J Intell Manuf (2014) 25:849–866
DOI 10.1007/s10845-013-0804-4

Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey

Mitsuo Gen · Lin Lin

Received: 31 July 2012 / Accepted: 5 June 2013 / Published online: 29 June 2013

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 order to find an optimal solution to scheduling problems it gives rise to complex combinatorial optimization problems. Unfortunately, most of them fall into the class of NP-hard combinatorial problems. In this paper, we focus on the design of multiobjective evolutionary algorithms (MOEAs) to solve a variety of scheduling problems. Firstly, we introduce fitness assignment mechanism and performance measures for solving multiple objective optimization problems, and introduce evolutionary representations and hybrid evolutionary operations especially for the scheduling problems. Then we apply these EAs to the different types of scheduling problems, included job shop scheduling problem (JSP), flexible JSP, Automatic Guided Vehicle (AGV) dispatching in flexible manufacturing system (FMS), and integrated process planning and scheduling (IPPS). Through a variety of numerical experiments, we demonstrate the effectiveness of these Hybrid EAs (HEAs) in the widely applications of manufacturing scheduling problems. This paper also summarizes a classification of scheduling problems, and illustrates the design way of EAs for the different types of scheduling problems. It is useful to guide how to design an effective EA for the practical manufacturing scheduling problems. As known, these practical scheduling problems are very complex, and almost is a combination of different typical scheduling problems.

M. Gen · L. Lin
Fuzzy Logic Systems Institute, Iizuka City, Japan

M. Gen
National Tsing Hua University, Hsinchu City, Taiwan

L. Lin (✉)
Dalian University of Technology, Dalian, Liaoning, China
e-mail: lin@dlut.edu.cn

Keywords Manufacturing scheduling · Multiobjective evolutionary algorithm (MOEA) · Hybrid evolutionary algorithm (HEA) · Job shop scheduling (JSP) · Flexible JSP (FJSP) · Advanced planning and scheduling (APS) · Automatic guided vehicle (AGV)

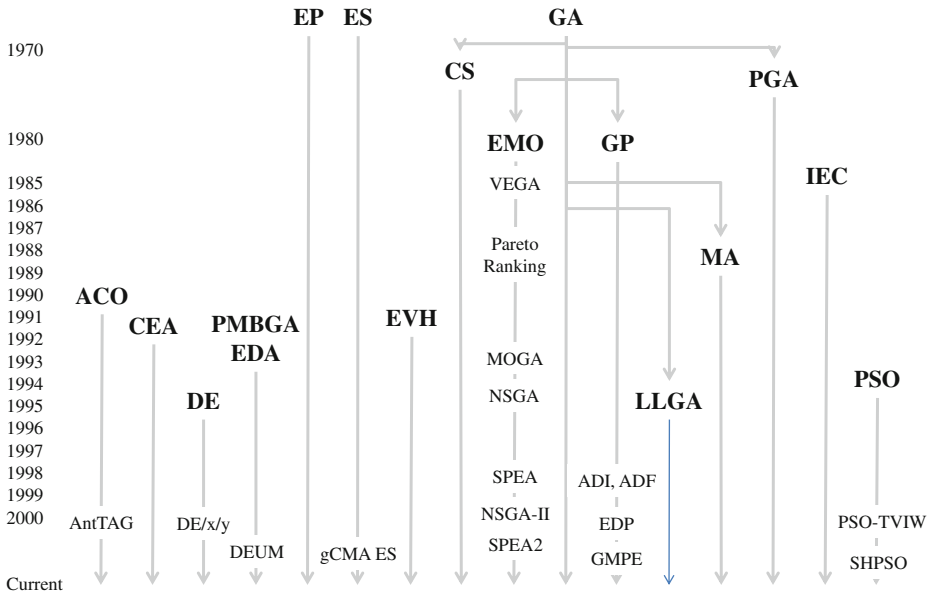
Introduction

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 Christodoulos 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) (Floudas and Lin 2005). The quality of the planning model and the integration scheme can be rendered inconsequential if the scheduling level does not rigorously model of 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 schedule 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.

Since the 1960s, there has been being an increasing interest in imitating living beings to solve the hard optimization problems. An evolutionary algorithm (EA) is a generic population-based meta-heuristic optimization algorithm. An EA uses some mechanisms inspired by biological evolution:

reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions "live" (see also cost function). Evolution of the population then takes place after the repeated application of the above operators. Handa et al. (2008) gave a comprehensive overview of recent advances of evolutionary computation (EC) studies, as shown in Fig. 1. EAs differ in the implementation details and the nature of the particular applied problem. EAs has attracted significantly attention with respect to complexity scheduling, which is referred to evolutionary scheduling, it is 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 algorithm for 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 neural network approach to generate initial feasible solutions and adapted a simulated annealing algorithm to improve the quality and performance of the solution in JSP. Meeran and Morshed (2012) proposed such as one solution method incorporating GA and Tabu Search for JSP. The rationale behind using such a hybrid method in case of other systems which use GA and TS is to combine the diversified global search and intensified local search capabilities of GA and TS respectively. 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 simulated annealing as a local search algorithm. Zhang and Gen (2005) proposed a multistage operation-based genetic algorithm to deal with the 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.



ACO	Ant Colony Optimization	**GA**	Genetic Algorithm
CEA	Coevolutionary Algorithm	**GP**	Genetic Programming
CS	Classifier System	**IEC**	Interactive Evolutionary Computation
DE	Differential Evolution	**LLGA**	Linkage Learning GA
EDA	Estimation of Distribution Algorithms	**MA**	Memetic Algorithm
EMO	Evolutionary Multiobjective Optimization	**PGA**	Parallel GA
EP	Evolutionary Programming	**PMBGA**	Probabilisti Model Building GA
ES	Evolution Strategies	**PSO**	Perticle Swarm Optimization
EVH	Evolvable Hardware		

Fig. 1 Evolution of evolutionary algorithms

Lopez and Ramirez (2005) newly describe 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 (FBS)-based heuristic algorithm which was proposed to study the dynamic scheduling problem in a 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 genetic algorithm-based approach have been developed to facilitate the integration and optimization of the two functions. In order to improve the optimized performance of the genetic algorithm-based approach, more efficient genetic representations and operator schemes have been developed. Li et al. (2010) proposed an agent-based approach for integrated process planning and scheduling. In this approach, the two functions are carried out simultaneously, and an optimization agent based on an evolutionary algorithm is used to manage the interactions and communications between agents to enable proper decisions to be made.

Furthermore, many researches are focusing on the multi-objectives manufacturing scheduling problems. Li and Huo (2009) proposed a GA for multi-objective 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 multi-objective model for the production smoothing problem on a single stage facility that some of the operating times could be determined in a time interval for. The proposed model was solved by a genetic algorithm, using a novel achievement function for exploring the solution space, based on LP-metric concepts. Li et al. (2012) proposed Nash equilibrium in game theory based approach has been used to deal with the multi-objective integrated process planning and scheduling. Gholami and Zandieh (2009) integrated simulation into GA to the dynamic scheduling of a flexible job shop with machines that suffer stochastic breakdowns. The objectives are the minimization of two criteria, expected makespan and expected mean tardiness. Elyn et al. (2011) proposed an ACI algorithm for a flowshop scheduling problem with minimizing both the makespan and the total completion time of jobs. Zandieh and Karimi (2011) considered a multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times by minimizing total weighted tardiness and maximum completion time simultaneously. They proposed a multi-population genetic algorithm. First stage applies combined objective of mentioned objectives and second stage uses previous stage's results as an initial solution. In the second stage sub-population will be generated by re-arrangement of solutions of first stage. Voratas and Siriwan (2011) proposed a two-stage genetic algorithm (2S-GA) for multi-objective Job Shop scheduling problems. The 2S-GA is proposed with three criteria: minimize makespan, minimize total weighted earliness, and minimize total weighted tardiness. The proposed algorithm is composed of two Stages: Stage 1 applies parallel GA to find the best solution of each individual objective function with migration among populations. In Stage 2 the populations are combined. The evolution process of Stage 2 is based on Steady-State GA using the weighted aggregating objective function.

Even if EAs have attracted significantly attention with respect to above complexity scheduling problems, it has a disadvantage: we have to design a specialized EA for each practical scheduling problem with the problem's specificity. So that means each class of EAs doesn't have a wide range of applications on manufacturing scheduling. Michalewicz (1994) summarized five basic components of EA:

(1) An evolutionary representation of potential solutions to the problem.
(2) A way to create a population (an initial set of potential solutions).
(3) An evaluation function rating solutions in terms of their fitness.
(4) Evolutionary operators that alter the genetic composition of offspring (crossover, mutation, selection, etc).
(5) Parameter values that evolutionary algorithm uses (population size, probabilities of applying evolutionary operators, etc).

In order to design an effective EA with the problem's specificity, we have to consider (1) how to design a representation and a way of population initialization; (2) how to evalu-

ate an individual by a fitness function; (3) how to improve initialization by evolutionary operators. In this paper, we focus on the effective multiobjective EA design for a wide range of applications on manufacturing scheduling. We will discuss the representation design of potential solutions to the scheduling problems, the fitness assignment mechanisms to the multiobjective scheduling problems, and the evolutionary operation design to improve solution simultaneously.

The rest of this paper is organized as follows: "Multiobjective evolutionary algorithm" section introduces multiobjective EA, and give fitness assignment mechanism, performance measures for multiobjective scheduling problems; How to present a solution of the scheduling problem into a chromosome is given in "Evolutionary representation" section; How to improve an individual by using hybrid evolutionary operations is given in "Hybrid evolutionary operations" section; "Scheduling formulations and EAs" section presents the brief mathematical formulations of the typical scheduling problems, and discusses the current state-of-the-art EAs for solving them. Finally, the conclusion of this paper and future researches are drawn in "Conclusion" section.

Multiobjective evolutionary algorithm

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, EAs have been received considerable attention as a novel approach to multiobjective optimization problems, resulting in a fresh body of research and applications known as evolutionary multiobjective optimization (EMO).

The inherent characteristics of EAs demonstrate why evolutionary 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 much mathematical requirements about the problems and can handle any kind of objective functions and constraints. Due to their evolutionary nature, the EAs can search for solutions without regard to the specific inner workings of the problem. Therefore, it is more hope for solving much complex problems than the conventional methods.

EAs are essentially a kind of meta-strategy methods. When applying the EAs to solve a given problem, it is necessary to refine upon each of the major components of EAs, such as encoding methods, recombination operators, fitness assignment, selection operators, constraints handling, and so on, in order to obtain a best solution to the given problem. Because the multiobjective optimization problems are the natural extensions of constrained and combinatorial optimization problems, so many useful methods based on EAs developed during the past two decades. One of special issues in the multiobjective optimization problems is fitness assignment mechanism. Since the 1980s, several fitness assignment mechanisms have been proposed and applied in multiobjective optimization problems (Gen et al. 2008). Although most fitness assignment mechanisms are just different approach and suitable to different cases of multiobjective optimization problems, in order to understanding the development of multiobjective EAs (MOEAs), we classify algorithms according to proposed years of different approaches:

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. The selection step in each generation becomes a loop. Each time through the loop the appropriate fraction of the next generation, or subpopulation, is selected on the basis of each objective. The entire population is shuffled thoroughly to apply crossover and mutation operators. This is performed to achieve the mating of individuals of different subpopulations.

Type 2 Pareto ranking + Diversity

Multiobjective genetic algorithm (MOGA: Fonseca and Fleming 1995): Fonseca and Fleming proposed multiobjective genetic algorithm (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 nondominated 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: Dev 1995): Srinivas and Deb also developed a Pareto ranking-based fitness assignment and it called Nondominated Sorting Genetic Algorithm (NSGA). In each method, the nondominated solutions constituting a nondominated 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 nondominated 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): Ishibuchi et al. proposed a weighted-sum based fitness assignment method, called random-weight

Genetic Algorithm (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. In RWGA, each objective $f_k(x)$ is assigned a weight $w_k = r_k / \sum_{j=1}^{q} r_j$, where r_j are nonnegative random number between [0, 1] with q objective functions. And the scalar fitness value is calculated by summing up the weighted objective value $w_k \cdot f_k(x)$. To search for multiple solutions in parallel, the weights are not fixed and able to uniformly the sample area towards to the whole frontier.

Strength Pareto evolutionary algorithm II (SPEA II: Zitzler and Thiele 2001): Zitzler and Thiele proposed strength Pareto Evolutionary Algorithm (SPEA: Zitzler and Thiele 1999) and an extended version SPEA II (Zitzler and Thiele 2001) that combines several features of previous multiobjective Genetic Algorithms (MOGA) in a unique manner. The fitness assignment procedure is a two-stage process. First, the individuals in the external nondominated set P' are ranked. Each solution $i \in P'$ is assigned a real value $s_i \in [0, 1)$, called *strength*; s_i is proportional to the number of population members $j \in P$ for which $i \succ j$. Let n denote the number of individuals in P that are covered by i and assume N is the size of P. Then s_i is defined as $s_i = n/(N+1)$. The fitness f_i of objective i is equal to its strength: $f_i = s_i$. Afterwards, the individuals in the population P are evaluated. The fitness of an individual $j \in P$ is calculated by summing the strengths of all external nondominated solutions $i \in P'$ that cover j. The fitness is $f_j = 1 + \sum_{i \in \{i \succ j\}} s_i$, where $f_j \in [1, N)$.

Adaptive-weight genetic algorithm (AWGA: Gen and Cheng 2000): Gen and Cheng proposed another weight sum-based fitness assignment method, called Adaptive-weight Genetic Algorithm (AWGA) which utilizes some useful information from the current population to readjust weights to obtain a search pressure toward the Pareto frontier. When considering the maximization problem with q objectives, we define two extreme points: the maximum extreme point $z^+ = \{z_1^{\max}, z_2^{\max} \ldots z_q^{\max}\}$ and the minimum extreme point $z^- = \{z_1^{\min}, z_2^{\min} \ldots z_q^{\min}\}$ in each generation. Each objective k is assigned a weight $w_k = 1/\left(z_k^{\max} - z_k^{\min}\right)$. And the scalar fitness value is calculated by $\sum_{k=1}^{q} (f_k(\boldsymbol{x}) - z_k^{\min})/(z_k^{\max} - z_k^{\min})$.

Non-dominated sorting genetic algorithm II (NSGA II: Deb 2001): Deb suggested a nondominated sorting-based approach, called non-dominated sorting Genetic Algorithm II (NSGA II), which alleviates the three difficulties: computational complexity, nonelitism approach, and the need for specifying a sharing parameter. The NSGA II was advanced from its origin, NSGA. In NSGA II, a nondominated sorting approach is used for each individual to create Pareto rank, and a crowding distance assignment method is applied to implement density estimation. In a fitness assignment between two individuals, NSGA II prefers the point with a lower rank value, or the point located in a region with fewer numbers of points if both of the points belong to the same front. Therefore, by combining a fast nondominated sorting approach, an elitism scheme and a parameterless sharing method with the original NSGA, NSGA II claims to produce a better spread of solutions in some testing problems.

Interactive adaptive-weight genetic algorithm (i-AWGA: Gen et al. 2008): Gen et al. proposed an interactive adaptive-weight genetic algorithm (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. Firstly, calculate the adaptive weight $w_i = 1/(z_i^{\max} - z_i^{\min})$ for each objective $i = 1, 2 \ldots q$ by using AWGA. Afterwards, calculate the penalty term $p(v_k) = 0$, if v_k is nondominated solution in the nondominated set P. Otherwise $p(v_{k'}) = 1$ for dominated solution $v_{k'}$. Last, calculate the fitness value of each chromosome by combining the method as follows and we adopted roulette wheel selection as supplementary to the i-AWGA.

$$eval(v_k) = \sum_{i=1}^{q} w_i (z_i^k - z_i^{\min}) + p(v_k), \ \forall k \in popSize \quad (1)$$

Hybrid sampling strategy-based EA (HSS-EA: Zhang et al. 2012a, b): Zhang et al. proposed a hybrid sampling strategy-based evolutionary algorithm (HSS-EA). A Pareto dominating and dominated relationship-based fitness function (PDDR-FF) is proposed to evaluate the individuals. The PDDR-FF of an individual S_i is calculated by the following function:

$$eval(S_i) = q(S_i) + \frac{1}{p(S_i) + 1}, \quad i = 1, 2, \ldots, popSize \quad (2)$$

where $p()$ is the number of individuals which can be dominated by the individual S. $q()$ is the number of individuals which can dominate the individual S. The PDDR-FF can set the obvious difference values between the nondominated and dominated individuals (Fig. 2).

The sampling strategy of VEGA prefers the edge rather than center regions of Pareto front that it causes VEGA cannot achieve better distribution performance. So it is natural, reasonable and possible to combine these two methods to improve the overall performance and reduce the computation time of the algorithm. Figure 3 shows the description of HSS-EA. The strong convergence capability of VEGA and PDDR-FF ensures that the HSS-EA has the ability to converge to the true Pareto front both in central and edge

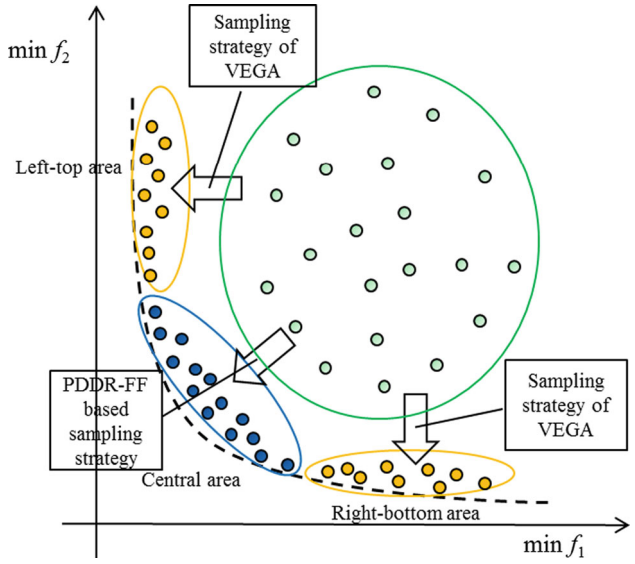


Fig. 2 The description of HSS-EA

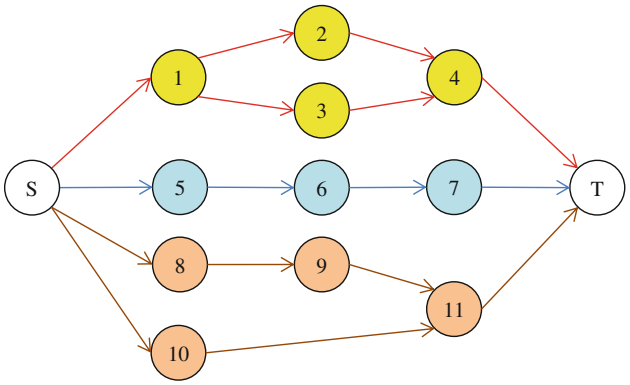


Fig. 3 Illustration of scheduling network with 3 job and 11 operations

regions. The preferences for the edge area of the Pareto front in VEGA and the central area of the Pareto front in PDDR-FF guarantee that the HSS-EA distributes along the Pareto front evenly. Moreover, less computing time makes that HSS-EA has higher efficiency than other approaches.

Evolutionary representation

How to present a solution of the scheduling problem into a chromosome is a key issue for EAs. For evaluating the effectiveness of the different chromosome representation, there are several critical issues are summarized by Gen et al. (2008)

- *Space:* Chromosomes should not require extravagant amounts of memory.
- *Time:* The time complexities of evaluating, recombining, and mutating chromosomes should be small.
- *Feasibility:* All chromosomes, particularly those generated by simple crossover (i.e., one-cut point crossover) and mutation, should represent feasible solutions.
- *Uniqueness:* The mapping from chromosomes to solutions (decoding) may belong to one of the following three cases: 1-to-1 mapping, n-to-1 mapping and 1-to-n mapping. The 1-to-1 mapping is the best one among three cases and 1-to-n mapping is the most undesired one.
- *Heritability:* Offspring of simple crossover (i.e., one-cut point crossover) should represent solutions that combine substructures of their parental solutions.
- *Locality:* A mutated chromosome should usually represent a solution similar to that of its parent.

We need to consider these critical issues carefully when designing an appropriate representation so as to build an effective EA. As known, scheduling problem is the implement of production plan, with considering production processes, lot-size, amount and customer requirements etc. And scheduling problem is how to decide the resources assignment to the production, with considering constrains of resources capabilities and capacities. There are two decision making parts for scheduling optimization: (1) operation sequencing and (2) resources assignment.

Representation for operation sequencing

In the past few decades, the following 6 representations for job-shop scheduling problem (JSP, an operation sequencing problem with considering the precedence constraints of operations) have been proposed:

- Operation-based representation (De Jong 1994)
- Job-based representation (Baker and Scudder 1990)
- Preference list-based representation (Croce et al. 1995)
- Priority rule-based representation (Dorndorf and Pesch 1995)
- Completion time-based representation (Yamada and Nakano 1992)
- Random key-based representation (Norman and Bean 1995)

The Flexible Job-shop Scheduling Problem (FJSP) is expanded from the traditional JSP, which possesses wider availability of machines for all the operations (a combinatorial optimization problem considering both of the operation sequence and the resource assignment). The following 4 representations for FJSP have been proposed:

- Parallel machine-based representation (Gen and Cheng 1997)
- Parallel jobs representation (Gen and Cheng 1997)

- Operations machines-based representation (Kacem et al. 2002a, b)
- Multistage operation-based representation (Zhang and Gen 2006)

Permutation-based representation is perhaps the most natural representation of operation sequences. Unfortunately because of the existence of precedence constraints, not all the permutations of the operations define feasible sequences. For job shop scheduling problem, Cheng et al. (1996, 1999) applied job-based representation: they name all operations for a job with the same symbol and then interpret them according to the order of occurrence in the sequence of a given chromosome. Gen and Zhang (2006) also applied this representation to advanced scheduling problem. The job-based representation can also be used to represent the operation sequences for the FJSP problem. Each job i appears in the operation sequence exactly n_i times to represent its n_i ordered operations. However, if the operation precedence is more complex than JSP or extend JSP problems, the job-based representation cannot be used directed.

Cheng and Gen (1994) proposed a priority-based representation firstly for solving Resource-constrained Project Scheduling Problem (rcPSP). This representation encodes a schedule as a sequence of operations and each gene stands for one operation. As known, a gene in a chromosome is characterized by two factors: locus, i.e., the position of the 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 operation ID and its value is used to represent the priority of the operation for constructing a schedule among candidates. A schedule can be uniquely determined from this encoding.

Figure 3 presents a scheduling network with 3 jobs and 11 operations. Illustration of priority-based representation is shown in Fig. 4. At the beginning, we try to find an operation for the position next to source node S. Operations 1, 5, 8 and 10 are eligible for the position, which can be easily fixed according to adjacent relation among operations. The priorities of them are 10, 5, 8 and 1, respectively. Operation 1 has the highest priority and is put into the schedule. The possible operations next to operation 1 are operations 2 and 3, and possible operations 5, 8 and 10. Because operation 3 has the largest priority value, it is put into the schedule. Then we form the set of operations available for next position and select the one with the highest priority among them. Repeat these steps until we obtain all operations into the schedule, $(N_1-N_3-N_8-N_5-N_6-N_2-N_4-N_7-N_9-N_{10}-N_{11})$.

locus	1	2	3	4	5	6	7	8	9	10	11
allele	10	4	9	5	7	11	3	8	2	1	6

Fig. 4 Illustration of priority-based representation

locus	1	2	3	4	5	6	7	8	9	10	11
allele	0.10	0.04	0.09	0.05	0.07	0.11	0.03	0.08	0.02	0.01	0.06

Fig. 5 Illustration of random key-based representation

However, the nature of the priority-based representation is a kind of permutation representations. Generally, this representation will yield illegal offspring when using one-cut point crossover or other simple crossover operators. That means some node's priority may be duplicated in the offspring. There are several crossover operators proposed for permutation representation, such as partial-mapped crossover (PMX), order crossover (OX), position-based crossover (PX), heuristic crossover, and so on (Gen et al. 2008). Norman and Bean (1995) proposed random key-based representation for JSP. In this paper, we combine the random key-based representation for operations sequencing. The example of representation is shown in Fig. 5, and we can obtain the same operations sequence into the schedule, $(N_1-N_3-N_8-N_5-N_6-N_2-N_4-N_7-N_9-N_{10}-N_{11})$.

Representation for resources assignment

After the operation sequence is fixed, the resources assignment can be formulated as a multi-stage graph problem. For each stage (operation), we decided the state number (which resource should be assigned). This multi-stage graph problem can be solved by dynamic programing. Yang (2001) proposed a GA-based discrete dynamic programming approach for scheduling in FMS environment. However, the IPPS problem is the combination of the operation sequencing (NP-hard problem) and resources assignment. Considering the computation times of the algorithm, and the most of practical IPPS problems are multi-resources assignment, the most of researches combined a state permutation representation into the chromosome (Okamoto et al. 2005; Gao et al. 2008; Gen et al. 2009), called multi-stage representation.

For each resource type h, generate a chromosome section by using this state permutation representation. In this representation, the position of a gene is used to represent operation ID, and its value is used to represent the resource id selected. The maximum value is equal to the number of resources $|R_h|$ of resource type h. In this paper, we use the real number in the state permutation representation.

For example, Fig. 6 presents resource R_h (machine) assignment for 11 operations. The number of resources $|R_h|$ is equal to 4. In particular, the operation N_3 is only achievable on a part of the available machines M_2 and M_4. And the operation N_4 is only achievable on a part of the available machines M_2, M_3 and M_4. Illustration of permutation representation is shown in Fig. 7. As the decoding process, we assign the machine M_u to operation j by using the following equation:

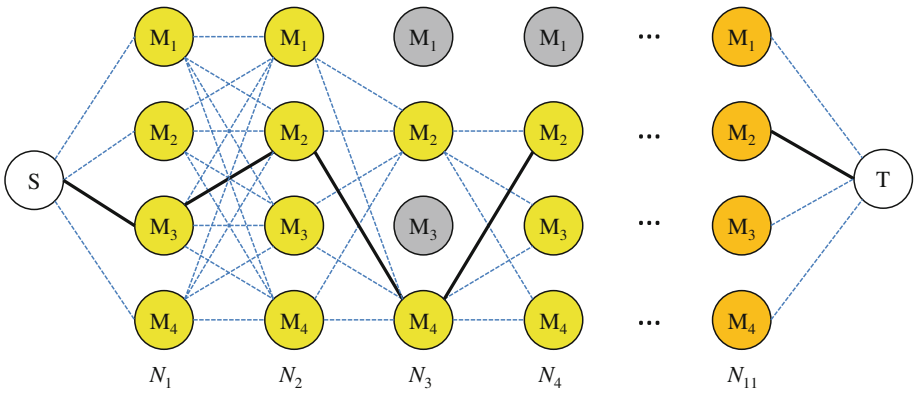


Fig. 6 Illustration of resource R_h (machine) assignment

locus	1	2	3	4	5	6	7	8	9	10	11
allele	0.56	0.36	0.78	0.23	0.34	0.70	0.50	0.41	0.85	0.15	0.32

Fig. 7 Illustration of permutation representation

$$u' = \mathrm{FIX}(v_j \cdot U_j') \tag{3}$$

where v_j is the value of the jth gene; U_j' is the number of available machines for operation j; FIX(x) rounds the elements of x to the nearest integers towards zero.

Hybrid evolutionary operations

There are many situations in which the simple EA does not perform particularly well, and various methods of hybridization have been proposed. One of most common forms of hybrid EA is to incorporate local optimization approach as an add-on extra to the canonical EA loop of evolutionary operations, such as recombination and selection etc. With the hybrid approach, local optimization is applied to each newly generated offspring to move it to a local optimum before injecting it into the population. EA is used to perform global exploration among a population while heuristic methods are used to perform local exploitation around chromosomes. Because of the complementary properties of EA and conventional heuristics, the hybrid approach often outperforms either method operating alone.

Bottleneck shifting

Gao et al. (2008) proposed a hybrid genetic algorithm for the flexible job shop scheduling problem (FJSP), by using effective evolutionary operation, called bottleneck shifting. As shown in Fig. 8, a feasible operation sequencing of scheduling can be represented with disjunctive graph $G = (N, A)$, with operations set N, operation sequence set A. In Fig. 8, the number above each node represents the processing time of that operation. The critical path is the longest path in a graph. The makespan is equal to the length of the critical path in the corresponding disjunctive graph. The critical path is highlighted with broad-brush arcs in Fig. 8. Any operation on the critical path is called a critical operation. A critical operation cannot be delayed without increasing the makespan of the schedule.

A improved schedule which is slightly different from the initial solution can be generated by changing the processing sequence of two adjacent operations performed on the same machine, i.e., reversing the direction of the disjunctive arc that links the two operations. The makespan of a schedule is defined by the length of its critical path, in other words, the makespan is no shorter than any possible path in the disjunctive graph. Only when these two adjacent operations are on the critical path, the new solution is possible to be superior to the old one. As shown in Fig. 9, the makespan is improved by swap operations 3 and 5 on critical path.

GA+PSO

In Particle Swarm Optimization (PSO), the representation code is called particle swarm, and each allele is called a particle. The evolutionary operation of PSO is that particle fly through the problem space by following the current optimum particles. In each iteration t, the algorithm updates positions (x_i^t) and velocities (v_i^t) of the particle i as follows:

$$v_i^t = \omega v_i^{t-1} + \phi_1 \left(g_i - x_i^{t-1}\right) + \phi_2 \left(l_i^{t-1} - x_i^{t-1}\right) \tag{4}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{5}$$

with $\phi_1 = r_1 a_g$, $\phi_2 = r_2 a_l$, r_1 and $r_2 \sim \mathrm{U}(0, 1)$, ω, a_l, $a_g \in \mathrm{R}$. l_i^{t-1} is the best position found so far by the i-1th particle and g_i is the global best position on the particle i. Guo et al. (2006, 2009) proposed PSO algorithms for integrated process planning and scheduling problems.

Lin et al. (2012) proposed a hybrid evolutionary operation by combining GA and PSO. They combine the evolutionary operation of PSO as a special crossover of GA.

In PSO, the particle is replaced by Eq. (5) in each iteration t. Different with the operation of PSO, the GA+PSO

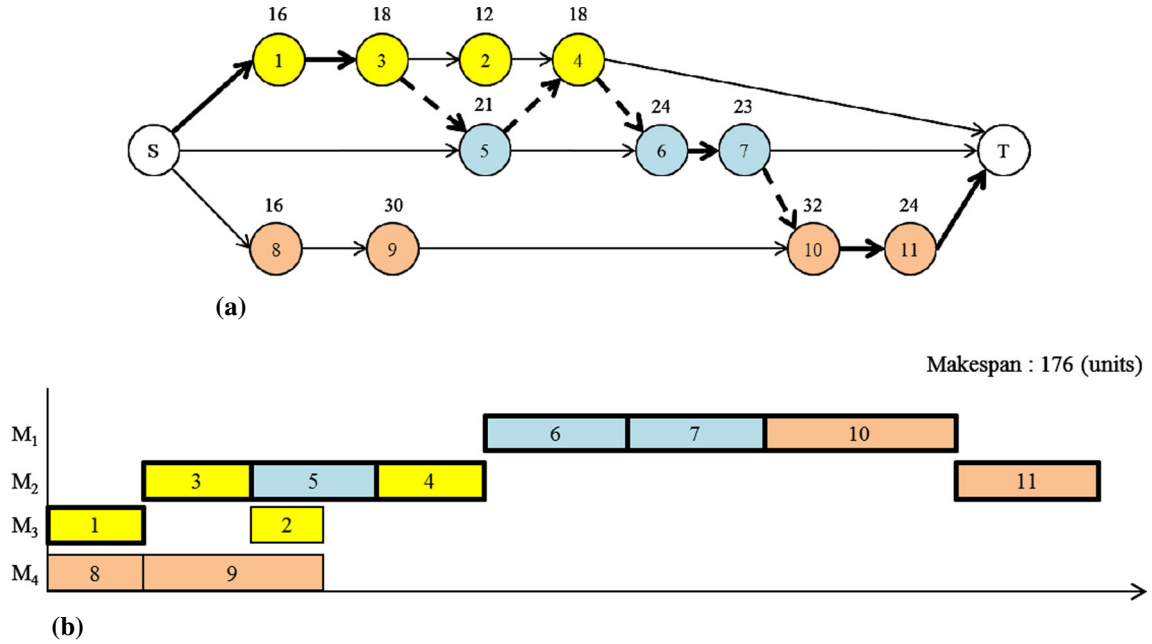


Fig. 8 Illustration of disjunctive graph. **a** A feasible solution based on evolutionary representation (Fig. 5 + Fig. 7), **b** Gantt chart

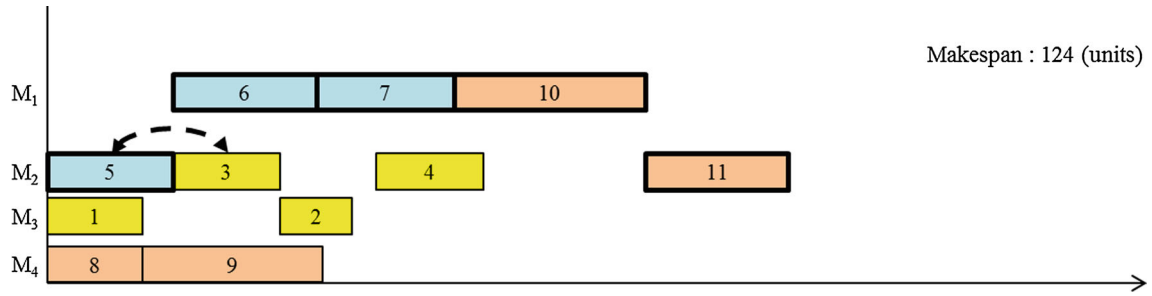


Fig. 9 Illustration of improved solution by bottleneck shifting operation

operation replace the chromosome if the generated chromosome is better than the original chromosome. The evolutionary operation process is shown as following steps:

Step 1: Calculate velocities (v_i^t) for each gene i in the chromosome j in generation t.

$$v_{ij}^t = \omega v_{ij}^{t-1} + \phi_1 \left(g_i - x_{ij}^{t-1}\right) + \phi_2 \left(l_{ij}^{t-1} - x_{ij}^{t-1}\right), \quad \forall i, j \tag{6}$$

where ω, a_l, a_g are parameters of the algorithm, $\phi_1 = r_1 a_g$, $\phi_2 = r_2 a_l$, r_1 and r_2 are random numbers between (0, 1). g_i is the value of gene i in the best chromosome, and l_{ij}^t is the value of gene i in the best chromosome j in generation t.

Step 2: Generate a new chromosome j' with calculating each gene i in generation t by following equation:

$$x_{ij'}^t = x_{ij}^t + v_{ij}^t, \quad \forall i, j \tag{7}$$

Step 3: Evaluate the fitness $fit_{j'}$ of the chromosome j'; if the fitness $fit_{j'}$ is better than the fitness fit_j of chromosome j, replace the chromosome j by $v_j^t = [v_{ij'}^t]_I$.

Scheduling formulations and EAs

As discussed above, manufacturing scheduling problem is how to decide the resources assignment to the production, considering constrains of resources capabilities and capacities; and is the implement of production plan, with considering production processes, lot-size, amount and customer requirements etc.

The notations of scheduling problem are shown as follows:

Indies:

$i \in I$	job ID
$j \in J$	operation ID
$l \in L$	material ID
$k \in K$	material type ID

$h \in H$ resource ID
$u \in U$ resource type ID

Parameters:
Material definitions

$M = \{M_l\}$	material set
$M_l = \{A_k^M,\ T_k^M\}$	{material attribute, constraint}

Resource definitions

$R = \{R_u\}$	resource set
$R_u = \{A_u^R,\ T_u^R\}$	{resource attribute, constraint}

Operation definitions

$N_j = \{t_j^{\mathrm{S}},\ t_j^{\mathrm{T}},\ p_j,\ C_j^{\mathrm{I}},\ C_j^{\mathrm{E}}\}$	operation j
t_j^{S}	starting time of operation j
t_j^{T}	ending time of operation j
p_j	processing time of operation j
$C_j^{\mathrm{I}} = \{M_j,\ R_j\}$	import of operation j
$C_j^{\mathrm{E}} = \{M_{j'},\ R_j\}$	export of operation j
$R_{jh} = \{A_{ju}^h,\ T_{ju}^h\}$	{resource attribute, constraint} of operation j

Operations relationship

$E_{jj'} = \{(N_j,\ N_{j'})\}$	precedence constraint of operation j and operation j'
$t_{jj'}^{\mathrm{L}}$	shipment time from operation j to operation j'
$t_{jj'}^{\mathrm{I}}$	idle time from operation j to operation j'

Decision variables:

$\boldsymbol{X}_i = \{x_{jh}\}$	operation resource assignment for each job i
$\boldsymbol{Z}_{jj'} = \{z_{jj'h}\}$	movement resource assignment between operation j and operation j'
$\boldsymbol{Y}_{jj'} = \{y_{jj'}\} = \{(0, 1)\},\ \forall j, j'$	0 or 1, precedence of operation j and operation j'

As shown in Fig. 10, for each operation N_j, resources requirement $\{R_h\}$, start time of operation j t_j^{S}, end time of operation j t_j^{T}, processing time of operation j p_j, import of operation j $C_j^I = \{M_j,\ R_j\}$, and export of operation j $C_j^E = \{M_{j'},\ R_j\}$ are assigned on the operation.

As shown in Fig. 11, a job can be defined as a sub-graph. In the sub-graph, (1) the directed arc is presenting the operations precedence $E_{jj'}$ for the product; (2) The variables on the arc are presenting the shipment time $t_{jj'}^{\mathrm{L}}$, idle time $t_{jj'}^{\mathrm{I}}$ and other specialized definitions between operations. Job i can be presented as a sub-graph G_i:

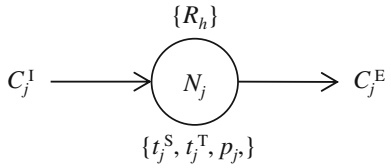


Fig. 10 Illustration of operation N_j

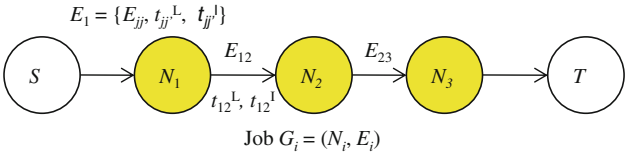


Fig. 11 Illustration of sub-graph G_i for job i

$G_i = (N_i,\ E_i)$	sub-graph of job i
$N_i = \{N_j\}$	operations set for completing job i
$E_i = \{E_{jj'},\ t_{jj'}^{\mathrm{L}},\ t_{jj'}^{\mathrm{I}}\}$	arc set from operation j to operation j'

A set of jobs can be defined as a directed graph $G = \{G_i\}$. G included all of jobs (with considering lot-size, amount etc.) manufactured. The notation of scheduling graph is shown as following:

$G = \{G_i\}$	scheduling network

The objectives of scheduling problems can be classified as operation sequencing problems, operations (or shipments) selection problems, resources assignment problems, operations (or shipments) grouping problems. The objective functions are (1) minimization/maximization of resource parameters (e.g., operation cost), or (2) minimization/maximization of operations parameters (e.g., makespan), (3) minimization/maximization of arc resource parameters (e.g., transportation cost). Sometimes, the special scheduling problems are considering the operation grouping problems (e.g., resources balancing problem). This paper does not consider the operation grouping problems.

Objective functions:

$$\min/\max \quad f_1(R_h, \boldsymbol{X}_i) \tag{8}$$

$$\min/\max \quad f_2\left(p_j, \boldsymbol{Y}_{jj'}\right) \tag{9}$$

$$\min/\max \quad f_3\left(E_{jj'}, \boldsymbol{Z}_{jj'}\right) \tag{10}$$

System constraints:

$$g_1\left(N_j, \boldsymbol{Y}_{jj'}\right) \geq 0 \tag{11}$$
$$g_2\left(E_{jj'}, \boldsymbol{Y}_{jj'}\right) \geq 0 \tag{12}$$
$$g_3\left(R_h, \boldsymbol{X}_i\right) \geq 0 \tag{13}$$
$$g_4\left(R_h, \boldsymbol{Z}_{jj'}\right) \geq 0 \tag{14}$$

Non-negative constraints:

$$\boldsymbol{X}_i = \left\{x_{jh}\right\} \geq 0, \forall i \tag{15}$$
$$\boldsymbol{Y}_{jj'} = \left\{y_{jj'}\right\} = \{(0,\ 1)\}, \forall j \tag{16}$$
$$\boldsymbol{Z}_{jj'} = \left\{x_{jh}\right\} \geq 0, \forall j \tag{17}$$

The constraints of scheduling problems can be classified as operation precedence constraints, shipments precedence constraints, and resources of operations (or shipments) capacity constraints. Therefore, the scheduling problems should consider the (11) operation precedence constraints, (12) the shipment precedent constraints, (13) resource usability constraints, and (14) shipment resource usability constraints.

Job shop scheduling

The job shop scheduling problem (JSP) is one of the typical scheduling problems, concerning determination of the operation sequences so that the makespan is minimized. It consists of several assumptions as follows:

A1. Every machine processes only one operation at a time.
A2. The execution of each operation requires one machine selected from a set of available machines for the operation.
A3. The operation sequence of a job is prespecified.
A4. The operations are not preemptable, that is, once an operation has started it cannot be stopped until it has finished.
A5. The set-up times for the operations are sequence-independent and are included in the processing times.
A6. 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 problem could be described as an n-job m-machine scheduling problem by simple equations as follows:

$$\min \quad t_{\mathrm{M}} = \max_j \left\{t_j^T\right\} \tag{18}$$
$$\text{s. t.} \quad t_{j'}^T - p_{j'} \geq t_j^T, \quad \forall \left\{j,\ j' \middle| E_{jj'} = 1\right\} \tag{19}$$
$$t_j^T \geq 0, \quad \forall j \tag{20}$$

The objective function by Eq. (18) is to minimize the makespan. The constraint by Eq. (19) is the operation precedence constraint, the j'th operation should be started after the jth operation finished in the same job.

Hybrid EA for JSP

The $P(t)$ and $C(t)$ are parents and offspring respectively in current generation t, the implementation structure of hybrid EA (HEA) for scheduling is described as follows:

```
procedure: HEA for JSP
input: scheduling data (N, E), HEA parameters
output: the best solution S
begin
    t←0;
    initialize operation sequence section P(t) by random key-based encoding routine;
    evaluate P(t) by decoding routine with eval(P) and keep the best solution;
    while (not terminating condition) do
        create P'(t) from P(t) by GA+PSO routine;
        replace v_j ← v_j' if eval(v_j) < eval(v_j') for all v_j' ∈ P'(t) and v_j ∈ P(t);
        create C(t) from P(t) by one-cut chromosome routine;
        evaluate P(t) by decoding routine with eval(P) and update the best solution;
        select P(t+1) from P(t) and C(t) by roulette wheel selection routine;
        t←t+1;
    end
    output a schedule S by the best solution
end;
```

Experimental comparisons

By using HEA, random key-based section (operation sequencing) is considered. We combine HEA with 3 EAs to solve JSP. A random key-based GA (rkGA) is proposed by Norman and Bean (1995), a priority-based GA (priGA) is proposed by Zhang and Gen (2006), a random key-based PSO (rkPSO) is proposed by Guo et al. (2009). We used Ta31-Ta70 well-known benchmarks of JSP from OR-Library. All of tests are conducted 30 runs on a machine running on Intel Xeon 2.00 GHz CPU and 4 GB of memory. Table 1 summarizes the experimental results. The result clearly indicates that all of results by HEA are better than each of the other EA approaches.

Multiobjective flexible JSP

Flexible job shop scheduling (FJSP) 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 is 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 assignment of operations to machines, and the second one is the operations sequencing problem (i.e., determining the starting time of each operation). The FJSP is NP-hard

Table 1 Performance comparisons with different EAs by 20 JSP tests

Problem	Size	rkGA	priGA	rkPSO	HEA
ta31	30 × 15	1,819	1,780	1,849	1,776
ta32	30 × 15	1,859	1,823	1,872	1,823
ta33	30 × 15	1,848	1,811	1,864	1,811
ta34	30 × 15	1,876	1,850	1,906	1,842
ta35	30 × 15	2,063	2,021	2,080	2,022
ta51	50 × 15	2,815	2,789	2,827	2,788
ta52	50 × 15	2,797	2,797	2,816	2,787
ta53	50 × 15	2,770	2,733	2,795	2,723
ta54	50 × 15	2,896	2,877	2,922	2,867
ta55	50 × 15	2,744	2,697	2,774	2,692
ta61	50 × 20	2,914	2,888	2,927	2,880
ta62	50 × 20	2,923	2,894	2,947	2,886
ta63	50 × 20	2,796	2,777	2,819	2,769
ta64	50 × 20	2,764	2,725	2,774	2,724
ta65	50 × 20	2,774	2,737	2,796	2,735
ta66	50 × 20	2,885	2,877	2,913	2,871
ta67	50 × 20	2,866	2,842	2,891	2,840
ta68	50 × 20	2,832	2,805	2853	2,806
ta69	50 × 20	3,139	3,094	3,151	3,092
ta70	50 × 20	3,040	3,015	3,058	3,008

rkGA random key based GA, *priGA* priority based GA, *rkPPS* random key based PSO, *HEA* hybrid EA

since it is an extension of the job shop scheduling problem (Garey et al. 1976).

The FJSP problem could be described as an n-job m-machine scheduling problem by simple equations as follows:

$$\min \quad t_{\mathrm{M}} = \max_j \left\{ t_j^T \right\} \tag{21}$$

$$\min \quad W_{\mathrm{M}} = \max_h \{W_h\} \tag{22}$$

$$\min \quad W_{\mathrm{T}} = \sum_h W_h \tag{23}$$

$$\text{where } W_h = \sum_j p_j \cdot x_{jh}$$

$$\text{s. t.} \quad t_{j'}^T - p_{j'} \geq t_j^T, \quad \forall \left\{ j,\ j' \,\middle|\, E_{jj'} = 1 \right\} \tag{24}$$

$$\sum_h \left(x_{jh} \cdot T_{ju}^h \right) = 1, \quad \forall h \tag{25}$$

$$t_j^T \geq 0, \quad \forall j \tag{26}$$

$$x_{jh} = \{0,\ 1\}, \quad \forall j,\ h \tag{27}$$

The objective functions accounts Eq. (21) is to minimize the makespan, Eq. (22) is to minimize the maximal machine workload (i.e., the maximum working time spent at any machine), Eq. (23) is to minimize the total workload (i.e., the total working time over all machines). Eq. (24) 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. (25) states that one machine must be selected for each operation.

Hybrid GA for FJSP

The $P(t)$ and $C(t)$ are parents and offspring respectively in current generation t, the implementation structure of hGA for scheduling is described as follows:

```
procedure: HGA for FJSP
input: scheduling data (N, E), HGA parameters
output: Pareto optimal solutions E
begin
    t←0;
    initialize operation sequence section P1(t) by priority-based encoding routine;
    initialize resources assignment sections P2(t) by permutation encoding routine;
                    // P(t)={P1(t) || P2(t)}: population of individuals
    calculate objective f(P(t)) by decoding routine and calculate eval(P) by fitness assignment;
    create Pareto E(P) by nodominated routine and keep the best solution;
    while (not termination condition) do
        create C(t) from P(t) by exchange crossover routine;
        create C(t) from P(t) by allele-based mutation routine;    //C(t) ={C1(t) || C2(t) }: offspring
        improve P(t) and C(t) to yield P'(t) and C'(t) by bottleneck shifting routine;
        calculate objective f(C) by decoding routine and calculate eval(C) by fitness assignment;
        update Pareto E(P,C) by nondominated routine and update the best solution;
        select P(t+1) from P(t) and C(t) by mixed sampling routine;
        t←t+1;
    end
    output Pareto optimal solutions E;
end;
```

Experimental comparisons

In order to test the effectiveness and performance of EAs, three representative instances (represented by problem $n \times m$) were selected for simulation. The works by (Kacem et al. 2002a), (Xia and Wu 2005), and (Gen et al. 2008) are among the most recent progresses made in the area of FJSP. All the simulation experiments were performed with Delphi on Pentium 4 processor (2.6-GHz clock).

Table 2 gives the performance of EAs. "Approach by Localization" and "AL+CGA" are two algorithms by Kacem

Table 2 Performance of multiobjective EAs for the 3 FJSP problems

Problem		Classical GA	AL+CGA		PSO+SA		HGA
8 × 8	t_{M}	16	15	16	15	16	15
	w_{M}				12	13	12
	w_{T}	77	79	75	75	73	75
10 × 10	t_{M}	7	7		7		7
	w_{M}	7	5		6		5
	w_{T}	53	45		44		43
15 × 10	t_{M}	23	24		12		11
	w_{M}	11	11		11		11
	w_{T}	95	91		91		91

et al. (2002a, b). "PSO+SA" is the algorithm by Xia and Wu (2005), and "HGA" is proposed by Gen et al. (2009).

AGV dispatching in FMS

Flexible manufacturing system (FMS) is a machinery manufacturing system controlled by a unified information system, material-handling system and a set of numerical control processing equipment components. It is able to use evolutionary computation technology to optimize production scheduling, rapidly adjust resource allocation, improve equipment utilization, and co-ordinating arrangements for the production schedule. On the other hand, an automated material handling is significant to integrated manufacturing, so it also should be considered during solving optimal scheduling problem. The state-of-art AGV is one of the automated material-handling systems (AMHS), but it is often used to facilitate automatic storage and retrieval system (AS/RS). So the AGV is rarely a material-handling system in the FMS.

In this field, many researchers have studied for decades to show the fact that evolutionary computation algorithms have been demonstrated to be suitable for the optimization in the FMS. For instance, recently Choudhury et al. (2009) tried to find appropriate GAs for planning & scheduling in batch mode. Zhao et al. (2011) used an effective Petri net modeling with GA and real-time modeling to solve Scheduling optimization problem. Wang et al. (2008) employed a hybrid algorithm of PSO and SA to solve optimal solution. Considering AGV, Song et al. (2008) came up with an AGV Dispatching Strategy based on theory of constraints (TOC) that can make bottleneck machine work fully without starvation and blocking. Vis (2006) give a survey of research in the design and control of AGV systems. Lin et al. (2006) presented the decision variables and system constraints of AGVs on the network. Let $G = (V,\ E)$ be a connected directed network with $n = |V|$ nodes and $m = |E|$ edges, nodes represent the tasks that transport the material from a pickup point to a delivery point. After the network structure is generated, all of the system constraints are included.

Assumptions are showed as follows:

A1. AGVs only carry one kind of products at same time.
A2. A network of guide paths is defined advance, and the guide paths have to through all of pickup/delivery points.
A3. The vehicles are assumed to travel at a constant speed.
A4. The vehicles just can travel forward, not backward.
A5. As many vehicles travel on the guide path simultaneously, collisions be avoided by hardware, not be considered in this paper.
A6. On each working stations, there are pickup space for store the operated material and delivery space for store the material for next operation.
A7. 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.

$$\min \quad t_{\mathrm{M}} = \max_j \left\{ t_j^T + t_{jj'}^L \right\} \tag{28}$$

$$\min \quad n_{AGV} = \max_{j,\, j'} \left\{ z_{jj'} \right\} \tag{29}$$

$$\text{s. t.} \quad t_{j'}^T - p_{j'} \geq t_j^T + t_{jj'}^L \cdot \min\left(1,\ z_{jj'}\right), \quad \forall \left\{ j,\ j' \,\middle|\, E_{jj'} = 1 \right\} \tag{30}$$

$$\sum_h \left(x_{jh} \cdot T_{ju}^h \right) = 1, \quad \forall h \tag{31}$$

$$t_a^T + t_{aj}^L \cdot \min\left(1,\ z_{aj}\right) \leq t_{j''}^T, \quad \forall \left\{ j,\ j'' \,\middle|\, z_{ja} = z_{j''a}, \quad \forall a \right\} \tag{32}$$

$$t_j^T \geq 0, \quad \forall j \tag{33}$$

$$x_{jh} = \{0,\ 1\}, \quad \forall j,\ h \tag{34}$$

$$z_{jj'} \geq 0, \quad \forall j \tag{35}$$

The objective functions accounts Eq. (28) is to minimize the time for completing all the tasks (i.e. makespan), Eq. (29) is to minimize the number of AGVs. In Eqs. (30) and (32), since one or the other constraint must be hold, it is called disjunctive constraint. It represents the operation un-overlapping constraint (Eq. 30) and the AGV non-overlapping constraint (Eq. 32).

Hybrid EA for AGV dispatching

AGV dispatching is a combination of AGV assignment and task sequencing. A solution can be described by the assignment of task on AGVs, and the task sequence that transports the material from a pickup point to a delivery point. Liang et al. (2012) proposed a random key-based hybrid EA for AGV dispatching. As we know, the real number can be separated with integer and decimal. We consider the part of integer decides AGV assignment, and consider the part of decimal gives the priority of each task. The detailed encoding and decoding processes are separate into following phases:

Phase 1: Random Key-based Encoding Process
Phase 2: Grouping Process (AGV Assignment)
2-1 Extraction of Integer Part
2-2 AGV Assignment
Phase 3: Task Sequencing Growth Process
3-1 Extraction of Decimal Part
3-2 Task Sequencing

The $P(t)$ and $C(t)$ are parents and offspring respectively in current generation t, the implementation structure of hybrid EA (HEA) for scheduling is described as follows:

procedure: HEA for AGV Dispatching
input: AGV dispatching and scheduling data (*N*, *E*), HEA parameters
output: the best solution *S*
begin
  t←0;
  initialize AGV sequence *P*(*t*) by random key-based encoding routine;
  calculate objective **f**(*P*) by decoding routine and calculate *eval*(*P*) by fitness assignment;
  create Pareto *E*(*P*) by nodominated routine and keep the best solution;
  while (**not** terminating condition) **do**
    create *P*'(*t*) from *P*(*t*) by *GA+PSO* routine;
    replace $v_j \leftarrow v_{j'}$ if $eval(v_j) < eval(v_{j'})$ for all $v_{j'} \in P'(t)$ and $v_j \in P(t)$;
    create *C*(*t*) from *P*(*t*) by one-cut chromosome routine;
    calculate objective **f**(*C*) by decoding routine and calculate *eval*(*C*) by fitness assignment;
    update Pareto *E*(*P*,*C*) by nondominated routine and update the best solution;
    select *P*(*t*+1) from *P*(*t*) and *C*(*t*) by roulette wheel selection routine;
    t←*t*+1;
  end
  output a AGV dispatching *S* by the Pareto best solution.
end;

Experimental comparisons

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). In a case study of FMS, 10 jobs are to be scheduled on 5 machines. The maximum number process for the operations is 4. Depend on Naso and Turchiano (2005), a layout of facility is given for the experiment in Fig. 12.

We combine HEA with GA to solve AGV dispatching. A priority-based GA (priGA) is proposed by Lin et al. (2006). GA parameter settings were taken as follows: population size, *popSize* = 20; crossover probability, $p_C = 0.70$; mutation probability, $p_M = 0.50$. The HEA parameter settings were taken as follows: population size, *popSize*=10; crossover probability, $p_C = 0.02$; mutation probability, $p_M = 0.02$; the velocity of advance x_i towards p_{pbest} with velocity $V_1 = c_1 = 0.6$; $V_2 = c_2 = 0.4$.; the velocity of Advance x_i towards p_{gbest} with velocity $V_2 = c_2$.

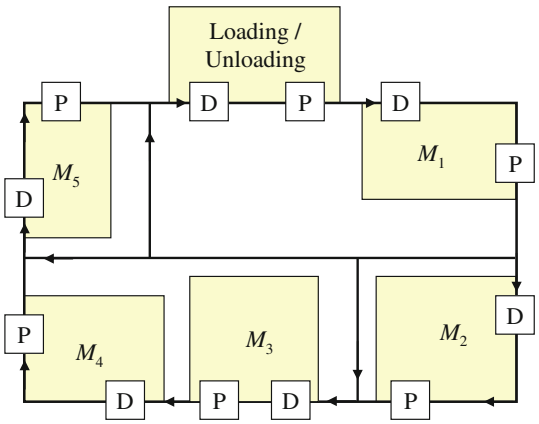


Fig. 12 Layout of facility (*P:* Pickup Point, *D:* Delivery Point)

Table 3 Experimental result summary

# of AGVs	priGA			HEA		
	Best	Worst	Average	Best	Worst	Average
4	590	636	614.5	574	602	581.8
5	524	561	541.0	512	543	524.5

The experimental results for 4 AGVs and 5 AGVs used are summarized in Table 3. Figure 13 gives the Gantt chart of the schedule with considering AGVs routing using 4 AGVs.

Advanced planning and scheduling

The advanced planning and scheduling (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. We find the process has been driven since the orders come from our customer. Moreover, to satisfy the requirements, some other constraints should be considered such as due date, setup time and shipping time. An integrated process planning and scheduling (IPPS) problem is proposed by Kim et al. (2003). Process planning is the determination of operations (machines, tools, and tool access directions) and their operation sequences for manufacturing are effective and economical partly. All the manufacturing resources are assumed as available in this phase. The scheduling is determination of the most appropriate moment to execute each operation in the shop over time with competitive resources. Figure 14 present two kinds of materials to be machined in a 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 4, which includes all the types of operations and the corresponding machines and tools selection.

The IPPS subjects to the following assumptions:

A1. Each machine can only handle one operation at each time.
A2. Each operation will be completed before another operation will be loaded.
A3. The sequence of the operations of each part complies with manufacturing constraints.
A4. All parts, machines and tools are available at time zero simultaneously.
A5. Each operation is performed on a single machine, and each machine can only execute an operation at a time.

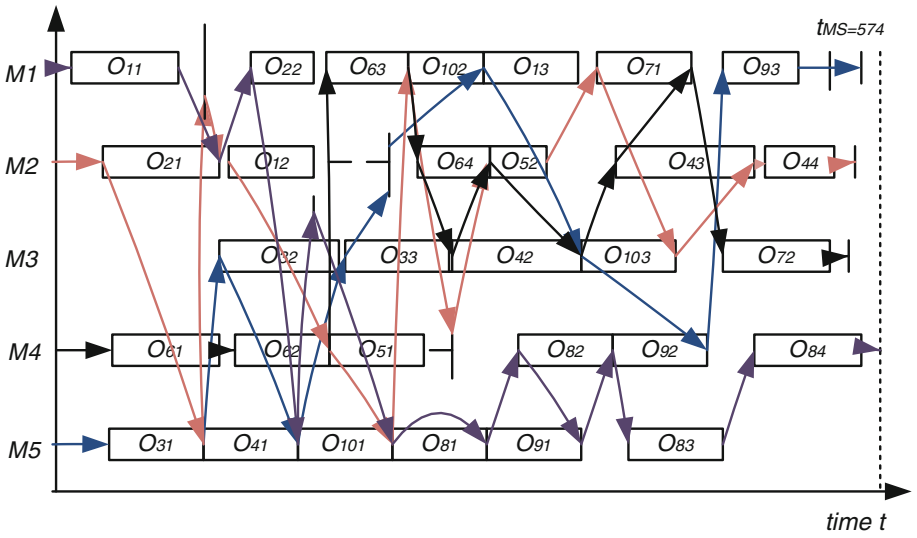


Fig. 13 Gantt chart of the schedule with considering AGVs routing

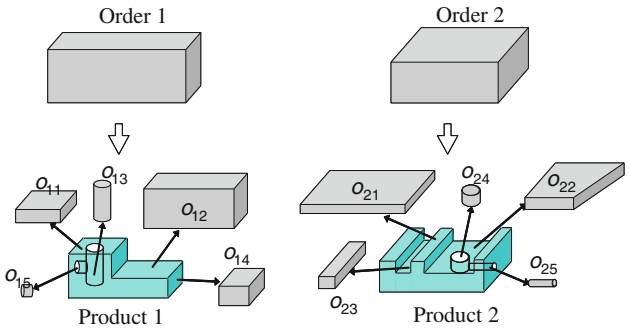


Fig. 14 A simple example for IPPS

A6. The time for a set-up is identical and independent of specific operations. The time for a machine change or a tool change follows the same value.

A7. Machines are continuously available for production.

$$\min \quad t_{\mathrm{M}} = \max_j \left\{ t_j^T \right\} \tag{36}$$

$$\min \quad W_{\mathrm{P}} = \sqrt{\frac{1}{T_1^R} \sum\nolimits_{h=1}^{T_1^R} W_h} \tag{37}$$

$$\text{where } W_h = \sum\nolimits_j p_j \cdot x_{jh}$$

$$\text{s. t.} \quad t_{j'}^T - p_{j'} \geq t_j^T, \quad \forall \left\{ j,\ j' \,\middle|\, E_{jj'} = 1 \right\} \tag{38}$$

$$\sum\nolimits_h \left(x_{jh}^1 \cdot T_{j1}^h \right) = 1, \quad \forall h \tag{39}$$

$$\sum\nolimits_h \left(x_{jh}^2 \cdot T_{j2}^h \right) = 1, \quad \forall h \tag{40}$$

$$t_j^T \geq 0, \quad \forall j \tag{41}$$

$$x_{jh}^1 = \{0,\ 1\}, \quad \forall j,\ h \tag{42}$$

$$x_{jh}^2 = \{0,\ 1\}, \quad \forall j,\ h \tag{43}$$

Table 4 Operation information for 2 order case

Operation ID (order, operation)	Operation type	Machine selection	Tool candidates
1 (1, o_{11})	Milling	M_1, M_4	T_6, T_9
2 (1, o_{12})	Milling	M_1, M_4	T_9, T_{10}
3 (1, o_{13})	Drilling	M_1, M_3, M_5	T_3
4 (1, o_{14})	Milling	M_1	T_1, T_3
5 (1, o_{15})	Drilling	M_2	T_2
6 (2, o_{21})	Milling	M_1, M_2	T_1, T_3
7 (2, o_{22})	Milling	M_1, M_3	T_1, T_6
8 (2, o_{23})	Milling	M_3, M_5	T_1
9 (2, o_{24})	Drilling	M_1, M_4, M_5	T_2, T_3
10 (2, o_{25})	Drilling	M_1, M_2	T_2

The objective functions accounts Eq. (36) is to minimize makespan, Eq. (37) is to minimize workload variance, defined as standard deviation of workload of all the machines. Equation (38) states that the successive operation has to be started after the completion of its precedent operation. Equation (39) states that one machine must be selected for each operation. Equation (40) states that assignment of tool must be selected for each operation.

Hybrid EA for IPPS

The $P(t)$ and $C(t)$ are parents and offspring respectively in current generation t, the implementation structure of HEA is described as follows:

Table 5 Performance comparisons with different EAs by 24 IPPS tests

Problem	1	2	3	4	5	6	7	8	9	10	11	12
# of Jobs	6	6	6	6	6	6	6	6	6	9	9	9
priGA	438.70	349.60	360.30	306.40	333.00	451.60	377.40	356.90	432.80	455.20	374.20	339.00
rkPSO	437.60	350.33	361.20	305.20	334.50	451.20	380.20	340.80	428.40	457.30	370.50	341.33
HEA	438.90	349.80	360.00	305.80	332.80	450.80	379.40	352.10	419.50	450.70	359.50	328.40
Problem	13	14	15	16	17	18	19	20	21	22	23	24
# of Jobs	9	9	9	12	12	12	12	12	12	15	15	18
priGA	455.40	395.50	444.60	472.10	437.50	391.70	447.40	440.50	482.60	521.80	497.00	564.80
rkPSO	450.33	398.20	440.50	468.72	435.50	385.66	440.50	435.50	473.40	513.60	487.50	560.40
HEA	436.80	378.80	424.20	447.60	412.50	370.40	410.80	420.00	454.33	494.50	451.50	520.60

procedure: HEA for IPPS
input: IPPS data (*N*, *E*), HEA parameters
output: Pareto optimal solutions *E*
begin
t←0;
initialize operation sequence section $P_1(t)$ by random key-based encoding routine;
initialize resource assignment sections $[P_0(t)]$ by state permutation encoding routine;
calculate objective $\boldsymbol{f}(P)$ by decoding routine and evaluate *eval*(*P*) by HSS-EA evaluation routine;
create Pareto E(P) by nondominated routine and keep the best solution;
while (**not** termination condition) **do**
create *P*'(*t*) from *P*(*t*) by *GA+PSO* routine;
replace $v_j \leftarrow v_{j'}$ if $eval(v_j) < eval(v_{j'})$ for all $v_{j'} \in P'(t)$ and $v_j \in P(t)$;
create *C*(*t*) from *P*(*t*) by one-cut crossover routine;
calculate objective $\boldsymbol{f}(P)$ by decoding and evaluate *eval*(*C*) by HSS-EA evaluation routine;
update Pareto *E*(*P*, *C*) by nondominated routine and update the best solution;
select *P*(*t*+1) from *P*(*t*) and *C*(*t*) by roulette wheel selection routine;
t←*t*+1;
end
output the best solution *S*;
end;

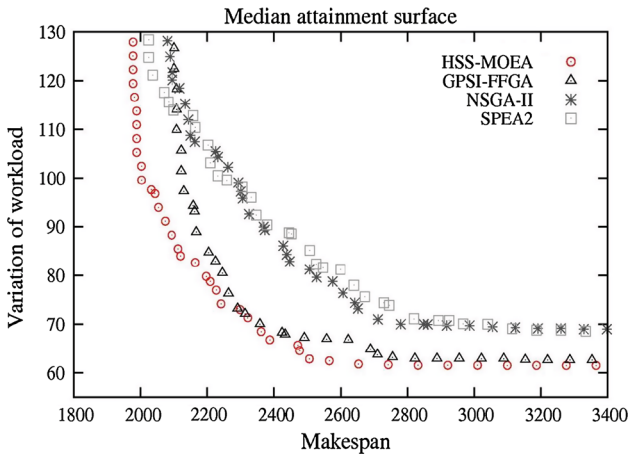


Fig. 15 50 % attainment surface by HSS-EA, GPSI-FFGA, NSGA-II and SPEA2

Experimental comparisons

To evaluate the effectiveness of random key-based representation and GA+PSO evolutionary operation, we compare HEA with a priority-based GA (priGA) which is proposed by Zhang and Gen (2006), a random key-based PSO (rkPSO) is proposed by Guo et al. (2009). We considered one objective of minimizing makespan, and generated 24 IPPS Tests with different number of jobs. All of tests are conducted 30 runs on a machine running on Intel Xeon 2.00 GHz CPU and 4 GB of memory. Table 5 summarizes the experimental results. Although considering the small-scale problems, HEA has the same performance with prGA and rkPSO, HEA has a high performance for solving large scale IPPS problems.

In order to investigate the effectiveness of multiobjective EAs to solve IPPS, HSS-EA was compared with GPSI-FF proposed by Ho et al. (2004), NSGA-II, and SPEA2. A 4 parts (each part having 20, 16, 14 and 7 operations, respectively) problem was suggested by Li and McMahon (2007). Figure 15 shows the 50 % attainment surface by using HSS-MOEA, GPSI-FFGA, NSGA-II and SPEA2 with 30 runs.

Conclusion

In this paper, we summarized a classification of manufacturing scheduling problems, and surveyed the design ways of EAs for the different types of scheduling problems. Firstly, we introduced multiobjective EA, and give fitness assignment mechanism, performance measures for multiobjective scheduling problems. Then we introduced how to design a representation, and how to improve initialization by evolutionary operators. We classified the scheduling problems as operation sequencing problems, operations (or shipments) selection problems, resources assignment problems, and operations (or shipments) grouping problems. We introduced the design ways to apply EAs to the different typical scheduling problems, including job shop scheduling problem (JSP) (operation sequencing), flexible JSP (operation sequencing with resources assignment), AGV dispatching in FMS (shipments grouping and assignment), integrated process planning and scheduling (operation sequencing with multiple resources assignment). Through a variety of numerical exper-

iments, we demonstrated the effectiveness of these EAs in the widely applications of manufacturing scheduling problems.

This paper summarized the optimization of manufacturing scheduling problems with different system constraints. As future researches, we will have in-depth investigation and research, analysis the impact of scheduling under the different environment, such as dynamic scheduling problems, robust scheduling etc. We will also have in-depth research on the convergence and stability of EAs to solve different types of scheduling problem under different environments, and the solution diversity for multiobjective scheduling problems.

Acknowledgments This work is partly supported by the Japan Society of Promotion of Science (JSPS):Grant-in-Aid for Scientific Research (C) (No. 24510219.0001), National Science Council (NSC 101-2811-E-007-004, NSC 102-2811-E-007-005), the Fundamental Research Funds (Software+X) of Dalian University of Technology (No. DUT12JR05, No. DUT12JR12), and supported by New Teacher Fund of Ministry of Education of China (No. 20120041120053).

References

Baker, k, & Scudder, G. (1990). Sequencing with earliness & tardiness penalties: A review. *Operations Research*, *38*, 22–36.

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.

Cheng, R., & Gen, M. (1994). Evolution program for resource constrained project scheduling problem. In *Proceedings of IEEE international conference of, evolutionary computation*, pp. 736–741.

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

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

Choudhury, B. B., Biswal, B. B., Mishra, D., & Mahapatra, R. N. (2009). Appropriate evolutionary algorithm for scheduling in FMS. *NaBIC World Congress on Nature & Biologically Inspired, Computing*, pp. 1139–1144.

Croce, F., Tadei, R., & Volta, G. (1995). A genetic algorithm for the job shop problem. *Computer & Operations Research*, *22*, 15–24.

Dahal, K., Tan, K. C., & Cowling, P. I. (2007). *Evolutionary scheduling*. Berlin: Springer.

De Jong, K. (1994). Genetic algorithms: A 25 year perspective. *Computational Intelligence: Imitating Life*, pp. 125–134.

Deb, K. (2001). *Multiobjective optimization using evolutionary algorithms*. Chichester, UK: Wiley.

Dev, K. (1995). *Optimization for engineering design: Algorithms and examples*. New Delhi: Prentice-Hall.

Dorndorf, W., & Pesch, E. (1995). Evolution based learning in a job shop scheduling environment. *Computer & Operations Research*, *22*, 25–40.

Elyn, L. Solano-Charris, Jairo, R. Montoya-Torres, & Carlos, D. Paternina-Arboleda. (2011). Ant colony optimization algorithm for a Bi-criteria 2-stage hybrid flowshop scheduling problem. *Journal of Intelligent Manufacturing*, *22*(5), 815–822.

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

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

Fonseca, C., & Fleming, P. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, *3*(1), 1–16.

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

Gao, J., Sun, L., & Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, *35*(9), 2892–2907.

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

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

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: Wiley.

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

Gen, M., & Zhang, H. (2006). Effective designing chromosome for optimizing advanced planning and scheduling. *Intelligent Engineering Systems Through Artificial Neural Networks*, *16*, 61–66.

Gen, M., Cheng, R., & Lin, L. (2008). *Network models and optimization: Multiobjective genetic algorithm approach*. Berlin: Springer.

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

Gholami, M., & Zandieh, M. (2009). Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. *Journal of Intelligent Manufacturing*, *20*(4), 481–498.

Guo, Y. W., Mileham, A. R., Owen, G. W., & Li, W. D. (2006). Operation sequencing optimization using a particle swarm optimization approach. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *220*(12), 1945–1958.

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

Handa, H., Kawakami, H., & Katai, O. (2008). Recent advances in evolutionary computation. *IEEJ Transactions on Electronics, Information & Systems*, *128*(3), 334–339.

Ho, S., Shu, L., & Chen, J. (2004). Intelligent evolutionary algorithms for large parameter optimization problems. *IEEE Transactions on Evolutionary Computation*, *8*(6), 522–541.

Hwang, C., & Yoon, K. (1981). *Multiple attribute decision making: Methods and applications*. Berlin: Springer.

Ishibuchi, H., & Murata, T. (1998). A multiobjective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, & Cybernetics*, *28*(3), 392–403.

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

Karimi-Nasab, M., & Aryanezhad, M. B. (2011). A multi-objective production smoothing model with compressible operating times. *Applied Mathematical Modeling*, *35*, 3596–3610.

Kim, Y., Park, K., & Ko, J. (2003). A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers and Operations Research, 30*, 1151–1171.

Kim, K., Yamazaki, G., Lin, L., & Gen, M. (2004). Network-based hybrid genetic algorithm to the scheduling in FMS environments. *Journal of Artificial Life and Robotics, 8*(1), 67–76.

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

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

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

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

Liang, Y., Lin, L., Gen, M., & Ohno, K. (2012). A hybrid evolutionary algorithm for FMS optimization with AGV dispatching. In *Proceedings of the 42nd international conference on computers and industrial engineering*, pp. 296.1–296.14.

Lin, L., Shinn, S. W., Gen, M., & Hwang, H. (2006). Network model and effective evolutionary approach for AGV dispatching in manufacturing system. *Journal of Intelligent Manufacturing*, *17*(4), 465–477.

Lin, L., Gen, M., Liang, Y., & Ohno, K. (2012). A hybrid EA for reactive flexible job-shop scheduling. *Complex Adaptive Systems.*, *12*, 110–115.

Lopez, O., & Ramirez, M. (2005). A STEP-based manufacturing information system to share flexible manufacturing resources data. *Journal of Intelligent Manufacturing*, *16*(3), 287–301.

Meeran, S., & Morshed, M. S. (2012). A hybrid genetic tabu search algorithm for solving job shop scheduling problems: A case study. *Journal of Intelligent Manufacturing*, *23*(4), 1063–1078.

Michalewicz, Z. (1994). *Genetic algorithm + data structures = evolution programs*. Berlin: Springer.

Najid, N. M., Dauzere-Peres, S., & Zaidat, A. (2002). A modified simulated annealing method for flexible job shop scheduling problem. *IEEE International Conference on Systems, Man and Cybernetics, 5*, 6–9.

Naso, D., & 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.

Norman, B., & Bean, J. (1995). *Random keys genetic algorithm for job-shop scheduling: Unabridged version*. Technical report, University of Michigan.

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

Okamoto, A., Gen, M., & Sugawara, M. (2005). Cooperation of scheduling agent and transportation agent in APS system. In *Proceedings of the JSLS Kyushu division conference*, pp. 1–11 (in Japanese).

Pareto, V. (1906). *Manuale di Economica Polittica*. Milan, Italy: Societa Editrice Libraia.

Pinedo, M. (2002). *Scheduling theory, algorithms and systems*. Upper Saddle River, NJ: Prentice-Hall.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of 1st international conference on GAs*, pp. 93–100.

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

Song, S.-G., Li, A.-p., & Xu, L.-Y. (2008). AGV dispatching strategy based on theory of constraints, automation and mechatronics. In *Proceedings of IEEE conference on, robotics*, pp. 922–925.

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

Verderame, P. M., & Christodoulos, A. F. (2008). Integrated Operational Planning and Medium-Term Scheduling for Large-Scale Industrial Batch Plants. *Industrial & 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.

Voratas, K., & Siriwan, S. (2011). A two-stage genetic algorithm for multi-objective job shop scheduling problems. *Journal of Intelligent Manufacturing*, *22*(3), 355–365.

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

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

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for muti-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, *48*, 409–425.

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

Yamada, T., & Nakano, R. (1992). A genetic algorithm applicable to large-scale job-shop problems. *Parallel Problem Solving from Nature: PPSN*, *II*, 281–290.

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

Zandieh, M., & Karimi, N. (2011). An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *22*(6), 979–989.

Zhang, H., & Gen, M. (2006). Effective genetic approach for optimizing advanced planning and scheduling in flexible manufacturing system. In *Proceedings of GECCO*, pp. 1841–1848.

Zhang, H., & Gen, M. (2005). Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Journal of Complexity International*, *11*, 223–232.

Zhang, W., Gen, M., & Jo, J.-B. (2012a). Hybrid sampling strategy-based multiobjective evolutionary algorithm for process planning and scheduling problem. In *Proceedings of international symposium on semiconductor manufacturing intelligence*.

Zhang, W., Lin, L., Gen, M., & Chien, C. F. (2012b). Hybrid sampling strategy-based multiobjective evolutionary algorithm. *Complex Adaptive Systems*, *12*, 96–101.

Zhao, Z.-X., Zhang, G.-S., & Bing, Z.-G. (2011). Scheduling optimization for FMS based on Petri net modeling and GA. In: *Proceedings of IEEE international conference on automation and logistics*, pp. 422–427.

Zitzler, E., & Thiele, L. (2001). *SPEA2: Improving the strength Pareto evolutionary algorithm*, Technical report 103, Computer Engineering and Communication Networks Lab (TIK).

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