APEM journal
Advances in Production Engineering & Management
Volume 13 | Number 3 | September 2018 | pp 279–296
https://doi.org/10.14743/apem2018.3.290
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Dynamic integration of process planning and scheduling using a discrete particle swarm optimization algorithm

Yu, M.R.[a,*], **Yang, B.**[a], **Chen, Y.**[b]

[a]School of Mechatronic Engineering, North University of China, Taiyuan, P.R. China
[b]North Automatic Control Technology Institute, Taiyuan, P.R. China

ABSTRACT

Because of the inherent relationship between process planning and scheduling, integration of process planning and scheduling (IPPS) provides a new path for further improvements of these two activities. Therefore, a novel two-phase IPPS approach is put forward in this paper. In the new method, the preplanning phase generates a process network for each job with consideration of the static shop floor status. After that, the final planning phase simultaneously creates the process plan of each job and the scheduling plan according to the current shop floor status. Based on the modified definition of IPPS and the proposed mathematical model, the IPPS problem and the dynamic IPPS problem can be solved together. Furthermore, a discrete particle swarm optimization (DPSO) algorithm is proposed to solve the IPPS optimization problem. In the DPSO algorithm, the particles update their positions by crossing with their own historical best positions (pbests) and the global best position of the population (gbest). In order to avoid local convergence, an external archive is introduced to keep more than one elite, and the gbest of each particle is randomly selected from the external archive. Furthermore, mutation operation is introduced to enhance the local search ability of DPSO algorithm. Finally, some comparative results are given to verify the efficiency and effectiveness of the proposed IPPS method and the DPSO algorithm as well as the dynamic IPPS method.

© 2018 CPE, University of Maribor. All rights reserved.

ARTICLE INFO

Keywords:
Process planning;
Scheduling;
Dynamic integration;
Mathematical model;
Optimization;
Discrete particle swarm optimization (DPSO)

**Corresponding author:*
yumingrang@163.com
(Yu, M.R.)

Article history:
Received 4 December 2017
Revised 21 August 2018
Accepted 24 August 2018

1. Introduction

In the common manufacturing systems, which transform the raw material or semi-finished product into final product using kinds of machines, some preparation activities such as materials, tools, process plans, scheduling plans and so on. In these activities, process planning and scheduling are two crucial functions which are usually carried out sequentially. In other words, the process plans, which are the outcomes of process planning, are transferred into the scheduling system, which assigns operations to specified machines at appropriate moments according to the precedence relations in the process plans, shop floor status, scheduling criteria, etc. Obviously, the effectiveness of the scheduling results should be strongly dependent on the process plans. However, in the past years, these two activities are often executed sequentially, in which the scheduling plans are generated separately according to the fix process plans obtained by the process planning.

However, when considering the disturbances in manufacturing process such as arrival of urgent jobs, due date change and machine breakdown, the traditional method, in which process planning and scheduling are separately treated, seems to be inadequate for following reasons [1]:

- Traditionally, the process plans are generated by process planners under some ideal assumptions, for example, the resources in the shop floor are always available, which is unrealistic in a real manufacturing environment;
- The conventional process planning methods provide deterministic process plans to scheduling system, which ignores the possibility of improvement for scheduling with alternative process plans;
- Because of the time delay between process planning phase and scheduling phase, even if the dynamic shop floor status is considered in advance, it may change greatly when the schedule plan is executed, thus the generated optimal process plans may become sub-optimal or even invalid;
- In traditional way, the process planning generates optimal plans with the consideration of process plan criterion, and the scheduling works in a similar manner, conflicts may appear in such a separate way.

To overcome these shortages, Chryssolouris *et al.* [2] first proposed the concept of integration of process planning and scheduling (IPPS). When process planning and scheduling are integrated, the manufacturing system can respond promptly to disruptions. As a result, the resource utilization and the productivity will be improved.

In the following sections, some literature reviews on IPPS are given in Section 2, and Section 3 describes the materials and methods. Then the discrete particle swarm optimization algorithm is presented in Section 3, which is followed by the application of DPSO in optimizing the proposed IPPS problem in Section 4, which is followed by some experiment results in Section 5 and conclusions in Section 6.

2. Literature review

Ever since the concept of IPPS was proposed by Chryssolouris *et al.* [2], a lot of research papers on IPPS could be found. Based on the descriptions in some existing publications on IPPS [1, 3-6], the IPPS methods can be divided into three categories: non-linear process planning (NLPP), closed loop process planning (CLPP) and distributed process planning (DPP). These IPPS approaches and related contributions are described below.

NLPP: The NLPP method attempts to generate as many process plans for each part as possible under ideal shop floor status with consideration of the process flexibility, sequence flexibility [7] and operation flexibility. Here, the process flexibility represents the possibility of machining a specified feature using alternative processes or series of processes, hereafter, use *macro-level plan* to stand for process or a serial of processes for machining a feature. And the sequence flexibility stands for possibility of different sequences of the selected macro-level plans. Finally, the operation flexibility means that a specified process may be arranged to different machines. After that, different priority levels are assigned to these process plans according to the optimization objectives of process planning, e.g. total machining cost/time, etc. And then, these process plans will be tested in the scheduling system according to their priority levels.

CLPP: In the CLPP method, the process plans are always feasible with respect to the current scheduling environment because they are generated just before the jobs are released to the shop floor, in the other words, the status of each resource in the shop floor is known to the process planner. Some of the CLPP approaches first generate off-line plans according to the static shop floor status, and then make some necessary on-line refinement on the basis of the availability of resources on the shop floor at the subsequent scheduling phase.

DPP: The DPP method performs both process planning and scheduling simultaneously in a hierarchic manner. In most of the DPP methods, they can be divided into two phases, which are preplanning and final planning. Some other DPP approaches may have three phases: preplan-

ning, pair planning and final planning. Similarly, at the preplanning phase, the process planning function recognizes the features and the feature relationships of each job, and determines the macro-level plan for each feature. Meanwhile, the scheduling function estimates the required machine capabilities. Then, at the final planning phase (which can be further divided into pair planning and final planning in some DPP methods), both the final process plan of each part and the scheduling plan are generated simultaneously, in this way, the integration occurs at the moment that the machining processes and the available resources are matched.

Both the NLPP and CLPP methods are one-way information flows as shown in Fig. 1, in which process planning is still executed before scheduling. In this way, the objectives of process planning and scheduling may be conflicted. Therefore, Both NLPP and CLPP are incapable of finding the global optimal solution. With this in mind, the DPP method, which performs the technical and capacity-related planning tasks simultaneously, may be the only one IPPS method that integrates the functions of process planning and scheduling. However, in most of the DPP methods [8, 9], the macro-level plan of each feature is determined in the preplanning stage; and the selected macro-level plans for all features are used for matching the operation capabilities of the available resources in the final planning stage. It means that the flexibility of the final planning phase is reduced because of ignoring the process flexibility.

As mentioned before, optimization algorithm is another research focus on IPPS because of its NP-hard characteristic [11, 12]. Kinds of optimization algorithms, such as genetic algorithm (GA) [13], particle swarm optimization (PSO) algorithm [8], etc. have been used for optimizing the IPPS problems. However, most of these algorithms are designed for one of the three existing IPPS methods. Therefore, the performances of those optimization algorithms may be restricted by the corresponding IPPS methods.

As is known to all, the basic purpose of IPPS is handling the inevitable disturbances in the manufacturing processes by treating process planning and scheduling as a whole. However, even in the newest publications on IPPS, dynamic IPPS is still less mentioned in the literatures for its complexity. For example, Zhang and Wong [14] proposed an object-coding genetic algorithm for IPPS, and Wang *et al.* [15] addressed a systematic approach for optimizing the IPPS problems in sustainable machining. The models in these two papers are still designed for static IPPS, in which the results of IPPS are final process plans and schedules. It is worth mentioning that even the optimal solution is obtained using integrated method, this solution may also be deteriorated in the execution stage because of the inevitable disturbances. Therefore, dynamic IPPS is necessary for further improving the performances of the manufacturing system.

To sum up, an effective IPPS method and the corresponding optimization algorithm are still necessary to be developed to get full integration of process planning and scheduling. Therefore, this paper proposes a novel IPPS method to overcome the shortages of the three existing IPPS methods. Based on the modified definition of IPPS, the mathematical model of the proposed IPPS method is presented, which is also suitable for dynamic IPPS method. Finally, a discrete particle swarm optimization is developed for optimizing the proposed IPPS problem.

Fig. 1 Information flows of: (a) NLPP and (b) CLPP [10]

3. Materials and methods

3.1 The proposed IPPS method

As shown in Fig. 2, the proposed IPPS method is a two-phase approach, which also contains preplanning and final planning. Be different from the existing DPP methods, at the preplanning phase, instead of determining the macro-level plan for each feature, the process network, which is widely used in representing the flexibility of process plan [16, 17], of each part to be machined is generated according to the static resources. A process network types of nodes, which are

starting node, intermediate node and ending node, more details for the process network can be found in the reference of Ho *et al.* [17]. Then the process networks of all the parts to be machined are stored and will be used in the final planning phase. With consideration of optimization criteria of both process planning and scheduling, an optimal plan, which includes the process plans of all parts to be machined and the scheduling plan, is generated according to the current shop floor status. Of course, an effective optimization algorithm is necessary because of the complexity of the problem.

After that, the optimal plan is released to the shop floor immediately. Suppose the makespan of the optimal plan is T and current time is t_0. Because disturbances such as machine breakdown, rush order and order cancellation may occur during the time interval t_0 to $t_0 + T$, a dynamic IPPS method, in which both the process plans of the jobs and the schedule are changeable, is presented to handle the disruptions.

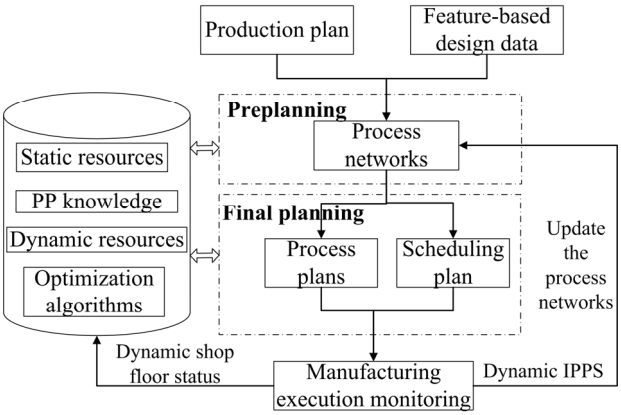


Fig. 2 The structure of the proposed IPPS method (PP stands for process planning)

3.2 Mathematical model of the proposed IPPS method

In order to solve the IPPS and the dynamic IPPS in a uniform manner, the definition of IPPS [18] is modified as follows.

Given a set of n jobs, at the time t, the number of the unfinished features of each part is N_{if}, the number of the available machine at time t is M_t. considering the flexibility of process, sequence and operation, determine the macro-level plan for each unfinished feature and the corresponding machine (machine set) as well as the sequences of the operations on each machine. Meanwhile, the precedence constraints in each part are satisfied and some optimization objectives can be achieved.

Based on the proposed IPPS method and the modified definition of IPPS, the mathematical model of the proposed IPPS method is founded. In this paper, the optimization objectives of process planning and scheduling are minimum machining time and minimum makespan, separately. Firstly, some reasonable assumptions and notations are given below:

- Jobs are independent. Job preemption is not allowed and each machine can handle only one job at any moment.
- The different operations of the same job cannot be machined simultaneously.
- The setup time and the transport time are negligible or included in the processing time.

M_t the total number of the available machines at the time t;
N the total number of jobs to be machined at the time t;
N_{if} the number of the unfinished features of the i-th job at the time t;
N_{ijo} the number of alternative macro-level plans of the j-th feature of i-th job;
O_{ijk} the k-th alternative macro-level plan of the j-th feature of i-th job;
M_{ijkl} the l-th alternative machine (machine set) of the macro-level plan O_{ijk};
t_{ijkl} the machining time of macro-level plan O_{ijk} on machine l;
c_{ijkl} the completion time of macro-level plan O_{ijl} on machine k;

$$X_{ijk} = \begin{cases} 1 & \text{the } k\text{-th alternative macro-level plan is selected for } j\text{-th feature of } i\text{-th job} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ijkl} = \begin{cases} 1 & \text{the } l\text{-th machine is selected for the macro-level plan } O_{ijkl} \\ 0 & \text{otherwise} \end{cases}$$

Objectives:

- Minimizing the total machining time:

$$F_1 = \text{Min} \sum_{i=1}^{N} \sum_{j=1}^{N_{if}} \sum_{k=1}^{N_{ijo}} \sum_{l=1}^{M_t} t_{ijkl} \times Y_{ijkl} \times X_{ijk} \tag{1}$$

- Minimizing the makespan:

$$F_2 = Min\{Makespan = Max\{c_{ijkl} \times X_{ijk} \times Y_{ijkl}\} \tag{2}$$

Subject to:

- The j-th feature of the i-th job can only choose one macro-level plan from its alternatives:

$$\sum_{k=1}^{N_{ijo}} X_{ijk} \tag{3}$$

- The macro-level plan O_{ijk} can only choose one machine (machine set) from its alternatives:

$$\sum_{k=1}^{N_{ijo}} X_{ijk} \tag{4}$$

Other constraints can be found in Li *et al*. [19]. It is worth mentioning that the numbers of the available machines and the jobs to be processed as well as the unfinished features of each job are varied as time goes by. Therefore, this mathematical model is also suitable for the dynamic IPPS method. And these kinds of information of dynamic shop floor status are provided by the manufacturing execution system.

3.3 Particle swarm optimization algorithm

The proposed IPPS method involves in four tasks, which are selection of macro-level plan for each unfinished feature, selection of machine (machine set) for each selected macro-level plan, sequencing the selected macro-level plans and determining the starting time of each process on its corresponding machine. It is a typical NP-hard problem and much more complex to solve than that of existing IPPS methods, which are partially involved in the four referred tasks. Particle swarm optimization (PSO) algorithm has been widely used in many optimization problems since it was proposed in 1995 [20]. However, the continuous characteristic of PSO algorithm restricts its applications in combinatory optimizations problems. Some researchers [21, 22] used modified encoding methods to transform the scheduling problems into continuous versions, and then the PSO can be used to solve these scheduling problems. However, they are incapable of solving the optimization problem of the proposed IPPS method since its complexity. Therefore, this paper proposes a discrete particle swarm optimization (DPSO) algorithm for the proposed IPPS method.

Standard PSO algorithm

In most of the researches, the standard PSO refers to a modified PSO algorithm [23], in which an inertia weight is introduced to improve the optimizing ability. In the PSO algorithm, a swarm, which contains a number of particles with certain positions and velocities, is initialized randomly. Then each particle dynamically adjusts its velocity and position according to its own and the population's experiences. This can be explained as follows:

$$V_{id}^{t+1}=\begin{cases}\omega V_{id}^{t}+c_1r_1\left(pbest_{id}-X_{id}^{t}\right)+c_2r_2\left(gbest_d-X_{id}^{t}\right) & V_{dmin}\le V_{id}^{t+1}\le V_{dmax}\\ V_{dmax} & V_{id}^{t+1}>V_{dmax}\\ V_{dmin} & V_{id}^{t+1}<V_{dmin}\end{cases} \tag{5}$$

$$X_{id}^{t+1}=\begin{cases}X_{id}^{t}+V_{id}^{t+1} & -X_{dmax}\ll X_{id}^{t+1}\ll X_{dmax}\\ X_{dmax} & X_{id}^{t+1}>X_{dmax}\\ -X_{dmax} & X_{id}^{t+1}<-X_{dmax}\end{cases} \tag{6}$$

$V_{id}^{t+1}$is the velocity of the *d*-th dimension of the *i*-th particle at the time *t*, and it means the distance to be traveled in the *d*-th dimension from its current position. ω is the inertial weight used for regulating the trade-off between the global exploration and local exploration abilities of the swarm. c_1 and c_2 are the acceleration constants, $r_1$and r_2 are two random functions within the range of [0,1]. $pbest_{id}$ represents the *d*-th dimension of the *i*-th particle's best historical position, and$gbest_d$ represents the *d*-th dimension of the swarm's global best position. The particles update their positions and velocities according to Eq. 5 and Eq. 6, and the *pbest* and *gbest* are also updated according to the fitness function. Thus all the particles move to the global best solution to finish the search process.

Discrete PSO algorithm

It can be found that the position and the velocity in standard PSO are both continuous variables from Eq. 5 and Eq. 6, which are meaningless for the IPPS optimization problems. This paper proposes a discrete version of PSO, in which the particles update their positions according to the following equation:

$$X_i^{t+1}=X_i^{t+1}\otimes P_{ibest}^{t}\otimes P_{gbest}^{t} \tag{7}$$

X_i^{t+1} and X_i^{t} are the positions of the *i*-th particle in current and the next iteration, P_{ibest}^{t} is the historical best position of the *i*-th particle, and P_{gbest}^{t} is the best position of the population in the *t*th iteration. $\otimes$ stands for the crossover operator. Eq. 7 shows that the particles adjust their positions according to their own and the population's experiences, which maintains the advantages of the standard PSO algorithm. The discrete particle swarm optimization algorithm can be explained as follow according to Fig. 3:

- The particleX_i^{t} and its *pbest* $P_{ibest}^{t}$are considered as two parents P_1 and P_2; O_1 and O_2 are two offspring of them. Select the better one of these two offspring for the next crossover, e.g. O_2 is selected;
- O_2 andP_{gbest}^{t}, which is the *gbest* of X_i^{t}, are treated as two new parents $P_1'$and P_2'; O_1' and O_{21}' are two offspring of them. Select the better one of these two offspring to be used as the new position of X_i^{t}, that is X_i^{t+1}.

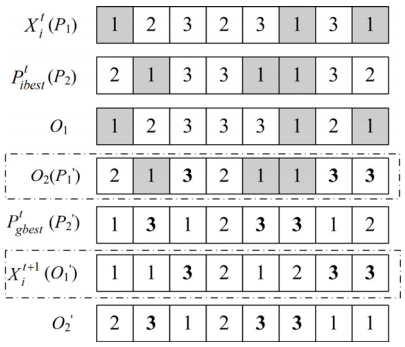


Fig. 3 Illustration of discrete particle swarm optimization algorithm

Generally, the *gbest* of the swarm is unique. Under this condition, each particle will cross with that *gbest*, which may lead to local convergence. For this reason, this paper introduces an external archive to keep the elites, which are the first N_{ea} (N_{ea} the size of the external archive) parti-

cles with lowest fitness values, generated in each iteration. Then each particle randomly selects an individual from the external archive as its *gbest* in each iteration, and the members of the external archive are also updated iteratively.

4. The application of discrete particle swarm optimization (DPSO) in optimizing the proposed IPPS problem

When applying the DPSO algorithm to optimize the IPPS problem, some necessary explanations and definitions should be given in advance. The information of a process network can also be listed in a table. Table 1 lists the information contained in four process networks of four jobs, in which '999' means that machine cannot finish the process, and O_{1211}-O_{1212} means the macro-level plan O_{121} contains two processes. It is worth mentioning that different macro-level plans of the same feature may have different machining time on the same machine. For example, the machining time of O_{111} and O_{112} on machine M_1 is 3 and 5 units, separately. The machine priority level for a macro-level plan is defined as follows. *Machine priority level*: For a macro-level plan, the machine priority level (1>2>3>4>5, etc.) is determined by the machining time of the machine or machine set. The longer the machining time of a machine is, the higher its machine priority level will be. And if the machining time of two machines (machine sets) for the same macro-level plan are equal, the bigger the number of the machine is, the higher its machine priority level will be.

Table 1 Information contained in the four process networks of four jobs

Jobs	Features	Alternative macro-level plans	Alternative machines and machining time				
			M_1	M_2	M_3	M_4	M_5
J_1	F_{11} (before F_{12})	O_{111}	999	3	4	999	4
		O_{112}	5	999	3	4	999
		O_{113}	3	4	999	999	4
	F_{12}	O_{1211}-O_{1212}	2/999	3/4	999/3	999/4	999/999
		O_{1221}-O_{1222}	3/2	999/3	4/999	4/3	3/999
	F_{13}	O_{131}	4	5	999	3	999
		O_{132}	999	999	3	4	5
		O_{133}	3	4	3	5	3
J_2	F_{21}	O_{211}	6	999	5	999	999
		O_{212}	999	6	999	6	5
	F_{22}	O_{2211}-O_{2212}	3/4	999/2	4/3	999/999	3/999
	F_{23} (before F_{24})	O_{231}	5	3	999	4	999
		O_{232}	4	999	4	5	5
		O_{233}	4	4	5	999	3
	F_{24}	O_{241}	5	3	4	999	4
		O_{242}	4	999	5	999	6
J_3	F_{31}	O_{3111}-O_{3112}-O_{3113}	999/3/999	4/5/3	4/4/6	999/4/5	5/999/4
	F_{32}	O_{321}	4	999	5	3	4
		O_{322}	5	5	4	6	4
	F_{33} (before F_{34})	O_{331}	4	6	4	5	999
		O_{332}	3	5	3	4	999
		O_{333}	999	4	999	3	5
	F_{34}	O_{3411}-O_{3412}	999/4	4/5	5/3	3/999	999/999
		O_{342}	10	10	9	999	999
J_4	F_{41}	O_{411}	4	999	5	999	4
	F_{42} (before F_{43})	O_{421}	3	4	999	3	5
		O_{422}	5	3	4	999	999
		O_{423}	3	4	999	4	3
	F_{43}	O_{4311}-O_{4312}-O_{4313}	5/3/999	4/5/4	5/999/6	999/5/999	4/6/5
	F_{44}	O_{441}	5	6	5	999	5
		O_{442}	6	999	5	6	4
		O_{4431}-O_{4432}	999/4	2/3	4/999	999/999	3/4

Table 2 Machine priority levels of the alternative machines

Jobs	Features	Alternative macro-level plans	Machine priority level(1>2>3>4>5)				
			1	2	3	4	5
J_1	F_{11}	O_{111}	$M_2(3)$	$M_3(4)$	$M_5(4)$	$M_1(999)$	$M_4(999)$
	(before	O_{112}	$M_3(3)$	$M_4(4)$	$M_1(5)$	$M_2(999)$	$M_5(999)$
	F_{12})	O_{113}	$M_1(3)$	$M_2(4)$	$M_5(4)$	$M_3(999)$	$M_4(999)$
	F_{12}	O_{1211}-O_{1212}	M_1+$M_3(5)$	M_1+$M_2(6)$	M_1+$M_4(6)$	M_2+$M_3(7)$	$M_2(7)$
		O_{1221}-O_{1222}	$M_1(5)$	M_5+$M_1(5)$	M_5+$M_2(6)$	$M_4(7)$	M_4+$M_2(7)$
	F_{13}	O_{131}	$M_4(3)$	$M_1(4)$	$M_2(5)$	$M_3(999)$	$M_5(999)$
		O_{132}	$M_3(3)$	$M_4(4)$	$M_5(5)$	$M_1(999)$	$M_2(999)$
		O_{133}	$M_1(3)$	$M_3(3)$	$M_5(3)$	$M_2(4)$	$M_4(5)$
J_2	F_{21}	O_{211}	$M_3(5)$	$M_1(6)$	$M_2(999)$	$M_4(999)$	$M_5(999)$
		O_{212}	$M_5(5)$	$M_2(6)$	$M_4(6)$	$M_1(999)$	$M_3(999)$
	F_{22}	O_{2211}-O_{2212}	M_1+$M_2(5)$	M_5+$M_2(5)$	M_1+$M_3(6)$	$M_1(7)$	$M_3(7)$
	F_{23}	O_{231}	$M_2(3)$	$M_4(4)$	$M_1(5)$	$M_3(999)$	$M_5(999)$
	(before	O_{232}	$M_1(4)$	$M_3(4)$	$M_4(5)$	$M_5(5)$	$M_2(999)$
	F_{24})	O_{233}	$M_5(3)$	$M_1(4)$	$M_2(4)$	$M_3(5)$	$M_4(999)$
	F_{24}	O_{241}	$M_2(3)$	$M_3(4)$	$M_5(4)$	$M_1(5)$	$M_4(999)$
		O_{242}	$M_1(4)$	$M_3(5)$	$M_5(6)$	$M_2(999)$	$M_4(999)$
J_3	F_{31}	O_{3111}-O_{3112}-O_{3113}	M_1+$M_2(10)$	M_2+$M_4(11)$	$M_2(12)$	M_1+$M_5(12)$	$M_3(14)$
	F_{32}	O_{321}	$M_4(3)$	$M_1(4)$	$M_5(4)$	$M_3(5)$	$M_2(999)$
		O_{322}	$M_3(4)$	$M_5(4)$	$M_1(5)$	$M_2(5)$	$M_4(6)$
	F_{33}	O_{331}	$M_1(4)$	$M_3(4)$	$M_4(5)$	$M_2(6)$	$M_5(999)$
	(before	O_{332}	$M_1(3)$	$M_3(3)$	$M_4(4)$	$M_2(5)$	$M_5(999)$
	F_{34})	O_{333}	$M_4(3)$	$M_2(3)$	$M_5(5)$	$M_1(999)$	$M_3(999)$
	F_{34}	O_{3411}-O_{3412}	M_4+$M_3(6)$	M_4+$M_1(7)$	M_2+$M_3(7)$	$M_3(8)$	$M_2(9)$
		O_{342}	$M_3(9)$	$M_1(10)$	$M_2(10)$	$M_4(999)$	$M_5(999)$
J_4	F_{41}	O_{411}	$M_1(4)$	$M_5(4)$	$M_3(4)$	$M_2(999)$	$M_4(999)$
	F_{42}	O_{421}	$M_1(3)$	$M_4(3)$	$M_2(4)$	$M_5(5)$	$M_3(999)$
	(before	O_{422}	$M_2(3)$	$M_3(4)$	$M_1(5)$	$M_4(999)$	$M_5(999)$
	F_{43})	O_{423}	$M_1(3)$	$M_5(3)$	$M_2(4)$	$M_4(4)$	$M_3(999)$
	F_{43}	O_{4311}-O_{4312}-O_{4313}	M_1+$M_2(12)$	$M_2(13)$	M_1+$M_5(13)$	M_1+$M_3(14)$	$M_5(15)$
	F_{44}	O_{441}	$M_5(4)$	$M_3(5)$	$M_1(6)$	$M_4(6)$	$M_2(999)$
		O_{442}	$M_1(5)$	$M_3(5)$	$M_5(5)$	$M_2(6)$	$M_4(999)$
		O_{4431}-O_{4432}	$M_2(5)$	M_2+$M_1(6)$	M_2+$M_5(6)$	$M_5(7)$	M_5+$M_1(7)$

Based on the definition of machine priority level, the machine priority levels of the machines shown in Table 1 can be obtained as shown in Table 2. As referred before, the process network is generated based on static shop floor status in the preplanning phase, thus in the final planning phase, the status of the shop floor as well as the machine priority levels should be updated.

4.1 Encoding and decoding

Encoding refers to problem mapping, which translates a problem to a particle. For the jobs listed in Table 2, a possible particle which contains three sections is shown in Fig. 4. The feature string, whose length equals to the total number of the features of all jobs, represents the manufacturing sequence of the features. For example, '1, 2, 3' represent the features of job 1, and '4, 5, 6, 7' represent the features of job 2, etc. The MLP string stands for the selected macro-level plans for corresponding features, e.g. the second 2 in the MLP string corresponds to the number 9 in the feature string. It means that the second feature of the third job (F_{32}) has selected its second alternative macro-level plan (O_{322}). The machine string represents the machine priority levels of the selected machines, e.g. the macro-level plan referred before (O_{322}) has selected the first prior machine (M_3) from its alternatives because the value of the corresponding position in the machine string is 1.

Feature string	3	1	2	4	9	10	12	15	8	11	6	7	5	13	14
MLP string	1	3	2	1	2	3	1	2	1	2	1	2	1	2	1
Machine string	2	1	2	2	1	1	2	2	1	2	2	1	2	3	2

Fig. 4 A particle for the four jobs listed in Table 2 (MLP stands for macro-level plan)

Decoding means solution generation, which translates a particle to a solution of the optimization problem. In the proposed IPPS method, the solution contains the process plan of each job and the scheduling plan. Based on the encoding method referred before, the decoding process is described as follows (take the particle shown in Fig. 4 for example):

Step 1: Translate the feature string to the features machining sequence of each job, for example, the four jobs' feature machining sequence are '3, 1, 2', '4, 6, 7, 5', '9, 10, 8, 11' and '12, 15, 13, 14', separately; meanwhile, the machining sequence of all the features is also determined, that is F_{13}-F_{11}-F_{12}-F_{21}-F_{32}-F_{33}-F_{41}-F_{44}-F_{31}-F_{34}-F_{23}-F_{24}-F_{22}- F_{42}-F_{43};
Step 2: Translate the MLP string to the selected macro-level plans for corresponding features;
Step 3: Translate the machine string to the selected machines for corresponding macro-level plans;
Step 4: Combine the results of steps 1, 2 and 3 to generate the process plan for each job.
Step 5: According to the results of steps 1, 2 and 3, the features machining sequence, selected macro-level plans and corresponding machines are known, and then the decoding method proposed by Zhang *et al.* [24] is introduced to decode the particle to an active scheduling plan.

4.2 Fitness function

In this paper, the optimizing objectives of process planning and scheduling are minimum total machining time and minimum makespan, respectively. Although these two objectives have the same dimension, their orders of magnitude may vary enormously. Thus the normalization of the data is still necessary. Let $D_i(T_i, M_i)$ $(1 \le i \le N)$ stand for the assessment of the solution corresponding to the i-th particle, in which N is the number of the particles in the swarm. T_i and M_i represent the total machining time and the makespan, respectively. The procedure of normalization can be explained as follows:

$$T_i' = \frac{T_i - \min\{T_j, 1 \le j \le N\}}{\max\{T_j, 1 \le j \le N\} - \min\{T_j, 1 \le j \le N\}} \tag{8}$$

$$M_i' = \frac{M_i - \min\{M_j, 1 \le j \le N\}}{\max\{M_j, 1 \le j \le N\} - \min\{M_j, 1 \le j \le N\}} \tag{9}$$

Thus, two new arrays of total machining time and makespan are obtained, which are $[T_1', T_2', \ldots, T_N']$ and $[M_1', M_2', \ldots, M_N']$, respectively. Then the final objective function of the IPPS optimization problem as well as the fitness function can be obtained as follow:

$$F = \omega_1 T_i' + \omega_2 M_i' \tag{10}$$

Where ω_1 and ω_2 stand for the weights of total machining time and makespan, respectively. Meanwhile, the relations $0 \le \omega_1 \le 1$, $0 \le \omega_2 \le 1$ and $\omega_1 + \omega_2 = 1$ should be satisfied.

4.3 Particle updating strategy

In the DPSO algorithm, the particles update their positions by crossing with their own historical best positions (*pbest*s) and the global best position of the population (*gbest*), and a mutation operator is designed to enhance the local search ability of the algorithm. Based on the encoding method referred before, the crossover and mutation operations are designed for the IPPS problem.

Crossover

As shown in Fig. 4, a particle contains three different strings: feature string, MLP string and machine string. The crossover operator acts on them separately. Thus three crossover operations are needed. The procedure of the crossover operation for the feature string is described as follows according to Fig. 5.

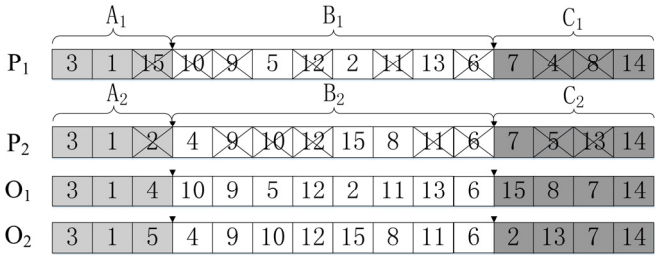


Fig. 5 The crossover operation for the feature string

Step 1: P_1 and P_2 are the feature strings of two particles, and generate two empty offspring: O_1 and O_2;
Step 2: Randomly select two different crossover points to divide P_1 to three sections: A_1, B_1 and C_1; this process is done to P_2 at the same positions of crossover points;
Step 3: Copy the elements in B_1 (B_2) into the same positions of O_1 (O_2);
Step 4: Delete the existing elements of O_1 (O_2) in P_2 (P_1), and then copy the remaining elements in P_2 (P_1) to the remaining empty positions in O_1 (O_2).

The typical two-point crossover operator is introduced and implemented on the MLP string and an example of this crossover operation for the MLP string is presented in Fig. 6. The crossover operation for machine string has the same procedure with that of MLP string.

It is worth mentioning that since the number of the alternative macro-level plans of different features may be different, the ranges of the values of different bits in MLP string may be different. It means that infeasible particles may be generated by the crossover operation, and a refinement procedure is necessary, which will be presented in Section *Refinement*.

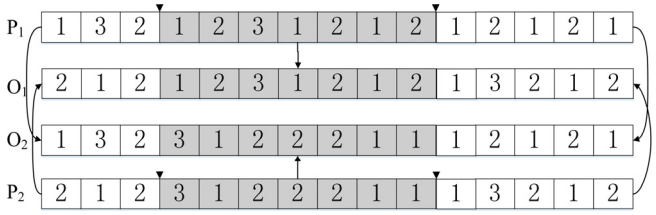


Fig. 6 The crossover operation for the MLP string

Mutation

Mutation operation is introduced in the DPSO algorithm to enhance its local search ability. Since the values of the elements in the MLP string are restricted by the values of corresponding elements in the feature string, the mutation operations should be sequentially conducted for the feature string as well as MLP string and the machine string. In order to save space, a partial particle is taken for example, and the procedure for mutation operations can be described as follows according to Fig. 7.

Step 1: Mutate the feature string: randomly select two different positions and exchange their elements in the feature string and the MLP string as well as the machine string;
Step 2: Mutate the MLP string: randomly select a bit from the MLP string and change its value within the range of that bit according to the value of its corresponding bit in the feature string;
Step 3: Mutate the machine string: randomly select a bit from the machine string and change its value.

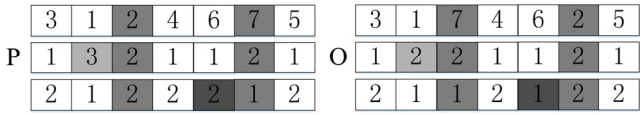


Fig. 7 The mutation operation for a partial particle

Refinement

As referred before, the crossover operation may generate infeasible particles, in which the values of the elements in the MLP string may exceed their value ranges. Additionally, the mutation operation and the random initialization may also generate invalid particles, in which the precedence relationships among features may be dissatisfied. The procedure of the refinement for the infeasible particles is described as follows:

Step 1: The constraint adjustment method of Li *et al.* [25] is introduced to adjust the feature string. In order to maintain the corresponding relationships among features, macro-level plans and machines, the adjustment operation is simultaneously implemented on the three strings. It means that if two elements in the feature string are swapped, the elements of corresponding positions in both the MLP string and the machine string should be swapped, simultaneously.

Step 2: Check the values of the elements in the MLP string. If the value of an element exceeds its value range, randomly choose a value within its value range to replace its previous value. For example, the numbers of the alternative macro-level plans of the features F_{12} and F_{13} are 2 and 3, separately. Suppose after crossover operation, the values of the elements in the MLP string correspond to these two features are 3 and 2, respectively. It means that the value of the element in the MLP string corresponds to the feature F_{12} has exceeded its value range [1, 2], and then the value of that element may be randomly replaced by 1 or 2.

4.4 The procedure of applying DPSO in optimizing the proposed IPPS problem

Based on the explanations before, the procedure of applying DPSO in optimizing the proposed IPPS problem is described as follows:

Step 1: Parameter setting and initialization: set the parameters and randomly generate the initial swarm, iteration number $i = 1$;

Step 2: Refinement: refine the infeasible particles using the refinement method presented in Section *Refinement*;

Step 3: Decoding: decode each particle to a solution, which contains the process plan of each job and the scheduling plan, using the decoding method referred before;

Step 4: Evaluate the fitness of each particle and sort the particles according to their fitness values in ascending order, then the first N_{ea} particles are selected for updating the members of external archive (EA);

Step 5: Update EA: if $i = 1$, insert the particles selected in step 4 into EA directly; otherwise, combine the particles selected in step 4 and the existing particles in the EA, and sort these $2N_{ea}$ particles according to their fitness values in ascending order, then the first N_{ea} particles are preserved in EA;

Step 6: Update the *pbest* and *gbest* of each particle, in which the *gbest* of each particle is randomly selected from the members of EA;

Step 7: If $i \leq Iter_{max}$ ($Iter_{max}$ is the maximum iteration number), go to step 8; otherwise, go to step 9;

Step 8: Update the positions of the particles using the updating strategy (crossover and mutation) presented in Section 5. 3, $i = i+1$, and go to step 2;

Step 9: Output the best solution: decode the best particle in EA, whose fitness value is the lowest, to a solution, which includes the process plan of each job and the scheduling plan.

5. Results and discussion

As referred before, the proposed IPPS method has combined the advantages of the three existing IPPS methods, which are NLPP, CLPP and DPP methods. Additionally, based on the modified definition and the mathematical model of IPPS, dynamic scheduling can be extended to dynamic IPPS for effectively handling the disruptions. Besides, a DPSO algorithm is proposed for optimizing the IPPS problem. In this section, the proposed DPSO algorithm is compared with some other

intelligent algorithms to show its effectiveness, and two test cases are designed to verify the effectiveness of the proposed IPPS method.

5.1 Case 1

Case 1 is designed to verify the efficiency and effectiveness of the proposed DPSO algorithm. Because there is no existing test instance, which is very similar with the IPPS problem in this paper, in order to compare the proposed algorithm with other intelligent algorithms, some classical benchmarks of flexible job-shop scheduling problem (FJSP), which are widely introduced in related researches, are used for instead. The sources of the benchmarks and the testing environment are given below:

Kacem data: five representative instances which cover a wide range from 4×5 to 15×10 taken from the works of Kacem *et al.* [26, 27] are used in the experiment without the consideration of release date. The instances are 4×5 problem (I_1), 8×8 problem (I_2), 10×7 problem (I_3), 10×10 problem (I_4) and 15×10 problem (I_5), and the details of them can be found in literatures [26, 27].

BRdata: the BRdata is consists of 10 test problems from Brandimarte [28] which were randomly generated using uniform distributions. In the selected BRdata, the number of machines ranges from 4 to 15, the number of jobs ranges from 10 to 20, the number of operations for each job ranges from 5 to 15, and the number of operations for all jobs ranges from 55 to 240.

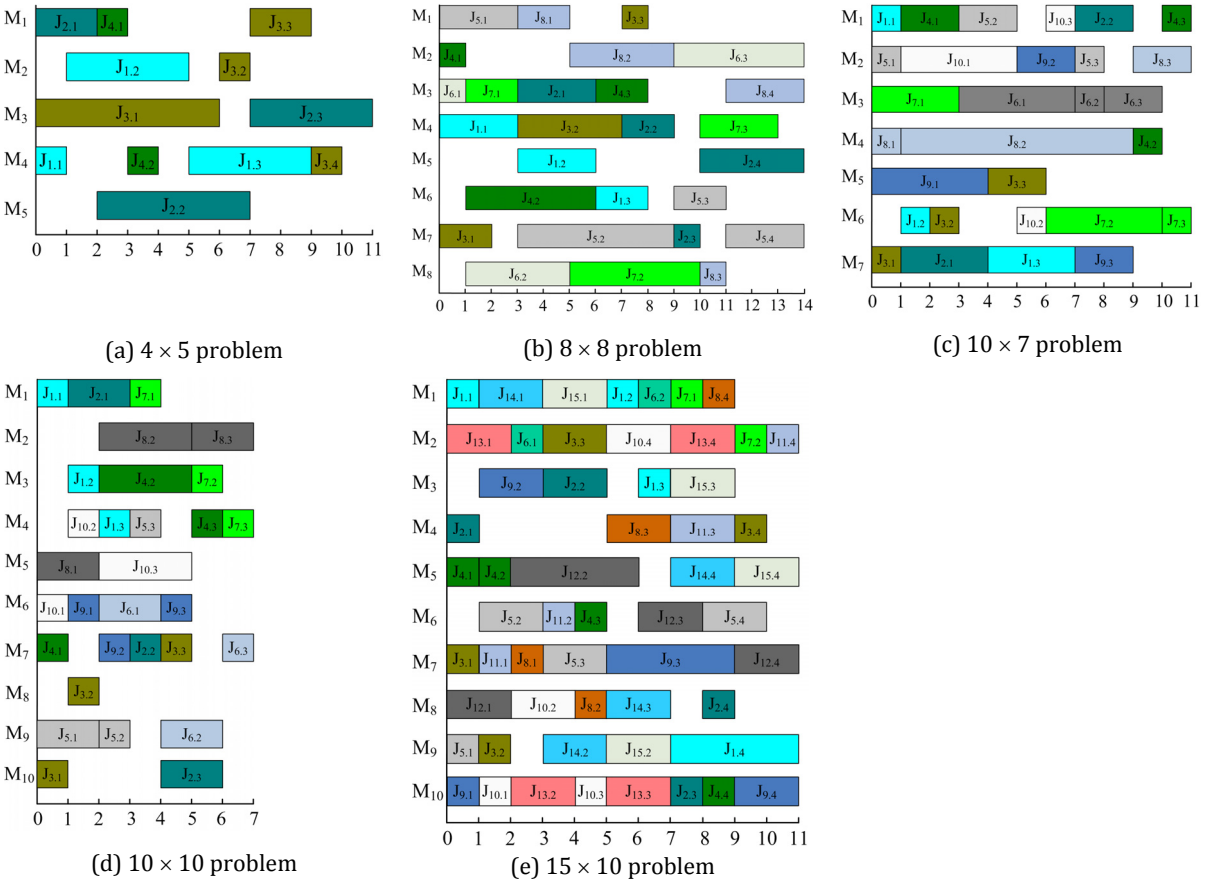
The programs are implemented using Matlab, running on a PC with 2.2GHz CPU and 4GB RAM. The main parameters are: the population size is $10 \times n$, where n is the number of jobs. The maximum number of iteration is $10 \times n \times m$, where m is the number of machines. The crossover probability $P_c = 1$, which means that each particle should cross with its own *pbest* and its selected *gbest*. Since the selected benchmarks are often used for testing multi-objective FJSP, and the multi-objective optimization is not the focus of this paper, the objectives are combined into a scalar function with equal weight of each objective.

Firstly, the proposed DPSO algorithm is applied on Kacem data and the results are compared with several other methods in recent publications, which include PDABC [29], MOGA [30], HPSO [31] and PSO/LS [32]. The comparative results are shown in Table 3. The three objectives are F_1 (makespan), F_2 (total workload) and F_3 (workload of the critical machine). The Gantt charts of the optimal solutions of these instances obtained by DPSO are shown in Fig. 8.

Table 3 Comparative results of the five instances of Kacem data

Instance	PDABC			MOGA			HPSO			PSO/LS			DPSO		
	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3
I_1	11	32	10	11	32	10	11	32	10	16	32	8	11	32	1
（4×5）	12	32	8	11	34	9				16	33	7			
	13	33	7	12	32	8									
I_2	14	77	12	15	81	11	14	77	12				14	77	1
（8×8）	15	75	12	15	75	12									
	16	73	13	16	73	13									
I_3	12	61	11							16	60	12	11	62	1
（10×7）	11*	63*	11*							15	61	11			
	12	60	12							15*	62*	10*			
I_4	8	41	7	8	42	5	7*	43*	6*	8	41	7	7	43	5
（10×10）	7	43	5	7	42	6				7*	44*	5*			
	8	42	5	8	41	7				7	42	6			
				7*	45*	5*									
I_5	12*	91*	11*	11	91	11	11*	93*	11*	23*	91*	11*	11	91	1
（15×10）	11*	93*	11*	12	95	10									
				11	98	10									

* Means that the solution is worse than the solution obtained by the proposed DPSO algorithm



(a) 4 × 5 problem (b) 8 × 8 problem (c) 10 × 7 problem

(d) 10 × 10 problem (e) 15 × 10 problem

Fig. 8 Gantt charts of the optimal solutions for (a) I_1, (b) I_2, (c) I_3, (d) I_4, and (e) I_5

Secondly, the proposed DPSO algorithm is applied on the BRdata, and the experiment results are compared with those of other two algorithms in recent publications, which are artificial immune (AI) [33] and efficient search (ES) [34], the comparative results are listed in Table 4.

From Table 3, it can be found that the results of the proposed DPSO algorithm are no worse (most of them are better) than those of other methods. Similarly, from the comparative results shown in Table 4, it is obvious that most of the solutions obtained by the DPSO algorithm are better than those of the AI algorithm, meanwhile, the CPU times are also reduced a lot. Although the CPU times of DPSO algorithm are longer than those of ES, the quality of the solutions has been greatly improved. In a word, the proposed DPSO algorithm stands out for its efficiency and effectiveness.

Table 4 Comparative results of the ten instances of BRdata

Instance	AI				ES				DPSO			
	F_1	F_2	F_3	T_{CPU}/S	F_1	F_2	F_3	T_{CPU}/S	F_1	F_2	F_3	T_{CPU}/S
MK01	40	171	36	97.21	42*	162*	42*	4.78	40	167	36	17.93
MK02	26*	154*	26*	103.46	28*	155*	28*	3.02	26	150	26	19.24
MK03	204*	1207*	204*	247.37	204*	852*	204*	26.14	204	850	204	56.41
MK04	60*	403*	60*	152.07	68*	372*	67*	17.74	60	372	60	45.87
MK05	173*	686*	173*	171.95	177*	702*	177*	8.26	172	687	172	92.54
MK06	63*	470*	56*	245.62	75*	431*	67*	18.79	58	429	55	89.46
MK07	140	695	140	161.92	150*	717*	150*	5.68	140	695	140	124.62
MK08	523*	2723*	523*	392.25	523	2524	523	67.67	523	2524	523	148.44
MK09	312*	2591*	306*	389.71	311*	2374*	299*	77.76	307	2282	299	263.74
MK10	214*	2121*	206*	384.54	227*	1989*	221*	122.52	197	2029	197	296.47

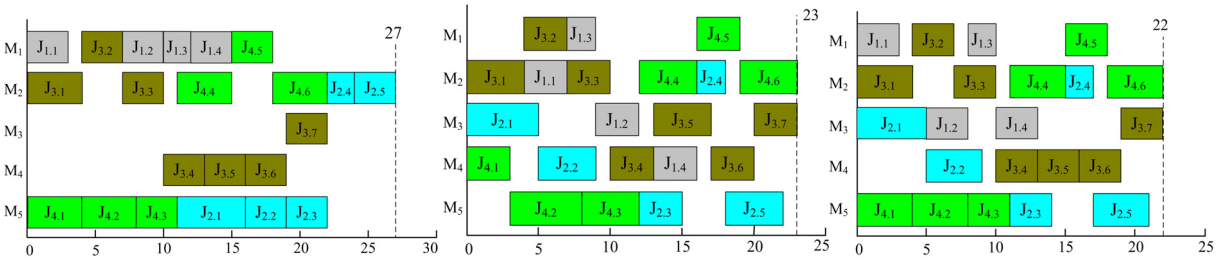
* Means that the solution is worse than the solution obtained by the proposed DPSO algorithm

5.2 Case 2

The efficiency and the effectiveness of the proposed DPSO algorithm have been verified by the experiment results in case 1. This section designs a test case to prove the effectiveness of the proposed IPPS method. Suppose the four jobs listed in Table 2 are released to the shop floor and all the machines in the shop floor are idle and available at the moment $t = 0$. Under this assumption, in the NLPP method, the first priory process plan for each part is feasible and will be selected at the scheduling phase. Thus the optimal process plan for a job obtained by the CLPP method will be the same as the process plan selected for that job in the NLPP method. Without loss of generality, in the DPP method, the first alternative macro-level plan is selected for each feature in the preplanning phase. The three existing IPPS methods and the proposed IPPS method are used to solve the IPPS problem listed in Table 2 under the referred assumption, separately. The comparative results are shown in Table 5 (process plans) and Fig. 9 (scheduling plans), respectively, in which T represents the total machining time. According to the comparative results we can find that the proposed IPPS method outperforms the DPP method because the DPP method ignores the process flexibility (possibility of machining the same feature with alternative macro-level plans). Furthermore, as referred before, actually, the NLPP and CLPP methods perform the process planning and scheduling sequentially, the optimization objectives of process planning and scheduling may be conflicted. For example, although the total machining time has reduced from 73 to 71 (about 2.74 %), the makespan time has increased from 22 to 27 (about 22.72 %), which means that the proposed IPPS method also does better than the NLPP and CLPP methods.

Table 5 Process plans of the three existing methods and the proposed IPPS method

Job	NLPP (CLPP)	DPP	The proposed IPPS
J_1	$O_{113}(M_1)$-$O_{1221}(M_1)$-$O_{1222}(M_1)$ -$O_{133}(M_1)$	$O_{111}(M_2)$-$O_{1211}(M_1)$-$O_{1212}(M_3)$-$O_{131}(M_4)$	$O_{113}(M_1)$-$O_{132}(M_3)$-$O_{1211}(M_1)$ -$O_{1212}(M_3)$
J_2	$O_{212}(M_5)$-$O_{233}(M_5)$-$O_{2211}(M_5)$ -$O_{2212}(M_2)$-$O_{241}(M_2)$	$O_{211}(M_3)$-$O_{231}(M_4)$-$O_{2211}(M_5)$-$O_{2212}(M_2)$-$O_{241}(M_5)$	$O_{211}(M_3)$-$O_{231}(M_4)$-$O_{2211}(M_5)$ -$O_{2212}(M_2)$-$O_{241}(M_5)$
J_3	$O_{3111}(M_2)$-$O_{3112}(M_1)$-$O_{3113}(M_2)$ -$O_{321}(M_4)$-$O_{333}(M_4)$-$O_{3411}(M_4)$ -$O_{3412}(M_3)$	$O_{3111}(M_2)$-$O_{3112}(M_1)$-$O_{3113}(M_2)$-$O_{321}(M_4)$-$O_{331}(M_3)$-$O_{3411}(M_4)$-$O_{3412}(M_3)$	$O_{3111}(M_2)$-$O_{3112}(M_1)$-$O_{3113}(M_2)$ -$O_{321}(M_4)$-$O_{333}(M_4)$-$O_{3411}(M_4)$ -$O_{3412}(M_3)$
J_4	$O_{441}(M_5)$-$O_{411}(M_5)$-$O_{423}(M_5)$ -$O_{4311}(M_2)$-$O_{4312}(M_1)$-$O_{4313}(M_2)$	$O_{421}(M_4)$-$O_{441}(M_5)$-$O_{411}(M_5)$-$O_{4311}(M_2)$-$O_{4312}(M_1)$-$O_{4313}(M_2)$	$O_{442}(M_5)$-$O_{411}(M_5)$-$O_{423}(M_5)$ -$O_{4311}(M_2)$-$O_{4312}(M_1)$-$O_{4313}(M_2)$
T	71	75	73



(a) Makespan = 27 (b) Makespan = 23 (c) Makespan = 22

Fig. 9 Gantt charts of the schedules (a) NLPP and CLPP, (b) DPP and (c) the proposed IPPS method

5.3 Case 3

As referred before, based on the modified definition and the mathematical model of IPPS, dynamic scheduling can be extended to dynamic IPPS for effectively handling the disruptions occurred in the execution process. In this section, two typical dynamic events, which are machine breakdown and arrival of urgent jobs, are designed to verify the effectiveness of the dynamic IPPS method.

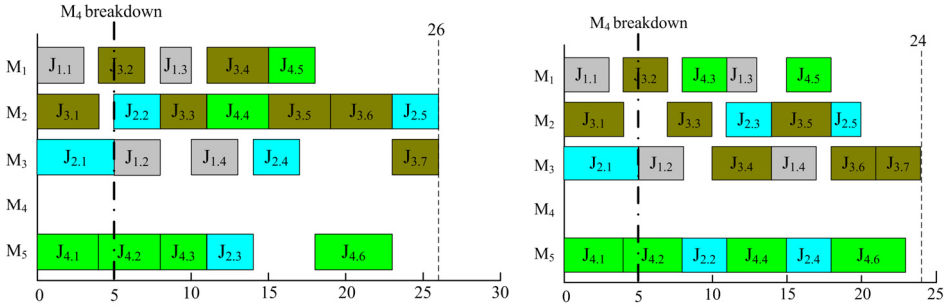
Machine breakdown

The optimal scheduling plan obtained by the proposed IPPS method is released to the shop floor. Suppose that at the time $t = 5$ the machine M_4 is broken-down with a relatively long repair time. Under this condition, traditionally, reactive scheduling methods often transfer the affected oper-

ations to their alternative machines for instead, which is the so called route changing rescheduling [35]. In the route changing rescheduling method, neither the alternative macro-level plan of each feature nor the operation sequence of each job is unchangeable. However, in the dynamic IPPS method, the process plans are also changeable, which may improve the flexibility of rescheduling. The comparative results are shown in Table 6 (process plans) and Fig. 10 (scheduling plans), respectively. From the results we can find that both the total machining time and makespan are reduced by the dynamic IPPS method.

Table 6 Process plans of the route changing rescheduling strategy and the dynamic IPPS method

Job	Route changing rescheduling	Dynamic IPPS method
J_1	$O_{113}(M_1)$-$O_{132}(M_3)$-$O_{1211}(M_1)$-$O_{1212}(M_3)$	$O_{113}(M_1)$-$O_{132}(M_3)$-$O_{1211}(M_1)$-$O_{1222}(M_3)$
J_2	$O_{211}(M_3)$-$O_{231}(M_2)$-$O_{2211}(M_5)$ -$O_{2212}(M_3)$-$O_{241}(M_2)$	$O_{211}(M_3)$-$O_{233}(M_5)$-$O_{241}(M_2)$ -$O_{2211}(M_5)$-$O_{2212}(M_2)$
J_3	$O_{3111}(M_2)$-$O_{3112}(M_1)$-$O_{3113}(M_2)$-$O_{321}(M_1)$-$O_{333}(M_2)$-$O_{3411}(M_2)$-$O_{3412}(M_3)$	$O_{3111}(M_2)$-$O_{3112}(M_1)$-$O_{3113}(M_2)$-$O_{322}(M_3)$ -$O_{3411}(M_2)$-$O_{3412}(M_3)$ -$O_{332}(M_3)$
J_4	$O_{442}(M_5)$-$O_{411}(M_5)$-$O_{423}(M_5)$-$O_{4311}(M_2)$ -$O_{4312}(M_1)$-$O_{4313}(M_5)$	$O_{442}(M_5)$-$O_{411}(M_5)$-$O_{423}(M_1)$-$O_{4311}(M_5)$ -$O_{4312}(M_1)$-$O_{4313}(M_5)$
T	76	74



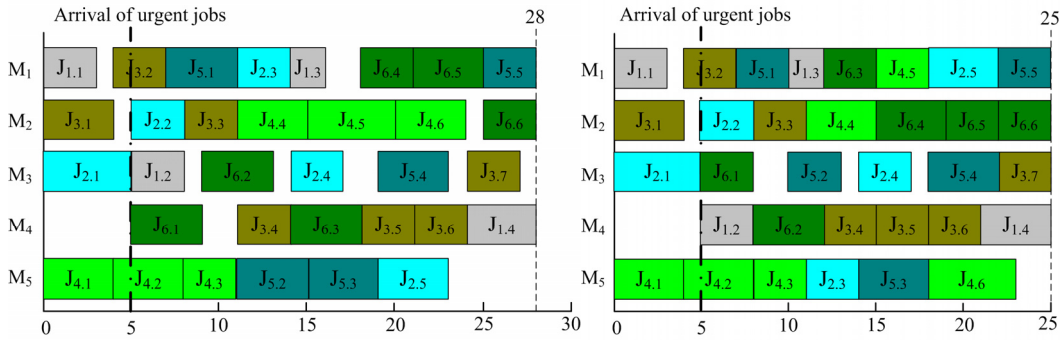
(a) Makespan = 26 (b) Makespan = 24

Fig. 10 Gantt chart of route changing rescheduling method (a) and the dynamic IPPS method (b)

Arrival of urgent jobs

The optimal scheduling plan obtained by the proposed IPPS method is released to the shop floor. Suppose two urgent jobs are released to the shop floor at the moment t = 5. The information of these two jobs is listed in Table 7. Generally, the urgent jobs should be completed as soon as possible. The comparative results are shown in Fig. 11, in which the dynamic IPPS method also outperforms the route changing rescheduling method.

From the above comparative results, it is obvious that the dynamic IPPS method is much more suitable than other rescheduling methods because it can increase the flexibility of rescheduling. Of course, the robustness and the stability of the dynamic IPPS method should be further studied.

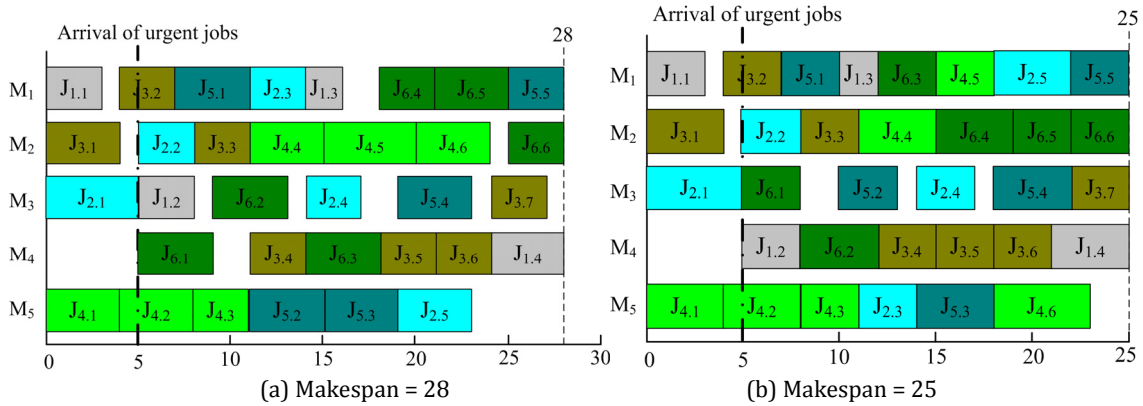


(a) Makespan = 28 (b) Makespan = 25

Fig. 11 Gantt charts of the route changing rescheduling method (a) and the dynamic IPPS method (b)

Table 7 Information of the two urgent jobs

Job	Feature	Alternative macro-level plan	Alternative machines and machining time				
			M_1	M_2	M_3	M_4	M_5
J_5	F_{51}	O_{511}	4	6	999	5	4
	(before F_{52})	O_{512}	5	999	3	4	999
		O_{513}	999	4	5	999	6
	F_{52}	O_{521}	5	6	999	5	4
		O_{522}	999	5	4	5	999
	F_{53}	O_{531}	3	5	4	999	4
		O_{532}	4	999	3	4	999
		O_{533}	999	4	999	5	3
	F_{54}	O_{5411}-O_{5412}	4/3	999/4	4/999	5/4	999/999
		O_{542}	999	6	7	999	7
J_6	F_{61}	O_{611}	4	999	5	4	999
		O_{612}	5	4	999	6	5
	F_{62}	O_{621}	999	5	4	6	5
		O_{622}	4	3	999	5	4
	F_{63}	O_{631}	5	4	5	4	999
	(before F_{64})	O_{632}	4	999	3	999	5
		O_{633}	4	4	5	6	999
	F_{64}	O_{6411}-O_{6412}-O_{6413}	3/4/999	999/4/3	4/999/5	5/4/4	999/999/4



(a) Makespan = 28 (b) Makespan = 25

Fig. 11 Gantt charts of the route changing rescheduling method (a) and the dynamic IPPS method (b)

6. Conclusion

This paper proposes a novel IPPS method which has combined the advantages of the three existing IPPS methods. Based on the modified definition of IPPS, the mathematical model of the proposed IPPS method is presented. This model is also suitable for the dynamic IPPS method, which can greatly improve the flexibility of rescheduling. Furthermore, a discrete version of particle swarm optimization algorithm is put forward to solve the optimization problem of the proposed IPPS method. In the DPSO algorithm, the particles update their positions by crossing with their own historical best positions (*pbests*) and the global best position of the population (*gbest*). Therefore, the continuous PSO algorithm can be used for the combinatory optimization problems such as optimization of the IPPS problems. In order to avoid local convergence, an external archive is introduced to keep more than one elite, and the *gbest* of each particle is randomly selected from the members of the external archive. Finally, the mutation operation is introduced to enhance the local search ability of the DPSO algorithm. As shown in the experiments and results of case 1, the proposed DPSO algorithm stands out from several typical intelligent optimization algorithms for its efficiency and effectiveness. Besides, the PDSO is designed for the proposed IPPS problem, which involves in four tasks as referred before, and the process planning, scheduling or the three existing IPPS methods are partially of the proposed IPPS problem, therefore, the DPSO algorithm could also be used to solve those problems.

In this work, we select the total machining time to judge the process plan and the makespan for scheduling plan, to simplify, we have combined these two objectives into one weighted function. However, other objectives such as total machining cost, total tardiness and utilization of machines may also be considered to complete the proposed IPPS method. It is no doubt that the Pareto-based optimization approaches will make this work more perfect, which will be studied in our future work. Furthermore, the robustness and the stability of the dynamic IPPS method should be further studied as well.

Acknowledgements

This work was supported by Shanxi Foundation Research Projects for Application (201701D221146) and Natural Science Foundation of North University of China (Grant No. 2017003).

References

[1] Phanden, R.K., Jain, A., Verma, R. (2011). Integration of process planning and scheduling: A state-of-the-art review, *International Journal of Computer Integrated Manufacturing*, Vol. 24, No. 6, 517-534, doi: 10.1080/0951192x.2011.562543.

[2] 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*, Vol. 1, No. 3-4, 315-319, doi: 10.1016/0736-5845(84)90020-6.

[3] Yu, M., Zhang, Y., Chen, K., Zhang, D. (2015). Integration of process planning and scheduling using a hybrid GA/PSO algorithm, *The International Journal of Advanced Manufacturing Technology,* Vol. 78, No. 1-4, 583-592, doi: 10.1007/s00170-014-6669-7.

[4] Tan, W., Khoshnevis, B. (2000). Integration of process planning and scheduling – A review, *Journal of Intelligent Manufacturing*, Vol. 11, No. 1, 51-63, doi: 10.1023/a:1008952024606.

[5] Li, X., Gao, L., Zhang, C., Shao, X. (2010). A review on integrated process planning and scheduling, *International Journal of Manufacturing Research*, Vol. 5, No. 2, 161-180, doi: 10.1504/IJMR.2010.031630.

[6] Zhang, H., Liu, S., Moraca, S., Ojstersek, R. (2017). An effective use of hybrid metaheuristics algorithm for job shop scheduling problem, *International Journal of Simulation Modelling*, Vol. 16, No. 4, 644-657, doi: 10.2507/IJSIMM 16(4)7.400.

[7] Huang, X.W., Zhao, X.Y., Ma, X.L. (2014). An improved genetic algorithm for job-shop scheduling problem with process sequence flexibility, *International Journal of Simulation Modelling*, Vol. 13, No. 4, 510-522, doi: 10.2507/IJSIMM13(4)CO20.

[8] 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*, Vol. 47, No. 14, 3775-3796, doi: 10.1080/00207540701827905.

[9] 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*, Vol. 59, No. 1, 166-180, doi: 10.1016/j.cie.2009.09.003.

[10] Zhang, H.-C., Merchant, M.E. (1993). IPPM – A prototype to integrate process planning and job shop scheduling functions, *CIRP Annals*, Vol. 42, No. 1, 513-518, doi: 10.1016/S0007-8506(07)62498-6.

[11] Dai, M., Tang, D., Xu, Y., Li, W. (2014). Energy-aware integrated process planning and scheduling for job shops, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 229, No. 1, 13-26, doi: 10.1177/0954405414553069.

[12] Galzina, V., Lujić, R., Šarić, T. (2012). Adaptive fuzzy particle swarm optimization for flow-shop scheduling problem, *Tehnički vjesnik - Tehnical Gazette,* Vol. 19, No. 1, 151-157.

[13] Chaudhry, I.A., Usman, M. (2017). Integrated process planning and scheduling using genetic algorithms, *Tehnički vjesnik – Technical Gazette*, Vol. 24, No. 5, 1401-1409, doi: 10.17559/TV-20151121212910.

[14] Zhang, L., Wong, T.N. (2015). An object-coding genetic algorithm for integrated process planning and scheduling, *European Journal of Operational Research*, Vol. 244, No. 2, 434-444, doi: 10.1016/j.ejor.2015.01.032.

[15] Wang, S., Lu, X., Li, X.X., Li, W.D. (2015). A systematic approach of process planning and scheduling optimization for sustainable machining, *Journal of Cleaner Production*, Vol. 87, No. 914-929, doi: 10.1016/j.jclepro.2014.10.008.

[16] 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*, Vol. 30, No. 8, 1151-1171, doi: 10.1016/S0305-0548(02)00063-1.

[17] Ho, Y.-C., Moodie, C.L. (1996). Solving cell formation problems in a manufacturing environment with flexible processing and routeing capabilities, *International Journal of Production Research*, Vol. 34, No. 10, 2901-2923, doi: 10.1080/00207549608905065.

[18] Guo, Y.W., Li, W.D., Mileham, A.R., Owen, G.W. (2009). Applications of particle swarm optimisation in integrated process planning and scheduling, *Robotics and Computer-Integrated Manufacturing*, Vol. 25, No. 2, 280-288, doi: 10.1016/j.rcim.2007.12.002.
[19] 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*, Vol. 37, No. 4, 656-667, doi: 10.1016/j.cor.2009.06.008.
[20] Kennedy, J., Eberhart, R. (1995). Particle swarm optimization, In: *Proceedings of ICNN'95 – International Conference on Neural Networks,* Perth, Australia, 1942-1948, doi: 10.1109/ICNN.1995.488968.
[21] Lin, T.-L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., Chen, R.-J., Lai, J.-L., Kuo, I.-H. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization, *Expert Systems with Applications*, Vol. 37, No. 3, 2629-2636, doi: 10.1016/j.eswa.2009.08.015.
[22] Sha, D.Y., Hsu, C.-Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem, *Computers & Industrial Engineering*, Vol. 51, No. 4, 791-808, doi: 10.1016/j.cie.2006.09.002.
[23] Shi, Y., Eberhart, R. (1998). A modified particle swarm optimizer, In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, Anchorage, USA, 69-73, doi: 10.1109/ICEC.1998.699146.
[24] Zhang, G., Gao, L., Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem, *Expert Systems with Applications*, Vol. 38, No. 4, 3563-3573, doi: 10.1016/j.eswa.2010.08.145.
[25] Li, W.D., Ong, S.K., Nee, A.Y.C. (2004). Optimization of process plans using a constraint-based tabu search approach, *International Journal of Production Research*, Vol. 42, No. 10, 1955-1985, doi: 10.1080/00207540310001652897.
[26] Kacem, I., Hammadi, S., Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic, *Mathematics and Computers in Simulation*, Vol. 60, No. 3-5, 245-276, doi: 10.1016/S0378-4754(02)00019-8.
[27] Kacem, I., Hammadi, S., Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* Vol. 32, No. 1, 1-13, doi: 10.1109/tsmcc.2002.1009117.
[28] Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, Vol. 41, No. 3, 157-183, doi: 10.1007/bf02023073.
[29] Li, J.-Q., Pan, Q.-K., Gao, K.-Z. (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems, *The International Journal of Advanced Manufacturing Technology*, Vol. 55, No. 9-12, 1159-1169, doi: 10.1007/s00170-010-3140-2.
[30] Wang, X., Gao, L., Zhang, C., Shao, X. (2010). A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem, *The International Journal of Advanced Manufacturing Technology*, Vol. 51, No. 5-8, 757-767, doi: 10.1007/s00170-010-2642-2.
[31] Xia, W., Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, *Computers & Industrial Engineering*, Vol. 48, No. 2, 409-425, doi: 10.1016/j.cie.2005.01.018.
[32] Moslehi, G., Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search, *International Journal of Production Economics*, Vol. 129, No. 1, 14-22, doi: 10.1016/j.ijpe.2010.08.004.
[33] Bagheri, A., Zandieh, M., Mahdavi, I., Yazdani, M. (2010). An artificial immune algorithm for the flexible job-shop scheduling problem, *Future Generation Computer Systems*, Vol. 26, No. 4, 533-541, doi: 10.1016/j.future.2009.10.004.
[34] Xing, L.-N., Chen, Y.-W., Yang, K.-W. (2009). An efficient search method for multi-objective flexible job shop scheduling problems, *Journal of Intelligent Manufacturing*, Vol. 20, No. 3, 283-293, doi: 10.1007/s10845-008-0216-z.
[35] He, W., Sun, D.-H. (2013). Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies, *The International Journal of Advanced Manufacturing Technology*, Vol. 66, No. 1-4, 501-514, doi: 10.1007/s00170-012-4344-4.