$	12.00	11.40	11.53	55	10
J_{27}	$O_5 \rightarrow O_8 \rightarrow O_4 \rightarrow O_3$	20.83	19.57	20.11	55	10
J_{28}	$O_1 \rightarrow O_7 \rightarrow O_3$	15.33	15.70	15.40	55	10
J_{29}	$O_4 \rightarrow O_1 \rightarrow O_5 \rightarrow O_2$	16.67	15.57	15.97	55	10
J_{30}	$O_1 \rightarrow O_4 \rightarrow O_9 \rightarrow O_5$	17.67	16.77	16.72	55	10
J_{31}	$O_2 \rightarrow O_5 \rightarrow O_1$	12.00	11.40	11.53	60	5
J_{32}	$O_4 \rightarrow O_6 \rightarrow O_5 \rightarrow O_3$	21.50	19.20	19.87	60	5
J_{33}	$O_3 \rightarrow O_1 \rightarrow O_5$	16.83	15.20	15.32	60	5
J_{34}	$O_3 \rightarrow O_4 \rightarrow O_7 \rightarrow O_2$	18.50	19.27	19.56	60	5
J_{35}	$O_5 \rightarrow O_3 \rightarrow O_1$	16.83	15.20	15.32	60	5
J_{36}	$O_3 \rightarrow O_2 \rightarrow O_9 \rightarrow O_4$	18.00	19.37	19.48	65	5
J_{37}	$O_5 \rightarrow O_7 \rightarrow O_3 \rightarrow O_2$	19.33	19.10	19.37	65	5
J_{38}	$O_3 \rightarrow O_2 \rightarrow O_1$	13.83	14.60	14.57	65	5
J_{39}	$O_4 \rightarrow O_1 \rightarrow O_2 \rightarrow O_5$	16.67	15.57	15.97	65	5
J_{40}	$O_5 \rightarrow O_3 \rightarrow O_8 \rightarrow O_2$	18.67	18.80	19.17	65	5

요약

FMS환경에서 다단계 일정계획문제를 위한 적응형혼합유전 알고리즘 접근법

윤영수* · 김관우**

본 논문에서는 유연제조시스템(FMS)에서 다단계스케쥴링 문제를 효율적으로 해결하기 위한 적응형혼합유전 알고리즘(ahGA) 접근법을 제안한다. 제안된 ahGA는 FMS의 해를 개선시키기 위하여 이웃탐색기법을 사용하며, 유전탐색과정에서의 수행도를 향상시키기 위해 유전알고리즘(GA)의 파라메터들을 조정하기 위한 적응형 구조를 사용한다. 수치실험에서는 제안된 ahGA와 기존의 알고리즘들 간의 수행도를 비교하기 위하여 두가지형태의 다단계스케쥴링문제를 제시한다. 실험결과는 제안된 ahGA가 기존의 알고리즘들 보나 더 뛰어난 수행도를 보여주고 있다.

* 조선대학교 경영학부
** 한양대학교 산업공학과