J Intell Manuf (2006) 17:509–522
DOI 10.1007/s10845-005-0010-0

Adaptive genetic algorithm for advanced planning in manufacturing supply chain

Chiung Moon · Yoonho Seo · Youngsu Yun · Mitsuo Gen

Received: June 2005 / Accepted: December 2005

Abstract A main function for supporting global objectives in a manufacturing supply chain is planning and scheduling. This is considered such an important function because it is involved in the assignment of factory resources to production tasks. In this paper, an advanced planning model that simultaneously decides process plans and schedules was proposed for the manufacturing supply chain (MSC). The model was formulated with mixed integer programming, which considered alternative resources and sequences, a sequence-dependent setup and transportation times. The objective of the model was to analyze alternative resources and sequences to determine the schedules and operation sequences that minimize makespan. A new adaptive genetic algorithm approach was developed to solve the model. Numerical experiments were carried out to demonstrate the efficiency of the developed approach.

C. Moon
Department of Information & Industrial Engineering,
Hanyang University, Ansan 425-791, Korea

Y. Seo (✉)
Department of Industrial Systems & Information
Engineering, Korea University, #1 Anamdong 5-ga,
Sungbukku, Seoul 136-713, Korea
e-mail: yoonhoseo@korea.ac.kr

Y. Yun
School of Business Administration,
Chosun University, Gwangju 501-759, Korea

M. Gen
Graduate School of Information,
Production & Systems, Waseda University,
Kitakyushu, Japan

Keywords Advanced planning · Manufacturing supply chain · Scheduling · Adaptive genetic algorithm

Introduction

Many manufacturing enterprises are developing into global chains covering multiple manufacturing sites and consisting of suppliers, fabrication and assembly shops, as well as outsourcing entities. The manufacturing supply chain (MSC) tries to optimize the total system to cope with global manufacturing. Increasingly, enterprises are being organized as multiple plant chains of different units. For that reason, planning and scheduling activities are very complex, and have to take place within the enterprise and across the entire supply chain in order to achieve high quality products at lower cost, lower inventory and high levels of performance. As a result, to efficiently provide global optimal solutions, manufacturing enterprises are migrating from separated planning processes toward more coordinated and integrated planning processes. However, state-of-the-art solutions do not effectively manage this planning and scheduling. Additionally, the transportation associated with manufacturing processes in several different plants and outsourcing plants becomes an important and practical issue (Guinet, 2001; Lutz, Helms, & Wiendahl, 1999; Moon, Kim, & Gen, 2004; Vercellis, 1999).

This paper presents an advanced planning model in a MSC to provide realistic process plans and schedules for manufacturing entities. The model was developed to determine a global optimal schedule of machine assignments and operation sequences of customer orders that minimizes the total processing time. The model treats one of the most important issues for supporting

management goals because the function takes part in the assignment of factory resources to production tasks. In addition, it is important that customer orders are met on time.

There has been some research completed for advanced planning. Tan (2000) presented a review of the research in the integrated process planning and scheduling area and discussed the extent of applicability of various approaches. Palmer (1996), Brandemarte and Calderini (1995) presented several operational planning models based on material resource planning (MRP) concepts. Saygin and Kilic (1999) and Morad and ZalZala (1999) proposed several operational planning models with the objective of reducing the completion time.

More recently, Thoney, Hodgson, King, and Taner (2002) developed a heuristic procedure for flexible scheduling, including inter-plant transportation. They considered batch processing, which includes both internal and external transportation operations. Moon, Kim, and Hur (2002) proposed an integrated process planning and scheduling model to minimize total tardiness through analysis of the alternative resource selection and the operation sequences in MSC using mixed integer programming. They developed a genetic algorithm (GA) approach to solve the model. Cochran, Horng, and Fowler (2003) proposed a two-stage multi-population GA to solve the parallel resource scheduling problems. Guinet (2001) formulated a production planning problem appearing as a network program solved with a primal-dual approach. Tan and Khoshnevis (2004) presented a linearized polynomial mixed integer programming model for the integration of process planning and scheduling problem. Moon and Seo (2005) developed an operational planning and scheduling model for a multi-plant composed of a network of production facilities, and of multiple products flow through manufacturers. They also developed a GA based heuristic approach to solve the model.

The models introduced above consider alternative machines for each operation with a fixed sequence or with a non-constraint operational sequence when generating a schedule. When assigning resources to orders, some models assume infinite resource capacities on the shop floor, a common due date for all orders, and an idle plant. As a result, the schedules may end up with an infeasible task, even before the start of manufacturing.

In this paper, an advanced planning problem is considered which minimizes the makespan for a MSC, and that is composed of many production facilities with outsourcings and with multiple product flows through manufacturers. The advanced planning problem includes a range of capabilities, from finite-capacity scheduling at the shop floor level, through constraint-based planning, to the advanced planning in a manufacturing supply chain. The model is formulated with mixed integer programming which considers alternative machines and sequences, a sequence-dependent setup and transportation times. It was assumed that the alternative machines had different capabilities and required unequal processing times for each operation. The operation sequences for each order included precedence constraints.

In order to obtain good solutions for planning and scheduling models, various GA approaches have been developed (Cochran et al., 2003; Moon & Seo, 2005; Moon et al., 2002). However, despite the successful application of various GAs to numerous planning and scheduling optimization problems, the identification of the correct settings of genetic parameters (such as population size, crossover and mutation rates) for these problems is not an easy task. This is mainly because the performance of GAs relies highly on the settings of the parameter values. Therefore, there have been many studies performed to identify the correct settings for these values (Angeline, 1995; Eiben, Hinterding, & Michalewicz, 1999; Fogel, Fogel, & Ohkura, 2001; Herrera & Lozano, 2003; Yun, 2002).

Consequently, this paper developed a new adaptive genetic algorithm (AGA) based on a heuristic approach. The suggested AGA approach has adaptive schemes that can automatically determine the use of the local search technique and adaptively regulate the rates of crossover and mutation operations in a GA; this is done during its search process.

Problem definition

The objective of the advanced planning problem is to determine an optimal schedule with resource selection for assignments, operations sequences, and allocations of variable transfer batches. This optimal schedule minimizes the makespan that consists of processing times, setup times and transportation times.

The advanced planning problem is defined as: given a set of n customer orders which are to be processed using m resources, with alternative operations sequences for orders and alternative resources for operations in the environment of a MSC; find an operation sequence for each order and a schedule in which orders pass through machines, as well as a schedule in which operations on the same orders are processed, such that it satisfies the precedence constraints and is optimal with respect to the minimization of the makespan criterion.

According to the production capacity, location and facility features of a plant; the transportation of material flows between plants in a MSC should be considered,

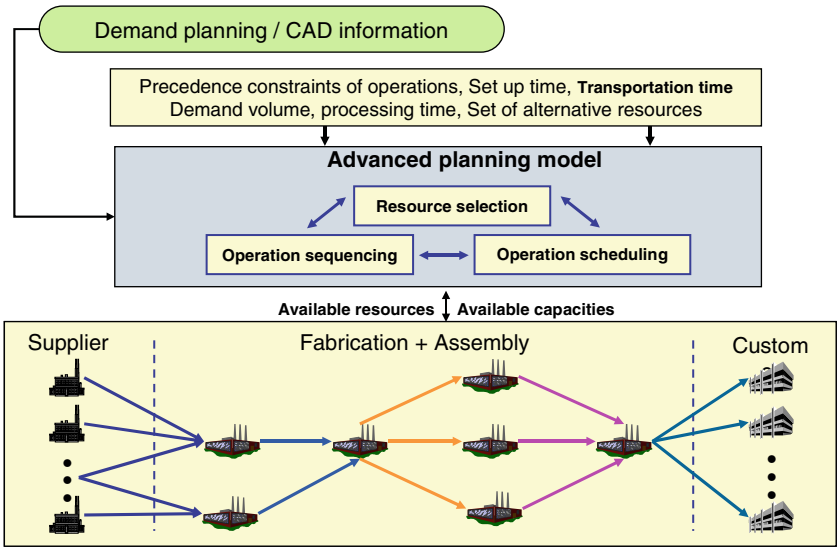


Fig. 1 Schematic diagram of the advanced planning problem

as each of them influences the performance measures such as total machining and transportation times. The resources have finite capacities, different capabilities and require unequal processing times for a specific operation. Alternative resources for an operation can be used to define flexible process plans and schedules for a customer order.

System characteristics are as follows: the system is composed of a network of plants. Transportation and setup times are sequence-dependent and all machines have a distinct initial load level. The schematic diagram of an advanced planning problem for a MSC is illustrated in Fig. 1.

Model development

The advanced planning problem includes the operation sequences that select a resource for each operation and determine the schedules for all parts. Operation sequence problems can be treated as multiple traveling salesman problems, each of which determines the operation sequences for each order. Since the sequence should obey the precedence relations intrinsic to the order, the traveling salesman problem with precedent constraints could be considered to represent the sequencing structure for an order. Additionally, the transition costs between the operations for an order, which correspond to the travel costs between nodes in the original traveling salesman problem, are not given, but should be obtained by solving a resource selection subproblem.

From an unordered set of the operation processes with precedence relations and alternative resources, an advanced planning problem determines a schedule with resource selection and an operation process plan with consideration of the combination of parallel processes and alternative resources for operations and resource constraints. Figure 2 shows an example of the process that determines a schedule with operation selection and operation sequence.

The following notations are used to describe the problem throughout this paper:

k,l	index for order, $k,l = 1,\ldots,K$, where K is the number of orders.
i,j	index for operation, $i,j = 1,\ldots,I$, where I is the number of all operations.
m,n	index for resource, $m,n = 1,\ldots,M$, where M is the number of resources.
R_k	set of precedence relations of two operations in order i, i.e., $R_k = \{(i,j)\|\forall(i,j) =$ pair of operations included in order k where operation i precedes operation $j\}$.
Q_k	set of operation pairs in order k without precedence relation, i.e., $Q_k = \{(i,j)\|\forall(i,j) =$ pair of operations included in order k where operations i and j can be processed in any order$\}$.
N_m	set of operations to be performed on resource m.
p_{kim}	processing time of operation i of order k on resource m.
M	an arbitrary large positive number.
C_m	available capacity in resource m.
t_{mn}	transportation time from resources m to n.
s_{kilj}	setup time required to transfer from operation i of order k to operation j of order l.
q_k	lot size for order k.
u_k	size of unit load for order k.

The variables are introduced to adapt the advanced planning model as follows:

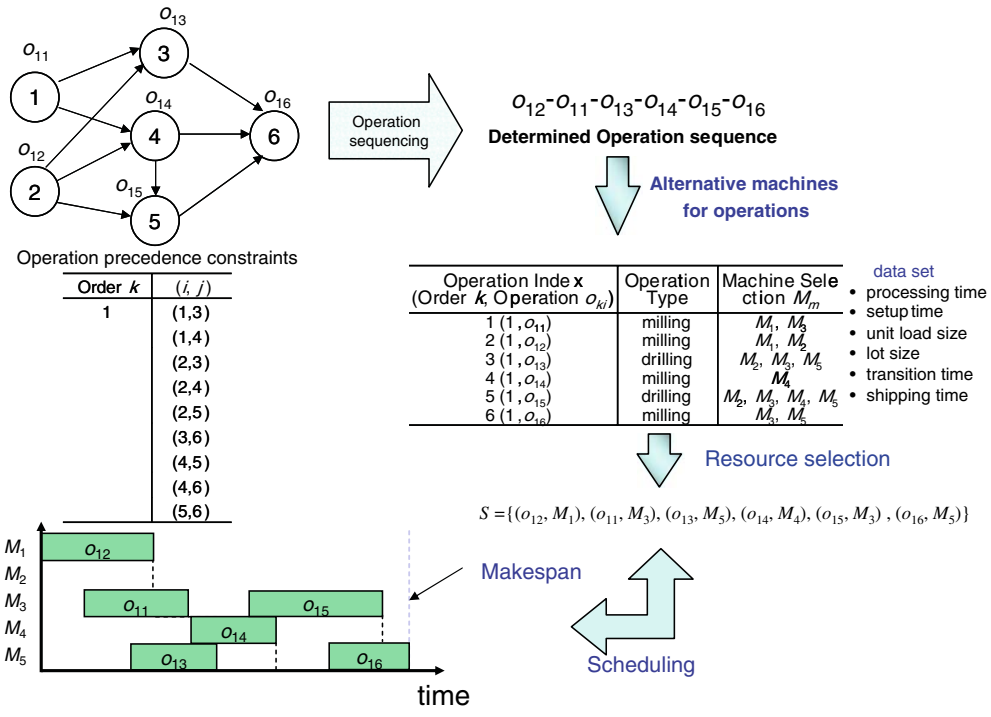


Fig. 2 Advanced planning process

$$x_{kim} = \begin{cases} 1, & \text{if operation } i \text{ in order } k \text{ is assigned to machine } m, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{kimljn} = \begin{cases} 1, & \text{if operation } i \text{ in order } k \text{ on machine } m \text{ precedes operation } j \text{ in order } l \text{ on machine } n, \\ 0, & \text{otherwise.} \end{cases}$$

c_{kim} completion time of operation i in order k on machine m

Generally, the transportation time between plants is larger than that between resources within a plant. In this paper, since both intra-plant transportation capacity between resources and inter-plant transportation capacity are assumed to be infinite, vehicle assignments for intra- and inter-transportations are not necessary. Therefore, transportation from resource m to resource n, t_{mn}, contains the transportation time between plants, if resources m and n are not in the same plant. If the transportation load size between resources is fixed (i.e., unit load size u_k), the number of sublots can be obtained by $\lceil q_k/u_k \rceil$, where $\lceil x \rceil$ represents the smallest integer greater than or equal to x. Because a unit load is handled as an individual part, the total number of items under scheduling consideration equals to the number of total sublots for all orders, $\sum_{\forall k} \lceil q_k/u_k \rceil$.

In general, the time to complete the orders for all customers can be calculated by summing setup times, transportation times, processing times, and waiting times. If two jobs on the same resource that are scheduled consecutively, then the setup time is added. If two operations in the same order are processed on two different machines, the transportation time is considered. The length required to complete the orders for all customers is called makespan. The makespan is important when the number of customer orders is finite. The makespan was denoted by E and defined as the time it takes for the last customers' order to leave the manufacturing system, that is,

$$X_k = \max_{\forall i \text{ and } m} \{c_{kim}\}, \text{ and} \tag{1}$$

$$E = \max_{\forall k} \{X_k\} = \max_{\forall k,i \text{ and } m} \{c_{kim}\}. \tag{2}$$

Minimizing the makespan in a MSC forces the planner to balance the workload over various resources and plants. The mixed integer programming model for the advanced planning problem in a MSC is presented below.

Minimize E

subject to

$$c_{kjn} - c_{kim} + M(2 - x_{kjn} - x_{kim}) \geq u_k p_{kjn} + t_{mn}, \quad \forall (i,j) \in R_k, k, m, n, m \neq n, \tag{3}$$

$$c_{kjn} - c_{kim} + M(1 - y_{kimkjn}) \geq u_k p_{kjn} + t_{mn}, \quad \forall (i,j) \in Q_k, k, m, n, \tag{4}$$

$$-c_{kjn+}c_{kim}+My_{kimkjn}+M(2-x_{kjn}-x_{kim}) \geq u_k p_{kim}+t_{nm}, \quad \forall(i,j)\in Q_k,k,m,n, \tag{5}$$

$$c_{ljm}-c_{kim}+M(1-y_{kimljm})\geq u_l p_{ljm}+s_{kilj}, \quad \forall(i,j)\in N_m,m,k,l,k\neq l, \tag{6}$$

$$-c_{ljm+}c_{kim}+My_{kimljm}+M(2-x_{ljm}-x_{kim}) \geq u_k p_{kim}+s_{ljki}, \quad \forall(i,j)\in N_m,m,k,l,k\neq l, \tag{7}$$

$$x_{ljn}+x_{kim}-2y_{kimljn}\geq 0, \quad \forall k,l,i,j,m,n, \tag{8}$$

$$\sum_k \sum_i p_{kim}x_{kim}\leq C_m, \quad \forall m, \tag{9}$$

$$\sum_{\forall m} x_{kim}=1, \quad \forall k,i, \tag{10}$$

$$c_{kim}\geq \begin{cases} p_{kim}, & \forall i\in Q_k,k,m, \\ 0, & \forall i\notin Q_k,k,m \end{cases} \tag{11}$$

$$x_{kim},y_{kimljn}\in\{0,1\}, \quad \forall k,l,i,j,m,n. \tag{12}$$

Constraint (3) means that the operations of the order of each customer are processed according to the required precedence constraint. Constraints (4) and (5) ensure that any two operations belonging to the same customer order cannot be processed simultaneously. Constraints (6) and (7) ensure that a resource cannot simultaneously process more than one customer order. These constraints, (3) through (7), are referred to as disjunctive constraints because only one or the other must hold. Constraint (8) guarantees that a precedence relation from operation i of order k on resource m to operation j of order l on resource n is possible only when the resource m and n are assigned to the operations i and j, respectively. Constraint (9) restricts the available capacity for each resource. Constraint (10) ensures that only one resource for each operation should be selected. Constraints (11) and (12) imply non-negativity and integrality of the corresponding variables.

Adaptive genetic algorithm approach

To solve the mixed integer programming model suggested in section "Model development", a new AGA approach with the adaptive scheme was proposed, which could automatically regulate GA operators (crossover and mutation rates) and could use a local search technique during its search process. The AGA reinforced by an adaptive scheme could appropriately regulate a balance between exploitation and exploration during its search process.

Therefore, first, some concepts and logics on exploitation and exploration in a genetic search process are discussed. Secondly, the detailed implementation procedure of the AGA follows.

Exploitation and exploration in genetic search process

In general, as a GA converges, the similarity among the chromosomes of a GA is increased and the variance of the fitness values of the chromosomes is decreased. If the GA search proceeds to an optimal solution, it can get a reliable solution. However, if it becomes fixed at a local optimum, there may be a premature convergence, which makes improvement of the solution and the search for an optimal solution very difficult.

Therefore, a reliable search method in GA requires that it: (i) continuously leads the convergence to an optimal solution and (ii) suitably regulates the inputs of new chromosomes into the current population in order to prevent premature convergence; this is usually referred to as exploitation and exploration of a solution. Exploitation uses knowledge acquired by exploration to reach better positions in the search space; while, exploration investigates new and unknown areas in the search space. By exploration, the diversity of the chromosomes in a GA population is maintained, and its premature convergence to local optima can be avoided. Alternatively, by exploitation, the searching ability for an optimal solution is enhanced, and the good chromosomes in a GA population are continuously preserved.

Many efforts for regulating a balance between exploitation and exploration in a GA have been performed. Srinivas and Patnaik (1994) developed a scheme that the crossover and mutation rates are increased when a population in GA tends to get stuck at a local optimum; they are decreased when the population is scattered in the search space of a GA. Other researchers (Herrena & Lozano, 2003; Mak, Wong, Wang, 2000; Wu, Cao, & Wen, 1998) have also suggested various schemes to adaptively regulate crossover and mutation rates.

A common characteristic of the studies above mentioned is to identify the status in which GA search process is converging or not converging, to adaptively regulate the crossover and mutation rates. As shown in Fig. 3, in a situation where the search is converging, the rates of mutation and crossover operations should be increased, respectively to prevent premature convergence and to reinforce the search to optimal solution. On the other hand, if GA is not conversing as in Fig. 4, then the rates of crossover and mutation operations should be decreased

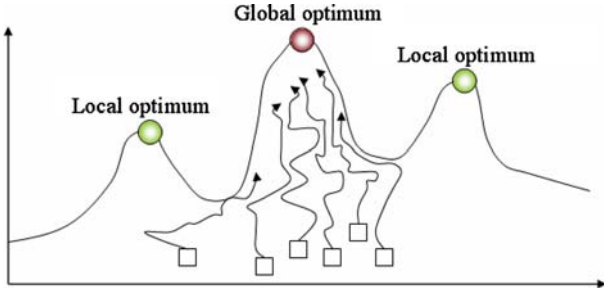


Fig. 3 The situation that GA is converging

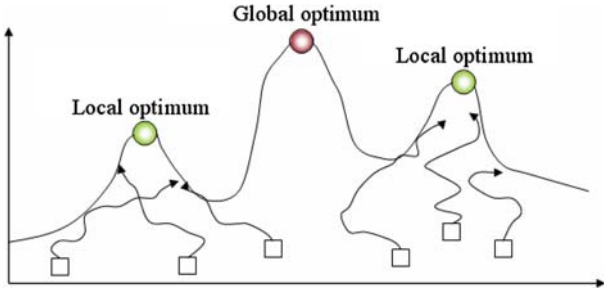


Fig. 4 The situation that GA is not converging

to restraint the introduction of new chromosomes and to guide the search toward convergence.

However, there are some weaknesses in the GA search strategies for these two situations. In the case of Fig. 3, where GA is converging, since the similarity among chromosomes of the GA population increases, the fitness values of the chromosomes are significantly similar to each other, and the variety of the population is reduced. This situation definitely deteriorates the performance of the GA, since it might be difficult for a GA search to avoid premature convergence even though the rates of crossover and mutation operations are increased. The technique that helps to improve this weakness of GA performance is to forbid the introduction of new chromosomes by crossover operations into the population when too many similar chromosomes already exist.

One possible alternative is to add new chromosomes into the current population. These new chromosomes should not be similar to the current population; certain high fitness values that are similar to that of the population should be maintained. The mutation operator in a GA is an example of such alternatives, but it usually generates random outputs, which is not necessarily an ideal approach. A local search technique that can search around the convergence area in a GA search at each generation is a possible alternative, since it is capable of having both a certain high fitness value, like the chromosomes generated by a GA, and being able to produce new chromosomes that discriminate the similarity with the chromosomes (Espinoza, Minsber, & Goldberg, 2001).

In the situation of Fig. 4, where the GA is not converging, detailed analysis divides the situation in two ways: one considers the case that the current fitness value of the GA is inferior to the previous fitness values during the genetic search process; the other is the case that the fitness values in continuous generations of the GA do not show any change.

In the first situation, the introduction of the inferior chromosomes in a current generation will be increased to the next generation, which deteriorates the performance of the GA. To overcome this weakness, the introduction of the new chromosomes should be reduced when constructing a new population of the next generation. Decreasing the rates of the crossover and mutation operations is a good choice for this purpose. By decreasing the rates, good chromosomes with a high fitness value in the current generation are kept, which can help the convergence of the solution.

The second case often occurs after the GA search has significantly progressed. At that time, any improvement or convergence of the solution may not occur, even though the introduction of new chromosomes to the next generation is reduced by decreasing the rates of crossover and mutation operations; which is mainly because the fitness values of the chromosomes at that time are significantly similar to each other. Therefore, a new technique for improving this situation is required. A possible alternative is to add the new chromosomes into the current population, resulting from the local search technique that is already explained when the GA is converging. Table 1 summarizes three resolutions to improve the searching capability of pure adaptive GA.

With the improved methods stated above, the three improved situations can be reached as shown in Table 1. To realize the three situations in Table 1, the change $C(t)$ of the population at generation t was used as follows:

$$C(t) = \frac{f_{\max}(t) - \bar{f}(t)}{f_{\max}(t) - f_{\min}(t)} \tag{13}$$

where $f_{\max}(t)$ and $f_{\min}(t)$ were the maximal and minimal fitness values of the population, respectively, and $\bar{f}(t)$ was the average of all the fitness values of the population, at generation t.

The change $C(t)$ in Eq. (13) can confirm the convergence situation of the solution in the GA population (Herrena & Lozano, 2003). Using Eq. (13), the change ratio (CR) of the fitness values in the continuous two generations could be formulated as follows:

$$\mathrm{CR} = \frac{C(t)}{C(t-1)} \tag{14}$$

Table 1 Three improved situations

Situation 1	If GA is converging, the rates of crossover and mutation operations are increased and a local search technique is also applied to the GA loop In this case, both the reservation of good chromosomes and the introduction of new chromosomes can be simultaneously achieved
Situation 2	If GA is not converging (the case where the current fitness value of the GA is inferior to the previous fitness values), the rates of crossover and mutation operations are decreased In this case, the introduction of new chromosomes is reduced and the convergence of the solution can be progressed.
Situation 3	If GA is not converging (the case where the fitness values in the continuous generations of the GA do not show any change at all), a local search technique is applied to the GA loop In this case, several new chromosomes, without the similarity of the current population and that keep a certain high fitness value similar to that of the population, are generated and inserted into the GA loop

Table 2 Three conditions

	Conditions
Situation 1	$CR < 1$
Situation 2	$CR > 1$
Situation 3	$CR = 1$

Using the CR in Eq. (14), when minimization was assumed, the three improved situations in Table 1 could be represented as shown in Table 2.

Using Tables 1 and 2, the procedure to adaptively regulate a balance between the exploration and exploitation during the GA search could be formulated. The procedure that uses the rate of the crossover operation (p_C), the rate of the mutation operation (p_M) and the local search technique is shown in Fig. 5.

The detailed approach using the GA and local search technique for implementing the proposed procedure are presented in section "Design of AGA"

Design of AGA

For designing the proposed AGA, both the GA and local search technique were used. The former was used to globally search the entire search space; while, the latter was used to locally search around the convergence area in the GA loop. By the mixed use of the two approaches, a balance between the exploitation and exploration during the search process of the AGA could be adaptively regulated. The detailed procedures for implementing both the GA and local search approaches are shown in the following sub-sections.

GA approach

The idea of the multistage operations considered in the proposed AGA came from the basic concept of a multistage decision making model, such as in dynamic programming. The multistage decision making model can be divided into stages, with a decision required at each stage. Each stage has a number of states associated with it. The decision at one stage transforms one state into a state in the next stage. After all the decisions were made for choosing states, a solution could be drawn, and the fitness of the result was in terms of the different decisions made along the route.

Since it was decided that both the operation sequence and the resource selection can affect the solution of an advanced planning problem, it was obvious that the chromosome presentation of the AGA for advanced planning problems consists of three main parts such as operation sequence, resource selection, and scheduling.

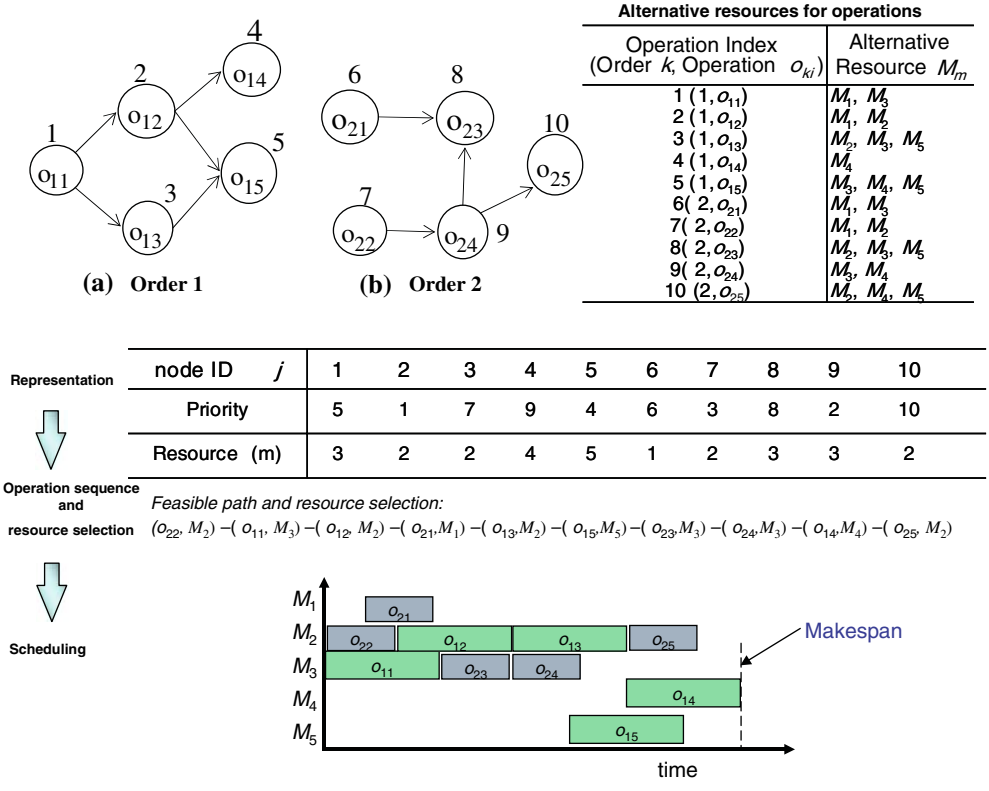
(1) Representation of feasible solution. The advanced planning problem includes various types of constraints such as precedent constraints and limited capacities. In order to generate a feasible solution, the representation scheme has to be capable of considering all possible constraints for a given problem. Therefore, a critical issue is the development of a representation scheme to represent a feasible solution. To generate a feasible schedule with operation sequence and resource selection, a representation scheme using the integrated concept of topological sort (Horowitz & Sahni, 1984) and random assignment of priority, and random resource selection is proposed. Suppose there are two customer orders, which consist of five operations for each order, o_{11} through o_{25}, where o_{ki} denotes operation i of order k. The chromosome structure can be represented as shown in Fig. 6.

In Fig. 6, the second row of the representation indicates the priority for candidate selection when operations with no incoming edges exist in a network of customer order with precedent constraints. The value of a gene is generated at random within $[1, J]$ exclusively, where J is the number of all operations. For example, operations o_{11}, o_{21}, and o_{22} have no precedent constraints

Fig. 5 Regulation of a balance between exploitation and exploration

procedure: *Regulation of a balance between exploitation and exploration*
input: CR, $p_C(t)$, $p_M(t)$
output: $p_C(t+1)$, $p_M(t+1)$
begin:
  if $CR < 1$ **then**
    $p_C(t+1) = p_C(t) + 0.1$;
    $p_M(t+1) = p_M(t) + 0.01$;
    apply local search technique to GA loop;
  if $CR > 1$ **then**
    $p_C(t+1) = p_C(t) - 0.1$;
    $p_M(t+1) = p_M(t) - 0.01$;
  if $CR = 1$ **then**
    apply local search technique to GA loop;
  end;
end.

Fig. 6 Chromosome representation



simultaneously as shown in Fig. 6. From the representation scheme, o_{22} is firstly selected for determining the operation sequence because the operation o_{22} has a high priority for selection. After selecting the o_{22}, we can remove the o_{22} and all arcs leading out of o_{22}. Now o_{11} and o_{21} have no predecessors and o_{11} is selected for the next operation because o_{11} has a higher priority than o_{21}. By the same manner, a final feasible path $o_{22}-o_{11}-o_{12}-o_{21}-o_{13}-o_{15}-o_{23}-o_{24}-o_{14}-o_{25}$ is determined.

The third row indicates the randomly selected resource number for each operation. Each operation randomly selects a resource number from the possible alternative resources. Let a feasible operation sequence with corresponding resources be $(o_{22}, M_2)-(o_{11}, M_3)-(o_{12}, M_2)-(o_{21}, M_1)-(o_{13}, M_2)-(o_{15}, M_5)-(o_{23}, M_3)-(o_{24}, M_3)-(o_{14}, M_4)-(o_{25}, M_2)$, where M_m denotes machine m. A feasible schedule is then obtained as shown in Fig. 6.

(2) *Overall procedure for generating a feasible solution.* The overall procedure to determine a feasible schedule and operation sequence with resource selection is shown in Fig. 7. The first step in the AGA is to initialize the population of feasible solutions. The initialization of the population is done by generation feasible solutions randomly as much as the population size.

Fig. 7 Overall procedure of feasible solution generation

```
procedure: feasible solution generation
  input: set of precedent constraints, alternative resources, available capacities, and processing
  and transportation times.
  begin:
  generate chromosome: generate a random priority number (1 to J) for operations with
  corresponding resources;
  generating an operation sequence:
  while (any operation remains) do
      if every operation has a predecessor then the network is infeasible: stop;
      else pick an operation o_kl without predecessors; output (o_kl, M_m); and
          delete o_kl and all arcs leading out of o_kl from the network;
  end;
  generate a schedule:
  while (any operation remains) do
      Allocate all operations to the corresponding resources sequentially;
  end;
end.
```

Local search approach

Local search techniques usually use local information about the current set of data (state) to determine a promising direction for moving some of the data set, which in turn is used to form the next set of data. The advantage of local search techniques is that they are simple and computationally efficient. However, they are easily entrapped in a local optimum. In contrast, global search techniques, such as GA, explore the global search space without any local information about a promising search direction. Consequently, they are less likely to become trapped in local optima; however, their computational cost is higher.

Many researchers have reported that hybrid approaches with both the GA and a local search technique produces certain benefits (Lee, Yun, & Gen, 2002; Li & Jiang, 2000; Yen Liao, Lee, & Randolph, 1998). The reason is that the hybrid approaches can combine a merit of the GA with that of the local search technique. That is, the hybrid approaches are less likely to become trapped in a local optimum than the local search technique. Due to the local search technique, the hybrid approach often converges faster than the GA does.

Therefore, in this paper, a method was used to hybridize the GA with the local search technique. This approach, as seen in most of conventional hybrid GAs, was to incorporate the local search technique into the GA loop (Gen & Cheng, 2000; Yen et al., 1998). With this approach, the local search technique was applied to each newly generated offspring to move it to a local optimum before injecting it into the new population.

For the proposed AGA, the iterative hill climbing method suggested by Michalewicz (1994) was used and improved upon. This method could guarantee the desired properties of a local search technique for hybridization as explained above. The main difference between the conventional iterative hill climbing method and the improved iterative hill climbing method was that the latter selects an optimal chromosome among the chromosomes satisfying the constraints of the hybrid GA, while the former selects a current chromosome at random. This allows the improved iterative hill climbing method to have various search abilities and good solutions unmet by the conventional method. The detailed procedure of the improved iterative hill climbing method, when minimization was assumed, is shown in Fig. 8.

Implementation procedure of AGA

According to the logic and scheme used in sections "GA approach" and "Local search approach", the overall procedure could be formulated for the APS problem using the AGA; the procedure is shown in Fig. 9. In this procedure, $P(t)$ and $C(t)$ were defined as populations for the parent and offspring at generation t, respectively. In the overall procedure, the appropriate functions of crossover and mutation were chosen in correspondence to the two vectors of the chromosome. There was only one crossover for phase 1 of section "GA approach", since after the operation sequence was decided, the stages (operations) position in phase 2 for choosing states (machines) did not have to be changed.

Numerical experiments

A benchmarking example (Moon et al., 2004) consisting of four customer orders with lot sizes (40, 70, 60, 30) and 6 machines in 2 plants was used to test the

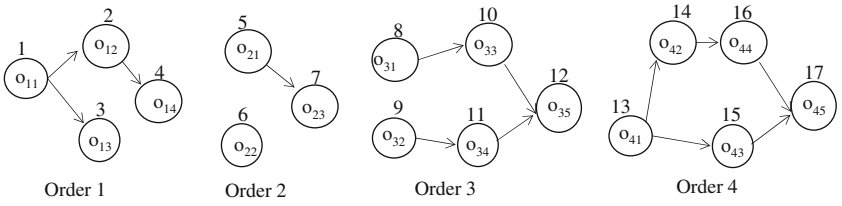
Fig. 8 Procedure of the improved iterative hill climbing method

procedure: improved iterative hill climbing method in GA loop
input: chromosome $\mathbf{v}_{c,}$
output: chromosome $\mathbf{v}_n$
begin
select the best chromosome $\mathbf{v}_c$ in current GA loop;
randomly generate as many chromosomes as the population size in the neighborhood of $\mathbf{v}_c$;
select the chromosome $\mathbf{v}_n$ with the optimal value of the objective function f among the set of the new chromosomes generated;
if $f(\mathbf{v}_c) > f(\mathbf{v}_n)$ **then** $\mathbf{v}_c \leftarrow \mathbf{v}_n$;
end.

Fig. 9 Overall procedure for APS using AGA

procedure: AGA for advanced planning
input: advanced planning data set, GA parameters
output: best solution
begin
t 0;
initialize $P_1(t)$ by priority encoding;
initialize $P_2(t)$ by machine permutation encoding;
fitness $eval(P_1, P_2)$;
while (not termination condition) **do**
crossover $P_1(t)$ to yield $C_1(t)$ by one-cut point crossover;
mutation $P_1(t)$ to yield $C_1(t)$ by swap mutation;
mutation $P_2(t)$ to yield $C_2(t)$ by neighbor search mutation;
fitness $eval(C_1, C_2)$; by priority decoding and machine permutation decoding;
calculate CR by equations (13) and (14) and adaptively change p_C and p_M by invoking the local search in Figure 5;
select $P(t+1)$ from $P(t)$ and $C(t)$ by roulette wheel selection;
$t \leftarrow t+1$;
end.

Fig. 10 Operations and precedence constraints for 4 orders



proposed AGA. Each plant had three resources; Plant $1 = \{M_1, M_2, M_3\}$ and Plant $2 = \{M_4, M_5, M_6\}$. The unit load size for transportation was assumed to be 10 for all orders. The operations and their precedence constraints for the 4 orders are given in Fig. 10. The transportation times between resources and the available capacities for the resources are given in Table 3, and the processing times for each operation and their alternative resources are given in Table 4. The setup times between operations are given in Table 5.

If the resources m and n are not included in a plant, the transportation time t_{mn} was assumed to be 50 (i.e., transportation time between plant 1 and plant 2), and the unit size per trip was equal to the lot size of each order. To solve the problem using the proposed AGA, the parameters used were set as follows:

Maximum generation maxGen = 200,
Population size popSize = 100,
Initial crossover probability $p_C = 0.6$,
Initial mutation probability $p_M = 0.2$,
Search range for improved iteration hill climbing method = 0.5

In the parameters of the AGA mentioned above, all the parameters except for the p_C and p_M were fixed

Table 3 Transition time t_{mn} between machines

		Plant 1					Plant 2		
		M_1	M_2	M_3			M_4	M_5	M_6
	M_1	0	5	6		M_4	0	5	6
Plant 1	M_2	5	0	7	Plant 2	M_5	5	0	7
	M_3	6	7	0		M_6	6	7	0
Available capacity		1000	1000	2000	Available capacity		2000	2000	2000

Table 4 Processing time p_{kim} for operations in alternative machines

	Operation	Order 1				Order 2				Order 3				Order 4				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	M_1	7	7	–	6	–	3	8	–	10	6	15	–	–	–	–	–	5
Plant 1	M_2	–	–	6	–	9	5	–	–	–	5	–	6	–	5	–	5	–
	M_3	–	–	5	–	–	–	12	5	–	–	–	–	6	–	6	–	–
	M_4	5	6	–	–	8	–	9	–	10	–	6	–	6	–	4	3	–
Plant 2	M_5	–	–	8	–	–	6	–	8	–	6	–	5	–	9	–	–	4
	M_6	–	–	–	5	–	–	8	–	7	–	5	–	8	–	–	5	–

Table 5 Setup time between operations

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	–	17	36	6	37	20	11	30	5	32	30	36	23	21	1	28	20
2	42	–	32	3	2	15	15	22	44	39	30	37	47	12	5	38	31
3	6	6	–	37	26	0	23	29	12	5	13	3	13	24	29	26	8
4	12	3	40	–	19	46	31	30	31	49	49	27	39	45	9	0	3
5	2	48	43	25	–	49	10	11	4	8	17	39	34	31	11	0	24
6	24	26	43	31	49	–	22	31	21	43	31	10	30	23	2	34	38
7	20	45	28	43	22	16	–	39	46	25	43	34	9	22	38	12	7
8	15	41	44	35	14	10	30	–	2	14	7	8	22	3	18	45	18
9	25	47	22	21	47	39	26	0	–	22	33	7	37	20	25	20	7
10	3	46	9	10	35	18	5	21	24	–	33	40	22	23	41	37	31
11	1	17	31	3	30	15	23	21	37	3	–	15	23	32	3	46	6
12	4	18	41	37	26	39	43	46	44	28	13	–	45	47	7	32	2
13	18	45	24	27	47	21	8	21	35	38	26	39	–	21	2	12	33
14	48	37	46	44	25	24	1	8	38	46	48	37	6	–	6	41	10
15	43	3	39	3	44	17	46	24	46	33	9	16	15	4	–	4	12
16	9	44	40	21	16	12	36	37	44	16	41	31	7	3	8	–	44
17	14	21	5	29	44	48	8	31	10	44	1	7	18	2	11	22	–

during its search process. However, the p_C and p_M were variously changed to automatrically regulate a suitable balance between the exploitation and exploration during the search process of the AGA.

By applying the proposed AGA to the example problem, the best solution was obtained as the makespan of 1,102 time units, and the corresponding chromosomes and schedules were as follows:

Table 6 Best schedule for the benchmarking problem

Plant d	Machine M_m	Operation O_{ki}(start time–finishing time)		
Plant 1	M_1	o_{22}(0–210)	o_{23}(232–792)	o_{45}(952–1102)
	M_2	o_{21}(35–665)	o_{42}(696–846)	o_{44}(890–1040)
	M_3	o_{31}(0–300)	o_{13}(551–751)	o_{43}(782–962)
	M_4	o_{11}(0–200)	o_{12}(217–457)	o_{34}(578–938)
Plant 2	M_5	o_{33}(350–710)	o_{35}(750–1050)	
	M_6	o_{41}(0–240)	o_{32}(452–872)	o_{14}(893–1093)

Table 7 Comparisons of Moo–Kim–Gen's and AGA approaches on different problem settings

Problem settings (No. of orders, No. of operations, No. of plants, No. of resources)	maxGen	popSize	Moon–Kim–Gen's approach		Proposed AGA approach	
			Makespan	Aver. comp. time (s)	Makespan	Aver. comp. time (s)
(5, 21, 2, 6)	100	100	1370	9	1370	2
	200	100	1370		1370	
	500	100	1370		1370	
(10, 40, 2, 6)	100	100	2264	56	2178	21
	200	100	2258		2178	
	500	150	2238		2178	
(15, 60, 3, 9)	200	100	1659	98	1598	48
	500	100	1623		1598	
	500	150	1620		1598	
(20, 80, 4, 16)	200	100	1330	132	1306	76
	500	100	1330		1306	
	500	150	1327		1306	

index j:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$v_1(j)$:	2	3	7	13	8	1	9	4	11	5	12	17	6	10	14	15	16
$v_2(j)$:	4	4	3	6	2	1	1	3	6	5	4	5	6	2	3	2	1

$$
\begin{aligned}
S &= \{\text{Order 1, Order 2, Order 3, Order 4}\} \\
&= \{(o_{11}, M_4 : 0-200), (o_{12}, M_4 : 217-457), \\
&\quad (o_{13}, M_3 : 551-751), (o_{14}, M_6 : 893-1093), \\
&\quad (o_{22}, M_1 : 0-210), (o_{21}, M_2 : 35-665), \\
&\quad (o_{23}, M_1 : 232-792), (o_{31}, M_3 : 0-300), \\
&\quad (o_{33}, M_5 : 350-710), (o_{32}, M_6 : 452-872), \\
&\quad (o_{34}, M_4 : 578-938), (o_{35}, M_5 : 750-1050), \\
&\quad (o_{41}, M_6 : 0-240), (o_{42}, M_2 : 696-846), \\
&\quad (o_{43}, M_3 : 782-962), (o_{44}, M_2 : 890-1040), \\
&\quad (o_{45}, M_1 : 952-1102)\}
\end{aligned}
$$

The best schedule, in detail, is shown in Table 6.

To prove the efficiency of the AGA approach, the experimental results were compared with Moon–Kim–Gen's approach (Moon et al. 2004) by using the same experimental data that were randomly generated. The Moon–Kim–Gen's approach is one of the traditional GAs that rely highly on the proper settings of genetic parameter values. The computational experiments for comparison were conducted on a computational environment with Pentium 4 CPU and 512 MB of RAM.

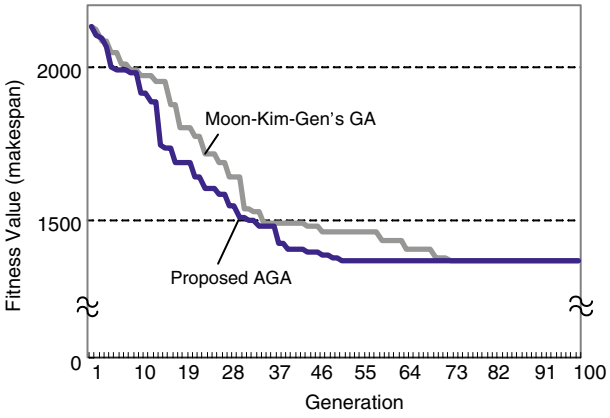


Fig. 11 Converging processes of the proposed AGA and Moon–Kim–Gen's GA

Table 7 shows the average makespan of the best solutions found over 10 runs for each of 12 cases, i.e., 4 different problems and 3 parameter combinations for each problem. In this set of experiments, the $p_C = 0.6$ and $p_M = 0.2$ were chosen. For the first, second and third problem settings, the experimental probabilities of getting the best solution were about 100%. For the

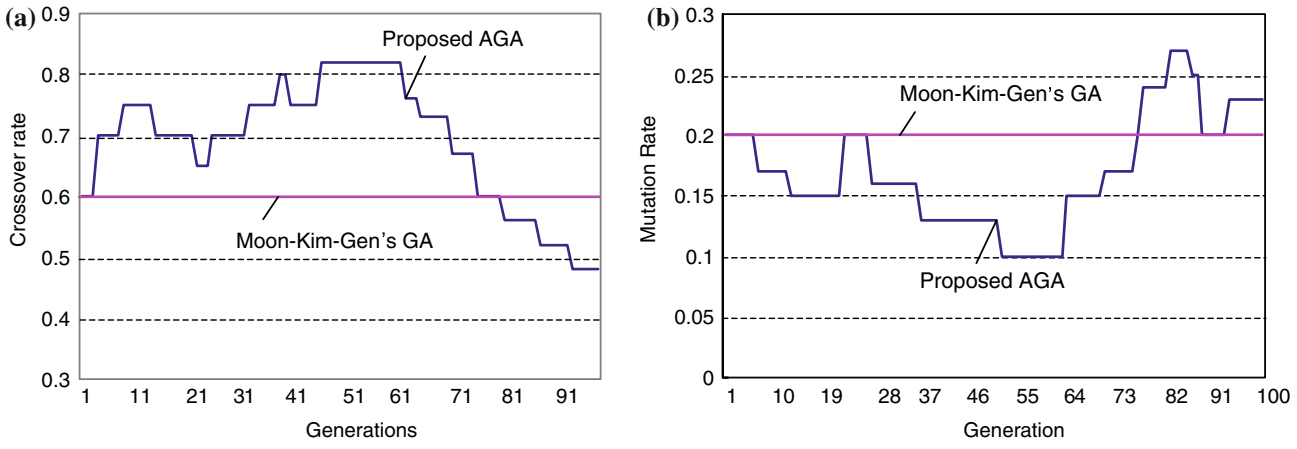


Fig. 12 Changing process of the (a) crossover probabilities and (b) mutation probabilities

forth experiment, the probability of getting the best solution was about 94%. However, Moon-Kim-Gen's approach could not reach the best solution for the second, third and forth cases. From the results, it could be seen that the AGA approach with a local search could find better solutions with a very high probability and in reasonable computational times. Also, the proposed AGA algorithm was relatively insensitive to the change of parameters, maxGen and popSize. This is because the proposed AGA algorithm is capable to adaptively control the balance of the crossover and mutation rates.

In Fig. 11, the average fitness values of 10 runs of the first 5-order problem setting in Table 7 are plotted to compare the converging processes of these two approaches. Figure 11 shows that the evolutionary search behavior of the proposed AGA is better than that of the Moon–Kim–Gen's GA approach.

In order to trace the adaptive capability of the proposed adaptive GA, the changing values of the crossover and mutation rates through the whole generations of the first 5-order problem in Table 7 were plotted in Fig. 12. Figure 12(a) and (b) illustrate that the crossover rate increases overall from 0.6 to 0.82 in the 61th generation, while the mutation rate decreases from 0.2 to 0.1 until around 64th generation. This controlled change of crossover and mutation rates by the adaptive scheme could bring out the outpeformance of the proposed AGA to Moon–Kim–Gen's approach shown in Fig. 11.

Conclusion

This paper addresses the adaptive genetic algorithm (AGA) approach with a local search to solve the advanced planning problems in a manufacturing supply chain. The objective was to find the optimal resource selection for assignments and operations sequences in order to minimize the makespan; this included the setup time, transportation time, and operations processing time. Some numerical experiments were offered to prove the efficiency of the AGA. The result showed that the AGA found a better solution when the problem sizes were enlarged. In the end, the experiments were analyzed in detail, and the appropriate parameter setting of the proposed AGA was determined. This approach was compared with Moon–Kim–Gen's approach using the same data. The results using various sizes of experiments have demonstrated the efficiency of the AGA by comparing it with the previous methods as well as the capability of adaptive regulation of exploration and exploitation using a local search technique embedded. In conclusion, the proposed approach can be effectively used to solve the complex and sizable problem of advanced planning in a manufacturing supply chain.

Acknowledgements This work was in part supported by Korea Research Foundation Grant funded by the Korean government (MOEHRD, Basic Research Promotion Fund: KRF-2005-206-D00024) and by the fund from the Basic Research Program (Grant No.: R01-2002-000-00232-0) of the Korea Science and Engineering Foundation.

References

Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computations. In: M. Palaniswami, Y. Attikiouzel, R. Markc, D. Fogel, & T. Fukuda (Eds.), *Computational intelligence: A dynamic systems perspective* (pp. 152–163). Piscataway, NJ: IEEE Press.

Bandeimarte, P, & Calderini, M. (1995). A heuristic bicriterion approach to integrated process plan selection and job shop scheduling. *International Journal of Production Research, 33*(1), 161–181.

Cochran, J. K., Horng, S., & Fowler, J. W. (2003). A multi-population GA to solve multi-objective scheduling problems for parallel resources. *Computers & Operations Research, 30*, 1087 – 1102.

Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolution Computation, 3*(2), 124–141.

Espinoza, F. P., Minsker, B. S., & Goldberg, D. E. (2001). A self adaptive hybrid genetic algorithm. *Proceedings on the Genetic and Evolutionary Computation Conference*. San Francisco: Morgan Kaufman Publishers.

Fogel, D. B., Fogel, G. B., & Ohkura, K. (2001). Multiple-vector self-adaptation in evolutionary algorithms. *BioSystems, 61*, 155–162.

Gen, M., & Cheng R. (2000) *Genetic algorithms & engineering optimization*. New York: John Wiley & Sons.

Guinet, A. (2001). Multi-site planning: A transshipment problem. *International Journal of Production Economics, 74*, 21–32.

Herrera, F., & Lozano, M. (2003). Fuzzy adaptive genetic algorithms: design, taxonomy and future directions. *Soft Computing, 7*(8), 545–562.

Horowitz, E., & Sahni, S. (1984). *Fundamentals of data structures in Pascal*. Computer Science Press.

Lee, C. Y., Yun, Y. S., & Gen, M. (2002). Reliability optimization design for complex systems by hybrid GA with fuzzy logic control and local search. *IEICE Transaction on Fundamentals*, *E85-A*(4), 880–891.

Li, B., & Jiang, W. (2000). A novel stochastic optimization algorithm. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 30*(1), 193–198.

Lutz, S., Helms, S. L., & Wiendahl H. P. (1999). Subcontracting in variable production networks. *Proceedings of the 15th International Conference on Production Research*, 1999, 597–600.

Mak, K. L., Wong, Y. S., & Wang, X. X. (2000). An adaptive genetic algorithm for manufacturing cell formation. *International Journal of Manufacturing Technology, 16*, 491–497.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution program*. Second Extended Edition, Spring-Verlag.

Moon, C., Kim, J, & Hur, S. (2002). Integrated process planning and scheduling with minimizing total tardiness in multiplants supply chain. *Computers & Industrial Engineering, 43*, 331–349.

Moon, C., & Seo, Y. (2005). Evolutionary algorithm for advanced process planning and scheduling in a multi-plant. *Computer and Industrial Engineering*, 48(2), 311–325.

Moon, C., Kim, J., & Gen, M. (2004). Advanced planning and scheduling based on precedence and resource constraints for e-plant chains. *International Journal of Production Research, 42*(15), 2941–2955.

Morad, N., & Zalzala, A. (1999). Genetic algorithm in integrated process planning and scheduling. *Journal of Intelligent Manufacturing, 10*, 169–179.

Palmer, G. J. (1996). A simulated annealing approach to integrated production scheduling. *Journal of Intelligent Manufacturing, 7*(3), 163–176.

Saygin, C., & Kilic, S. E. (1999). Integrating flexible process plans with scheduling in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology, 15*, 265–280.

Srinvas, M., & Patnaik, L. M. (1994). Adaptive *probabilities* of crossover and mutation in genetic algorithms. *IEEE Transaction on Systems, Man and Cybernetics, 24*(4), 656–667.

Tan, W. (2000), Integration of process planning and scheduling – a review, *Journal of Intelligent Manufacturing, 11*, 51–63.

Tan, W., & Khoshnevis, B. (2004). A linearized polynomial mixed integer programming model for the integration of process planning and scheduling. *Journal of Intelligent Manufacturing, 15*, 593–605.

Thoney, K. A., Hodgson, T. J., King, R. E., & Taner, M. R. (2002). Satisfying due-dates in large multi-factory supply chain. *IIE Transactions, 34*, 803–811.

Vercellis, C. (1999). Multi-plant production planning in capacitated self-configuring two-stage serial systems. *European Journal of Operational Research, 119*, 451–460.

Wu, Q. H., Cao, Y. J., & Wen, J. Y. (1998). Optimal reactive power dispatch using an adaptive genetic Algorithm. *Electrical Power and Energy Systems, 20*(8), 563–569.

Yen, J., Liao, J. C., Lee, B. J., & Randolph, D. (1998). A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 28*(2), 173–191.

Yun, Y. (2002). Genetic algorithm with fuzzy logic controller for preemptive and non-preemptive job shop scheduling problems. *Computers and Industrial Engineering, 43*(3), 623–644.