Industrial Engineering
& Management Systems
Vol 11, No 4, December 2012, pp.310-330
ISSN 1598-7248 | EISSN 2234-6473 |
http://dx.doi.org/10.7232/iems.2012.11.4.310


# Multiobjective Genetic Algorithm for Scheduling Problems in Manufacturing Systems

**Mitsuo Gen***
Fuzzy Logic Systems Institute (FLSI), Fukuoka, Japan; National Ting Hua University, Hsinchu, Taiwan

**Lin Lin**
Fuzzy Logic Systems Institute (FLSI), Fukuoka, Japan; Dalian University of Technology, Dalian, China

(Received: September 1, 2012 / Accepted: September 10, 2012)

ABSTRACT

Scheduling is an important tool for a manufacturing system, where it can have a major impact on the productivity of a production process. In manufacturing systems, the purpose of scheduling is to minimize the production time and costs, by assigning a production facility when to make, with which staff, and on which equipment. Production scheduling aims to maximize the efficiency of the operation and reduce the costs. In order to find an optimal solution to manufacturing scheduling problems, it attempts to solve complex combinatorial optimization problems. Unfortunately, most of them fall into the class of NP-hard combinatorial problems. Genetic algorithm (GA) is one of the generic population-based metaheuristic optimization algorithms and the best one for finding a satisfactory solution in an acceptable time for the NP-hard scheduling problems. GA is the most popular type of evolutionary algorithm. In this survey paper, we address firstly multiobjective hybrid GA combined with adaptive fuzzy logic controller which gives fitness assignment mechanism and performance measures for solving multiple objective optimization problems, and four crucial issues in the manufacturing scheduling including a mathematical model, GA-based solution method and case study in flexible job-shop scheduling problem (fJSP), automatic guided vehicle (AGV) dispatching models in flexible manufacturing system (FMS) combined with priority-based GA, recent advanced planning and scheduling (APS) models and integrated systems for manufacturing.

Keywords: Manufacturing Scheduling, Evolutionary Algorithm, Genetic Multiobjective Optimization, Flexible Job-shop Scheduling Problem, Automatic Guided Vehicle, Flexible Manufacturing System, Advanced Planning and Scheduling

* Corresponding Author, E-mail: gen@flsi.or.jp

## 1. INTRODUCTION

Scheduling is one of the most important fields in manufacturing optimization. Scheduling involves determining the allocation of plant resources. Tasks must be assigned to the process units, and the duration and amount of processed material related to those assigned tasks must be determined (Verderame and Floudas, 2008). For a more extensive explanation of the various aspects of the scheduling model, the reader is directed to the reviews of Floudas and Lin (2004, 2005). The quality of the planning model and the integration scheme can be rendered inconsequential if the scheduling level does not rigorously model the production capacity of the plant, which is greatly dependent on the chosen time representation. Bidot *et al.* (2009) gave detail definitions to avoid ambiguity of terms commonly used by different communities: complete schedule, flexible schedule, conditional schedule, predictive schedule, executable schedule, adaptive scheduling system, robust predictive sche-

dule and table predictive schedule.

However, to find the optimal solutions of manufacturing scheduling gives rise to complex combinatorial optimization problems; unfortunately, most of them fall into the class of NP-hard combinatorial problems. Depending on the common sense "from easy to difficult" and "from simple to complex", Gen *et al*. (2008) gave a widely surveyed scheduling models as shown in Figure 1.

Genetic algorithm (GA) has attracted significant attention with respect to complexity scheduling, which is referred to genetic scheduling, it is a vital research domain at interface of two important sciences–artificial intelligence and operational research (Dahal, *et al*., 2007). In the last decade, Nowicki and Smutnicki (2005) provided an approximate Tabu search (TS) algorithm for job-shop scheduling problem (JSP) that is based on the big valley phenomenon, and uses some elements of so-called path relinking technique as well as new theoretical properties of neighborhoods. Tavakkoli-Moghaddam *et al*. (2005) used a neural network approach to generate initial feasible solutions and adapted a simulated annealing (SA) algorithm to improve the quality and performance of the solution in JSP. Wu and Weng (2005) considered the problem with job earliness and tardiness objectives, and proposed a multiagent scheduling method. Xia and Wu (2005) treated this problem with a hybrid of particle swarm optimization (PSO) and SA as a local search algorithm. Zhang and Gen (2005) proposed a multistage operation-based GA to deal with the flexible JSP (fJSP) problem from the point view of dynamic programming. Kacem *et al*. (2002a) proposed a GA controlled by the assigned model which is generated by the approach of localization. Najid *et al*. (2002) used simulated annealing for optimizing the flexible assignment of machines in fJSP. Lopez and Moramay (2005) newly described the design and implementation of a step-based manufacturing information system to share flexible manufacturing resources data.

Recently, manufacturing scheduling problems are also formulated in distributed and dynamic environments. Xiang and Lee (2008) proposed an ant colony intelligence algorithm for multi-agent dynamic manufacturing scheduling. Ant colony intelligence (ACI) is proposed to be combined with local agent coordination so as to make autonomous agents adaptive to changing circumstances and to give rise to efficient global performance. Wang *et al*. (2008) proposed a multi-agent approach integrated with a filtered beam-search-based heuristic algorithm to study the dynamic scheduling problem in a flexible manufacturing system (FMS) shop floor consisting of multiple manufacturing cells.

Framinan and Ruiz (2010) gave a research review for architecture of manufacturing scheduling systems. Process planning and scheduling were regarded as separate tasks performed sequentially, where scheduling was implemented after process plans had been generated. However, their functions are usually complementary. If the two systems can be integrated more tightly, greater performance and higher productivity of manufacturing system can be achieved. Shao *et al*. (2009) proposed an integration process planning and scheduling model and gave a GA-based approach developed to facilitate the integration and optimization of the two functions. In order to improve the optimized performance of the GA-based approach, more efficient genetic representations and operator schemes have been developed. Li *et al*. (2010) proposed an agent-based approach to integrated process planning and scheduling. In this approach, the two functions are carried out simultaneously, and an optimization agent based on an evolutionary algorithm (EA) is used to manage the interactions and communications between agents to enable proper decisions to be made. Gen *et al*. (2009b) surveyed evolutionary techniques for various optimization problems in integrated

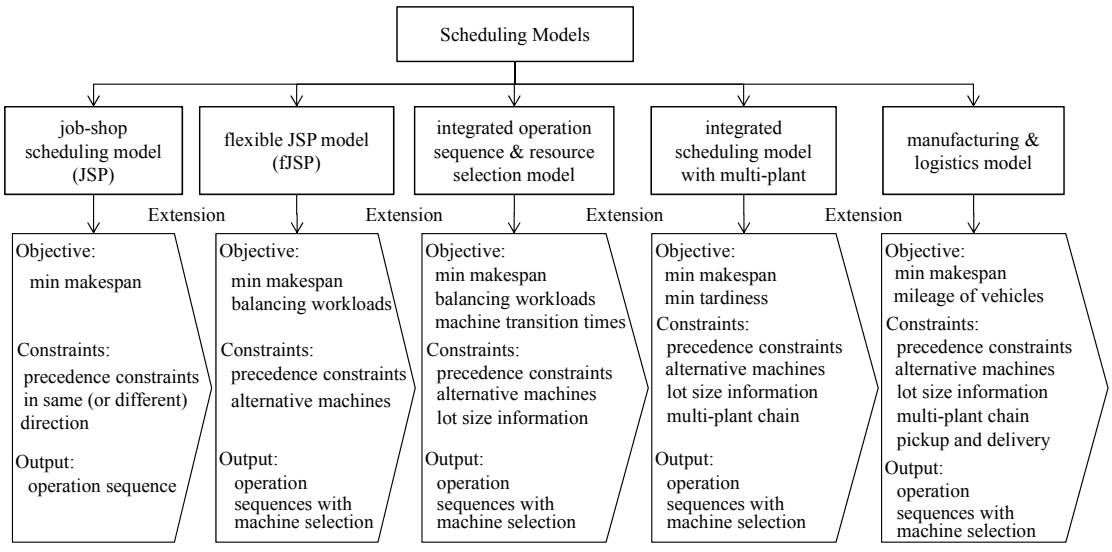



**Figure 1.** The core models of scheduling.

manufacturing system. Recently, Zhang and Gen (2010) proposed multiobjective genetic algorithm for solving process planning and scheduling problem in distributed manufacturing system, and Zhang *et al.* (2012) proposed hybrid sampling strategy-based multiobjective EA for process planning and scheduling problem. Lin *et al.* (2012a) reported a network modeling technique to formulate the complex scheduling problems in manufacturing, focus on how to model the scheduling problems to mathematical formulation, and proposed a multisection EA for the scheduling models formulated by network modeling.

Furthermore, many researches are focusing on the multiobjectives manufacturing scheduling problems. Li and Huo (2009) proposed a GA for multiobjective fJSP with consideration of maintenance planning, intermediate inventory, and machines in parallel, which had a background of practical scheduling problem in seamless steel tube production. Geiger (2011) proposed a heuristic search, intensification through variable neighborhoods, and diversification through perturbations and successive iterations in favorable regions of the search space, and successfully tested on permutation flow shop scheduling problems under multiple objectives. Karimi-Nasab and Aryanezhad (2011) introduced a multi-product multi-period production planning problems. A novel multiobjective model for the production smoothing problem on a single stage facility for which some of the operating times could be determined in a time interval. The proposed model was solved by a GA, which uses a novel achievement function for exploring the solution space, based on LP-metric concepts. Li *et al*. (2012) proposed Nash equilibrium in a game theory based approach that has been used to deal with the multiobjective integrated process planning and scheduling.

The rest of this survey paper is organized as follows: in Section 2, we introduce multiobjective GA, and give fitness assignment mechanism, and performance measures for multiple objective optimization problems, that are useful for designing multiobjective GAs. In Section 3, we give fJSP and propose a multistage operation-based GA (moGA) approach for solving fJSP. In Section 4, we introduce automatic guided vehicle (AGV) dispatching in flexible manufacturing system (FMS) combined with priority-based GA and in Section 5 a recent advanced planning and scheduling (APS) model will be introduced.

## 2. MULTIOBJECTIVE GENETIC ALGORITHM

Optimization deals with the problems of seeking solutions over a set of possible choices to optimize certain criteria. If there is only one criterion to be taken into consideration, it becomes single objective optimization problems, which have been extensively studied for the past 50 years. If there are more than one criterion which must be treated simultaneously, we have multiple objective optimization problems (Steuer, 1986; Deb, 2005). Multiple objective problems arise in the design, modeling, and planning of many complex real systems in the areas of industrial production, urban transportation, capital budgeting, forest management, reservoir management, layout and landscaping of new cities, energy distribution, etc. It is easy to find that almost every important real world decision problem involves multiple and conflicting objectives which need to be tackled while various constraints are leading to overwhelming problem complexity. The multiple objective optimization problems have been receiving growing interest from researchers with various backgrounds since early 1960 (Hwang and Yoon, 1981). There are a number of scholars who have made significant contributions to the problem. Among them, Pareto is perhaps one of the most recognized pioneers in the field (Pareto, 1906). Recently, GAs have been received considerable attention as a novel approach to multiobjective optimization problems, resulting in a fresh body of research and applications known as genetic multiobjective optimization (EMO).

The inherent characteristics of EAs demonstrate why genetic search is possibly well suited to the multiple objective optimization problems. The basic feature of EAs is the multiple directional and global searches by maintaining a population of potential solutions from generation to generation. The *population-to-population* approach is hopeful to explore all Pareto solutions.

EAs do not have many mathematical requirements about the problems and can handle any kind of objective functions and constraints. Due to their genetic nature, the EAs can search for solutions without regard to the specific inner workings of the problem. Therefore, it offers more hope for solving many more complex problems than the conventional methods.

Because EAs, as a kind of metaheuristics, provide us a great flexibility to hybridize with conventional methods into their main framework, we can take both advantages of the EAs and the conventional methods to make much more efficient implementations for the problems. The ingrowing researches on applying EAs to the multiple objective optimization problems present a formidable theoretical and practical challenge to the mathematical community (Gen *et al*., 2008).

### 2.1 Multiobjective Optimization Model

A single objective optimization problem is usually given in the following form:

$$\begin{aligned} &\max \quad z = f(\boldsymbol{x}) \\ &\text{s.t.} \quad g_i(\boldsymbol{x}) \le 0, \quad i = 1, 2, \cdots, m \end{aligned} \tag{1}$$

where $\boldsymbol{x} \in R^n$ is a vector of $n$ decision variables, $f(\boldsymbol{x})$ is the objective function, and $g_i(x)$ are inequality constraint $m$ functions, which form the area of feasible solutions. We usually denote the feasible area in decision space

with the set $S$ as follows:

$$S = \left\{ \boldsymbol{x} \in R^n \,\middle|\, g_i(x) \le 0, \;\; i = 1, 2, \cdots, m, \, \boldsymbol{x} \ge 0 \right\} \tag{2}$$

Without loss of generality, a multiple objective optimization problem (MOP) can be formally represented as follows:

$$\begin{aligned} &\max \;\; \left\{ z_1 = f_1(\boldsymbol{x}), z_2 = f_2(\boldsymbol{x}), \cdots, z_q = f_q(\boldsymbol{x}) \right\} \\ &\text{s.t.} \;\; g_i(\boldsymbol{x}) \le 0, \;\; i = 1, 2, \cdots, m \end{aligned} \tag{3}$$

We sometimes graph the multiple objective problem in both decision space and criterion space. $S$ is used to denote the feasible region in the *decision space* and $Z$ is used to denote the feasible region in the *criterion space*.

$$Z = \left\{ z \in Q_q \,\middle|\, z_1 = f_1(\boldsymbol{x}), z_2 = f_2(\boldsymbol{x}), \cdots, z_q = f_q(\boldsymbol{x}), \, \boldsymbol{x} \in S \right\} \tag{4}$$

where $\boldsymbol{x} \in R^k$ is a vector of values of $q$ objective functions. In the other words, $Z$ is the set of images of all points in $S$. Although $S$ is confined to the nonnegative region of $R^n$, $Z$ is not necessarily confined to the nonnegative region of $R^q$.

In principle, MOPs are very different from single objective optimization problems. For the single objective case, one attempts to obtain the best solution, which is absolutely superior to all other alternatives. In the case of multiple objectives, there does not necessarily exist such a solution that is the best with respect to all objectives because of incommensurability and conflict among objectives. A solution may be best in one objective but worst in other objectives. Therefore, there usually exists a set of solutions for the multiple objective cases which cannot be simply compared with each other. Such kind of solutions are called *non-dominated solutions* or *Pareto optimal solutions*, for which no improvement in any objective function is possible without sacrificing on at least one of the other objective functions. For a given non-dominated point in the criterion space $Z$, its image point in the decision space $S$ is called *efficient* or *non-inferior*. A point in $S$ is efficient if and only if its image in $Z$ is non-dominated.

***Definition* 1:** For a given point $z_0 \in Z$, it is non-dominated if and only if there does not exist another point $z \in Z$ such that for the maximization case,

$$\begin{aligned} &z_k > z_k^0, \;\; \text{for some} \;\; k \in \left\{ 1, 2, \cdots, q \right\} \\ &z_l > z_l^0, \;\; \text{for all} \;\; l \ne k \end{aligned}$$

where $z_0$ is a dominated point in the criterion space $Z$ with $q$ objective functions.

***Definition* 2:** For a given point $x_0 \in S$, it is efficient if and only if there does not exist another point $x \in S$ such that for the maximization case,

$$\begin{aligned} &f_k(\boldsymbol{x}) > f_k(\boldsymbol{x}_0), \;\; \text{for some} \;\; k \in \left\{ 1, 2, \cdots, q \right\} \\ &f_l(\boldsymbol{x}) > f_l(\boldsymbol{x}_0), \;\; \text{for all} \;\; l \ne k \end{aligned}$$

where $x_0$ is an inefficient in the decision space $S$ with $q$ objective functions.

## 2.2 Fitness Assignment Mechanism

GAs are essentially a kind of meta-strategy methods. When applying the GAs to solve a given problem, it is necessary to refine upon each of the major components of GAs, such as *encoding methods*, *recombination operators*, *fitness assignment*, *selection operators*, *constraints handling*, and so on, in order to obtain the best solution to the given problem. Because the MOPs are the natural extensions of constrained and combinatorial optimization problems, so many useful methods are based on GAs developed during the past two decades. One of the special issues in the MOPs is fitness assignment mechanism. Since the 1980s, several fitness assignment mechanisms have been proposed and applied in MOPs (Gen *et al*., 2008).

***Type 1: Vector evaluation approach***
Vector evaluated genetic algorithm (VEGA; Schaffer, 1985) is the first notable work to solve multiobjective problems in which it uses a vector fitness measure to create the next generation.

***Type 2: Pareto ranking + Diversity***
***Multiobjective genetic algorithm*** (MOGA): Fonseca and Fleming (1995) proposed a MOGA in which the rank of a certain individual corresponds to the number of individuals in the current population by which it is dominated. Based on this scheme, all the non-dominated individuals are assigned rank 1, while dominated ones are penalized according to the population density of the corresponding region of the tradeoff surface.
***Non-dominated sorting genetic algorithm*** (NSGA): Srinivas and Deb (1995) also developed a Pareto ranking-based fitness assignment and it is called NSGA. In each method, the non-dominated solutions constituting a non-dominated front are assigned the same dummy fitness value. These solutions are shared with their dummy fitness values (phenotypic sharing on the decision vectors) and ignored in the further classification process. Finally, the dummy fitness is set to a value less than the smallest shared fitness value in the current non-dominated front. Then the next front is extracted. This procedure is repeated until all individuals in the population are classified.

***Type 3: Weighted Sum + Elitist Preserve***
***Random-weight genetic algorithm*** (RWGA): Ishibuchi

and Murata (1998) proposed a weighted-sum based fitness assignment method, called RWGA to obtain a variable search direction toward the Pareto frontier. Weighted-sum approach can be viewed as an extension of methods used in the multiobjective optimizations to GAs. It assigns weights to each objective function and combines the weighted objectives into a single objective function.

***Strength Pareto genetic algorithm II*** (SPEA II): Zitzler and Thiele (1999) proposed SPEA and an extended version SPEA II (Zitzler *et al*., 2001) that combines several features of previous MOGA in a unique manner.

***Adaptive-weight genetic algorithm*** (AWGA): Gen and Cheng (2000) proposed another weight sum-based fitness assignment method, called AWGA which utilizes some useful information from the current population to readjust weights to obtain a search pressure toward the Pareto frontier.

***Non-dominated sorting genetic algorithm II*** (NSGA II): Deb (1989, 2001) suggested a non-dominated sorting-based approach, called NSGA II, which alleviates the three difficulties: computational complexity, non-elitism approach, and the need for specifying a sharing parameter. The NSGA II was advanced from its origin, NSGA. In NSGA II, a non-dominated sorting approach is used for each individual to create Pareto rank, and a crowding distance assignment method is applied to implement density estimation.

***Interactive adaptive-weight genetic algorithm*** (i-AWGA): Gen *et al*. (2008) proposed an i-AWGA, which is an improved adaptive-weight fitness assignment approach with the consideration of the disadvantages of weighted-sum approach and Pareto ranking-based approach. They combined a penalty term to the fitness value for all of dominated solutions.

### 2.3 Procedure of Multiobjective Genetic Algorithm

The $P(t)$ and $C(t)$ are parents and offspring respectively in current generation $t$, the implementation structure of multiobjective hybrid GA with combining the fuzzy logic method (Lin and Gen, 2008), local search routine and multiobjective fitness assignment method is described as follows:

**procedure**: multiobjective hybrid GA
**input**: MOP problem data, GA parameters
**output:** Pareto optimal solutions $E$
**begin**
  $t \leftarrow 0$;  // $t$: generation number
  initialize $P(t)$ by encoding routine; // $P(t)$: population
  calculate objectives $z_i(P)$, $i = 1, \cdots, q$ by decoding routine;
  create Pareto $E(P)$ by non-dominated routine;
  calculate $eval(P)$ by fitness assignment routine;
  **while** (**not** termination condition) **do**
    create $C(t)$ from $P(t)$ by crossover routine;
    create $C(t)$ from $P(t)$ by mutation routine; //$C(t)$: offspring
    update $C(t)$ by local search routine;
    calculate objectives $z_i(C)$, $I = 1, \cdots, q$ by decoding routine;
    update Pareto $E(P, C)$ by non-dominated routine;
    calculate $eval(P, C)$ by fitness assignment routine;
    tune GA parameters by fuzzy logic routine;
    select $P(t+1)$ from $P(t)$ and $C(t)$ by selection;
    $t \leftarrow t+1$;
  **end**
  **output** Pareto optimal solutions $E(P, C)$;
**end**;

## 3. FLEXIBLE JOB SHOP SCHEDULING MODELS

### 3.1 Background and Mathematical Model

Flexible job shop is a generalization of the job shop and the parallel machine environment (Pinedo, 2002), which provides a closer approximation to a wide range of real manufacturing systems. In particular, there are a set of parallel machines with possibly different efficiency. The fJSP allows an operation to be performed by any machine in a work center. This presents two problems. The first one is the routing problem (i.e., the assignment of operations to machines), and the second one is the scheduling problem (i.e., determining the starting time of each operation). The fJSP is NP-hard since it is an extension of the JSP (Garay *et al*., 1976). It is a combined assignment and scheduling decision.

- Every machine processes only one operation at a time.
- The execution of each operation requires one machine selected from a set of available machines for the operation.
- The operation sequence of a job is prespecified.
- The operations are not preemptable, that is, once an operation has started it cannot be stopped until it has finished.
- The set-up times for the operations are sequence-independent and are included in the processing times.
- The problem is to assign each operation to an available machine and sequence the operations assigned on each machine in order to minimize the makespan, that is, the time required to complete all jobs. The notations used in this section are summarized as below:

Before introduce the mathematical model some symbols and notations have been defined as follows:

*Indices*
  $i$: index of jobs, $i, h = 1, 2, \cdots, n$;
  $j$: index of machines, $j = 1, 2, \cdots, m$;
  $k$: index of operations, $k = 1, 2, \cdots, K_i$
*Parameters*

$n$: total number of jobs
$m$: total number of machines
$K_i$: total number of operations in job $i$ (or $J_i$)
$J_i$: the $i^{th}$ job
$o_{ik}$: the $k^{th}$ operation of job $i$ (or $J_i$)
$M_j$: the $j^{th}$ machine
$p_{ikj}$: processing time of operation $o_{ik}$ on machine $j$ (or $M_j$)
$U$: a set of machines with the size $m$
$U_{ik}$: a set of available machines for the operation $o_{ik}$
$W_j$: workloads (total processing time) of machine $M_j$

*Decision variables*

$$x_{ikj} = \begin{cases} 1, & \text{if machine } j \text{ is selected for the operation } o_{ik} \\ 0, & \text{otherwise} \end{cases}$$

$c_{ik}$ : completion time of the operation $o_{ik}$

The fJSP model will be formulated as a 0-1 mixed integer programming as follows:

$$\min \quad t_M = \max_{1\le i\le n}\left\{\max_{1\le k\le K_i}\{c_{ik}\}\right\} \tag{5}$$

$$\min \quad W_M = \max_{1\le i\le n}\{W_j\} \tag{6}$$

$$\min \quad W_T = \sum_{j=1}^{m} W_j \tag{6}$$

$$\text{s.t.} \quad c_{ik} - t_{ikj}x_{ikj} - c_{i(k-1)} \ge 0,\ k = 2, \cdots, K_i;\ \forall i,\ j \tag{7}$$

$$\sum_{j=1}^{m} x_{ikj} = 1,\ \forall\ k, i \tag{8}$$

$$x_{ikj} \in \{0, 1\},\ \forall\ j, k, i \tag{9}$$

$$c_{ik} \ge 0,\ \forall\ k, i \tag{8}$$

The first objective function accounts for makespan, Eq. (5) combining with Eq. (6) give a physical meaning to the fJSP, which refer to reducing total processing time and dispatching the operations averagely for each machine. Considering both of the two equations, our objective is to balancing the workloads of all machines. Eq. (7) states that the successive operation has to be started after the completion of its precedent operation of the same job, which represents the operation precedence constraints. Eq. (8) states that one machine must be selected for each operation.

**Table 1.** Processing time of operations

| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ |
|---|---|---|---|---|---|
| $J_1$ | $o_{11}$ | 1 | 3 | 4 | 1 |
| | $o_{12}$ | 3 | 8 | 2 | 1 |
| | $o_{13}$ | 3 | 5 | 4 | 7 |
| $J_2$ | $o_{21}$ | 4 | 1 | 1 | 4 |
| | $o_{22}$ | 2 | 3 | 9 | 3 |
| | $o_{23}$ | 9 | 1 | 2 | 2 |
| $J_3$ | $o_{31}$ | 8 | 6 | 3 | 5 |
| | $o_{32}$ | 4 | 5 | 8 | 1 |

To demonstrate fJSP model clearly, we firstly prepare a simple example. Table 1 give the data set of an fJSP including 3 jobs operated on 4 machines. It is obviously a problem with *total flexibility* because all the machines are available for each operation ($U_{ik} = U$).

## 3.2 Encoding

As mentioned above, fJSP is a combination of assignment and scheduling decisions. It is rather easy to represent the machine selection in a chromosome. We can simply record the index of the machine assigned for an operation in the place where the operation is indicated in a chromosome. Originally, the encoding ideas of fJSP are followed Cheng *et al*. (1996, 1999)'s research work on a tutorial survey of GA for JSP.

Once the assignment decision has been made, fJSP reduces to JSP. Hence, the representation schemes designed for JSP can be used directly to represent the scheduling decision in fJSP. Permutation representation is perhaps the most natural representation of operation sequence, where operations are represented by their operation ID and listed in the order in which they are scheduled. Unfortunately, because of the existence of the precedence constraints, not all the permutations of natural numbers define feasible schedules. During the past few years, the following representations for JSP have been proposed (Gen *et al*., 2008).

***Parallel machine representation (PM-R)***
The chromosome is a list of machines placed in parallel. For each machine, we associate operations to execute. Each operation is coded by three elements: operation $k$, job $J_i$, and $t_{ikj}^S$ (starting time of operation $o_{ik}$ on the machine $M_j$).

***Parallel job representation (PJ-R)***
The chromosome is represented by a list of jobs. Information of each job is showed in the corresponding row where each case is constituted of two terms: machine $M_j$ which executes the operation and corresponding starting time $t_{ikj}^S$.

***Kacem' approach***
Kacem *et al*. (2002b) proposed a new coding: operations machine coding (OMC). It consists in representing the schedule in table $S = (s_{ikj})$. The case $s_{ikj} = 0$ indicates that the $k^{th}$ operation of job $i(o_{ik})$ is not processed on machine $j(M_j)$. In case $M_j$ is assigned for $o_{ik}$, $s_{ikj}$ is filled with the couple ($s_{ik}$, $c_{ik}$), where $s_{ik}$ is the starting time and $c_{ik}$ is the completion time.

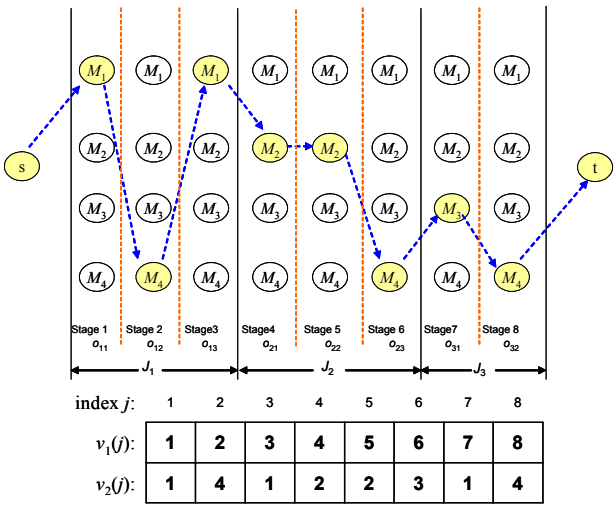
***Priority rule-based representation (PDR)***
In Tanev *et al*. (2004), a PDR-based indirect representation of the schedule for a real-world fJSP is used, where the alleles in chromosome represent the PDR used for assigning the order to the specified machine. Each chromosome (the genotype) is represented as a string $g_0$, $g_1$, $g_2, \cdots, g_{N-1}$, where $N$ is the amount of submitted orders. Schedule builder implements the gene expression mecha-

nism by mapping the chromosome into the corresponding schedule (the phenotype) during the chromosome evaluation phase. Each of the genes of the chromosome is interpreted by schedule builder as follows: "for the currently becoming free machine $M_k$, select all the unscheduled orders that can be currently processed on $M_k$ and range them in accordance with the $g_i^{th}$ PDR; then select the first order $o_j$ from the arranged list of unscheduled orders and assign $o_j$ to $M_k$."

Yang (2001) proposed a new GA-based discrete dynamic programming (DDP) approach for generating static schedules in a FMS environment. Given a sequence of jobs, the operations are scheduled using heuristic method, in which DDP is used to reduce the calculations required to schedule the operations of jobs. Hence, the chromosome only needs to represent the job sequence.

***Multistage operation-based approach***

Zhang and Gen (2005) invented an effective approach to represent the chromosome and also proved to get better performance in solutions. An example of multistage operation-based encoding is shown in Figure 2. Also they reported an effective designing chromosome for optimizing advanced planning and scheduling problem (Brandimarte, 1993).




| index $j$: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $v_1(j)$: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $v_2(j)$: | 1 | 4 | 1 | 2 | 2 | 3 | 1 | 4 |

**Figure 2.** Multistage operation-based representation of flexible job-shop scheduling problem.

### 3.3 Test Problems

Because GAs are of the kind of possibility searching techniques, experiments are necessary in order to test the efficiency of those algorithms. Test problems used in the early works, e.g., (Dauzere-Pares and Paulli, 1997; Gao *et al*., 2006) are taken as benchmarks for the successive works. Realistic FMS contexts usually have many more complexities that must be handled during the static scheduling. Other factors that could be of importance are the setup times at each station, the number of loading and unloading docks available, the time taken by each dock to perform its loading and or unloading operation, the topology and buffer capacities of the material handling systems, and the transfer time from one station to another (where applicable). For example, the transfer times between different stations are considered in Kacem *et al*. (2002b), and randomly generated test problems are used to test the efficiency of the proposed approach. Moreover, genetic techniques have all been proven to be effective methods for multiobjective optimization problem. In Yang (2001), the objective considered is to minimize the overall completion time (makespan), the total workload of machines and the workload of the most loaded machine. A 10-job 10-machine problem is presented by Kacem *et al*. (2002b).

Table 2 illustrates the results comparing with other researcher's approaches with proposed approach in Gao *et al*. (2006) by using published for the benchmark problems, where "a" denotes the approach "AL+CGA" proposed by Kacem *et al*. (2002b), while "b" denotes the approach "PSO+SA" proposed by Xia and Wu (2005).

**Table 2.** Results comparing with other approaches

| | sGA | AL+CGAa | PSO+SAb | Proposed approach |
|---|---|---|---|---|
| Makespan | 7 | 7 | 7 | 7 |
| $W_T$ | 53 | 45 | 44 | 43 |
| Max ($W_k$) | 7 | 5 | 6 | 5 |

GA: genetic algorithm, AL: approach by localization, CGA: controlled genetic algorithm, PSO: particle swarm optimization, SA: simulated annealing.

## 4. AGV DISPATCHING MODELS

### 4.1 Background and Mathematical Model

For a recent review on AGV problems and issues, the reader is referred to Vis (2006), Le-Ahn and De Koster (2006), Lim (2004), and Hwang *et al*. (2002). An AGV is a driverless transport system used for horizontal movement of materials. AGVs were introduced in 1955. The use of AGVs has grown enormously since their introduction. AGV systems are implemented in various industrial contexts: container terminals, part transportation in heavy industry, manufacturing systems (Kim and Hwang, 1999). In fact, new analytical and simulation models need to be developed for large AGV systems to overcome: large computation times, NP-completeness, congestion, deadlocks and delays in the system and finite planning horizons (Kim and Hwang, 2001; Moon and Hwang, 1999; Proth *et al*., 1997).

We consider a dispatching AGV system that each AGV transports the material (or semi-product) between working stations. Assumptions considered in this paper are as follows:

1) AGVs only carry one kind of products at the same time.
2) A network of guide paths is defined in advance, and the guide paths have to travel through all of pickup/delivery points.
3) The vehicles are assumed to travel at a constant speed.
4) The vehicles just can travel forward, not backward.
5) As many vehicles travel on the guide path simultaneously, collisions be avoided by hardware, not be considered in this paper.
6) On each working stations, there are pickup space for store the operated material and delivery space for store the material for next operation.
7) The operation can be started any time after an AGV take the material to come. And also the AGV can transport the operated material form the pickup point to next delivery point any time.

In this paper, the problem is to dispatch AGVs for transports the product between different machines in a FMS. At first stage, we model the problem by using network structure.

Assumptions considered in this paper are as follows:

For scheduling:
1) In a FMS, $n$ jobs are to be scheduled on $m$ machines.
2) The $i^{th}$ job has $n_i$ operations that have to be processed.
3) Each machine processes only one operation at a time.
4) The set-up time for the operations is sequence-independent and is included in the processing time.

For AGV dispatching:
1) Each machine is connected to the guide path network by a pick-up/delivery (P/D) station where pallets are transferred from/to the AGVs.
2) The guide path is composed of aisle segments on which the vehicles are assumed to travel at a constant speed.
3) As many vehicles travel on the guide path simultaneously, collisions be avoided by hardware, not be considered in this paper.

Subject to the constraints that,

For scheduling:
1) The operation sequence for each job is prescribed;
2) Each machine can process only one operation at a time;
3) Each AGV can transport only one kind of products at a time.

For AGV dispatching:
1) AGVs only carry one kind of products at same time.
2) The vehicles just can travel forward, not backward.

The objective function is minimizing the following two criteria:
1) Time required to complete all jobs (i.e., make-span): $t_{MS}$
2) Number of AGVs: $n_{AGV}$

The notation used in this paper is summarized in the following:

Indices
$i, i'$ :index of jobs, $i$, $i' = 1, 2, \cdots, n$;
$j, j'$ :index of processes, $j$, $j' = 1, 2, \cdots, n$;

Parameters
$n$ : total number of jobs;
$m$ : total number of machines;
$n_i$ : total number of operation of job $j$;
$o_{ij}$ : the $j^{th}$ operation of job $i$;
$p_{ij}$ : processing time of operation $o_{ij}$;
$M_{ij}$ : machine assigned for operation $o_{ij}$
$T_{ij}$ : transition task for operation $o_{ij}$;
$t_{ij}$ : transition time from $M_{ij-1}$ to $M_{ij}$;

Decision variables
$x_{ij}$ : assigned AGV number for task $T_{ij}$
$t_{ij}^S$ : starting time of task $T_{ij}$;
$c_{ij}^S$ : starting time of operation $o_{ij}$;

**Example 1:** An FMS has four machines 1, 2, 3, and 4. Three job types $J_1$, $J_2$, and $J_3$ are to be carried out, and Table 3 shows the requirements for each job (Naso and Turchiano, 2005).

The first process of $J_1$ is carried out at machine 1 with $p_{11}$ (60 processing times). The second process of $J_1$ be carried out at machine 2 with $p_{12}$ (80 processing), this table also gives the precedence constraints among the operations $O_{ij}$ in each job $J_i$. For instance, the second process of $J_1$ can be carried out only after the first process of $J_1$ is complete. Notice that $J_2$ has only two processes to be completed. Figure 3 shows Gantt chart of the schedule of Example 1 without considering AGVs routing.

**Table 3.** Job requirements of Example 1 (Goncalves *et al.*, 2005)

| $O_{ij}$ | $M_{ij}$ | | | | $p_{ij}$ | | | |
|---|---|---|---|---|---|---|---|---|
| $J_i$ \ $P_j$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ |
| $J_1$ | 1 | 2 | 3 | 4 | 60 | 80 | 100 | 70 |
| $J_2$ | 4 | 3 | - | - | 100 | 40 | - | - |
| $J_3$ | 1 | 4 | 2 | - | 70 | 20 | 50 | - |

$M_{ij}$: machine # of assigned $P_j$-th operation $O_{ij}$ of job $J_i$.
$p_{ij}$: processing time of operation $O_{ij}$.

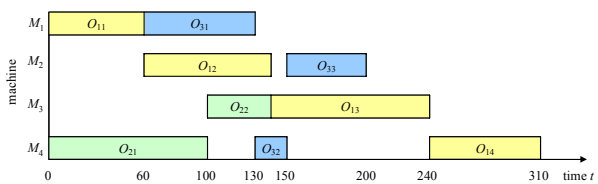



**Figure 3.** Gantt chart of the schedule of Example 1 without considering automatic guided vehicles routing.

For designing an AGV system in manufacturing system, the transition time $t_{uv}/c_{vu}$ between pick-up point on machine $u$ and delivery point on machine $v$ are defined in Table 4, and they depend on Naso and Turchiano (2005). We give a layout of facility for Example 1 in Figure 4. Note, although the network of guide paths is unidirectional, it has to take a very large transition time from pickup point (P) to delivery point (D) on same machine. It is unnecessary in the real application. So we defined an inside cycle for each machine, that is the transition time is the same as P to D and D to P. We give a routing example to carry out job $J_1$ by using one AGV in Figure 5.

***Definition* 3:** A node is defined as task $T_{ij}$ that presents a transition task of $j^{th}$ process of job $J_i$ for moving pick-up point of machine $M_{i,\,j\text{-}1}$ to delivering point of machine $M_{ij}$.

***Definition* 4:** An arc can be defined as many decision variables such as capacity of AGVs, precedence constraints among the tasks, costs of movement. In this paper, we defined an arc as precedence constraint, and give a transition time $c_{jj'}$ from delivery point of machine $M_{ij}$ to pick-up point of machine $M_{i'j'}$ on the arc.

**Table 4.** Transition time between pick-up point on machine $u$ and delivery point on machine $v$

| $t_{uv}/c_{uv}$ | | Loading/ Unloading | Machine number | | | |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| Loading/Unloading | | 1/1 | 1/7 | 8/13 | 16/23 | 18/20 |
| Machine number | 1 | 13/18 | 3/3 | 2/9 | 10/19 | 13/18 |
| | 2 | 18/22 | 22/28 | 2/2 | 4/13 | 12/18 |
| | 3 | 8/14 | 12/20 | 18/26 | 3/3 | 2/10 |
| | 4 | 5/7 | 9/12 | 15/18 | 23/28 | 2/2 |

$t_{uv}$: transition time from pick-up point on machine $u$ to delivery point on machine $v$.
$c_{uv}$: transition time from delivery point on machine $u$ to pick-up point on machine $v$.

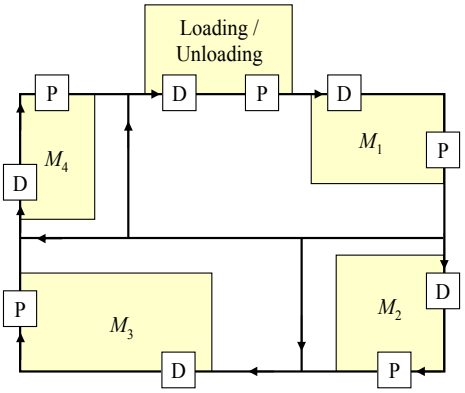



**Figure 4.** Layout of facility. P: pick-up point, D: delivery point.

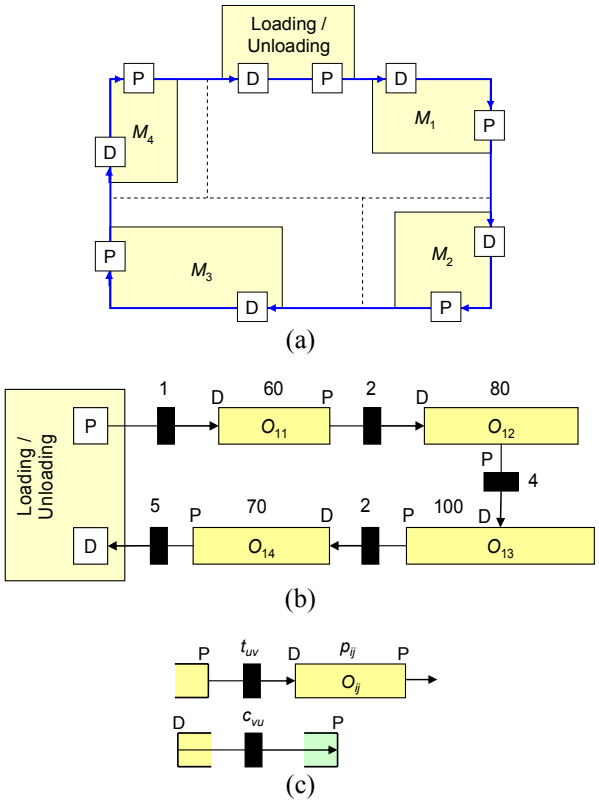



**Figure 5.** A routing example for carry out job $J_1$ by using one automatic guided vehicle: (a) illustration of facility layout, (b) illustration of transition flow, and (c) symbol definitions. P: pick-up point, D: delivery point.

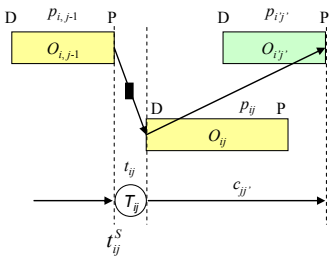



**Figure 6.** Illustration of problem representations.

***Definition* 3:** We define the task precedence for each job.

Job $J_1$: $T_{11} \succ T_{12} \succ T_{13} \succ T_{14}$
Job $J_2$: $T_{21} \succ T_{22}$
Job $J_3$: $T_{31} \succ T_{32} \succ T_{33}$

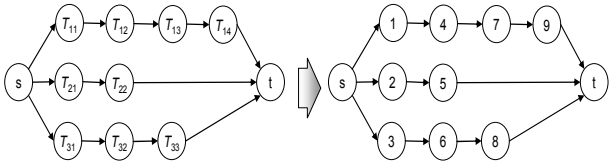



**Figure 7.** Illustration of the network structure of Example 1.

We can draw a network (as Figure 7) depending on the precedence constraints among tasks $\{T_{ij}\}$. The objective of this network problem assigns all of tasks to several AGVs, and gives the priority of each task to make the AGV routing sequence. A result of Example 1 is shown as follows, and the final time required to complete all jobs (i.e., makespan) is 321, and 3 AGVs are used. Figure 8 shows the result on Gantt chart.

AGV1: $T_{11} \rightarrow T_{12} \rightarrow T_{13} \rightarrow T_{14}$
AGV2: $T_{21} \rightarrow T_{22}$
AGV3: $T_{31} \rightarrow T_{32} \rightarrow T_{33}$

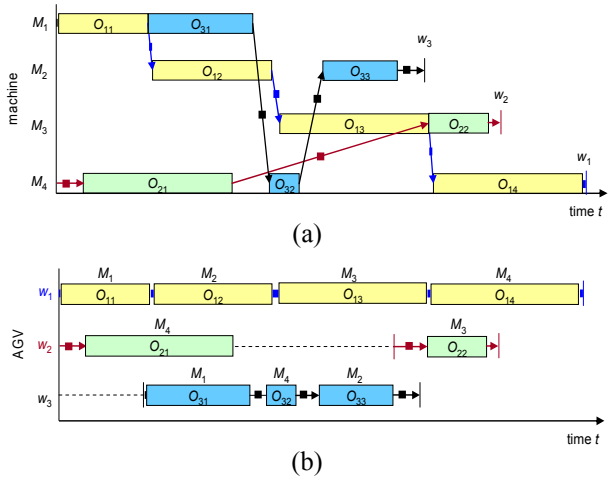



**Figure 8.** Gantt chart of the schedule of Example 1 with considering automatic guided vehicles (AGVs) routing ($e_i$: AGV). Based on operations processing (a) and AGVs dispatching (b).

The AGV dispatching problem can be formulated by the multiobjective optimization model as follows:

$$\min \quad t_{\mathrm{MS}} = \max_i \left\{ t^S_{i,n_i} + t_{M_{i,n_i},0} \right\} \tag{11}$$

$$\min \quad n_{\mathrm{AGV}} = \max_{i,j} \left\{ x_{ij} \right\} \tag{12}$$

$$\text{s.t.} \quad c^S_{ij} - c^S_{i,j-1} \geq p_{i,j-1} + t_{ij}, \quad \forall i,\ j = 2, \cdots, n_i \tag{13}$$

$$\left( c^S_{ij} - c^S_{i'j'} - p_{i'j'} + \Gamma \left| M_{ij} - M_{i'j'} \right| \geq 0 \right) \vee \left( c^S_{i'j'} - c^S_{ij} - p_{ij} + \Gamma \left| M_{ij} - M_{i'j'} \right| \geq 0 \right),\ \forall (i, j), (i', j') \tag{14}$$

$$\left( t^S_{ij} - t^S_{i'j'} - t_{i'j'} + \Gamma \left| x_{ij} - x_{i'j'} \right| \geq 0 \right) \vee \left( t^S_{i'j'} - t^S_{ij} - t_{ij} + \Gamma \left| x_{ij} - x_{i'j'} \right| \geq 0 \right),\ \forall (i, j), (i', j') \tag{15}$$

$$\left( t^S_{i,n_i} - t^S_{i'j'} - t_{i'j'} + \Gamma \left| x_{ij} - x_{i'j'} \right| \geq 0 \right) \vee \left( t^S_{i'j'} - t^S_{i,n_i} - t_i + \Gamma \left| x_{ij} - x_{i'j'} \right| \geq 0 \right),\ \forall (i, n_i), (i', j') \tag{16}$$

$$c^S_{ij} \geq t^S_{i,j+1} - p_{ij},\ \forall i,\ j \tag{17}$$

$$x_{ij} \geq 0,\ \forall i,\ j \tag{18}$$

$$t^S_{ij} \geq 0,\ \forall i,\ j \tag{19}$$

where $\Gamma$ is a very large number, and $t_i$ is the transition time for pick-up point of machine $M_{i,n_i}$ to delivery point of loading/unloading. Inequality (13) describes the operation precedence constraints. In inequities (14)-(16), since one or the other constraint must hold, it is called disjunctive constraint. It represents the operation unoverlapping constraint (Inequality 14) and the AGV nonoverlapping constraint (Inequalities 15, 16).

## 4.2 Priority-Based GA

For solving the AGV dispatching problem in FMS, the special difficulty arises from 1) the task sequencing is *NP-hard problem*, and 2) a random sequence of AGV dispatching usually does not correspond to the operation precedence constrain and routing constrain.

In this paper, we firstly give a priority-based encoding method that is an indirect approach: encode some guiding information for constructing a sequence of all tasks. As it is known, a gene in a chromosome is characterized by two factors: locus, i.e., the position of gene located within the structure of chromosome, and allele, i.e., the value the gene takes. In this encoding method, the position of a gene is used to represent task ID and its value is used to represent the priority of the task for constructing a sequence among candidates. A feasible sequence can be uniquely determined from this encoding by considering operation precedence constraint. An example of generated chromosome and its decoded path is shown in Figure 9, for Example 1 (in Section 2).

| Task ID : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Priority : | 1 | 5 | 7 | 2 | 6 | 8 | 3 | 9 | 4 |

$T_{11} \rightarrow T_{12} \rightarrow T_{13} \rightarrow T_{14} \rightarrow T_{21} \rightarrow T_{22} \rightarrow T_{31} \rightarrow T_{32} \rightarrow T_{33}$

**Figure 9.** Example of generated chromosome and its decoded task sequence.

At the beginning, we try to find a task for the position next to source node s. Task $T_{11}$, $T_{21}$, and $T_{31}$ (task ID: 1, 2, and 3) are eligible for the position, which can be easily fixed according to adjacent relation among tasks. Their priorities are 1, 5, and 7, respectively. The node 1 has the highest priority and is put into the task sequence. The possible tasks next to task $T_{11}$, is task $T_{12}$ (task ID: 4) and unselected task $T_{21}$ and $T_{31}$ (task ID: 2 and 3). Because node 4 has the largest priority value, it is put into the task sequence. Then we form the set of tasks available for the next position and select the one with the highest priority among them. Repeat these steps until all of the tasks are selected,

$$T_{11} \rightarrow T_{12} \rightarrow T_{13} \rightarrow T_{14} \rightarrow T_{21} \rightarrow T_{22} \rightarrow T_{31} \rightarrow T_{32} \rightarrow T_{33}$$

After generating the task sequence, we secondly separate tasks to several groups for assigning different AGVs. First, separate tasks with a separate point in which the task is the final transport of job $i$ form pick-up point

of operation $O_{i,n_i}$ to delivery point of loading/unloading. Afterward, unite the task groups in which finished time of a group is faster than the starting time of another group. The particular is introduced in next subsection. An example of grouping is shown as follows by using the chromosome (Figure 9), for Example 1.

AGV1: $T_{11} \rightarrow T_{12} \rightarrow T_{13} \rightarrow T_{14}$
AGV2: $T_{21} \rightarrow T_{22}$
AGV3: $T_{31} \rightarrow T_{32} \rightarrow T_{33}$

### 4.3 Case Study

For evaluating the efficiency of the AGV dispatching algorithm suggested in a case study, a simulation program was developed by using Java on Pentium 4 processor (3.2-GHz clock). The problem was used by Yang (2001) and Kim *et al.* (2004). GA parameter settings were taken as follows: population size, *popSize* = 20; crossover probability, $p_C$ = 0.70; mutation probability, $p_M$ = 0.50; immigration rate, $\mu$ = 0.15.

In a case study of FMS, 10 jobs are to be scheduled on 5 machines. The maximum number process for the operations is 4. Table 5 gives the assigned machine numbers and process time. And Table 6 gives the transition time among pick-up points and delivery points. Depending on Naso and Turchiano (2005), we give a layout of facility for the experiment in Figure 10.

**Table 5.** Job requirements of Example 2

| $O_{ij}$ | $M_{ij}$ | | | | $p_{ij}$ | | | |
|---|---|---|---|---|---|---|---|---|
| $J_i$ \ $P_j$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ |
| $J_1$ | 1 | 2 | 1 | - | 80 | 120 | 60 | - |
| $J_2$ | 2 | 1 | - | - | 100 | 40 | - | - |
| $J_3$ | 5 | 3 | 3 | - | 70 | 100 | 70 | - |
| $J_4$ | 5 | 3 | 2 | 2 | 70 | 100 | 100 | 40 |
| $J_5$ | 4 | 2 | - | - | 90 | 40 | - | - |
| $J_6$ | 4 | 4 | 1 | 2 | 90 | 70 | 60 | 40 |
| $J_7$ | 1 | 3 | - | - | 80 | 70 | - | - |
| $J_8$ | 5 | 4 | 5 | 4 | 70 | 70 | 70 | 80 |
| $J_9$ | 5 | 4 | 1 | - | 70 | 70 | 60 | - |
| $J_{10}$ | 5 | 1 | 3 | - | 70 | 60 | 70 | - |

**Table 6.** Transition time between pick-up point *u* and delivery point *v*

| $t_{uv}/c_{uv}$ | Loading/ Unloading | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ |
|---|---|---|---|---|---|---|
| Loading/ Unloading | 1/1 | 1/7 | 8/13 | 14/18 | 16/23 | 18/20 |
| $M_1$ | 13/18 | 3/3 | 2/9 | 8/14 | 10/19 | 13/18 |
| $M_2$ | 18/22 | 22/28 | 2/2 | 2/7 | 4/12 | 12/18 |
| $M_3$ | 13/11 | 17/22 | 24/29 | 1/1 | 1/6 | 7/11 |
| $M_4$ | 8/14 | 12/20 | 18/26 | 24/29 | 3/3 | 2/10 |
| $M_5$ | 5/7 | 9/12 | 15/18 | 19/23 | 23/28 | 2/2 |

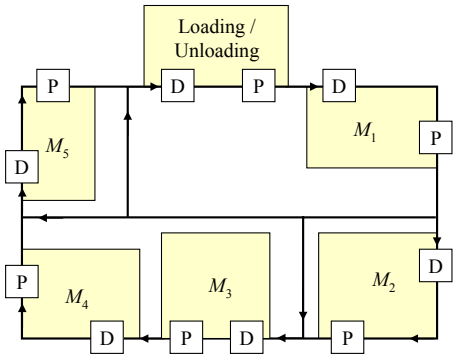



**Figure 10.** Layout of facility. P: pick-up point, D: delivery point.

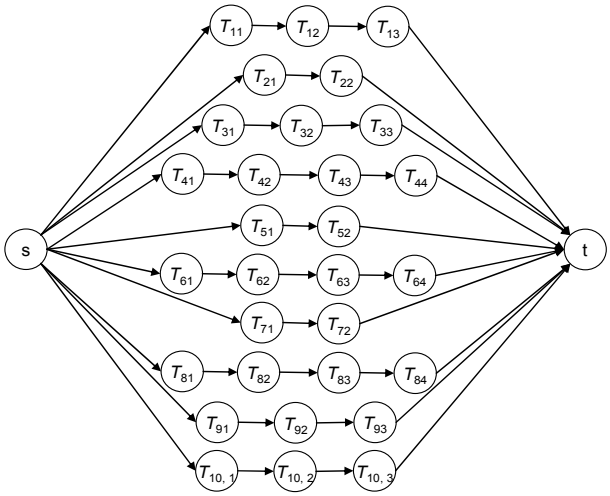
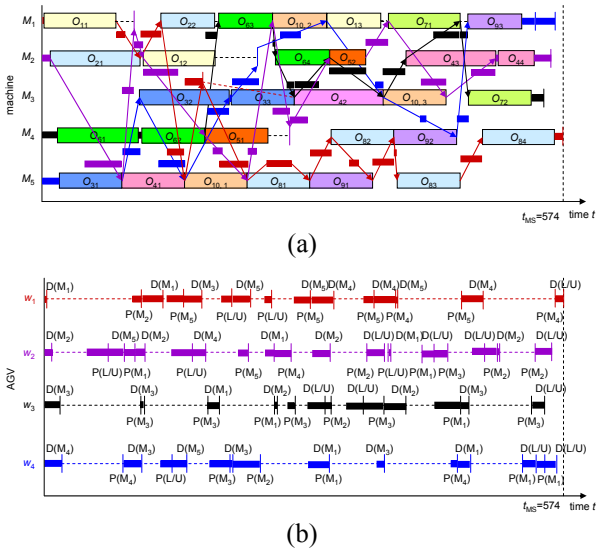



**Figure 11.** Illustration of the network structure of Example 2.

We can draw a network (as Figure 11) depending on the precedence constraints among tasks $\{T_{ij}\}$ of Example 2. The best result of Example 2 is shown as follows, and the final time required to complete all jobs (i.e., makespan) is 574, and 4 AGVs are used. Figure 12 shows the result on Gantt chart.

AGV1: $T_{11} \rightarrow T_{22} \rightarrow T_{41} \rightarrow T_{81} \rightarrow T_{91} \rightarrow T_{82} \rightarrow T_{92} \rightarrow T_{83} \rightarrow T_{84}$
AGV2: $T_{21} \rightarrow T_{41} \rightarrow T_{12} \rightarrow T_{15} \rightarrow T_{10,2} \rightarrow T_{52} \rightarrow T_{71} \rightarrow T_{44}$
AGV3: $T_{61} \rightarrow T_{62} \rightarrow T_{63} \rightarrow T_{64} \rightarrow T_{43} \rightarrow T_{72}$
AGV3: $T_{31} \rightarrow T_{32} \rightarrow T_{10,1} \rightarrow T_{33} \rightarrow T_{13} \rightarrow T_{10,3} \rightarrow T_{93}$

For this case, the makespan and the number of AGVs used are two conflicting elements. Yet, we can use 4 AGVs to achieve the minimum makespan, that is, more AGVs are no use in decreasing the makespan. When we use three or less AGVs, the makespan is unacceptably long, hence 4 vehicles is quite a satisfactory number of AGVs used. To test the performance of our algorithm, it is necessary to conduct larger-size problems and more realistic factors should be considered. These issues are among our future research subjects.




(a)

(b)

**Figure 12.** Gantt chart of the schedule of Example 2 with considering automatic guided vehicles (AGVs) routing. Based on operations processing (a) and AGVs dispatching (b).

Lin *et al.* (2012b) recently proposed a random key-based PSO algorithm with crossover and mutation operation to avoid premature convergence and to maintain diversity of the swarm. Numerical analyses for case study show the effectiveness of the proposed approach comparing with GA.

## 5. ADVANCED PLANNING AND SCHEDULING MODELS

### 5.1 Background and Mathematical Model

Recently, Gen *et al.* (2009b) surveyed EAs in APS. The APS problem includes finding the optimal resource selection for operations, operations sequences, allocation of variable transfer batches, and schedules considering flexible flows, resources status, capacities of plants, precedence constraints, and workload balance. The relationship between all the main parts of APS problems in MPC is shown in Figure 13. We find the process is driven since the orders come from our customer. Moreover, to satisfy the requirements, some other constraints should be considered such as due date, set-up time and shipping time. The main integrated model involves the time allocation for operations based on the selected operations sequences to minimize the makespan. Thus, the operations sequencing problem should be integrated into the scheduling problem. We allow any two operations belonging to the same order proceed concurrently if resources are available. In multi-plant environment, we should also consider the situation of outsourcing that means the same order can also be delivered to some other plant for assigning the resources in different locations; therefore, a lot of orders can be divided further for the subsequent operations if it is allowed by the load size of transfer device. In this case, the transfer batch and process batch may not coincide. On the other hand, if inter-plants transportation is needed, then the transfer batch should be equal to the process batch.

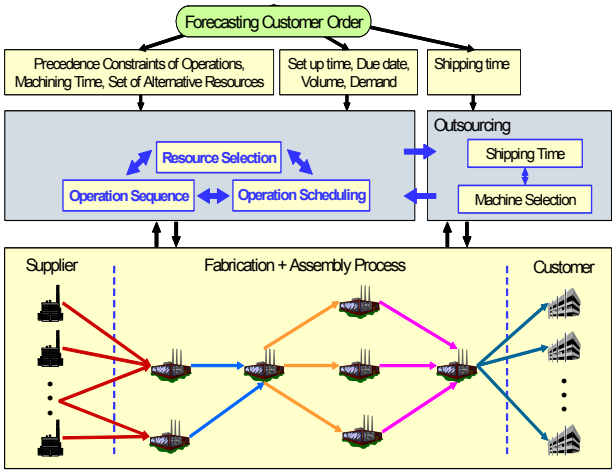



**Figure 13.** Modern advanced planning and scheduling system.

To clearly demonstrate the process planning in the APS problem in detail, we create a simple example shown in Figure 14, which presents the two kinds of materials to be machined in a modern manufacturing system with the lot sizes 40 and 50 orders by the customer. Concretely, 10 volumes should be removed from two materials for obtaining the final two products. All the manufacturing plans of this example are offered in Table 7, which includes all the types of operations and the corresponding machine selection.

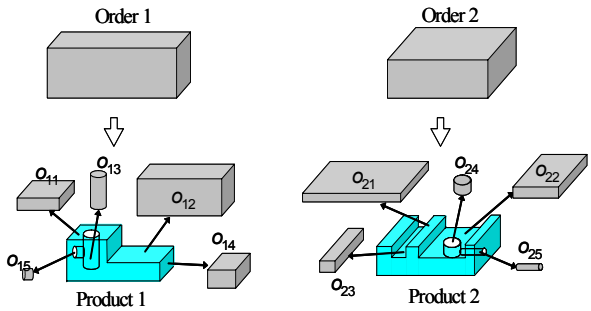



**Figure 14.** Simple example for process planning problem.

In Figure 15, we describe the operation sequence constrains of the two orders by using node graphs, and, we can confirm that the operation sequence follows some precedence constraints for each order. For instance, in order 1: both {o11, o13, o12, ⋯} and {o11, o12, o13, ⋯} are legal, while {o13, o11, o12, ⋯} is illegal.

Tables 8 and 9 prepare the data set of processing time for each operation and transition time between different machines. Table 9 shows that five machines have

different capabilities for each corresponding operation, and are not available for some of them. In addition, concerning this simple processing planning example, we do not consider the setup time, and both of the orders are completed in one plant.

**Table 7.** Manufacturing plan

| Operation Index (Order $k$, Operation $O_{ki}$) | Operation Type | Machine Selection $M_m$ |
|---|---|---|
| 1(1, $o_{11}$) | milling | $M_1$, $M_4$ |
| 2(1, $o_{12}$) | milling | $M_1$, $M_4$ |
| 3(1, $o_{13}$) | dilling | $M_2$, $M_3$, $M_5$ |
| 4(1, $o_{14}$) | milling | $M_1$ |
| 5(1, $o_{14}$) | dilling | $M_2$ |
| 6(2, $o_{21}$) | milling | $M_1$, $M_2$ |
| 7(2, $o_{22}$) | milling | $M_1$, $M_3$ |
| 8(2, $o_{23}$) | milling | $M_3$, $M_5$ |
| 9(2, $o_{24}$) | dilling | $M_1$, $M_4$, $M_5$ |
| 10(2, $o_{25}$) | dilling | $M_1$, $M_2$ |

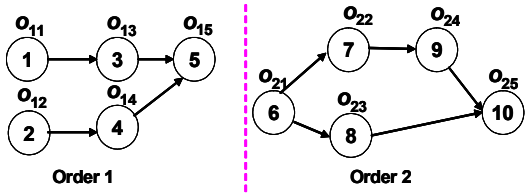



**Figure 15.** Operation precedence constraints of two orders.

**Table 8.** Processing time $p_{kim}$ of operations

| | Order 1 | | | | | Order 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Operation / Machine | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $M_1$ | 7 | 7 | - | 6 | - | 3 | 8 | - | 10 | 6 |
| $M_2$ | - | - | 6 | - | 9 | 5 | - | - | - | 5 |
| $M_3$ | - | - | 5 | - | - | - | 12 | 5 | - | - |
| $M_4$ | 5 | 6 | - | - | - | - | - | - | 10 | - |
| $M_5$ | - | - | 8 | - | - | - | - | 8 | 7 | - |

**Table 9.** The transition time $t^S_{mn}$ between different machines

| $M_m$ \ $M_n$ | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ |
|---|---|---|---|---|---|
| $M_1$ | - | 7 | 27 | 26 | 17 |
| $M_2$ | 19 | - | 15 | 19 | 10 |
| $M_3$ | 5 | 17 | - | 36 | 27 |
| $M_4$ | 8 | 20 | 4 | - | 30 |
| $M_5$ | 18 | 18 | 14 | 5 | - |
| Available capacity | 1500 | 1500 | 1500 | 1500 | 1500 |

A process plan should also be able to represent all the possible precedents that occur during the planning and processing decisions. From an unordered set of operations with precedence relations, the operations sequencing is to determine a sequence considering the combination of parallel processes and alternative resources for operations. Nevertheless, we should clarify that the simple example in this section only indicates the planning horizon in a single plant, while the APS in multi-plant chain will be solved in the experimental section.

As shown previously, the objective of an APS system in a multi-plant chain is usually to determine an optimal schedule with operation sequences for all the orders (jobs). That is, the problem we are treating can be defined as: there are a set of $K$ orders which are to be processed on $N$ machines with alternative operations sequences and alternative machines for operations in the environment of the multi-plant chain, we want to find an operations sequence for each job and a schedule in which jobs pass between machines and a schedule in which operations on the same jobs are processed such that it satisfies the precedence constraints and it is optimal with respect to the makespan minimization.

To formulate the mathematical model, some notations and symbols are defined firstly as follows:

*Indices*

$i, j$: index of operation number, $i, j = 1, 2, \cdots, J_k$
$k, l$: index of orders, $k, l = 1, 2, \cdots, K$
$m, n$: index of machines, $m, n = 1, 2, \cdots, N$
$d, e$: index of plants , $d, e = 1, 2, \cdots, D$

*Parameters*

$K$: number of orders
$D$: number of plants
$N$: number of machines
$O_k$: set of operations for order $k$, i.e.,
$O_k = \{o_{ki} | i = 1, 2, \cdots, J_k\}$
$o_{ki}$: the $i^{\text{th}}$ operation for order $k$
$J_k$ : number of operations for order $k$
$M_m$: the $m^{\text{th}}$ machine
$q_k$: lot size of order $k$
$A_m$: set of operations that can be processed on machine $m$
$p_{kim}$: unit processing time of operation $o_{ki}$ on machine $m$
$L_m$: capacity of machine $m$
$r_{kij}$: precedence constraints

$$r_{kij} = \begin{cases} 1, & \text{if } o_{ki} \text{ is predecessor of } o_{kj} \text{ in order } k \\ 0, & \text{otherwise} \end{cases}$$

$$r_{kij} = \begin{cases} 1, & \text{if } o_{ki} \text{ is predecessor of } o_{kj} \text{ in order } k \\ 0, & \text{otherwise} \end{cases}$$

$b_{md}$: capability of resources in plant $d$

$$b_{md} = \begin{cases} 1, & \text{if } m\text{th machine belongs to plant } d \\ 0, & \text{otherwise} \end{cases}$$

$t^U_{kilj}$ : set-up time from operation $o_{ki}$ to operation $o_{lj}$
$u_{kij}$: unit load size of order $k$ from operation $o_{ki}$ to operation $o_{kj}$
$c_M$: total makespan for all the orders

$d_D$: due date
$B_d$: set of machines that are included in plant $d$
$t_{mn}$ : unit shipping time between machine $m$ to machine $n$

$$t_{mn} = \begin{cases} t_{mn}^S, \text{ if resources } m \text{ and } n \text{ are included in the same plant,} \\ t_{mn}^D, \text{ otherwise.} \end{cases}$$

$v_{kij}$: number of shipping times from operation $o_{ki}$ to operation $o_{kj}$

$$v_{kij} = \begin{cases} \left\lceil \frac{q_k}{u_{kij}} \right\rceil, \text{ if resources } m \text{ and } n \\ \qquad \text{are included in the same plant,} \\ 1, \text{ otherwise.} \end{cases}$$

$T_{kij}$: transition time from operation $o_{ki}$ to operation $o_{kj}$

$$T_{kij} = \sum_{m=1}^{N} \sum_{n=1}^{N} \{q_k p_{kim} x_{kim} + v_{kij} t_{mn} x_{kim} x_{kjn} + t_{kij}^U y_{kij}\}$$

$c_{ki}$ : completion time of operation $o_{ki}$

$$c_{ki} = s_{ki} + q_k \sum_{m=1}^{N} p_{kim} x_{kim}$$

*Decision variables*

$$y_{kilj} = \begin{cases} 1, & \text{if opration } o_{ki} \text{ is performed} \\ & \text{immediately before operation } o_{lj} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{kim} = \begin{cases} 1, & \text{if machine } m \text{ is selected for operation } o_{ki} \\ 0, & \text{otherwise} \end{cases}$$

$s_{ki}$ : starting time of operation $o_{ki}$

The objective to APS problem in this paper is to minimize total makespan $t_M$, the overall model can be described as follows:

$$\min \quad c_M = \max_{k,i} \{c_{ki}\} \tag{20}$$

$$\text{s. t.} \quad \left\{s_{lj} - \left(c_{ki} + t_{kilj}^U\right)\right\} x_{kim} x_{ljm} y_{kilj} \geq 0 \quad \forall (k,i),(l,j),m \tag{21}$$

$$\left\{s_{kj} - \left(s_{ki} + u_{kij} p_{kim} + t_{mn}^S\right)\right\} r_{kij} x_{kim} x_{kjn} \geq 0 \quad \forall i,j,k,\forall m,n \in B_d,\forall d \tag{22}$$

$$\left\{c_{kj} - u_{kij} p_{kjn} - \left(c_{ki} + t_{mn}^S\right)\right\} r_{kij} x_{kim} x_{kjn} \geq 0 \quad \forall i,j,k,\forall m,n \in B_d,\forall d \tag{23}$$

$$\left\{s_{kj} - \left(c_{ki} + t_{mn}^D\right)\right\} r_{kij} x_{kim} x_{kjn} \geq 0 \quad \forall i,j,k,\ \forall m \in B_d,\ \forall n \in B_e,\ \forall d,e \tag{24}$$

$$\sum_{k=1}^{K} \sum_{i=1}^{J_k} q_k p_{kim} x_{kim} \leq L_m \quad \forall m \tag{25}$$

$$r_{kij} y_{kjki} = 0 \quad \forall i,j,k \tag{26}$$

$$y_{kiki} = 0 \quad \forall i,k \tag{27}$$

$$y_{kilj} + y_{ljki} = 1 \quad \forall (k,i),(l,j),\ (k,i) \neq (l,j) \tag{28}$$

$$\sum_{m=1}^{N} x_{kim} = 1 \quad \forall i,k \tag{29}$$

$$x_{kim} = 0 \quad \forall (k,i) \notin A_m,\ \forall m \tag{30}$$

$$s_{ki} \geq 0 \quad \forall i,k \tag{31}$$

$$y_{kilj} \in \{0,1\} \quad \forall (k,i),(l,j) \tag{32}$$

$$x_{kim} \in \{0,1\} \quad \forall i,k,m \tag{33}$$

Eq. (21) imposes that for any resource (machine), it cannot be selected for one operation until the predecessor is completed and also set-up time must be considered (Figure 16).

Eqs. (22) and (23) impose the transportation instances in local plant. Both of the two constraints must be satisfied simultaneously to ensure operations can be run uninterrupted on one machine (Figure 17).

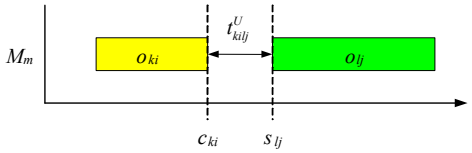



**Figure 16.** Time chart for the constraints on the same machine.

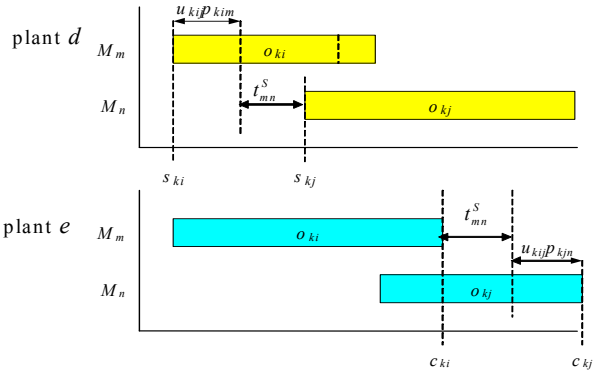



**Figure 17.** Time chart for the constraints in the same plant.

Eq. (24) restricts the other transportation between different plants (in MPC environment), which indicates that the operation cannot move to another plant until all the lot size have been finished (Figure 18).

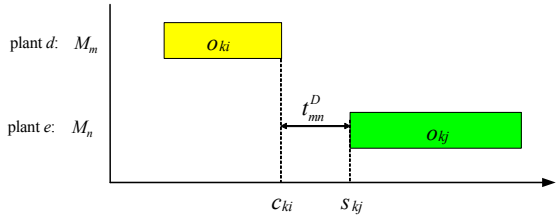



**Figure 18.** Time chart for the constraints in different plants.

Eq. (25) restricts the available capacity for each machine. Eq. (26) ensures that the precedence constraints are not violated. Eqs. (27) and (28) ensure the feasible operation sequence. Eqs. (29) and (30) ensure the feasible resource selection. Eqs. (31)–(33) impose nonnegative condition.

### 5.2 Multistage Operation-Based GA

Following the development of GA, neither the optimization of GA's parameter (crossover rate and mutation rate) setting, nor the approach of GA's operators (crossover approach and mutation approach) can significantly improve the effectiveness of the algorithm. Hence, more and more researchers tried to find an optimal designing of chromosome, which contains more information and can also improve both effectiveness and efficiency of the algorithm to the corresponding combinatorial optimization problem. Especially, some researchers used two dimensional schemes: Ulusoy *et al*. (1997) used a two-array representation (one for operation sequencing and the other for AGV assignment), Goncalves *et al*. (2005) also used a two string representation, one for operation priorities and the other for delay times.

Originally, our idea of moGA comes from the basic concept of a multistage decision making model shown in Figure 19. There are several stages separating the route from the starting node to the terminal node, and in each stage several states are offered to be chosen from. After we make all the decisions for choosing states, we can draw a solution, and the fitness of the result is in terms of the different decisions made along the route.

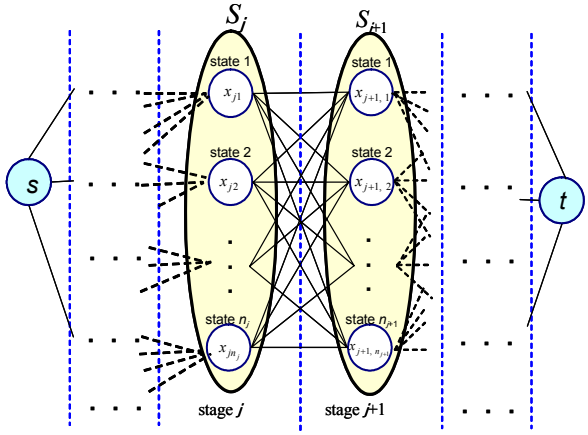



**Figure 19.** Basic concept of multistage decision making model.

Such kind of optimization has already been used in Zhang and Gen (2005), especially for solving fJSP.

Since both operation sequence and machine selection can affect the solution in an APS problem, the chromosome presentation of moGA for APS problems consists of two parts:

- Priority-based encoding for operation sequence;
- Machine permutation encoding for machine selection;

**Phase 1: Sequencing Operations**

Phase 1 is a procedure to get a feasible fixed operation sequence, hence we input the operation set for all the orders, and the precedence constraints in each order. After legalization (making precedence feasible), we can output the legal fixed operation sequence.

In this phase, we use the priority encoding procedure to formulate chromosomes, and draw a chromosome for the simple example in Section 2 as shown in Figure 20.

| *Node ID j* : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Priority $v_1(j)$: | 5 | 1 | 7 | 9 | 4 | 6 | 3 | 8 | 2 | 10 |

**Figure 20.** Chromosome $v_1$ drawn by priority-based encoding.

Following the precedence constraints, one feasible operation sequence for the simple example in Section 2 will be obtained as follows:

a final feasible path = ($v_2$, $v_1$, $v_6$, $v_7$, $v_9$, $v_3$, $v_8$, $v_4$, $v_5$, $v_{10}$)
operation sequences = {$o_{12}$, $o_{11}$, $o_{21}$, $o_{22}$, $o_{24}$, $o_{13}$, $o_{23}$, $o_{14}$, $o_{15}$, $o_{25}$}

**Phase 2: Selecting Machines**

After finishing Phase 1, we draw a fixed operation sequence, which means that the position of all stages (operations) has been decided. Hence, in Phase 2 we input the fixed operation sequence, processing time data, setup time data, and transition time.

**Phase 3: Designing Schedule**

After assigning states (machines) in Phase 2, we will output the whole schedule in Gantt chart with a solution of makespan.

Since the APS problem involves the flexible machine selection, we use a machine permutation coding procedure in this paper to make another part of the chromosome. That is, we will firstly build a multistage operation-based frame shown in Figure 21.

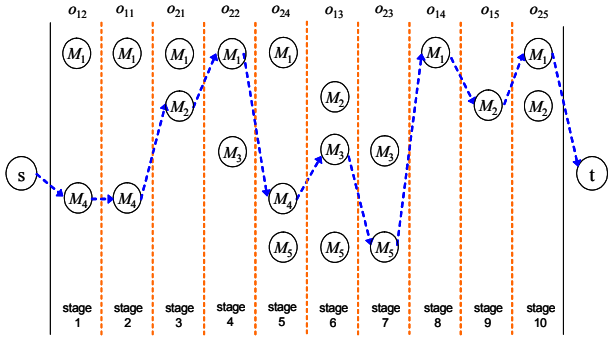



**Figure 21.** Node graph of machine selection in multistage operation-based genetic algorithm.

It is obvious to find the number of stages is just the total operation number, and also in each stage, the machines available are treated as the corresponding state. We can randomly generate second moGA chromosome as follows in Figure 22.

So, we can assign the machine selection to the fixed stage sequence offered by Phase 1, and finally

draw the feasible solution as follows:

Schedule:
S = {Order 1, Order 2}
= {($o_{12}$, $M_4$: 0-240), ($o_{11}$, $M_4$: 240-440),
($o_{13}$, $M_3$: 294-494), ($o_{14}$, $M_1$: 469-709),
($o_{15}$, $M_2$: 536-896), ($o_{21}$, $M_2$: 0-250),
($o_{22}$, $M_1$: 69-469), ($o_{24}$, $M_4$: 440-940),
($o_{23}$, $M_5$: 650-1050), ($o_{25}$, $M_2$: 828-1128)}

| Node ID $j$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| machine $v_2(j)$: | 4 | 4 | 3 | 1 | 2 | 2 | 1 | 5 | 4 | 1 |

**Figure 22.** Chromosome $v_2$ drawn by machine permutation encoding.

After calculating the makespan using the data in Tables 8 and 9, we can draw the Gantt chart shown in Figure 23.

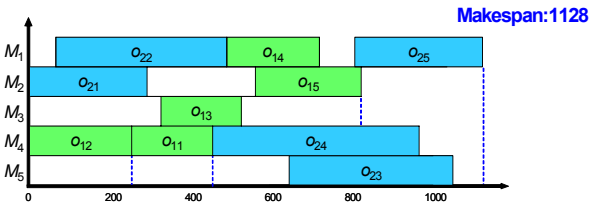



**Figure 23.** Gantt chart for the best solution.

## 5.3 Case Study

In this model, we firstly consider a relatively small size problem with 2 plants with 6 resources to treat four orders. The lot sizes for the orders are $q$ = (40, 70, 60, 30) and each plant has three resources. Plant 1 = {$M_1$, $M_2$, $M_3$} and Plant 2 = {$M_4$, $M_5$, $M_6$}. Their available capacities are:

$$L_1 = 1000,\ L_2 = 1000,\ L_3 = 2000,$$

$$L_4 = 2000,\ L_5 = 2000,\ L_6 = 2000,$$

The unit load size for transportation is assumed to be 10 for all orders. The operations and their precedence constraints for the 4 orders are given in Figure 24. The transportation times between resources are given in Table 10, and the processing times for each operation and their alternative resources are given in Table 11. The set-up times between operations are given in Table 12.

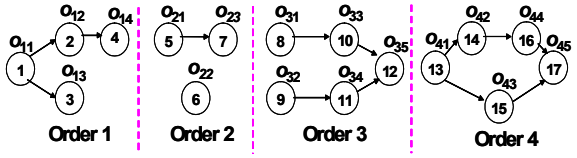



**Figure 24.** Operation precedence constraints.

The transportation time $t_{mn}^{D}$ per trip between plants is assumed to be 50, and the unit size per trip is equal to the lot size of each order. To solve the problem using the moGA approach, the genetic parameters are set to maximum generation, *maxGen* = 200; population size, *popSize* = 100; crossover probability, $p_C$ = 0.7; and mutation probability, $p_M$ = 0.3.

**Table 10.** Transition time $t_{mn}^{S}$ between machines

| | | Plant 1 | | |
|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ |
| Plant 1 | $M_1$ | 0 | 5 | 6 |
| | $M_2$ | 5 | 0 | 7 |
| | $M_3$ | 6 | 7 | 0 |
| Available capacity | | 1000 | 1000 | 2000 |

| | | Plant 1 | | |
|---|---|---|---|---|
| | | $M_4$ | $M_5$ | $M_6$ |
| Plant 2 | $M_4$ | 0 | 5 | 6 |
| | $M_5$ | 5 | 0 | 7 |
| | $M_6$ | 6 | 7 | 0 |
| Available capacity | | 2000 | 2000 | 2000 |

**Table 11.** Processing time $p_{kim}$ for operations in alternative machines

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

**Table 12.** Set-up time $t_{kij}^{U}$ 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 | - |

The makespan of the best solution is 1102, with the corresponding chromosomes and schedules are shown as follows:

| 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 |

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

The best schedule in detail is shown in Table 13. Furthermore, we can also draw the resource utilization factor for this experiment shown in Figure 25. As in the Gant chart shown in Figure 26, we compare the best results with the solution obtained by Moon-Kim-Gen's approach (Moon *et al*., 2004). The figure obviously presents the reason why they miss chances to find the best solution. It is because they only consider the minimum processing time. For example, operations in order 4 changed plants twice, but the best schedule of ours only once. They neglected the transportation time between plant 1 and plant 2.

Moreover, we set the parameter $p_C$ = 0.7 (for crossover), $p_M$ = 0.3 (for mutation), but changing the *maxGen* and *popSize* into 4 different tests within each experimental dataset, the comparison of the results is shown in Table 14.

**Table 13.** Best schedule

| 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) | $o_{22}$(782-962) |
| Plant 2 | $M_4$ | $o_{11}$(0-200) | $o_{12}$(217-457) | $o_{34}$(578-938) |
| | $M_5$ | $o_{33}$(3500-710) | $o_{35}$(750-1050) | |
| | $M_6$ | $o_{41}$(0-240) | $o_{32}$(452-872) | $o_{14}$(893-1093) |

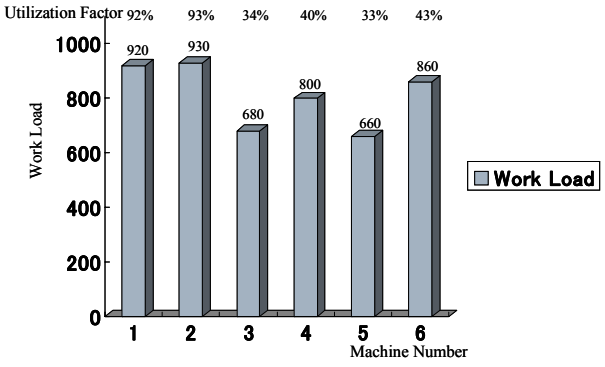



**Figure 25.** Resource utilization.

Furthermore, to prove the efficiency of our approach, we compared our experimental results with Moon-Kim-Gen's approach (Moon *et al*., 2004) by using the same experimental data. All the data in shadow and marked with "*" are best known solution (by enumerative algorithm operated on multiprocessor computer).

The solution shown in Table 15 illustrates that using moGA can get better solution in some large cases. That means, in some large cases, the assignment of machines may not obey the strategy for minimum processing time selection.

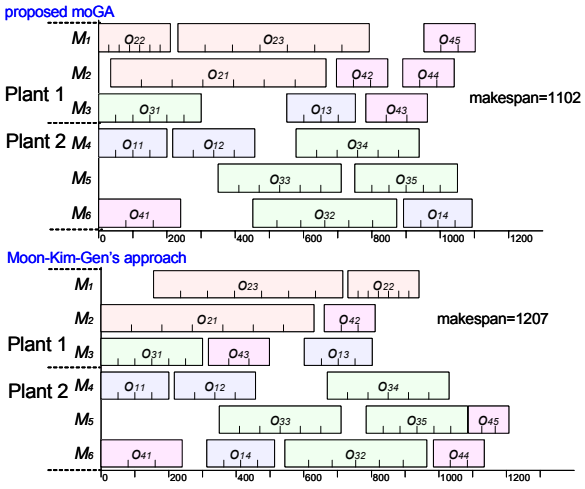



**Figure 26.** Gant chart of the best schedule. moGA: multi-stage operation-based genetic algorithm.

**Table 14.** Experimental result comparisons for different maxGen and popSize

| No. of orders | No. of operations | No. of plants | No. of machines | *maxGen* | *popSize* | $C_M$ |
|---|---|---|---|---|---|---|
| 4 | 17 | 2 | 6 | 100 | 20 | **1102*** |
| | | | | 100 | 50 | **1102*** |
| | | | | 200 | 100 | **1102*** |
| | | | | 500 | 100 | **1102*** |
| 5 | 21 | 2 | 6 | 100 | 50 | 1402 |
| | | | | 100 | 100 | **1370*** |
| | | | | 200 | 100 | **1370*** |
| | | | | 500 | 100 | **1370*** |
| 8 | 33 | 3 | 9 | 100 | 50 | 1680 |
| | | | | 100 | 100 | 1557 |
| | | | | 200 | 100 | **1513*** |
| | | | | 500 | 150 | **1513*** |
| 15 | 64 | 3 | 9 | 200 | 50 | 3408 |
| | | | | 200 | 100 | 2356 |
| | | | | 500 | 100 | 2239 |
| | | | | 500 | 150 | **2207*** |
| 29 | 124 | 5 | 15 | 200 | 50 | 3106 |
| | | | | 200 | 150 | 2920 |
| | | | | 500 | 150 | 2862 |
| | | | | 500 | 200 | **2798*** |

**Table 15.** Comparisons of experimental result

| No. of orders | No. of operations | No. of plants | No. of machines | Moon-Kim-Gen's approach | moGA |
|---|---|---|---|---|---|
| 4 | 17 | 2 | 6 | 1207 | **1102*** |
| 5 | 21 | 2 | 6 | 1561 | **1370*** |
| 8 | 33 | 3 | 9 | 1902 | **1513*** |
| 15 | 64 | 3 | 9 | 2640 | **2207*** |
| 29 | 124 | 5 | 15 | 3246 | **2798*** |

## 6. CONCLUSION

In this survey paper, we addressed MOGAs for three crucial issues in the manufacturing scheduling including the mathematical models, GA-based solution methods and case studies.

The fJSP is expanded from the traditional JSP, which possesses wider availability of machines for all the operations. Firstly, with loss of generality, we considered the total flexibility of fJSP, assuming that each operation is achievable on any machine. We presented an effective genetic approach to represent the chromosome and also proved to get better performance in solutions. We also gave the performance of the proposed method in comparison with other algorithms.

Secondly, we focused on the simultaneous scheduling and routing of AGVs in a FMS. We modeled an AGV system by using the network structure. This network model of an AGV dispatching has simplexes decision variables that consider most AGV problem's constraints. For applying a genetic approach to this multicriteria case of AGV problem that minimizes time required to complete all jobs (i.e., makespan) and minimizes the number of AGVs, simultaneously, a priority-based GA had been proposed. Numerical analyses for case study proved the effectiveness of the proposed approach.

Thirdly, we have addressed the moGA approach to solve the APS problems in multi-plant chain. In order to minimize the makespan, we should find an optimal resource selection for assignments and operations sequences simultaneously. Hence, we used the concept of multistage to formulate an efficient model, and divided the problem into 2 main phases by analyzing the character of APS problems. The results of various sizes of numerical experiments have demonstrated the efficiency of moGA by comparing it with the previous methods.

## ACKNOWLEDGMENTS

This work is partly supported by the Japan Society of Promotion of Science (JSPS): Grant-in-Aid for Scientific Research (C) (No. 245102190001), the Fundamental Research Funds (Software+X) of Dalian University of Technology (No. DUT12JR05) and National Tsing Hua University (NSC 101-2811-E-007-004).

## REFERENCES

Bidot, J., Vidal, T., Laborie, P., Beck, J. C. (2009), A theoretic and practical framework for scheduling in a stochastic environment, *Journal of Scheduling*, **12**(3), 315-344.

Brandimarte, P. (1993), Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, **41**(3), 157-183.

Cheng, R., Gen, M., and Tsujimura, Y. (1996), A tutorial survey of job-shop scheduling problems using genetic algorithms. I: representation, *Computers and Industrial Engineering*, **30**(4), 983-997.

Cheng, R., Gen, M., and Tsujimura, Y. (1999), A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies, *Computers and Industrial Engineering*, **36**(2), 343-364.

Dahal, K. P., Tan, K. C., and Cowling, P. I. (2007), *Evolutionary Scheduling*, Springer, London, UK.

Dauzere-Peres, S. and Paulli, J. (1997), An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search, *Annals of Operations Research*, **70**(1), 281-306.

Deb, K. (1989), *Genetic algorithms in multimodal function optimization*, MS Thesis, University of Alabama, Tuscaloosa, AL.

Deb, K. (2001), *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley and Sons, New York, NY.

Deb, K. (2005), *Optimization for Engineering Design: Algorithms and Examples*, Prentice-Hall of India Private Ltd., New Delhi, India.

Floudas, C. A. and Lin, X. (2004), Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review, *Computers and Chemical Engineering*, **28**(11), 2109-2129.

Floudas, C. A. and Lin, X. (2005), Mixed integer linear programming in process scheduling: modeling, algorithms, and applications, *Annals of Operations Research*, **139**(1), 131-162.

Framinan, J. M. and Ruiz, R. (2010), Architecture of manufacturing scheduling systems: Literature review and an integrated proposal, *European Journal of Operational Research*, **205**(2), 237-246.

Gao, J., Gen, M., and Sun, L. (2006), Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, *Journal of Intelligent Manufacturing*, **17**(4), 493-507.

Garey, M. R., Johnson, D. S., and Sethi, R. (1976), The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research*, **1**(2), 117-129.

Geiger, M. J. (2011), Decision support for multi-objective flow shop scheduling by the Pareto iterated local search methodology, *Computers and Industrial Engineering*, **61**(3), 805-812.

Gen, M. and Cheng, R. (2000), *Genetic Algorithms and Engineering Optimization*, Wiley, New York, NY.

Gen, M., Cheng, R., and Lin, L. (2008), *Network Models and Optimization: Multiobjective Genetic Algorithm Approach*, Springer, London, UK.

Gen, M., Gao, J., and Lin, L. (2009), Multistage-based

genetic algorithm for flexible job-shop scheduling problem, *Intelligent and Evolutionary Systems, Studies in Computational Intelligence*, **187**, 183-196.

Gen, M., Zhang, W., and Lin, L. (2009), Survey of evolutionary algorithms in advanced planning and scheduling, *Journal of the Korean Institute of Industrial Engineering*, **35**(1), 15-39.

Goncalves, J. F., de Magalhaes Mendes, J. J., and Resende, M. G. C. (2005), A hybrid genetic algorithm for the job shop scheduling problem, *European Journal of Operational Research*, **167**(1), 77-95.

Hwang, C. L. and Yoon, K. (1981), *Multiple Attribute Decision Making: Methods and Applications*, Springer, Berlin, Germany.

Hwang, H., Moon, S., and Gen, M. (2002), An integrated model for the design of end-of-aisle order picking system and the determination of unit load sizes of AGVs, *Computers and Industrial Engineering*, **42**(2-4), 249-258.

Ishibuchi, H. and Murata, T. (1998), A multi-objective genetic local search algorithm and its application to flowshop scheduling, *IEEE Transactions on Systems, Man, and Cybernetics Part C*, **28**(3), 392-403.

Kacem, I., Hammadi, S., and Borne, P. (2002), Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems, *IEEE Transactions on Systems, Man, and Cybernetics Part C*, **32**(1), 1-13.

Kacem, I., Hammadi, S., and Borne, P. (2002), Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic, *Mathematics and Computers in Simulation*, **60**(3-5), 245-276.

Kim, D. B. and Hwang, H. (2001), A dispatching algorithm for multiple-load AGVs using a fuzzy decision-making method in a job shop environment, *Engineering Optimization*, **33**(5), 523-547.

Kim, S. H. and Hwang, H. (1999), An adaptive dispatching algorithm for automated guided vehicles based on an evolutionary process, *International Journal of Production Economics*, **61-62**, 465-472.

Le-Ahn, T. and De Koster, M. B. M. (2006), A review of design and control of automated guided vehicle systems, *European Journal of Operational Research*, **171**(1), 1-23.

Li, L. and Huo, J. Z. (2009), Multi-objective flexible job-shop scheduling problem in steel tubes production, *Systems Engineering-Theory and Practice*, **29** (8), 117-126.

Li, X., Gao, L., and Li, W. (2012), Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling, *Expert Systems with Applications*, **39**(1), 288-297.

Li, X., Zhang, C., Gao, L., Li, W., and Shao, X. (2010), An agent-based approach for integrated process planning and scheduling, *Expert Systems with Applications*, **37**(2), 1256-1264.

Lim, J. K. (2004), *Study on guide path design and path planning in automated guided vehicle system*, PhD Thesis, Waseda University, Tokyo, Japan.

Lin, L. and Gen, M. (2008), Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation, *Soft Computing*, **13**(2), 157-168.

Lin, L., Hao, X. C., Gen, M., and Jo, B. J. (2012), Network modeling and evolutionary optimization for scheduling in manufacturing, *Journal of Intelligent Manufacturing*, **23**(6), 2237-2253.

Lin, L., Liang, Y., Gen, M., and Chien, C. F. (2012), A hybrid evolutionary algorithm for FMS optimization with AGV dispatching, *Proceedings of the 42th International Conference on Computers and Industrial Engineering*, Cape Town, South Africa.

Lopez-Ortega, O. and Moramay, R. (2005), A STEP-based manufacturing information system to share flexible manufacturing resources data, *Journal of Intelligent Manufacturing*, **16**(3), 287-301.

Moon, C., Kim, J. S., and 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.

Moon, S. W. and Hwang, H. (1999), Determination of unit load sizes of AGV in multi-product multi-line assembly production systems, *International Journal of Production Research*, **37**(15), 3565-3581.

Najid, N. M., Dauzere-Peres, S., and Zaidat, A. (2002), A modified simulated annealing method for flexible job shop scheduling problem, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Hammamet, Tunisia.

Naso, D. and Turchiano, B. (2005), Multicriteria meta-heuristics for AGV dispatching control based on computational intelligence, *IEEE Transactions on Systems, Man, and Cybernetics Part B*, **35**(2), 208-226.

Nowicki, E. and Smutnicki, C. (2005), An advanced Tabu search algorithm for the job shop problem, *Journal of Scheduling*, **8**(2), 145-159.

Pareto, V. (1906), *Manuale di economia politica*, Societa Editrice, Milano, Italy.

Pinedo, M. (2002), *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, Upper Saddle, NJ.

Proth, J. M., Sauer, N., and Xie, X. (1997), Optimization of the number of transportation devices in a flexible manufacturing system using event graphs, *IEEE Transactions on Industrial Electronics*, **44**(3), 298-306.

Schaffer, J. D. (1985), Multiple objective optimization with vector evaluated genetic algorithms, *Proceed-*

*ings of the 1st International Conference on Genetic Algorithms*, Pittsburgh, PA, 93-100.

Shao, X., Li, X., Gao, L., and Zhang, G. (2009), Integration of process planning and scheduling: a modified genetic algorithm-based approach, *Computers and Operations Research*, **36**(6), 2082-2096.

Srinivas, N. and Deb, K. (1995), Multiobjective optimiza-tion using nondominated sorting in genetic algorithms, *Journal of Evolutionary Computation*, **2**(3), 221-248.

Steuer, R. E. (1986), *Multiple Criteria Optimization: Theory*, *Computation*, *and Application*, Wiley, New York, NY.

Tamaki, H., Kita, H., and Kobayashi, S. (1996), Multiobjective optimization by genetic algorithms: a review, *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 517-522.

Tanev, I. T., Uozumi, T., and Morotome, Y. (2004), Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach, *Applied Soft Computing*, **5**(1), 87-100.

Tavakkoli-Moghaddam, R., Jolai, F., Vaziri, F., Ahmed, P. K., and Azaron, A. (2005), A hybrid method for solving stochastic job shop scheduling problems, *Applied Mathematics and Computation*, **170**(1), 185-206.

Ulusoy, G., Sivrikaya-Serifoglu, F., and Bilge, U. (1997), A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles, *Computers and Operations Research*, **24**(4), 335-351.

Verderame, P. M. and Floudas, C. A. (2008), Integrated operational planning and medium-term scheduling for large-scale industrial batch plants, *Industrial and Engineering Chemistry Research*, **47**(14), 4845-4860.

Vis, I. F. A. (2006), Survey of research in the design and control of automated guided vehicle systems, *European Journal of Operational Research*, **170**(3), 677-709.

Wang, S. J., Xi, L. F., and Zhou, B. H. (2008), FBS-enhanced agent-based dynamic scheduling in FMS, *Engineering Applications of Artificial Intelligence*, **21**(4), 644-657.

Wu, Z. and Weng, M. X. (2005), Multiagent scheduling method with earliness and tardiness objectives in flexible job shops, *IEEE Transactions on Systems, Man, and Cybernetics Part B*, **35**(2), 293-301.

Xia, W. and Wu, Z. (2005), An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, *Computers and Industrial Engineering*, **48**(2), 409-425.

Xiang, W. and Lee, H. P. (2008), Ant colony intelligence in multi-agent dynamic manufacturing scheduling, *Engineering Applications of Artificial Intelligence*, **21**(1), 73-85.

Yang, J. B. (2001), GA-based discrete dynamic programming approach for scheduling in FMS environments, *IEEE Transactions on Systems, Man, and Cybernetics Part B*, **31**(5), 824-835.

Zhang, W. and Gen, M. (2010), Process planning and scheduling in distributed manufacturing system using multiobjective genetic algorithm, *IEEE Transactions on Electrical and Electronic Engineering*, **5**(1), 62-72.

Zhang, W., Lin, L., Gen, M., and Chien, C. F. (2012), Hybrid sampling strategy-based multiobjective evolutionary algorithm, *Procedia Computer Science*, **12**, 96-101.

Zitzler, E. and Thiele, L. (1999), Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE Transactions on Evolutionary Computation*, **3**(4), 257-271.

Zitzler, E., Laumanns, M., and Thiele, L. (2001), SPEA2: Improving the Strength Pareto Evolutionary Algorithm, *TIK-Report 103*, Computer Engineering and Networks Laboratory (TIK), Zurich, Switzerland.

Bean, J. C. (1994), Genetic algorithms and random keys for sequencing and optimization, *INFORMS Journal on Computing*, **6**(2), 154-160.

Ding, L., Yue, Y., Ahmet, K., Jackson, M., and Parkin, R. (2005), Global optimization of a feature-based process sequence using GA and ANN techniques, *International Journal of Production Research*, **43** (15), 3247-3272.

Gen, M. and Zhang, H. (2006), Effective designing chromosome for optimizing advanced planning and scheduling. In: Dagli, C. H., Buczak, A. L., Enke, D. L., Embrechts, M., and Ersoy, O. (ed.), *Intelligent Engineering Systems through Artificial Neural Network*, ASME Press, New York, NY, 61-66.

Gen, M., Lin, L., and Zhang, H. (2009), Evolutionary techniques for optimization problems in integrated manufacturing system: State-of-the-art-survey, *Computers and Industrial Engineering*, **56**(3), 779-808.

Guo, Y. W., Li, W. D., Mileham, A. R., and Owen, G. W. (2009), Applications of particle swarm optimisation in integrated process planning and scheduling, *Robotics and Computer-Integrated Manufacturing*, **25**(2), 280-288.

Guo, Y., Li, W., Mileham, A. R., and Owen, G. W. (2009), Optimisation of integrated process planning and scheduling using a particle swarm optimisation approach, *International Journal of Production Research*, **47**(14), 3775-3796.

Kim, Y. K., Kim, J. Y., and Shin, K. S. (2007), An asymmetric multileveled symbiotic evolutionary algorithm for integrated FMS scheduling, *Journal of Intelligent Manufacturing*, **18**(6), 631-645.

Lee, C. Y. and Chen, Z. L. (2001), Machine scheduling with transportation considerations, *Journal of Scheduling*, **4**(1), 3-24.

Li, W. D. and McMahon, C. A. (2007), A simulated annealing-based optimization approach for integrated process planning and scheduling, *International Journal of Computer Integrated Manufacturing*, **20**(1), 80-95.

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

Qui, L., Hsu, W. J., Huang, S. Y., and Wang, H. (2002), Scheduling and routing algorithms for AGVs: a survey, *International Journal of Production Research*, **40**(3), 745-760.

Zhang, F., Zhang, Y. F., and Nee, A. Y. C. (1997), Using genetic algorithms in process planning for job shop machining, *IEEE Transactions on Evolutionary Computation*, **1**(4), 278-289.