European Journal of Operational Research 244 (2015) 434–444

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An object-coding genetic algorithm for integrated process planning and scheduling

Luping Zhang, T. N. Wong*

Department of Industrial & Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong, Pokfulam Road, China

ARTICLE INFO

Article history:
Received 31 March 2014
Accepted 18 January 2015
Available online 23 January 2015

Keywords:
Process planning
Jobshop scheduling
Genetic algorithm
Object-coding

ABSTRACT

Process planning and jobshop scheduling problems are both crucial functions in manufacturing. In reality, dynamic disruptions such as machine breakdown or rush order will affect the feasibility and optimality of the sequentially-generated process plans and machining schedules. With the approach of integrated process planning and scheduling (IPPS), the actual process plan and the schedule are determined dynamically in accordance with the order details and the status of the manufacturing system. In this paper, an object-coding genetic algorithm (OCGA) is proposed to resolve the IPPS problems in a jobshop type of flexible manufacturing systems. An effective object-coding representation and its corresponding genetic operations are suggested, where real objects like machining operations are directly used to represent genes. Based on the object-coding representation, customized methods are proposed to fulfill the genetic operations. An unusual selection and a replacement strategy are integrated systematically for the population evolution, aiming to achieve near-optimal solutions through gradually improving the overall quality of the population, instead of exploring neighborhoods of good individuals. Experiments show that the proposed genetic algorithm can generate outstanding outcomes for complex IPPS instances.

1. Introduction

Both process planning and scheduling functions are responsible for the efficient allocation and utilization of resources in manufacturing systems. Generally, these two functions are conducted sequentially (Usher & Fernandes, 1996). Process planning determines the selection and sequence of production operations, and essential manufacturing resources including machines and tools. The purpose of scheduling is to allocate the jobs and operations to limited manufacturing recourses in accordance with the process plan.

The dynamic manufacturing environment is susceptible to disturbances such as machine breakdowns, rush order arrivals and order cancellations. These uncertainties will cause deviations to the original process plans and schedules, and will affect the performance of a manufacturing system (Kumar & Rajotia, 2003). To cope with the unexpected disturbances, it is necessary to revise the process plan and the schedule dynamically. Integrated process planning and scheduling (IPPS) is proposed to integrate the process design, resource allocation and scheduling into a cohesive function to enhance the manufacturing feasibility and performance (Chryssolouris, Chan, & Suh, 1985).

For a manufacture order to produce **n** jobs (parts) on **m** machines in a jobshop or similar kind of flexible manufacturing environment, an IPPS system intends to generate the process plans for all **n** parts and the overall job-shop schedule concurrently, with the aim to optimize a manufacturing objective such as makespan. Massive research has been done respectively in the fields of process planning and jobshop scheduling. In contrast, there is only a limited number of research on IPPS. Due to the combination of the two optimization problems and the flexibilities in the manufacturing systems, large-scale IPPS problems are difficult to be solved with analytical approaches. Researchers have attempted to solve IPPS problems with various approximate approaches, for instance, evolutionary algorithms, tabu search, simulated annealing and ant algorithms. The authors' research group has investigated on the applications of multi-agent system (MAS) in IPPS. Recently, work has been conducted on the combination of MAS and metaheuristics in solving IPPS problems. This hybrid approach is to adopt the excellent search ability of metaheuristics and autonomy of agents for dynamic adaptation in the manufacturing environment. A MAS structure has been designed to support a variety of metaheuristics for process planning and scheduling/rescheduling in dynamic manufacturing environments. For instance, an enhance ACO algorithm has been established to demonstrate the applicability of the hybrid MAS-ACO solution approach for IPPS problems. In addition to ACO, a novel genetic algorithm (GA) has also been implemented in the MAS architecture to evaluate the hybrid MAS-GA approach for IPPS.

* Corresponding author. Tel.: +86 852 2859 7055; fax: +86 852 2858 6535.
E-mail address: tnwong@hku.hk (T.N. Wong).

http://dx.doi.org/10.1016/j.ejor.2015.01.032
0377-2217/

Descriptions on the design and implementation of the MAS architecture are available in Zhang et al. (Zhang, Wong, & Fung, 2013). This current paper presents details of the GA-based IPPS solution approach. The GA is denoted as an object-coding genetic algorithm (OCGA). In essence, the classical GA has been enhanced in three aspects to cater for the IPPS problem domain: (1) real objects (machining operations in IPPS) are used to represent chromosomes, and corresponding genetic operations are proposed; (2) an unusual selection method, based on the inferior selection mechanism, is adopted to give inferior solutions more chances to reproduce; (3) a simplified version of crowding is proposed to cater for the object-coding representation and release the selection pressure; and (4) a population degeneration mechanism is introduced to further preserve the population diversity.

The rest of the paper is arranged as follows: Section 2 is a review of relevant literature on GA implementation in process planning and scheduling; representation of the IPPS problem is presented in Section 3; the OCGA algorithm is elaborated in Section 4; experiments and results on benchmark problems are displayed in Section 5; the last section is the conclusion which restates the findings and suggests future research.

2. Literature review

Several earlier research attempts on the IPPS problem were reported in 1980s and 1990s. For instance, Chryssolouris, Chan, and Cobb (1984) modelled the IPPS as two decision-making stages and used a decision matrix for the combination of process planning and scheduling. In Shaw and Whinston (1985), a knowledge-based system was established to collaborate process planning and scheduling. Chen and Khoshnevis (1993) embedded a process planning module into a scheduling system. Some researchers established IPPS models in terms of different levels of flexibilities, and they also attempted to solve these models with analytical approaches (Li, Gao, Shao, Zhang, & Wang, 2010; Tan, 1998). However, because of the manufacturing and product complexities, process planning and scheduling problems are NP-hard which are impossible to achieve optimal solutions by traditional accurate algorithms with reasonable efforts (Kim, Park, & Ko, 2003). A wide variety of approximate algorithms have been adopted to find near-optimal solutions efficiently. Various heuristics of this kind have been tried, such as Genetic Algorithms (Morad & Zalzala, 1999), Simulated Annealing (Li & McMahon, 2007), Particle Swarm Optimization (Guo, Li, Mileham, & Owen, 2009), ant algorithm (Leung, Wong, Mak, & Fung, 2010), a new hybrid algorithm (HA) (Li, Shao, Gao, & Qian, 2010), an imperialist competitive algorithm (ICA) (Lian, Zhang, Gao, & Li, 2011), etc. Reviews of literature relevant to IPPS and MAS-based IPPS solution approaches are available in previous publications (Wong, Leung, Mak, & Fung, 2006) and not repeated here. The review in this section is focused on GA applications in IPPS problems.

2.1. GA-based applications in process planning and scheduling

Davis reported one of the first attempts to use GAs for jobshop problems (Davis, 1985). Thereafter GAs have been widely implemented in this field (Falkenauer & Bouffouix, 1991; Fang, Ross, & Corne, 1993; Gen, Tsujimura, & Kubota, 1994; Zhang, Zhang, & Nee, 1997). Some researchers contributed their efforts to improve GA-based jobshop scheduling systems, examples include GA for jobshop scheduling problems with alternate routings (Hussain & Joshi, 1998), and population-diversity-oriented selection procedures (Brizuela & Sannomiya, 1999), etc. A significant amount of relevant research was identified with implementation approaches based on different GA encoding schemes and different genetic operators. GAs were also hybridized with local search algorithms for the performance enhancement (Cai, Wu, & Yong, 2000; Liu & Xi, 2006; Resende, Goncalves, & Mendes, 2005). Particularly, Simulated Annealing (SA) was hybridized with GA to enhance the neighborhood search and avoid premature convergence (Wang & Zheng, 2002). A Tabu Search (TS) approach was also applied to enhance the performance of GA (Ombuki & Ventresca, 2004). Wong, Chan, and Lau (2003) used a hybrid of fuzzy and genetic approach for solving the process sequencing problem, which was based on a fuzzy expert system incorporated with GA for machining cost optimization according to the cost–tolerance relationship.

In IPPS consideration, GAs were traditionally applied individually (Morad & Zalzala, 1999), or combined with other optimization methods (Ye & Zhang, 2000). Lee and Kim (2001) reported the application of a simulation based GA to deal with the IPPS problem. Kim et al. (2003) proposed a symbiotic evolutionary algorithm (SEA) based on the idea of parallel search. Moreover, GA-based IPPS systems were also enhanced by adjusting the evolutionary procedure adaptively (Zhang & Gen, 2010), or improving the genetic representations and operator schemes (Li, Gao, Shao, Zhang, & Wang, 2010). Qiao and Lv (2012) proposed an improved genetic algorithm (IGA) which could generate good results for an IPPS benchmark problem.

2.2. GA implementation issues

Majority of GA implementations in constrained problems adopt an encoding/decoding scheme with which genes on chromosomes are represented by numbers or letters (Michalewicz, 1996). Commonly used encoding approaches include binary encoding, permutation encoding, real number encoding, and general data structure encoding (Gen & Cheng, 2000). According to the structure of encodings, encoding approaches can be classified into two types: one-dimensional encoding and multi-dimensional encoding (Gen & Cheng, 2000). With one-dimensional encoding, genes are arranged orderly in a succession. One-dimensional encoding is commonly adopted in previous implementations. However, for some complex problems, multi-dimensional encoding has to be adopted to encode chromosomes into a more complex structure. For instance, with two-dimensional encoding, a chromosome mostly looks like a matrix or a graph, for which Vignaux and Michalewicz (1991) provided a successful implementation in a linear transportation problem. Fixed-length and binary coded strings for encodings have dominated GA implementations. Theoretically, it is evident that they are the most appropriate approaches since they are amenable and could be easily implemented (Goldberg, 1990). Other categories of representation scheme include the real number representation (Lucasius & Kateman, 1989), and such an application for JSP problem has been provided by Gen et al. (1994). This kind of approach to represent chromosomes with strings of numerical numbers is generically named *numerical-number representation*.

Some researchers found that the numerical-number representation scheme might severely limit the ability of GAs to explore the search space (Koza, 1994). This limitation is intensified in the IPPS problem domain. Zhang et al. (1997) proposed a GA for the process planning and scheduling problem. They used operations directly for the representation of genes on chromosomes, however, only one job was considered in their approach. Wong et al. (2003) also applied a non-numerical-number representation scheme in their fuzzy-GA approach for machining process sequencing problems. Their approach behaved much like permutation encoding since all operations were labeled with consecutive integer numbers, and the algorithm followed the paradigm of permutation-encoding based GAs.

2.3. Issues affecting the implementation of GAs

In summary, conventional GAs possess several core functions and traditional practices which have some unavoidable weaknesses in the IPPS problem domain.

(i) Representation schemes – for a practical IPPS model with consideration of alternative machines, alternative processes and

alternative sequences, the transformation between practical operations and numerical numbers is usually quite difficult and rigid, which substantially ruins the flexibility of practical applications. Besides, not properly designed encoding schemes may have negative impacts on the algorithm performance.

(ii) Selection methods – inspired by the mechanism of nature selection, major GAs usually adopt the analogy of natural evolution strategy directly where relatively good individuals are more likely to be selected to breed the next generation. "Elitism" is regarded as one of the best strategies for genetic evolution (De Jong, 1975). However, the analogy of natural evolution may cause premature convergence or even search stagnation.

(iii) Crossover and mutation – Crossover and mutation are two main methods for neighborhood construction. Dedicated crossover and mutation approaches have to be defined when applying GAs in different areas. For instance, the one-point or two-point crossover which is commonly used for numerical optimization is not suitable for the IPPS problem, restricted by complex relationships among operations. To ensure that the process precedence constraints were not violated, job-based crossover approaches were adopted in many of the GA-based IPPS implementations (Pinedo, 2012; Qiao & Lv, 2012).

(iv) Replacement schemes – the replacement scheme is to define how to yield the next generation from current population. A large proportion of GAs adopts a complete replacement scheme with which the current population is completely replaced by the offspring. De Jong (1975) proposed another kind of replacement schemes called crowding, where only a fraction of the population propagate offspring and replace old ones at each generation, aiming to preserve the population diversity. There are several variants of De Jong's crowding scheme, the most widely-known ones are deterministic crowding (Mahfoud, 1995), restricted tournament selection (Harik, 1995), and the struggle genetic algorithm (Grüninger & Wallace, 1996). The complete replacement scheme is commonly used in the field of numerical optimization problems and performs very well. However, for large-scale discrete problems, the evolution process becomes more unpredictable. The GA may quickly converge or get trapped in a suboptimal solution, especially for those instances with high flexibility. Crowding and its variations are proposed to tackle this issue through preserving population diversity and maintaining multiple solutions, but they also introduce selection pressure to measure the similarity of individuals.

It is difficult or almost impossible to extend conventional GA approaches with numerical-number representations to solve the IPPS problem. Therefore, the GA approach proposed in this paper abandons the numerical encoding and uses a sequence of operations directly a chromosome. The significance of the proposed approach is not only to save the computational efforts, but also to enhance the search abilities.

3. Representation of the IPPS problem

This paper is to deal with the **n** jobs **m** machines IPPS problems in flexible jobshop type of manufacturing systems. Each job refers to the production of a single piece of product of a given design, comprising production operations which have to be processed on several non-identical machines. That is, a part or product is regarded as a job, and each job is composed of a series of operations. An operation can be processed on a dedicated set of alternative machines, and operations may have conjunction, disjunction and precedence relationships between them. Processing is non-preemptive. Setup and internal logistic consumption is deemed negligible or included in the processing time of the corresponding operation. For a job, operations are not allowed to overlap, that is, it is not legitimate to produce more than one operations of the same job at the same time. Also a machine is not allowed to process more than one operations at a time.

One aim of the IPPS is to increase manufacturing flexibilities. Three kinds of flexibilities: operation flexibility, sequencing flexibility and processing flexibility are taken into consideration in this paper. The IPPS system normally works in static environment, but it should also have abilities to react to changes and disruptions like machine breakdowns or rush order, updating current schedule or rescheduling if necessary.

To represent an IPPS problem, the complete set of feasible and alternative production processes and sequences for each part are recorded in an **AND/OR** graph. As an illustrative example, Fig. 1(a) and (b) presents feature descriptions of part 1 and part 2, respectively. Sequence relationships between features are given in Fig. 1(c). Corresponding machining operations include turning (F1.1–F1.3, F2.1–F2.6), milling (F1.4–F1.5, F2.1, F2.7), and drilling/reaming/boring (F1.6). An OR-link depicts a pair of alternative features' sequences for part 2, which are

(i) F2.1, F2.2 (turing) → F2.7 (milling) → …
(ii) F2.1, F2.7 (milling) → F2.2 (turning) → …

Assume that a lathe m_1, an NC lathe m_2, a milling machine m_3 and a drilling machine m_4 are available for processing all machining operations.

The AND/OR graphs in Fig. 2 are used to depict the relevant operations, sequence relationships and alternative process routes for the two sample parts respectively. The AND/OR graph model is denoted as D = (O, A), where O is a set of nodes, and A is a set of directed arcs. Two types of nodes are introduced in our AND/OR graph representation: operation nodes and dummy nodes (Wong et al., 2006). The operation nodes correspond to machining operations denoted by $O^k_{j.i}$, which means an operation with index number i of job j to be processed on machine k. The start and end dummy nodes of job j are denoted by S_j and E_j, respectively. A directed arc from $O^{k_1}_{j_1.i_1}$ to $O^{k_2}_{j_2.i_2}$ implies that $O^{k_1}_{j_1.i_1}$ is an immediate predecessor of $O^{k_2}_{j_2.i_2}$. An OR relationship in the AND/OR graph represents a pair of alternative process routes, where only one route is needed to finish the job. Additional input data like alternative machines and corresponding processing times are listed in Table 1.

4. Object-coding GA (OCGA)

The OCGA is a variant of GAs with object-coding representation. Its features include the real-object represented chromosome, customized genetic operations, and reconstructed evolutionary strategies. The "object" refers to practical components, such as the job, the machine, the machining operation, etc. Specifically in the IPPS problem, an "object" refers to a machining operation on a dedicated processing machine.

4.1. The overall procedure of the OCGA

The OCGA follows the classical GA approach in the overall procedure whose single iteration is mainly composed of the population maintenance, selection of parent individuals, genetic operations, and population evolution, as shown in Fig. 3.

4.2. Genetic representations

In major cases, the numerical-number representation can easily be implemented in GAs to solve the jobshop type of scheduling problems where the process plans for each job are usually fixed. Qiao and Lv (2012) proposed a dual-numerical-number coding for their improved

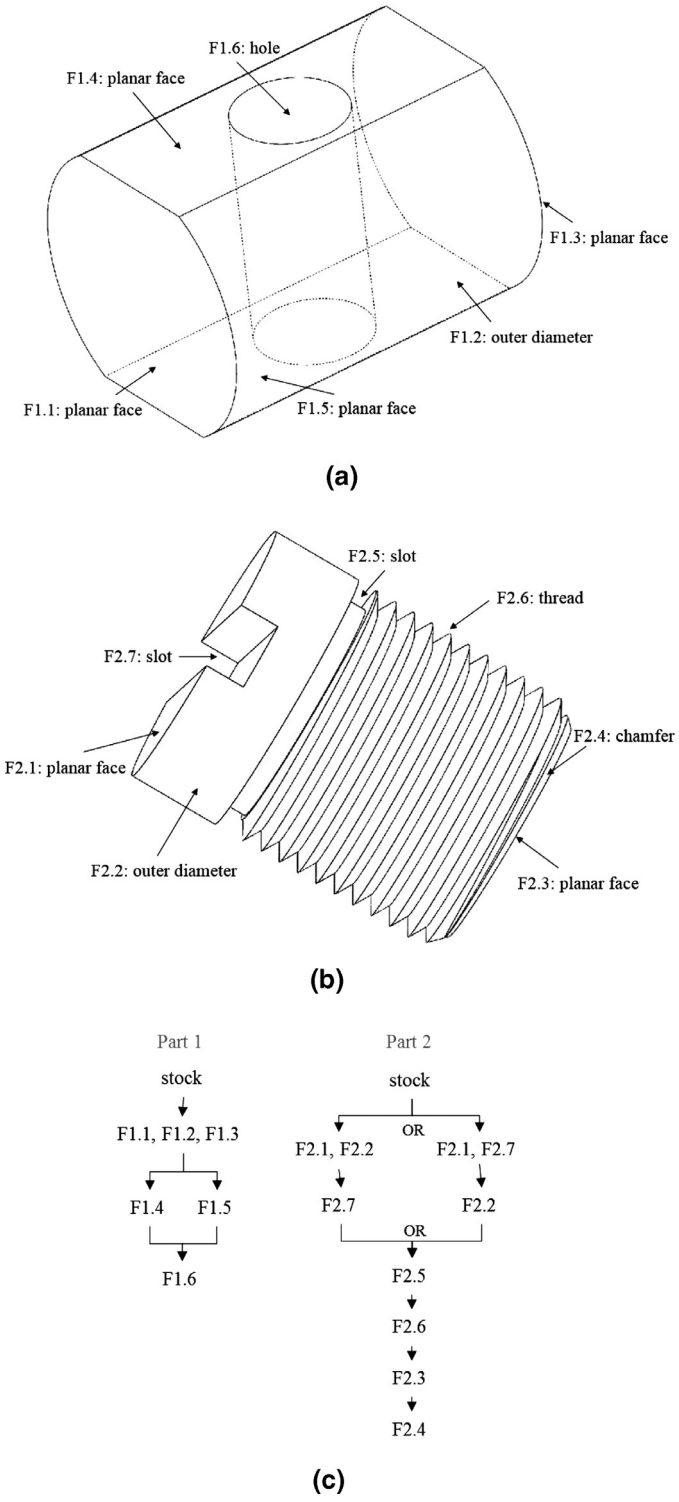


Fig. 1. (a) Part 1 and its features. (b) Part 2 and its features. (c) Sequence relationship of features.

genetic algorithm to cope with the IPPS model with full consideration of flexibilities of process plans. The dual-numerical-number coding uses two integer numbers to represent a gene and has two preliminary stages to generate operation sequences which are then encoded as chromosomes.

According to Qiao and Lv (2012), all operations involved in AND or OR relationships are combined and sequenced in different ways to form basic process routes for each job. All feasible process plans of the job can be yielded from shifting processing machines of each operation on these basic process routes. Shifting the processing machine of any operation results a new process plan. After that, a subsequent stage is carried out to combine process plans of different jobs to get a complete operation sequence for the IPPS instance which is then encoded as a chromosome. A chromosome based on Qiao and Lv's approach for the sample IPPS instance given in Section 3 is as follows:

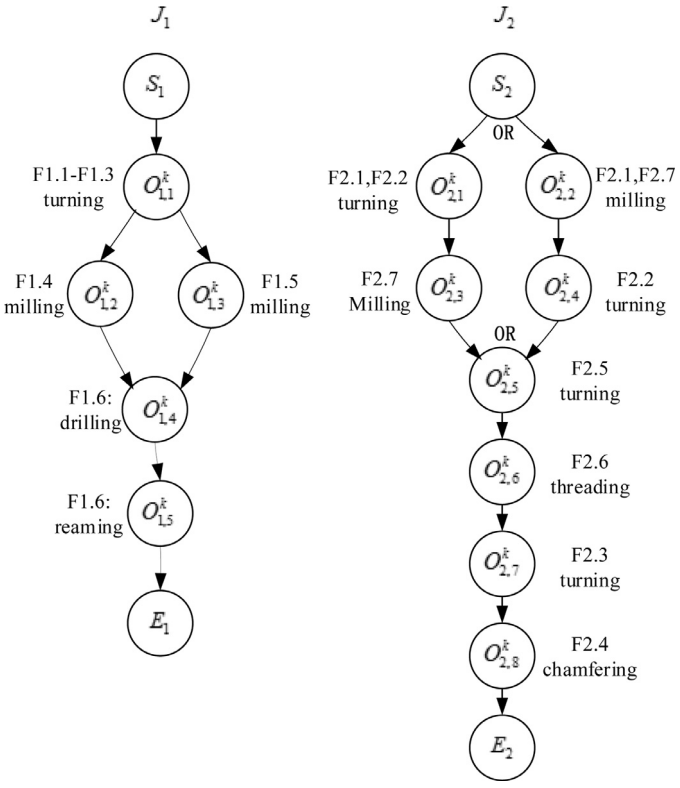


Fig. 2. AND/OR graph of the sample IPPS instance with 2 parts.

1,1	2,3	1,1	1,1	2,3	2,3	1,1	2,3	2,3	1,1	2,3

where the first number of a gene is a job index, and the second is the index of a chosen process plan.

With such encoding approach, the process plan information is not visible in the chromosome representation and extra means is required to store and interpret the detailed meanings of codes. In general, the information held by each gene may be used by genetic operators during the search procedure. With the numerical-number representation, however, this kind of information is hidden by codes or not directly accessible. It may then be necessary to prepare all feasible solutions before exploring the whole search space. This is a tedious task and the heuristic ability will be affected significantly. Furthermore, in cases where an operation can be processed by alternative machines, even if it is only required to choose another machine for the operation, the mutation operator will affect a series of genes on the chromosome, and hence the process plan for the job hosting this operation will be entirely replaced. Apparently this encoding/decoding scheme is rigid and not efficient for IPPS problems. It also infers that the GA approach with numerical-number representation is not potentially suitable for dynamic situations.

In this paper, genetic representation in OCGA is based on real objects. That is, a chromosome is expressed in terms of the operation sequence directly. A chromosome representing the sample IPPS instance for the two parts is presented in Fig. 4.

With the AND/OR graph model, the GA is required to handle a population of existing sequences to obtain improved sequences in terms of incorporated criteria. Obviously the object-coding chromosome

Table 1
Related processing information of the sample IPPS instance.

Job	J_1					J_2							
$O^k_{j,i}$	$O^k_{1,1}$	$O^k_{1,2}$	$O^k_{1,3}$	$O^k_{1,4}$	$O^k_{1,5}$	$O^k_{2,1}$	$O^k_{2,2}$	$O^k_{2,3}$	$O^k_{2,4}$	$O^k_{2,5}$	$O^k_{2,6}$	$O^k_{2,7}$	$O^k_{2,8}$
F	F1.1–F1.3	F1.4	F1.5	F1.6	F1.6	F2.1, F2.2	F2.1, F2.7	F2.7	F2.2	F2.5	F2.6	F2.3	F2.4
k	1,2	3	3	3,4	3,4	1,2	3	3	1,2	1,2	1,2	1,2	1,2
$\tau^k_{j,i}$	20,13	15	15	10,8	18,16	23,14	18	8	13,8	18,14	30,28	10,8	8,6

F: feature.
k = 1: lathe; k = 2: NC lathe; k = 3: milling machine; k = 4: drilling machine.
$\tau^k_{j,i}$: the processing time of $O^k_{j,i}$.

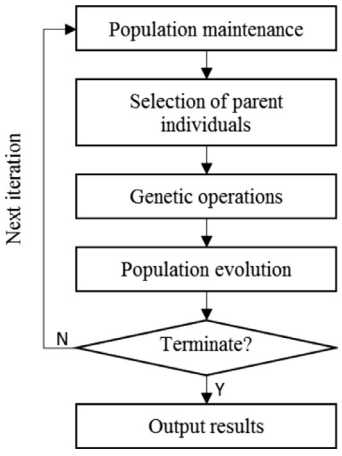


Fig. 3. Overall procedure of the OCGA.

$O^1_{1,1}$	$O^2_{2,1}$	$O^3_{1,2}$	$O^3_{1,3}$	$O^3_{2,3}$	$O^1_{2,5}$	$O^2_{2,6}$	$O^4_{1,4}$	$O^4_{1,5}$	$O^1_{2,7}$	$O^1_{2,8}$

Fig. 4. A sample chromosome for the sample IPPS instance.

exposes much more information which is directly accessible during the heuristic process. The relative positions of any two operations on a chromosome reflect the precedence relationship between them. For instance, if $O^{k1}_{j1.i1}$ is a predecessor of $O^{k2}_{j2.i2}$, $O^{k1}_{j1.i1}$ must precede $O^{k2}_{j2.i2}$ on any chromosomes.

4.3. Population maintenance

This function is to stabilize the size of the population in order to avoid population degeneration (population size keeps decreasing) or overgrowth (population size keeps increasing). Assume that the population size has to be stabilized to a constant N ($N > 1$). At the beginning of the algorithm, an initial population has to be prepared (N individuals have to be generated). Besides, at the beginning of subsequent iterations, if the population size is larger than N, a number of individuals have to be eliminated whilst if the number of individuals in the population is less than N, new individuals have to be generated to fill the vacancies.

With regard to the generation of new individuals, a random search based approach is adopted in this paper. Assume that an invisible crawler is put onto the AND/OR graph for an IPPS instance. The crawler starts from a start node of any job, and then crawls over the whole graph under the restriction of precedence relationships, until it collects a series of operations to finish all jobs. The trace of the crawler presents a feasible solution.

4.4. Individual evaluation and selection of parents

Individual evaluation is closely related to the optimization criteria. In this paper, the objective of the OCGA is to minimize the makespan which is the primary criterion. For evaluating individuals, in addition, machine utilization is also adopted as a secondary criterion. The affinity value for an individual S_i is calculated by

$$A(S_i) = \alpha \times C_i + \beta \times MU_i \tag{1}$$

where $A(S_i)$ is the affinity value of S_i; C_i and MU_i are the makespan and machine utilization of S_i respectively. α and β are two positive constants and α is usually much bigger than β. Intuitively, an individual solution with a smaller makespan is a better solution. Thus, a larger affinity value implies a poor quality in S_i. In case if the makespans of two individuals are equal, the one possessing smaller machine utilization is better.

A given number of operators carry out the genetic operations simultaneously, each of which selects a pair of parent individuals according to the affinity values. The OCGA adopts an inferior selection method. Worse individuals are given more chances to yield offsprings. A linear rank approach is implemented for the selection. Firstly, all individuals are ordered by their affinity values in ascending order, so that each individual owns a fixed position in the sequence (the position index) and better individuals are positioned at the front. The position index of S_i is denoted as $pos(S_i)$ and the probability of taking S_i is calculated by

$$p(S_i) = \frac{2 \times pos(S_i)}{N \times (N+1)} \tag{2}$$

where N is the population size. It is clear that a worse solution possesses a higher probability to be selected since the corresponding position index $pos(S_i)$ is relatively larger. The inferior selection method intends to mine good genes in poor individuals before they die out. Better individuals are remained in the population. When the overall quality of the population is improved, old individuals become relatively worse and their selection probabilities grow accordingly. The inferior selection method can obviously preserve the population diversity and can somehow maintain multiple individuals.

4.5. Genetic operations

In general, genetic operations in GA include crossover and mutation. With regard to the IPPS problem, the operation flexibility, sequencing flexibility and processing flexibility bring in three different dimensions. Hence the genetic operations must correspond to all three flexibilities to fulfill search toward different dimensions. In OCGA, the following GA operators are implemented for neighborhood construction: crossover (to recombine process plans of different jobs); mutation of shifting genes' loci (to reorder partial operations in a parent sequence); and mutation of shifting processing machines (to alter processing machine for some operations).

4.5.1. Crossover

Crossover is one of the most important neighborhood construction methods for GAs. With regard to the IPPS consideration, due to the sequencing and processing flexibilities, a job can have alternative process plans, that is, different sets of operations o_i or the same set of operations are sequenced in different orders. The crossover for the

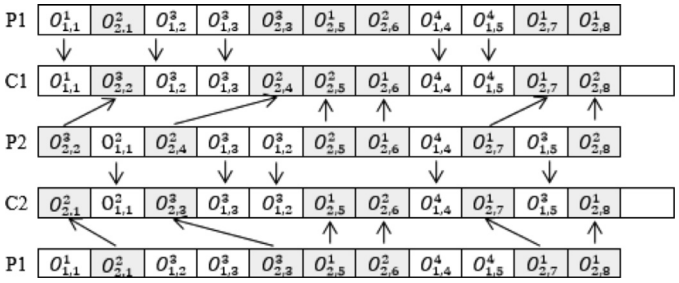


Fig. 5. Illustration of the crossover operator.

OCGA is similar to the Precedence Preserving Order-based Crossover (POX) (Lee, Yamakawa, & Lee, 1998), with the aim to exchange operations of a same job between two individuals, as well as preserve precedence relationships. As there may exist alternative process plans in a job, the crossover operator in OCGA also provides a chance for a job to choose an alternative process plan.

Fig. 5 depicts the approach of the crossover operator. P1 and P2 are two parent chromosomes, and C1 and C2 are initially two empty arrays. In essence, crossover for the OCGA is to copy operations of different jobs from two parent chromosomes in a specific way to fulfill the recombination. The approach is summarized as follows:

- firstly select a job randomly as a feature job (job 1 in the case);
- copy operations of job 1 from P_1 to corresponding positions on C_1, and copy operations of job 1 from P_2 to corresponding positions on C_2;
- fill vacancies in C_1 with operations other than those of job 1 from P_2, and fill vacancies in C_2 with operations other than those of job 1 on P_1 orderly from left to right;
- eliminate vacancies on C_1 and C_2, and two child individuals can then be obtained.

4.5.2. Shifting genes' loci

The mutation of shifting genes' loci is to rearrange partial operations' order. However, relative positions of two operations of the same job cannot be shuffled since it may violate the precedence relationship. Hence an orderly moving method is proposed for the OCGA. The shifting genes' loci perform on one child individual obtained from the crossover, say, C_1 in Fig. 5.

- Initially, make a copy of C_1, named C_3;
- randomly select two loci on C_3, say L_1 and L_2. If operations on L_1 and L_2 belong to the same job, eliminate C_3 and terminate.
- for operations on L_1 and L_2 not belonging to the same job, handle C_3 as follows: assume that the two operations on L_1 and L_2 belong to J_1 and J_2 respectively
  - eliminate the operation on L_1, and then move those operations of J_2 backwards to fill the vacancies one by one until the operation on L_2 is moved. A vacancy will then appear on L_2;
  - move operations of J_1 between L_1 and L_2 forward to fill the vacancies one by one; the last vacancy is then filled by the original operation on L_1. Those operations between L_1 and L_2 but not belonging to J_1 or J_2 are remained.

Fig. 6 illustrates the shifting genes' loci, where the directed arcs show the movements of operations, numbers on which indicate the movement priorities. Finally, a new mutated individual C_3 is obtained.

4.5.3. Shifting processing machine

The mutation of shifting processing machines is exerted on the other child individual which is not mutated with shifting genes' loci, that is, C_2 in Fig. 5. It denotes that each gene on the chromosome has

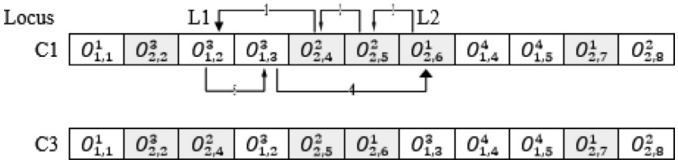


Fig. 6. Illustration of shifting genes' loci mutation.

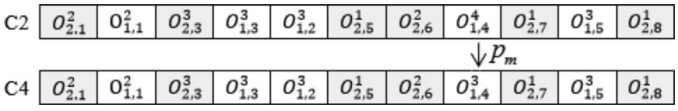


Fig. 7. Shifting processing machine.

a very small probability p_m (usually $p_m < 0.1$) to change its processing machine. An overview of this mutation is given in Fig. 7, where the process machine for $O^4_{1,4}$ shifts from machine No. 4 to machine No. 3.

Intuitively, crossover and shifting genes' loci will not change the relative positions of any two operations of the same job. Hence all children are feasible since no genetic operations that violate precedence relationships are carried out. However, strict proof for this issue needs to be investigated intensively in future research.

4.6. Population evolution

A simplified version of crowding is implemented for the population replacement. Only a fraction of the population is selected to reproduce, and partial old individuals are replaced at each generation. De Jong's crowding and its variations usually need to measure the similarity of children to the old individuals, and most similar old individuals are most likely to be replaced. But in the context of discrete problems with high flexibility and real-object representation, the similarity measurement may be quite difficult or it may introduce heavy computing pressure. In the IPPS problem domain, alternative process plans for each job and non-identical machine set for each operation substantially increase the difficulty to use Hamming-distance, Euclidean distance or Common-edge distance for the similarity measurement.

The simplified crowding approach for the OCGA is:

- r pairs of parents are selected in the parent selection stage;
- around $4 \times r$ children are reproduced in the stage of genetic operations (as discussed in Section 4.5.2, shifting gene's loci may not produce new individuals, so the number of children cannot be exactly $4 \times r$);
- all $2 \times r$ selected parents are replaced by the offspring.

Obviously, the new population size will exceed the original size N. Therefore, after population replacement, the worst individuals are eliminated immediately until the new population size is dropped to $N \times (1 - R) : 0 \le R < 1$ where R is the *degeneration ratio*. This is another mechanism to further enhance the ability to maintain population diversity, that is, the degeneration ratio R is introduced to reduce the population size below the original level. The shortage of individuals will be replenished at the next population maintenance stage.

The procedure of processing the sample IPPS instance by OCGA is summarized as follows. Assume that the set of individuals in the population at iteration i is denoted by G_i and only one pair of parent individuals P_1 and P_2 are selected for propagation at each iteration. Three child individuals C_1, C_2, C_4 and possibly the fourth child C_3 are generated. The overall procedure of the OCGA can be presented as follows:

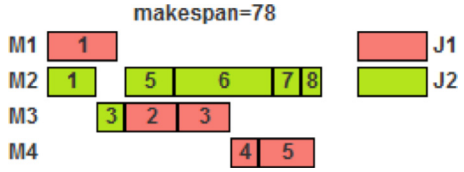


Fig. 8. Gantt chart for the sample IPPS instance.

Iteration i starts:
While(the population size<N){
  Generate a new solution to cover the individual shortage.
}
Select a given number of parent individuals for propagation. (P_1 and P_2 are selected.)
Perform genetic operations and generate offspring. (C_1, C_2, C_3^*, C_4 are generated.)
Replace parent individuals with their offspring. ($G_i \leftarrow (G_i \backslash \{P_1, P_2\}) \cup \{C_1, C_2, C_3, C_4\}$)
While($currentpopulationsize > N \times (1 - R) : 0 \leq R < 1$){
  Eliminate the worst individual. ($G_i \leftarrow G_i \backslash \{the\text{worstindividual}\}$)
}
If(Terminate criteria are not fulfilled){
  $i \leftarrow i + 1$. Go to next iteration.
}
  Else {Terminate!}

*C_3 may not exist.

The proposed approach will be repeated for a number of iterations until the terminate criteria are all fulfilled. A run of the OCGA on the sample IPPS instance (given in Figs. 1 and 2) generates the schedule in Fig. 8, from which it is easy to verify that all assumptions and constraints given in Section 3 are satisfied.

5. Experiments and discussions

For benchmarking, the IPPS problem test bed proposed by Kim et al. (2003) is adopted for the experiments. The test bed is composed of 18 separate jobs including 300 operations. Different combinations of the 18 jobs form 24 problem sets of different problem complexity and strength of flexibilities.

For system implementation, the software programs of OCGA were developed in JAVA with MySQL database. Experiments in this paper were conducted on a personal computer equipped with Intel core i-5 CPU (M520 2.4 gigahertz) and 6 gigabytes RAM.

The objective of OCGA was to minimize the makespan. The coefficients were set as follows: population size $N = 100$; number of genetic operators working synchronously $r = 7$; probability for crossover $p_c = 1$; probability for shifting genes' loci $p_e = 1$; probability for shifting process machines $p_m = 0.06$; degeneration ratio $R = 0.09$. The termination criterion was that the algorithm would terminate if a better solution could not be obtained in successive 2500 iterations. Every problem of the test bed was repeated for 11 times and best solutions with smallest makespans at each time were recorded. The main reason for conducting 11 runs of simulations was to compare the performance of OCGA with the other algorithms which also recorded results of similar simulation runs.

5.1. Experiment 1: The global performance of OCGA

A series of regular experiments with fixed coefficients setting as listed above were conducted on all 24 problems, and best and mean results were recorded and compared with other three algorithms: the cooperative co-evolutionary genetic algorithm (CCGA) (Potter & De Jong, 1994), symbiotic evolutionary algorithm (SEA) (Kim et al., 2003), and an improved genetic algorithm (IGA) (Qiao & Lv, 2012). This experiment is to figure out the global performance of the proposed approach.

Table 2 lists the best makespans in 11 runs of repeated experiments. It shows the results of 14 problems have reached their lower bounds, which implies that the OCGA has found optimal solutions for these 14 problems. For the other 10 problems, the best results were just a bit higher than their lower bounds, which would be even closer to optimal solutions. In conclusion, Table 2 reveals that the OCGA possesses wonderful search ability.

Table 3 accounts for mean makespans of SEA, CCGA, IGA and OCGA. The column "improved rate" depicts the relative improved ratio of the OCGA result compared to the minimum result yielded by the other three algorithms. A positive improved rate indicates that the OCGA presents the best result for the corresponding problem. It shows that the OCGA achieved significant improvements in 12 out of the 24 problems whose improved ratios were larger than 3.5 percent. And for another 11 problems the OCGA just made a slight improvement or did a little worse. It is because that most results of these 11 problems obtained by all four algorithms almost reached the

Table 2
Comparison of best makespans in benchmark test.

Problem	Jobs	CCGA	SEA	IGA	OCGA	Lower bound
1	1, 2, 3, 10, 11, 12	458	428	427	427	427
2	4, 5, 6, 13, 14, 15	363	343	343	343	343
3	7, 8, 9, 16, 17, 18	366	347	344	344	344
4	1, 4, 7, 10, 13, 16	312	306	306	306	306
5	2, 5, 8, 11, 14, 17	327	319	304	318	304
6	3, 6, 9, 12, 15, 18	476	438	427	427	427
7	1, 4, 8, 12, 15, 17	378	372	372	372	372
8	2, 6, 7, 10, 14, 18	363	343	342	343	342
9	3, 5, 9, 11, 13, 16	464	428	427	427	427
10	1, 2, 3, 5, 6, 10, 11, 12, 15	476	443	427	427	427
11	4, 7, 8, 9, 13, 14, 16, 17, 18	410	369	368	348	344
12	1, 4, 5, 7, 8, 10, 13, 14, 16	360	328	312	318	306
13	2, 3, 6, 9, 11, 12, 15, 17, 18	498	452	429	427	427
14	1, 2, 4, 7, 8, 12, 15, 17, 18	420	381	386	372	372
15	3, 5, 6, 9, 10, 11, 13, 14, 16	482	434	427	427	427
16	1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15	512	454	433	427	427
17	4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18	466	431	415	370	344
18	1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17	396	379	364	351	306
19	2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18	535	490	450	427	427
20	1, 2, 4, 6, 7, 8, 10, 12, 14, 15, 17, 18	450	447	429	384	372
21	2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 18	501	477	433	427	427
22	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18	567	534	491	446	427
23	1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18	531	498	465	394	372
24	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18	611	587	532	458	427

Table 3
Comparison of mean makespans in benchmark test.

Problem	SEA	CCGA	IGA	OCGA	Best	Improved rate (percent)	CPU time of OCGA (seconds)
1	437.6	470.4	427	427	–	–	4.5
2	349.7	369.2	344.5	343.5	OCGA	0.3	6.5
3	355.2	382	351	346.4	OCGA	1.3	7.9
4	306.2	321.5	307.4	310.1	SEA	−1.3	4.4
5	323.7	337.8	309.8	323	IGA	−4.3	6
6	443.8	485.6	427	427	–	–	7.4
7	372.4	385.6	372.7	373.3	SEA	−0.2	4.1
8	348.3	373.8	357	343.5	OCGA	1.4	6.2
9	434.9	474.5	427	427	–	–	5.7
10	456.5	502.6	431.6	427.1	OCGA	1.0	10.9
11	378.9	423.8	379.7	350.6	OCGA	7.5	12.2
12	332.8	379.5	323.7	324.7	IGA	−0.3	8.7
13	469	511.1	442.8	427.2	OCGA	3.5	15.3
14	402.4	433.4	415.3	377.4	OCGA	6.2	11.2
15	445.2	493.9	427.4	427	OCGA	0.1	10.7
16	478.8	549.7	449.4	428.1	OCGA	4.7	27.8
17	448.9	496.9	426	383.1	OCGA	10.1	27.5
18	389.6	419.8	373.6	354.5	OCGA	5.1	26.4
19	508.1	557	471.3	433.7	OCGA	8.0	30.5
20	453.8	482.7	446.6	390.5	OCGA	12.6	25.9
21	483.2	534	447.8	427	OCGA	4.6	26.5
22	548.3	587.5	508.1	452.9	OCGA	10.9	33.5
23	507.5	557.9	477.8	410	OCGA	14.2	31.5
24	602.2	633.3	548.5	471.2	OCGA	14.1	48.5

optimal, and insignificant differences in these results were basically due to the randomness of heuristics. Furthermore, the OCGA performed better in more complex problems: problems No. 16–24. It also shows that the computing time for each problem increased steadily as the complexity of problems increases, but it was always controlled at an acceptable level.

An extra experiment was performed on the most complex problem No. 24 under an extreme situation where the algorithm was allowed to run for unlimited number of iterations so that the OCGA could sufficiently explore the solution space. It can be known from Table 3 that in regular experiments, the OCGA can usually find good solutions for the problem No. 24 in around 50 seconds. In the extreme test, the OCGA ran for around 400 seconds (around 20,000 iterations) and yielded the solution with makespan equaled to 457. The schedule of this solution can be found in Fig. 9.

The outstanding search ability and high performance of the OCGA are mainly due to the following factors. First of all, the real-object representation enables the algorithm to easily perform neighborhood search, and the whole solution space can be covered. Then, three genetic operations can explore the search space in three different directions. Crossover helps the algorithm to move to unexplored solution, whilst two kinds of mutation enable the algorithm to extensively search an area. Finally, the population can be gradually evolved and a high population diversity can always be maintained under the two evolutionary mechanisms: the inferior selection and population degeneration.

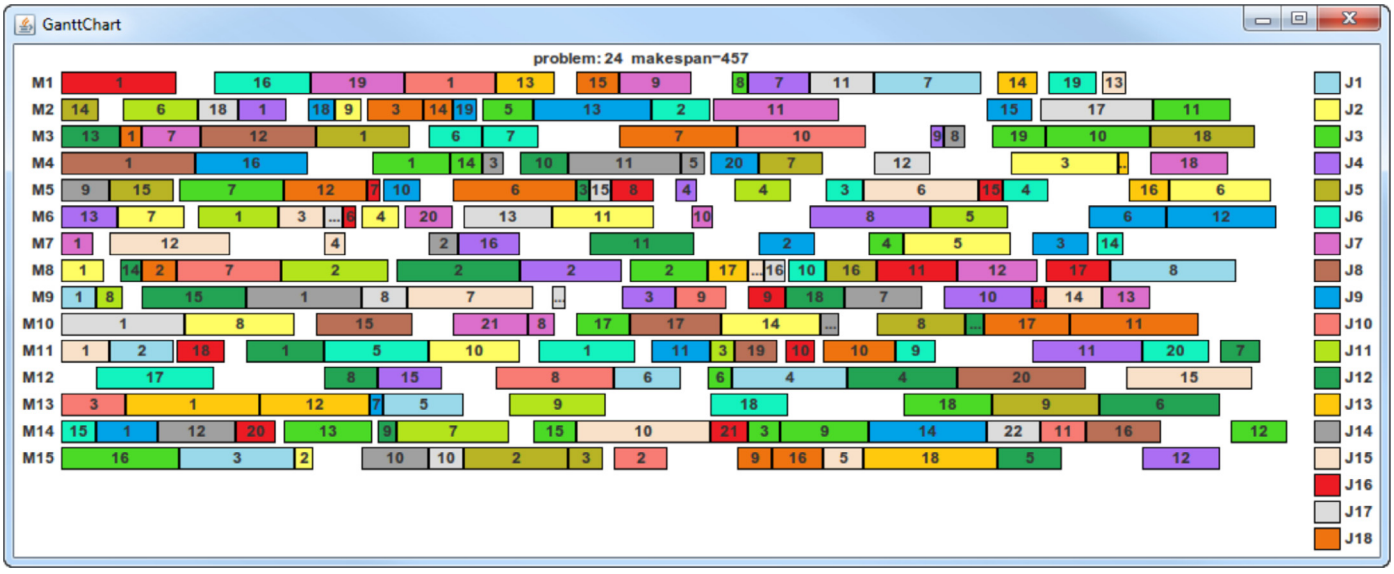


Fig. 9. Schedule for problem No. 24 generated in an extreme test.

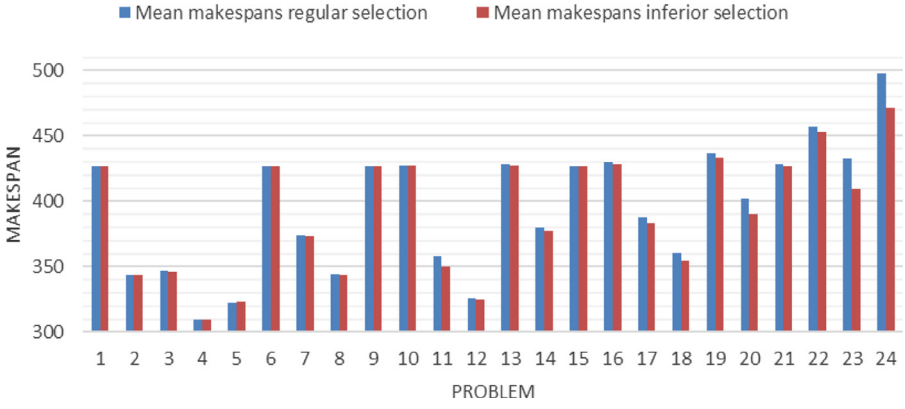


Fig. 10. Comparison of mean makespans for 24 problems using different selection methods.

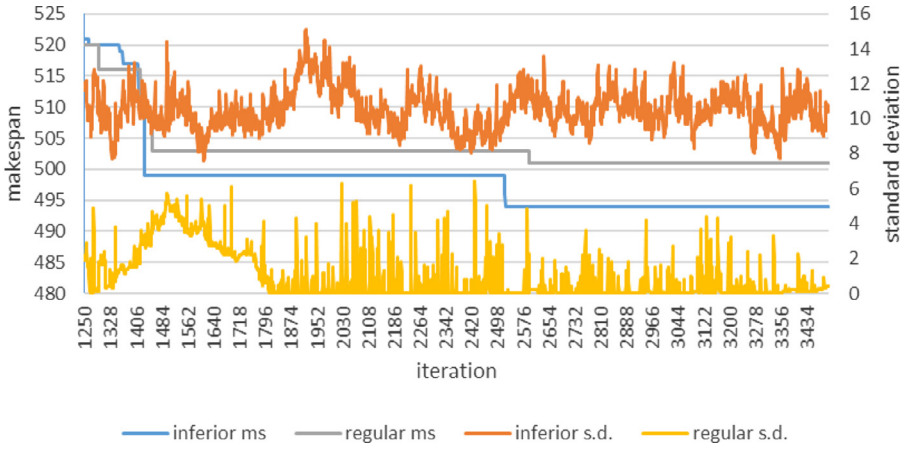
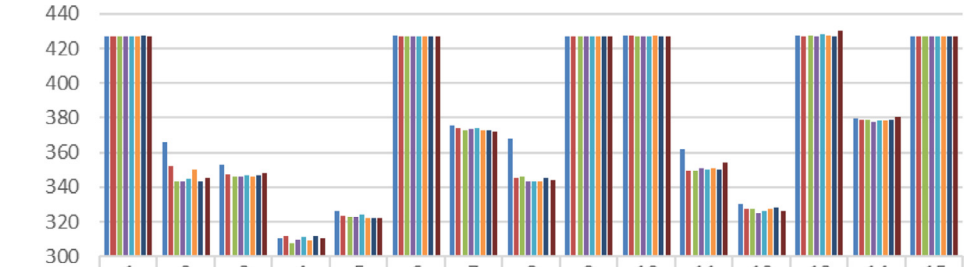
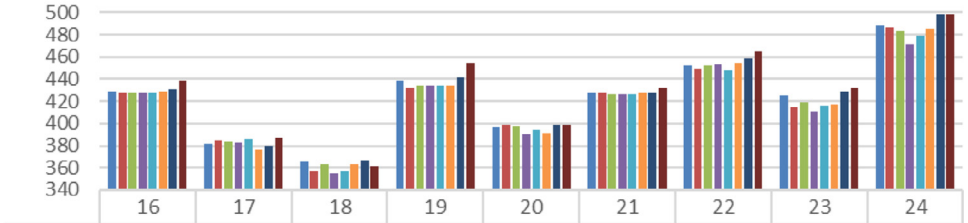


Fig. 11. Segmentary data of two experiments using different selection methods.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R=0	427.2	365.9	353	310.4	325.9	427.4	375.4	368.3	427	427.4	361.9	330.2	427.7	379.5	427
R=0.04	427	352.3	347.5	312	323.2	427.1	374.3	345.1	427	427.4	349.5	327.3	427.1	378.7	427
R=0.07	427	343.5	346.1	307.9	323.1	427.1	372.7	346	427	427.1	349.4	327.3	427.6	379.1	427
R=0.09	427	343.5	346.4	310.1	323	427	373.3	343.5	427	427.1	350.6	324.7	427.2	377.4	427
R=0.1	427	344.5	347	310.9	324.1	427	374.2	343.6	427	427	350.2	326.1	428.2	378.4	427
R=0.12	427	350.2	346.4	309.2	322.2	427	372.7	343.3	427	427.3	350.7	327.5	427.3	378.4	427
R=0.14	427.3	343.3	346.9	312	321.9	427	372.9	345.5	427	427	350.3	328.1	427.1	379.1	427
R=0.17	427	345.2	348	310.4	322.1	427	372.4	344.2	427	427.2	354.5	326.4	430.4	380.4	427

(a)



	16	17	18	19	20	21	22	23	24
R=0	429	381.1	365.9	437.9	396.2	427.1	452.1	425.5	488.5
R=0.04	427.7	385.1	357.2	431.9	398.5	427.1	449.2	414.7	486.6
R=0.07	427.5	383.5	363.5	433.7	397.8	427	452.5	418.9	482.7
R=0.09	428.1	383.1	354.5	433.7	390.5	427	452.9	410	471.2
R=0.1	428.1	385.4	357.5	434.5	394.1	427	448.3	416.3	479.2
R=0.12	428.5	376.3	363	434.2	391.5	427.1	453.8	416.5	485.5
R=0.14	431.2	379.4	367	441.3	398.5	427.5	458.8	428.6	497.8
R=0.17	438.8	387.1	361.4	454.2	398.8	432.3	464.7	431.5	498.3

(b)

Fig. 12. (a) Mean makespans for simple problems with different degeneration ratios. (b) Mean makespans for complex problems with different degeneration ratios.

5.2. Experiment 2: The impact of the inferior selection method

Experiments under the ordinary selection method with the same context setting as experiment 1 were repeated. But in the phase of selecting parent individuals, all individuals were sequenced in descending order so that good individuals possessing higher probabilities to reproduce. Fig. 10 depicts the comparison of mean makespans under the inferior and ordinary selection methods.

With regard to problems No. 1–10, the OCGA could easily achieve good solutions close to the lower bounds, hence the difference in selection methods is not significant. However, for more complex problems No. 11–24, the improvements of mean makespans using the inferior selection method are obvious. Particularly, iterative data from iteration 1250–3500 (when the search procedure becomes relatively stable) of two experiments conducted on problem No. 24 using the ordinary and inferior selection methods are compared to demonstrate the transition of population diversity during a relatively stable evolutionary process.

The best-so-far makespans and makespans' standard deviation of the population at each iteration are recorded and mapped on a sequence diagram as shown in Fig. 11. The best-so-far makespan and standard deviation are abbreviated to ms and s.d. respectively in this graph. It shows that the standard deviation of the population using inferior selection fluctuates around 11 and shows no downward trend, whilst standard deviation using ordinary selection usually falls to 0 and stays at a very low level. This phenomenon reflects that the inferior selection method can maintain much higher population diversity during the heuristic process.

5.3. Experiment 3: The analysis of degeneration ratio

The experiment similar to the one in experiment 1 was repeated under the setting of different degeneration ratios. Here simple problems No. 1–15 and complex problems No. 16–24 are considered separately, and the results are presented in Fig. 12(a) and (b), respectively.

In Fig. 12(a) and (b), $R = 0$ means there is no population degeneration. As R grows, more inferior individuals in the population are eliminated and more new individuals are introduced at each iteration. Fig. 12(a) shows that mean makespans at $R = 0$ for problem No. 2, 3, 8 and 11 are significantly much higher. It reveals that population degeneration does have effects to prevent quick premature convergence. For other simple problems, there are no significant differences. The reason may be that optimal solutions to these problems can be easily reached, then the impact of population degeneration is not that obvious.

In Fig. 12(b), as the degeneration ratio increases, mean makespans for major problems go downward and then upward. The valley point usually appears at around $R = 0.09$. The reason for the downward trend is clear that newly introduced individuals at each iteration increase the probability to find better solutions. But the following upward trend is mainly because that a high degeneration ratio may disturb the evolutionary coherence, that is, offspring may be wiped out of the population quickly and too many new individuals successively pour into the population, which will certainly impede the population's evolution and convergence.

6. Conclusion

A variant of GA, named object-coding GA (OCGA), has been implemented to solve the IPPS problem. The OCGA uses operation sequences directly as chromosomes. Genetic operators including crossover and mutation, therefore, have to be customized to cope with the new genetic representation. In addition, the OCGA adopts an inferior selection method to select parents for propagation. A simplified version of crowding and a degeneration ratio factor are introduced to accommodate the population evolution. The combined effect of the unique approaches, including object-coding, inferior selection, modified replacement and degeneration ratio, leads to the preservation of high population diversity and yield of outstanding results. Experiments on benchmark problems have confirmed the performance of the systematic approach.

The focus of the current article is on the design and implementation of OCGA. From simulation experiments, it is obvious that OCGA is able to generate process plans and schedules for complex jobshop problems. The results, in terms of makespans, are very well converged and outperform the other algorithms in many of the test problems. Regarding the validity of the solutions, the process priority and precedence relationship of each job is governed by the AND/OR graph representation, and these process constraints are incorporated in the genetic operators in OCGA. In our IPPS research, we have attempted to deal with the validity issue in the AND/OR graph model, and the correctness of the solution algorithms are evaluated with the graph modelling and corresponding mathematical programming methods. This is not the focus of the current paper and will be reported in separate publication.

As future work, OCGA can be extended to cope with the process planning and scheduling requirements of other manufacturing scenarios, for instance, assembly operations, dynamic process planning and rescheduling.

Furthermore, there is great potential to extend and apply the object-coding representation to other discrete problems where decision variables are related to discrete elements. For example, "cities" in the traveling Salesman Problem (TSP) where can be conveniently be regarded as "objects" with the approach in OCGA.

Acknowledgment

The work described in this paper is fully supported by a grant from the Research Grants Council, University Grants Committee, Hong Kong (Project Code HKU 718809E).

Reference

Brizuela, C. A., & Sannomiya, N. (1999). A diversity study in Genetic Algorithms for Job Shop Scheduling Problems. *Gecco-99: Proceedings of the genetic and evolutionary computation conference* (pp. 75–82).

Cai, L. W., Wu, Q. H., & Yong, Z. Z. (2000). A genetic algorithm with local search for solving job shop problems. *Real-World Applications of Evolutionary Computing, Proceedings, 1803*, 107–116.

Chen, Q., & Khoshnevis, B. (1993). Scheduling with flexible process plans. *Production Planning & Control, 4*, 333–343.

Chryssolouris, G., Chan, S., & Cobb, W. (1984). Decision making on the factory floor: An integrated approach to process planning and scheduling. *Robotics and Computer-Integrated Manufacturing, 1*, 315–319.

Chryssolouris, G., Chan, S., & Suh, N. P. (1985). An integrated approach to process planning and scheduling. *CIRP Annals – Manufacturing Technology, 34*, 413–417.

Davis, L. (1985). Job shop scheduling with genetic algorithms. In *Proceedings of an international conference on genetic algorithms and their applications: Vol. 140*.

De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems:* Ph.D. Dis. Univ. of Michigan.

Falkenauer, E., & Bouffouix, S. (1991). A genetic algorithm for job shop. *1991 IEEE international conference on robotics and automation: Vols. 1–3* (pp. 824–829).

Fang, H. L., Ross, P., & Corne, D. (1993). A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. *Proceedings of the fifth international conference on genetic algorithms* (pp. 375–382).

Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization: Vol. 7*. John Wiley & Sons.

Gen, M., Tsujimura, Y., & Kubota, E. (1994). Solving job-shop scheduling problems by genetic algorithm. *1994 IEEE international conference on systems, man, and cybernetics – Humans, information and technology: Vols. 1–3* (pp. 1577–1582).

Goldberg, D. E. (1990). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems, 5*, 139–167.

Grüninger, T., & Wallace, D. (1996). Multimodal optimization using genetic algorithms. *MIT CADlab-Technical Report: 96.02.*

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

Harik, G. (1995). Finding multimodal solutions using restricted tournament selection. In L. J. Eshelman (Ed.), *Proc. 6th Int. Conf. on Genetic Algorithms (ICGA-95)* (pp. 24–31). San Mateo, CA: Pittsburgh, Morgan Kaufmann.

Hussain, M. F., & Joshi, S. B. (1998). A Genetic Algorithm for job shop scheduling problems with alternate routing. *1998 IEEE international conference on systems, man, and cybernetics: Vols. 1–5* (pp. 2225–2230).
Kim, Y. K., Park, K., & Ko, J. (2003). A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers & Operations Research*, *30*, 1151–1171.
Koza, J. R. (1994). Genetic programming as a means for programming computers by natural-selection. *Statistics and Computing*, *4*, 87–112.
Kumar, M., & Rajotia, S. (2003). Integration of scheduling with computer aided process planning. *Journal of Materials Processing Technology*, *138*, 297–300.
Lee, H., & Kim, S. S. (2001). Integration of process planning and scheduling using simulation based genetic algorithms. *International Journal of Advanced Manufacturing Technology*, *18*, 586–590.
Lee, K. -M., Yamakawa, T., & Lee, K. -M. (1998). A genetic algorithm for general machine scheduling problems. In *Knowledge-based intelligent electronic systems, 1998. Proceedings KES'98. 1998 second international conference on: Vol. 2* (pp. 60–66) IEEE.
Leung, C. W., Wong, T. N., Mak, K. L., & Fung, R. Y. K. (2010). Integrated process planning and scheduling by an agent-based ant colony optimization. *Computers & Industrial Engineering*, *59*, 166–180.
Li, W. D., & McMahon, C. A. (2007). A simulated annealing-based optimization approach for integrated process planning and scheduling. *International Journal of Computer Integrated Manufacturing*, *20*, 80–95.
Li, X., Gao, L., Shao, X., Zhang, C., & Wang, C. (2010). Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. *Computers & Operations Research*, *37*, 656–667.
Li, X., Shao, X., Gao, L., & Qian, W. (2010). An effective hybrid algorithm for integrated process planning and scheduling. *International Journal of Production Economics*, *126*, 289–298.
Li, X. Y., Gao, L., Shao, X. Y., Zhang, C. Y., & Wang, C. Y. (2010). Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. *Computers & Operations Research*, *37*, 656–667.
Lian, K., Zhang, C., Gao, L., & Li, X. (2011). Integrated process planning and scheduling using an imperialist competitive algorithm. *International Journal of Production Research*, *50*, 4326–4343.
Liu, L., & Xi, Y. G. (2006). A hybrid genetic algorithm for job shop scheduling problem to minimize makespan. *WCICA 2006: Sixth world congress on intelligent control and automation, Conference proceedings: Vols. 1–12* (pp. 3709–3713).
Lucasius, C. B., & Kateman, G. (1989). Application of genetic algorithms in chemometrics. In *Proceedings of the 3rd international conference on genetic algorithms* (pp. 170–176) Morgan Kaufmann Publishers Inc.
Mahfoud, S. W. (1995). *Niching methods for genetic algorithms:* Ph.D. Dis. Univ. of Illinois at Urbana-Champaign.
Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs.* Springer.
Morad, N., & Zalzala, A. (1999). Genetic algorithms in integrated process planning and scheduling. *Journal of Intelligent Manufacturing*, *10*, 169–179.
Ombuki, B. M., & Ventresca, M. (2004). Local search genetic algorithms for the job shop scheduling problem. *Applied Intelligence*, *21*, 99–109.
Pinedo, M. (2012). *Scheduling: Theory, algorithms, and systems.* Springer.
Potter, M. A., & De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. *Parallel problem solving from nature – Ppsn Iii – International conference on evolutionary computation, Proceedings: 866* (pp. 249–257).
Qiao, L. H., & Lv, S. P. (2012). An improved genetic algorithm for integrated process planning and scheduling. *International Journal of Advanced Manufacturing Technology*, *58*, 727–740.
Resende, M. G. C., Goncalves, J. F., & Mendes, J. J. D. M. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, *167*, 77–95.
Shaw, M. J. P., & Whinston, A. B. (1985). Automatic planning and flexible scheduling: A knowledge-based approach. In *Robotics and automation. Proceedings. 1985 IEEE international conference on: Vol. 2* (pp. 890–894).
Tan, W. (1998). *Integration of process planning and scheduling functions: A mathematical programming approach.* University of Southern California.
Usher, J. M., & Fernandes, K. J. (1996). Dynamic process planning – The static phase. *Journal of Materials Processing Technology*, *61*, 53–58.
Vignaux, G. A., & Michalewicz, Z. (1991). A genetic algorithm for the linear transportation problem. *IEEE Transactions on Systems, Man and Cybernetics*, *21*, 445–452.
Wang, L., & Zheng, D. Z. (2002). A modified genetic algorithm for job shop scheduling. *International Journal of Advanced Manufacturing Technology*, *20*, 72–76.
Wong, T. N., Chan, L. C. F., & Lau, H. C. W. (2003). Machining process sequencing with fuzzy expert system and genetic algorithms. *Engineering with Computers*, *19*, 191–202.
Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006). An agent-based negotiation approach to integrate process planning and scheduling. *International Journal of Production Research*, *44*, 1331–1351.
Ye, C. M., & Zhang, F. D. (2000). Working out system of production planning and job-shop scheduling based on hybrid genetic algorithms. *Proceedings of the seventh international conference on industrial engineering and engineering management* (pp. 226–229).
Zhang, F., Zhang, Y. F., & Nee, A. Y. C. (1997). Using genetic algorithms in process planning for job shop machining. *IEEE Transactions on Evolutionary Computation*, *1*, 278–289.
Zhang, L., Wong, T., & Fung, R. (2013). A multi-agent system to support heuristic-based dynamic manufacturing rescheduling. *Intelligent Decision Technologies*, *7*, 197–211.
Zhang, W. Q., & Gen, M. S. (2010). Process planning and scheduling in distributed manufacturing system using multiobjective genetic algorithm. *IEEJ Transactions on Electrical and Electronic Engineering*, *5*, 62–72.