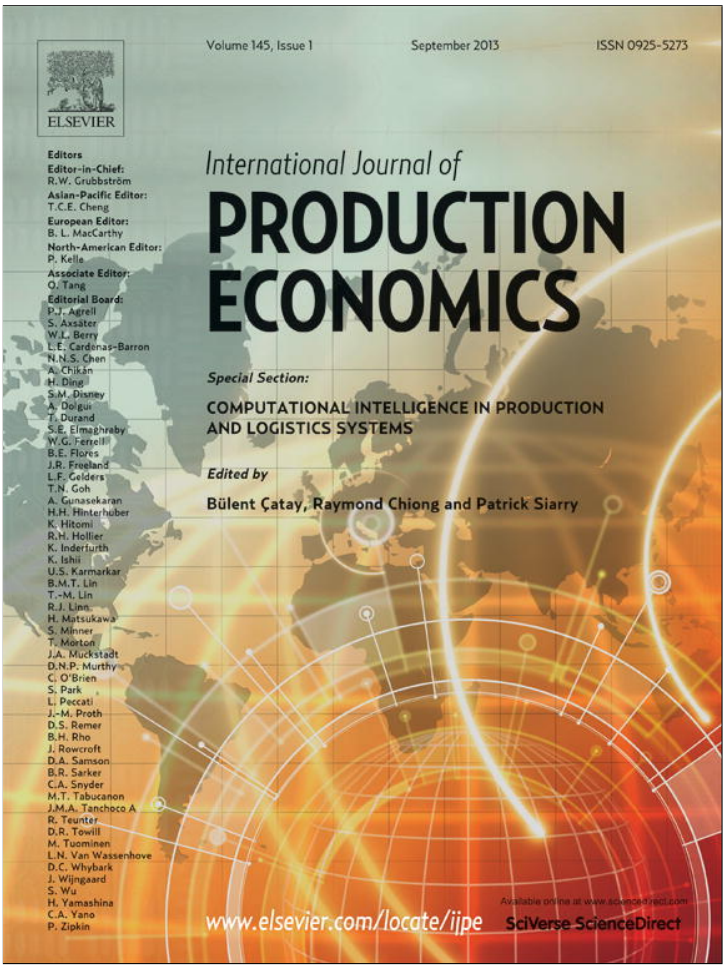

Int. J. Production Economics 145 (2013) 4–17

Contents lists available at SciVerse ScienceDirect

Int. J. Production Economics

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

Chemical-reaction optimization for solving fuzzy job-shop scheduling problem with flexible maintenance activities ☆

Jun-qing Li [a], Quan-ke Pan [a,b,*]

[a] College of Computer Science, Liaocheng University, Liaocheng 252059, PR China
[b] State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, ShenYang 110819, PR China

ARTICLE INFO

Article history:
Received 24 January 2012
Accepted 9 November 2012
Available online 20 November 2012

Keywords:
Fuzzy job-shop scheduling problem
Chemical-reaction optimization
Tabu search
Flexible maintenance activity

ABSTRACT

This paper proposes a hybrid chemical-reaction optimization (HCRO) algorithm for solving the job-shop scheduling problem with fuzzy processing time. The flexible maintenance activities under both resumable and non-resumable situations are also considered to make the problem more close to the reality. In the proposed algorithm, each solution is represented by a chemical molecule. Four elementary reactions, i.e., on-wall ineffective collision, inter-molecular ineffective collision, decomposition, and synthesis, are imposed. A well-designed crossover function is introduced in the synthesis and decomposition operators. In order to balance the exploitation and exploration, HCRO divides the evolution phase into two loop bodies: the first loop body contains on-wall ineffective collision and inter-molecular ineffective collision, while the second loop body includes all the four elementary reactions. Tabu search (TS) based local search is embedded in the proposed algorithm to enhance the convergence capability. A novel decoding approach is utilized to schedule each operation, while considering each flexible preventive maintenance activity on each machine. The proposed algorithm is tested on sets of the well-known benchmark instances. Through the analysis of experimental results, the highly effective performance of the proposed HCRO algorithm is shown against three efficient algorithms from the literature, i.e., SMGA (Sakawa and Mori, 1999), GPSO (Niu et al., 2008), and RKGA (Zheng et al., 2010).

1. Introduction

Recently, job-shop scheduling problem (JSSP) has received more and more research attentions (Armentano and Scrich, 2000). In the classical JSSP, n jobs are to be scheduled on m machines with predefined sequence and constraints. The processing time for each operation on each machine is generally deterministic. However, in most practical industries, the processing time for each operation is just a fuzzy value, because various factors are involved in the real-world problems. This is particularly true in the practical situations when human-centred factors are incorporated into the problems. Kuroda and Wang (1996) proposed a branch-and-bound algorithm for solving both the static and the dynamic JSSP. After that, many researchers have conducted different approaches for solving the fuzzy JSSP (FJSSP). Genetic algorithm (GA) is one of the most popular algorithms which have been utilized for the problem. Sakawa and Mori (1999) designed an efficient GA for solving the FJSSP with fuzzy processing time and fuzzy due date. Again, in 2000, a fuzzy programming based GA was presented by Sakawa and Kubota for the multi-objective FJSSP. Song et al. (2006) developed a hybrid algorithm combining GA, Ant colony optimization (ACO), and a local search approach. Inés et al. (2010) solved the multi-objective JSSP with uncertain durations and crisp due dates by an efficient GA embedded with a fuzzy programming approach. Lei (2010a) developed a random key GA algorithm for the problem. The other swarm intelligent algorithms have also been introduced for solving the FJSSP. Wu et al. (2006) designed an efficient algorithm by combining the fuzzy ranking method and shifting bottleneck procedure. Lei (2007) proposed an algorithm to apply particle swarm optimization (PSO) to solve the FJSSP with three objectives. Niu et al. (2008) introduced a hybrid algorithm combining PSO and GA for the problem. Very recently, a modified differential evolution (DE) algorithm was conducted by Hu et al. (2011) for solving the FJSSP with fuzzy processing time and fuzzy due date.

Most literature considering JSSP assumes that each machine is available in the production horizon. However, in reality, operations may be interrupted by the preventive maintenance (PM) activity on the processing machines. Schmidt (2000) has summarized most results related to deterministic scheduling problems

* Corresponding author.
E-mail address: panquanke@gmail.com (Q.-k. Pan).

☆This research is partially supported by the National Science Foundation of China under Grants 61104179 and 61174187, and the Science Research and Development of Provincial Department of Public Education of Shandong under Grants (J12LN39, J11LG02 and J10LG25).

0925-5273/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2012.11.005

with PM constraints published before 1998. Ma et al. (2010) surveyed the scheduling problems with maintenance activity constraints during very recent years. It shows that most literature considered machine availability constraints in solving single machine problems, parallel machine problems, and flow shop scheduling problems. There are few literature considering the availability constraints in the job-shop scheduling context. For solving the job-shop scheduling problem with deterministic processing time under PM situation, Gao et al. (2006) proposed a hybridization of GA and the local search method for solving the multi-objective flexible job-shop scheduling problems (FJSP) with PM tasks. Zribi et al. (2008) considered the MPM job-shop scheduling problem with maintenance activity constraints. Wang and Yu (2010) investigated a filtered beam search (FBS) based algorithm for FJSPs with PM tasks. For the fuzzy job-shop scheduling problem with PM activities (FJSSP-PM), Lei (2010b) solved the FJSSP under availability constraints with the objective to maximize the minimum agreement index subject to periodic maintenance. Zheng et al. (2010) proposed a random key based GA. Lei (2011) again developed an efficient swarm-based neighborhood search algorithm for the problem. However, the above literature considers PM tasks at a fixed interval in FJSSP context. The FJSSP with PM tasks at a flexible interval are more close to the production reality (Gao et al. 2006; Wang and Yu, 2010), and therefore should be given more research focuses.

Very recently, by simulating the behavior of molecules in chemical reactions, an efficient chemical-reaction optimization (CRO) algorithm was proposed by Lam and Li (2010b) to optimize combinatorial problems. CRO has four elementary reactions, namely, on-wall ineffective collision, inter-molecular ineffective collision, decomposition, and synthesis. The first two reaction operators perform the exploitation function, while the last two reactions complete the exploration tasks. Meanwhile, CRO has a buffer to collect energy produced by on-wall ineffective collision, and help the molecules to escape from the local optima. Based on the above characteristics, CRO is suitable for solving problems with many local optima, such as scheduling problems, and continuous optimization problems. Experimental comparisons demonstrated that the performance of CRO is competitive to other swarm intelligent algorithms (Lam and Li, 2010b; Lam et al., 2010a; Xu et al., 2010; Lam and Li, 2012b). Due to its ability to escape from the local optima, CRO has been applied for solving many scheduling problems, such as peer-to-peer live streaming scheduling (Lam et al., 2010a), resource-constrained project scheduling problem (Lam and Li, 2010b), grid scheduling (Xu et al., 2010), task scheduling in grid computing (Xu et al., 2011), and continuous optimization problems (Lam et al., 2012a).

Although the canonical CRO has many advantages, it should be improved in many aspects, especially its exploitation capability. Therefore, in this study, we propose a hybrid CRO (HCRO) for solving the fuzzy job-shop scheduling problem with flexible maintenance constraints. The main differences between CRO and HCRO are as follows: (1) TS-based local search is embedded in HCRO to enhance the exploitation capability; (2) A well-designed crossover operator is developed in HCRO; (3) HCRO divides the evolution phase into two loop bodies: the first loop body contains two reactions, i.e., on-wall ineffective collision and inter-molecular ineffective collision; the second loop body includes all the four elementary reactions. The evolution phase begins with the first loop body, that is, the exploitation task is firstly been performed. When the exploitation cannot be continued after certain number of generations, the second loop body starts to perform both exploration and exploitation functions. Then, the two loop bodies run alternatively.

The rest of this paper is organized as follows: Section 2 briefly describes the problem. Then, the canonical CRO is presented in Section 3. Section 4 gives the framework of the proposed algorithm. Section 5 illustrates the experimental results and compares to the present performing algorithms from the literature to demonstrate the superiority of the proposed algorithm. Finally, Section 6 gives the concluding remarks and future research direction.

2. Problem descriptions

2.1. Fuzzy number and operations

In this study, the fuzzy processing time is denoted by a triangular fuzzy number (TFN), which is represented by a triplet (similar to Sakawa and Mori, 1999). Given an operation O_{ij}, which should be processed on the machine M_k, then, $\tilde{p}_{ijk}=(s_1, s_2, s_3)$ denotes the fuzzy processing time of O_{ij}.

2.1.1. Fuzzy addition and fuzzy maximum

Given two fuzzy numbers: $\tilde{p}_{ijk}=(s_1, s_2, s_3)$, and $\tilde{p}_{uvh}=(t_1, t_2, t_3)$. The addition of the two triangular fuzzy numbers $\tilde{p}_{ijk}$ and $\tilde{p}_{uvh}$ is shown by the following formula, similar to Sakawa and Mori (1999).

$$\tilde{p}_{ijk}+\tilde{p}_{uvh}=(s_1,s_2,s_3)+(t_1,t_2,t_3)=(s_1+t_1,\ s_2+t_2,\ s_3+t_3). \quad (1)$$

The maximum ($\vee$) of the two triangular fuzzy numbers is computed by the following formula:

$$\tilde{p}_{ijk}\vee\tilde{p}_{uvh}\cong(s_1\vee t_1,\ s_2\vee t_2,\ s_3\vee t_3) \quad (2)$$

For example, given two triplet numbers $\tilde{p}_{ijk}=(3, 4, 5)$ and $\tilde{p}_{uvh}=(1, 5, 6)$. Then, $\tilde{p}_{ijk}+\tilde{p}_{uvh}=(3, 4, 5)+(1, 5, 6)=(4, 9, 11)$; $\tilde{p}_{ijk}\vee\tilde{p}_{uvh}=(3\vee1, 4\vee5, 5\vee6)=(3, 5, 6)$.

2.1.2. The method of ranking the fuzzy time

In this study, the objective of the problem is to minimize the maximum fuzzy completion time. Therefore, the ranking of the fuzzy time is critical in the proposed algorithm. Suppose two fuzzy numbers $\tilde{s}$ and $\tilde{t}$ are represented by triplets (s_1, s_2, s_3) and (t_1, t_2, t_3), respectively, the ranking process is performed according to the following conditions (Sakawa and Mori, 1999):

i) *Condition* 1. If $c_1(\tilde{s})=s_1+2s_2+s_3/4>(<)\ c_1(\tilde{t})=t_1+2t_2+t_3/4$, then $\tilde{s}>(<)\ \tilde{t}$; otherwise, check condition 2.
ii) *Condition* 2. If $c_2(\tilde{s})=s_2>(<)\ c_2(\tilde{t})=t_2$, then $\tilde{s}>(<)\ \tilde{t}$; otherwise, check condition 3.
iii) *Condition* 3. If $c_3(\tilde{s})=(s_3-s_1)>(<)\ c_3(\tilde{t})=(t_3-t_1)$, then $\tilde{s}>(<)\ \tilde{t}$.

For example, suppose four fuzzy numbers, $\tilde{s}_1=(2, 4, 6)$, $\tilde{s}_2=(1, 5, 8)$, $\tilde{s}_3=(3, 4, 5)$, $\tilde{s}_4=(1, 5, 9)$. According to the above ranking method, $c_1(\tilde{s}_1)=4$, $c_1(\tilde{s}_2)=4.75$, $c_1(\tilde{s}_3)=4$, $c_1(\tilde{s}_4)=5$. Therefore, the first two fuzzy number are $\tilde{s}_4$ and $\tilde{s}_2$. According to $c_2(.)$, we cannot decide the sequence of $\tilde{s}_1$ and $\tilde{s}_3$. At last, $c_3(\tilde{s}_1)=4$, $c_3(\tilde{s}_3)=2$. Therefore, the last ranking order of the four fuzzy numbers is $\tilde{s}_4>\tilde{s}_2>\tilde{s}_1>\tilde{s}_3$.

2.2. Problem formulation

In the classical JSSP, there are n jobs to be processed on m machines. Each job consists of m operations, which should be processed in a pre-defined order. Each machine can process only one operation at a time. Each operation can be processed on one machine at a time. If the processing time for each operation on each machine is a fuzzy number rather than a deterministic number, then the problem becomes a FJSSP. If some PM activities occur on at least one machine during the planning horizon, then the problem is a FJSSP-PM. There are two kinds of PM activities, i.e., fixed PM and flexible PM. The fixed PM assumes that each PM activity has a fixed

predefined time interval. The flexible PM assigns a time window for each PM activity. Each PM activity should be scheduled in the given time window, i.e., the PM task should start after the earliest starting time of the time window, and complete before the end of the time window. Therefore, FJSSP with flexible PM activities become harder than the problem with fixed PM tasks. Suppose that the processing of an operation is interrupted by a PM activity on a machine, if the operation can be continued after the completion of the PM activity, the operation is resumable; if the operation has to be started from beginning when the machine becomes available, the operation is non-resumable (Lei, 2011).

In this study, we consider the job-shop scheduling with fuzzy processing time and flexible maintenance activities. In the present literature for solving the FJSSP, fixed PM tasks are considered in (Lei, 2010b, 2011). To the best of our knowledge, there is no work for solving the FJSSP with flexible PM tasks. The problem considered in this study is described as follows.

Let O_{ij} be the jth operation of job J_i; Let $\tilde{p}_{ijk}$ be the fuzzy processing time of O_{ij} on machine M_k. The objective of the problem is to sequence each operation on each machine in order to minimize the maximum fuzzy completion time.

The notation used in this paper is summarized in the following, similar to Schmidt (2000):

- Indices
  i: index of jobs, $i=1,2\ldots,n$;
  k: index of machines, $k=1,2,\ldots,m$;
  j: index of operation sequence, $j=1,2,\ldots, m$;
  l: index of maintenance tasks, $l=1,2,\ldots,L_k$;
- Parameters
  n: total number of jobs;
  m: total number of machines;
  L_k: total number of preventive maintenance tasks of machine M_k;
  PM_l^k: the lth preventive maintenance activity on M_k;
  $\tilde{p}_{ijk}$: fuzzy processing time of O_{ij} on machine M_k;
  $[pm_l^{kS}, pm_l^{kE}]$: time window associated with PM_l^k, where pm_l^{kS} is the earliest starting time, and pm_l^{kE} is the latest completion time;
  d_{kl}: Duration of the maintenance task PM_l^k.
  z_{kl}: Completion time of the maintenance task PM_l^k.
- Decision variables
  $\tilde{c}_{ij}$: fuzzy completion time of O_{ij};

In this paper, the following objective is considered:

$$\min f = \tilde{C}_{\max} = \max_{i=1,2,\ldots n}\{\tilde{c}_{im}\} \quad (3)$$

3. The canonical CRO

CRO was introduced by Lam and Li (2010b), which loosely mimics what happens to molecules in a chemical reaction system and tries to capture the energy in the reaction process. The molecules represent the solutions for the considered problem, which possess two kinds of energies, i.e., potential energy (*PE*) and kinetic energy (*KE*). *PE* corresponds to the objective function of a molecule while *KE* of a molecule symbolized its ability of escaping from a local minimum. CRO is a population based swarm intelligent algorithm. The main difference between CRO and other swarm intelligent algorithms is that CRO is a swarm algorithm with variable population size.

In the canonical CRO, there are four elementary reactions, i.e., the on-wall ineffective collision, decomposition, inter-molecular ineffective collision, and synthesis. These elementary reactions can be categorized into single molecular reactions and multiple molecular reactions. The on-wall ineffective collision and decomposition reactions are single molecular reactions, while the inter-molecular ineffective collision and synthesis reactions are of the latter category.

- The on-wall ineffective collision reaction occurs when a molecule hits the wall and then bounces back. After the on-wall collision, some attributes of the molecule (ω) will change, and thus the molecule becomes a new molecule (ω') if the given condition satisfies. After the on-wall ineffective collision, the molecule ω will lose some percent of *KE* to the *buffer*. By losing kinetic energy to the environment, the molecule can improve its local search ability and enhance the convergence ability.
- The decomposition reaction is used to mimic the process of hitting the wall and then decomposing into two or more pieces. Two situations should be considered for the decomposition reaction: (1) the molecule has enough energy to complete the decomposition; (2) otherwise, the molecule should get energy from the energy buffer.
- The process of two or more molecules to share information with each other and then produce two or more other different molecules is called inter-molecular ineffective collision. The inter-molecular ineffective collision mimics the process that two molecules collide with each other and then bounce away.
- The synthesis is the process when more than one molecule collides and combines together. Suppose two molecules ω_1 and ω_2 collide with each other, and then a new molecule ω' is produced.

4. The proposed HCRO

In this section, we give the detailed implementation of the proposed HCRO algorithm, which includes the encoding and decoding, the well-designed crossover operator, the improved CRO reaction operators, and the framework of the proposed algorithm.

4.1. Encoding

In this study, the job-shop scheduling problem with fuzzy processing time is considered. The solution is represented by a string of integer values (Li et al., 2010; Wang, 2003) in the

Table 1
Fuzzy processing time table.

Job	Machine (fuzzy processing time)			
1	3(8,9,10)	4(6,8,10)	2(7,8,12)	1(3,6,8)
2	4(3,4,5)	1(9,9,11)	2(7,8,11)	3(10,12,14)
3	3(10,10,14)	1(4,5,7)	4(4,7,11)	2(2,3,6)
4	2(9,11,15)	4(3,5,7)	1(10,13,14)	3(10,13,14)

Table 2
Preventive maintenance tasks.

PM tasks		Time window		Duration (d_{kl})
		pm_l^{kS}	pm_l^{kE}	
M_1	PM_{11}	0	7	4
M_2	PM_{21}	1	10	1
M_3	PM_{31}	5	15	2
M_4	PM_{41}	10	18	6

proposed algorithm. The length of the string is equal to $n \times m$, where n and m represent the number of jobs and the number of machines, respectively. In the solution string, each job number occurs exactly m times. Each occurrence of the job number represents the operation sequence of the corresponding job. For example, given a 4 jobs-4 machines problem, the fuzzy processing time for each operation on each given machine is displayed in Table 1, while Table 2 gives the preventive maintenance constraints. Suppose a solution is represented by a string {3, 4, 2, 1, 2, 2, 4, 1, 3, 1, 3, 3, 4, 2, 1, 4}. In the solution string, each integer value denotes the job number. The occurrence time of each job represents the operation number of the corresponding job. The sequence of each job number denotes the scheduling of the corresponding operation. For example, in the given solution string, the first job number is 3, which is the first operation of the job J_3. Then the operation O_{31} will be first scheduled on M_3, with the fuzzy processing time (10,10,14). The following operation to be scheduled is the operation O_{41}, then O_{21}. The last operation in the example solution string is O_{44}, which is the last operation of the job J_4. From the above string, we can conclude the scheduling sequence of all operations, that is, $O_{31} > O_{41} > O_{21} > O_{11} > O_{22} > O_{23} > O_{42} > O_{12} > O_{32} > O_{13} > O_{33} > O_{34} > O_{43} > O_{24} > O_{14} > O_{44}$.

4.2. Decoding with maintenance constraints

It is notable that the solution encoding given above contains no scheduling information for the maintenance tasks. In this subsection, we decode the maintenance activities under both non-resumable and resumable cases.

4.2.1. Non-resumable case

In non-resumable case, if an operation is interrupted with a PM task, the work of the operation before the PM task cannot be remained during the maintenance time window. Therefore, in this case, the operation has to be scheduled to start just after the PM task. The maintenance tasks are scheduled dynamically by using the following steps:

Step 1: Schedule the maintenance tasks on each machine at the end of their time window, that is, set $z_{kl}=pm_l^{kE}$, for each $k \in (1,m)$.
Step 2: When schedule an operation O_{ij} on machine M_k, denote the fuzzy starting time and completion time of O_{ij}, without considering the PM tasks, $(s_{ij}^1, s_{ij}^2, s_{ij}^3)$ and $(e_{ij}^1, e_{ij}^2, e_{ij}^3)$, respectively.
Step 3: If each maintenance task PM_l^k, does not overlap with the operation O_{ij}, then schedule O_{ij}. Otherwise, perform Step 4.
Step 4: If O_{ij} is overlapped with a maintenance task PM_l^k, shift PM_l^k to left as possible as compact, then schedule O_{ij} just after PM_l^k. The new starting time of O_{ij} is $(\max(s_{ij}^1, pm_l^{kS}+d_{kl}), \max(s_{ij}^2, pm_l^{kS}+d_{kl}), \max(s_{ij}^3, pm_l^{kS}+d_{kl}))$.

Fig. 1 gives the Gantt chart for decoding of the example solution explained before.

4.2.2. Resumable case

In resumable case, the work of an interrupted operation can be remained and continued after the machine repair. Therefore, if an operation is interrupted with a PM task, the processing will be divided into two or three phases, i.e., the phase before the PM task (if exist), the phase of the PM task, and the phase after the PM task. Fig. 2 gives all the possible conditions which divide the processing time window into three phases.

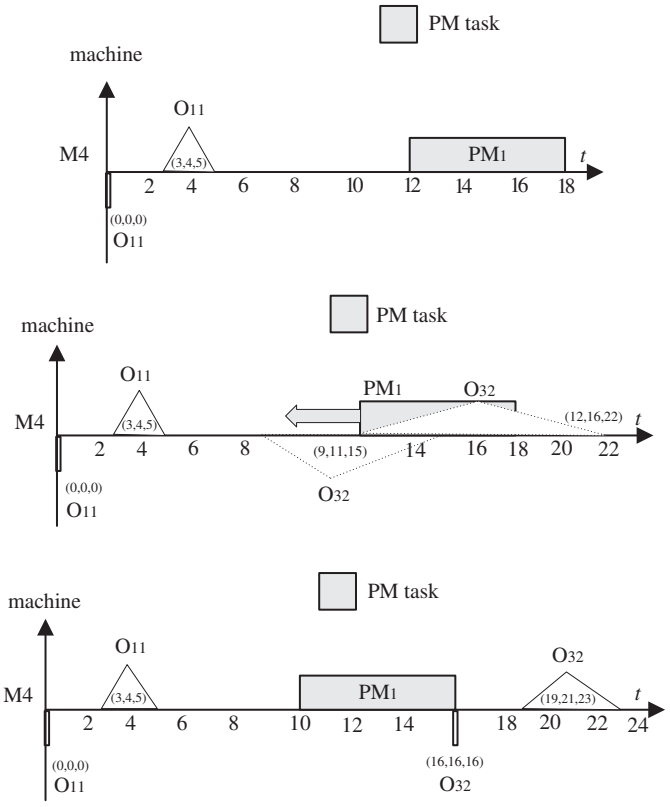

Fig. 1. Gantt chart for the decoding process of the non-resumable case. (a) The PM task is scheduled, (b) Before the operation O_{32} which is overlapped with the PM task (PM_1) is scheduled and (c) After the operation O_{32} is scheduled.

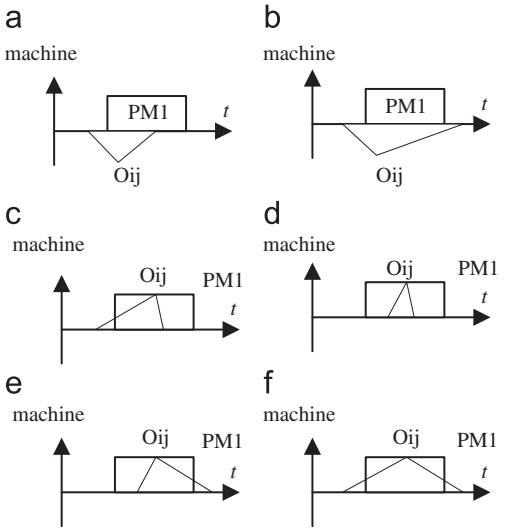

Fig. 2. The possible conditions which divide the processing time window into three phases (resumable case).

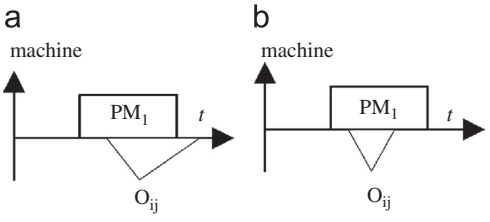

Fig. 3. The possible conditions which divide the processing time window into two phases (resumable case).

In Fig. 2, (a) and (b) tell the status that the fuzzy starting time of an operation O_{ij} is interrupted with the PM task, while (c)–(f) illustrate the truth that the fuzzy completion time of the operation O_{ij} is interrupted with the PM task. Suppose that the fuzzy starting time, completion time, and processing time of O_{ij} is $(s_{ij}^1, s_{ij}^2, s_{ij}^3)$, $(e_{ij}^1, e_{ij}^2, e_{ij}^3)$, and $(p_{ij}^1, p_{ij}^2, p_{ij}^3)$. Shift PM_l^k to right as possible as compact, i.e., $pm_l^{kS} = z_{kl} - d_{kl}$, $pm_l^{kE} = z_{kl}$. The conditions when the operation O_{ij} is interrupted with the PM task can be concluded as follows:

- The condition for (a) and (b) is that $(s_{ij}^1 < pm_l^{kS}) \wedge (s_{ij}^3 > pm_l^{kS})$;
- The condition for (c) and (d) is that $(pm_l^{kS} < e_{ij}^3 \leq pm_l^{kE})$;
- The condition for (b) and (e) is that $(pm_l^{kS} \leq e_{ij}^1 < pm_l^{kE})$;
- The condition for (f) is that $(e_{ij}^1 < pm_l^{kS}) \wedge (e_{ij}^3 > pm_l^{kE})$.

The processing time window will be divided into three phases, which are given as follows:

- The phase before the PM task (similar to Lei (2011) and Zheng et al. (2010)). The starting time and the completion time of this phase are (s_{11}, s_{12}, s_{13}) and (e_{11}, e_{12}, e_{13}), respectively, where $s_{11} = \min(s_{ij}^1, pm_l^{kS})$, $s_{12} = \min(s_{ij}^2, pm_l^{kS})$, $s_{13} = \min(s_{ij}^3, pm_l^{kS})$, $e_{11} = \min(e_{ij}^1, pm_l^{kS})$, $e_{12} = \min(e_{ij}^2, pm_l^{kS})$, $e_{13} = \min(e_{ij}^3, pm_l^{kS})$. Then the work of O_{ij} will be completed partially. The completed processing time of O_{ij} is (c_{11}, c_{12}, c_{13}), where $c_{11} = e_{11} - s_{11}$, $c_{12} = e_{12} - s_{12}$, and $c_{13} = e_{13} - s_{13}$. Then, the remaining

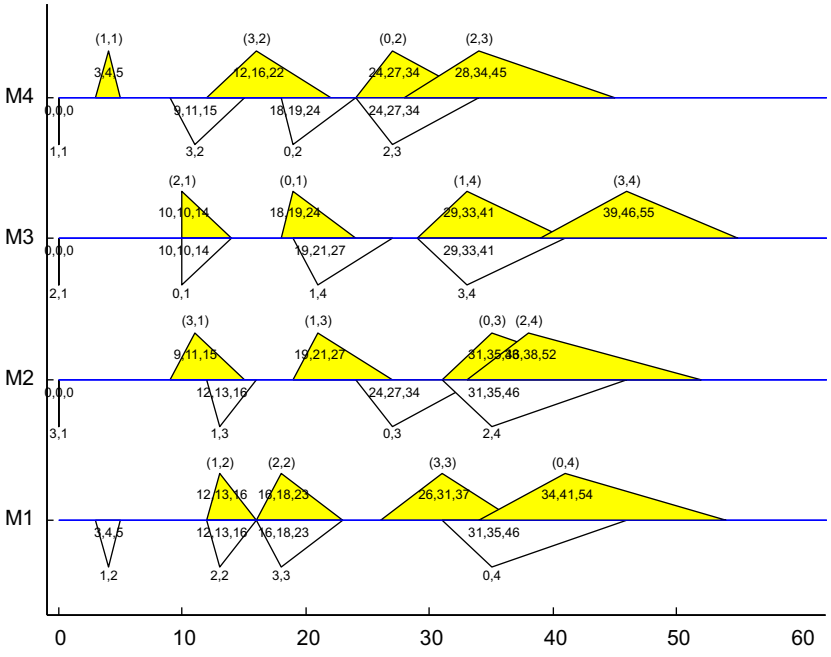

Fig. 4. Fuzzy Gantt chart without considering the maintenance constraints (fuzzy makepsan=(39, 46, 55)).

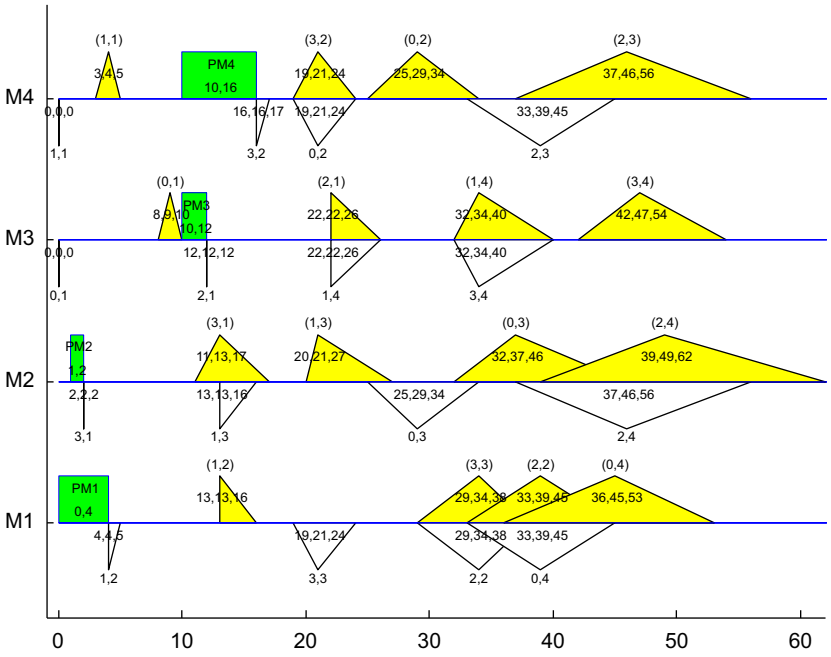

Fig. 5. Fuzzy Gantt chart with maintenance constraints (non-resumable) (fuzzy makepsan=(39, 49, 62)).

processing time of O_{ij} is (r_{11}, r_{12}, r_{13}), where $r_{11}=p_{ij}^{1}-c_{11}$, $r_{12}=p_{ij}^{2}-c_{12}$, and $r_{13}=p_{ij}^{3}-c_{13}$.

- The phase of the PM task. The starting time and the completion time of this phase are (s_{21}, s_{22}, s_{23}) and (e_{21}, e_{22}, e_{23}), respectively, where $s_{21}=s_{22}=s_{23}=pm_l^{kS}$, and $e_{21}=e_{22}=e_{23}=pm_l^{kS}+d_{kl}$.
- The phase after the PM task. The starting time and the completion time of this phase are (s_{31}, s_{32}, s_{33}) and (e_{31}, e_{32}, e_{33}), respectively, where $s_{31}=\max(e_{ij}^{1}, e_{21})$, $s_{32}=\max(e_{ij}^{2}, e_{22})$, $s_{33}=\max(e_{ij}^{3}, e_{23})$, $e_{31}=s_{31}+r_{11}$, $e_{32}=s_{32}+r_{12}$, and $e_{33}=s_{33}+r_{13}$.

Fig. 3 gives all the possible conditions which divide the processing time window into two phases. In this case, the condition is $(s_{ij}^{1}\geq pm_l^{kS})\wedge(s_{ij}^{1}<pm_l^{kE})$. Then, the two phases can be described as follows:

- The phase of the PM task. The starting time and the completion time of this phase are (s_{11}, s_{12}, s_{13}) and (e_{11}, e_{12}, e_{13}), respectively, where $s_{11}=s_{12}=s_{13}=pm_l^{kS}$, and $e_{11}=e_{12}=e_{13}=pm_l^{kS}+d_{kl}$.
- The phase after the PM task. The starting time and the completion time of this phase are (s_{21}, s_{22}, s_{23}) and (e_{21}, e_{22}, e_{23}), respectively, where $s_{21}=\max(e_{ij}^{1}, e_{11})$, $s_{22}=\max(e_{ij}^{2}, e_{12})$, $s_{23}=\max(e_{ij}^{3}, e_{13})$, $e_{21}=s_{21}+p_{ij}^{1}$, $e_{22}=s_{22}+p_{ij}^{2}$, and $e_{23}=s_{23}+p_{ij}^{3}$.

For the given example problem in Tables 1 and 2, the fuzzy Gantt chart without considering the maintenance constraints is given in Fig. 4. Fig. 5 gives the fuzzy Gantt chart for the same problem with maintenance constraints under non-resumable situation, while Fig. 6 gives the Gantt chart under resumable situation. In the three figures, the fuzzy starting time and fuzzy completion time of each operation are given in the Gantt chart. Both starting time and completion time are represented by a triangle, respectively. The fuzzy starting time of each operation is given below the base line for each machine, while the fuzzy completion time is placed up the base line (Lei, 2010b). For example, in Fig. 4, on the machine M_4, O_{11} starts at (0,0,0) and completes at (3,4,5), while O_{32} starts at (9,11,15) and completes at (12,16,22). The last operation to be scheduled in Fig. 4 is O_{34}, and its fuzzy completion time is {39, 46, 55}.

For FJSSP with maintenance constraints under non-resumable situation, Fig. 5 gives the fuzzy starting time and fuzzy completion time for each operation. The maintenance activities for each machine are also given in Fig. 5. For example, in Fig. 5, the preventive maintenance PM_4 starts from 10 and ends at 16 for machine M_4. Without considering PM_4, O_{32} may start at (11,13,17), which is the completion time of O_{31}. However, in this case, the operation O_{32} is interrupted with PM_4. So, the new fuzzy starting time of O_{32} is (16,16,17). The other three PM tasks are interrupted with O_{21}, O_{31}, and O_{12}, respectively. The fuzzy makespan for the problem under non-resumable situation given in Fig. 5 is (39,49,62), which is obviously greater than the fuzzy makespan in Fig. 4.

Fig. 6 gives the fuzzy starting time and fuzzy completion time for each operation under resumable situation. From Fig. 6, we can see that if an operation is interrupted with a PM task, the remaining work of the operation can be restarted after the completion of the PM task. For example, O_{21} is interrupted with the PM task on M_3. Then, the processing duration of O_{21} is divided into two phases, i.e., the phase before the PM task and the phase after the PM task. In Fig. 6, O_{21} starts from (0,0,0) and ends at (5,5,5) in the first phase, while starts from (7,7,7) and ends at (12,12,16) in the second phase. The fuzzy makespan for the problem under resumable situation given in Fig. 6 is (40,46,55), which is slightly greater than the makespan in Fig. 4.

4.3. Crossover operator

Like other swarm intelligent algorithm, such as GA and PSO, CRO also has several operators to generate new solutions by learning information from two or more solutions. In the canonical PSO, each particle corresponds to a solution in the current population. Each particle learns information from both the local best and the global best to direct the search process to optimal spaces. Therefore, in this study, similar to PSO, we propose a novel crossover function named A-LOX (Non-ABEL and Linear Order Crossover), which combines the two commonly used crossover approaches, i.e., the linear order crossover (LOX) and the Non-ABEL approach (Wang, 2003). In the

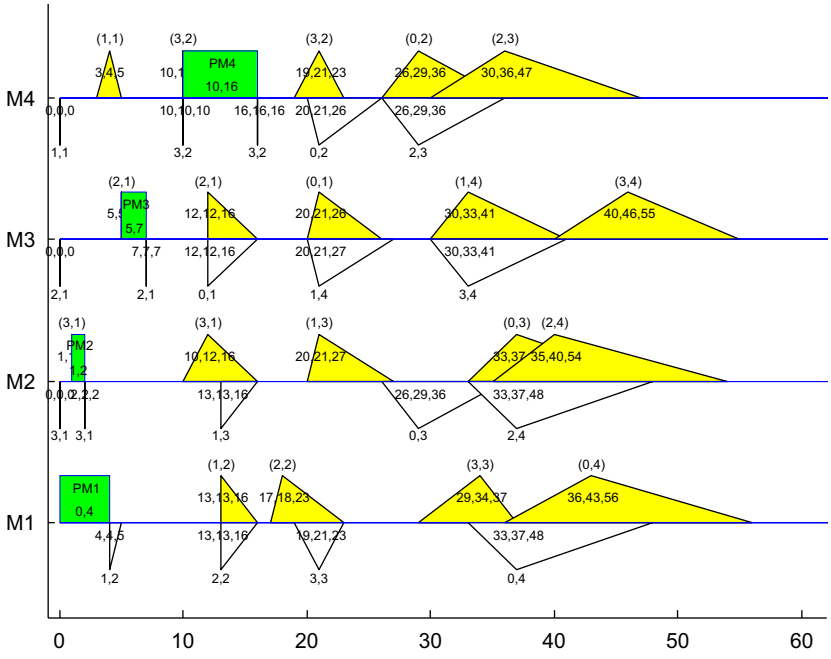

Fig. 6. Fuzzy Gantt chart with maintenance constraints (resumable) (fuzzy makepsan=(40, 46, 55)).

proposed algorithm, A-LOX is applied in two elementary collisions, i.e., synthesis and decomposition.

Suppose that the two parent solutions are p_1 and p_2; the new produced child solutions are c_1 and c_2; the local best for p_1 and p_2 are b_1 and b_2, respectively; the global best is g_s. Each child solution is divided into three parts. The first part is to learn information from the local best with possibility q_1, the second part is copied from the other parent solution, while the last part is to study from the global best with possibility q_2. The detailed crossover operator A-LOX is given in Fig. 7.

From the pseudo code in Fig. 7, we can conclude that the proposed A-LOX function holds several advantages as follows. (1) By learning information from both the local best and the global best, the A-LOX operator will direct the search process to optimal

Procedure *A-LOX* ()

Input: two molecules p_1 and p_2, two crossover possability with the best solutions q_1 and q_2.

Output: two new molecules c_1 and c_2.

Begin

Step 1. Randomly generate two numbers h_1 and h_2, where h_1 and h_2 are between [0, 1]. If $h_1 \le q_1$, then $pa_{11}=b_1$, otherwise, $pa_{11}=p_1$. If $h_2 \le q_2$, then $pa_{12}=g_s$, otherwise, $pa_{12}=p_1$. If $h_1 \le q_1$, then $pa_{21}=b_2$, otherwise, $pa_{21}=p_2$. If $h_2 \le q_2$, then $pa_{22}=g_s$, otherwise, $pa_{22}=p_2$.

Step 2. Randomly generate two positions r_1 and r_2, where $r_1 < r_2$;

Step 3. Copy the elments between $[r_1, r_2]$ from p_1 and p_2 into the corresponding position of c_2 and c_1, respectively;

Step 4. Delete the elements which have occured in c_1 from pa_{11}. Delete the elements which have occured in c_2 from pa_{21}.

Step 5. Let i=1. Perform steps 6 to 12 until there is none element in pa_{12}.

Step 6. If $i<r_1$, Let $r=p_1$[i] mod $len(pa_{11})$, where $len(pa_{11})$ is the length of pa_{11}. Let $t[i]= pa_{11}[r]$. Otherwise, go to step 8.

Step 7. Delete the element at position r from pa_{11}, repleace i with i+1, go back to step 6.

Step 8. Insert each element in t into the empty position in c_1. Delete all elements in t.

Step 9. Let $i=r_2$+1. Delete the elements which have occured in c_1 from pa_{12}.

Step 10. Let $r=p_1$[i] mod $len(pa_{12})$. Let $t[i]= pa_{12}[r]$.

Step 11. Delete the element at position r from pa_{12}, repleace i with i+1, go back to step 10.

Step 12. Insert each element in t into the empty position in c_1. Delete all elements in t.

Step 13. Replace pa_{11} and pa_{12} with pa_{21} and pa_{22}, respectively. Produce c_2 by performing steps 5 to 12.

End

Fig. 7. Pseudo code of A-LOX function.

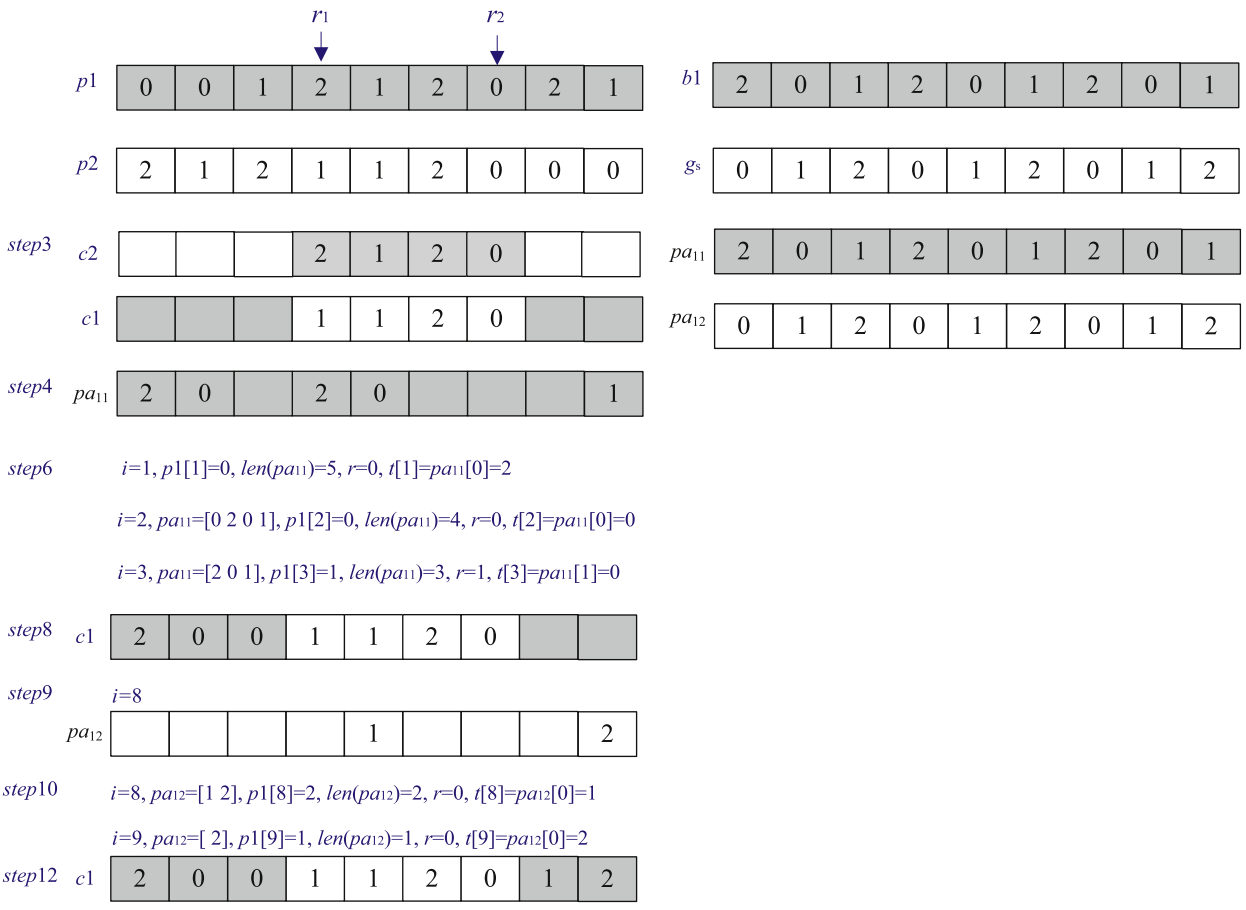

Fig. 8. The example process of A-LOX.

spaces with fast speed. (2) In learning from the two best solutions, A-LOX arranges remaining elements according to their order in the local best or the global best with slightly perturbation. This feature gives the searching process the capability to escape from the local optima. (3) Like the classical Non-ABEL method, the computational process of the proposed A-LOX is simple and easy of implement.

Given a 3 jobs-3 machines problem, suppose the two parent solutions to be crossover with each other are p_1=[0 0 1 2 1 2 0 2 1] and p_2=[2 1 2 1 1 2 0 0 0], respectively; the two randomly generated positions are r_1=4 and r_2=7, respectively; the two crossover possibility with the best solutions are q_1=0.3 and q_2=0.5, respectively; the randomly generated two numbers h_1=0.2 and h_2=0.4, respectively; the local best b_1 for p_1 is [2 0 1 2 0 1 2 0 1], while the global best g_s is [0 1 2 0 1 2 0 1 2]. Then the process of generating the child solution c_1 is given in Fig. 8.

4.4. Improved elementary collisions

In the canonical CRO, the four elementary reactions can be divided into two sets: the first set includes on-wall ineffective collision and inter-molecular ineffective collision, while the second set includes decomposition and synthesis. The first set completes the exploitation task, while the second set performs the exploration function. In this study, we implemented the four elementary reactions for solving the FJSSPs as follows.

4.4.1. On-wall ineffective collision

In the canonical CRO, like the mutation operator in the genetic algorithm (GA), the on-wall ineffective collision is to generate a new neighboring molecule around a given one. In this study, several neighboring approaches are used in the on-wall ineffective collision function to make the algorithm with high level of expolitation capability. Given a molecule ω, randomly generate two positions r_1 and r_2, where $1 \le r_1 < r_2 \le n \times m$, then, the neighboring approaches are described as follows:

- Reverse approach. To reverse each element between r_1 and r_2;
- Swap approach. To swap the two elements at r_1 and r_2;
- Insert approach. To insert the element at r_2 before the element at r_1.

4.4.2. Inter-molecular ineffective collision

The inter-molecular ineffective collision occurs when two molecules collide and then produce two new molecules. Similar to the canonical CRO, in this study, the inter-molecular ineffective collision is realized by performing independently on-wall ineffective collision for the two selected molecules. That is, in parallel, the two molecules perform a slight change in their structure.

4.4.3. Decomposition

The decomposition reaction is to produce two or more molecules based on one molecule. In the proposed algorithm, the decomposition is realized as follows: firstly, randomly select a molecule from the current population, and randomly generate a molecule; secondly, apply the crossover operator on the two selected molecules.

4.4.4. Synthesis

The synthesis is used to produce one molecule by combining two molecules. In the proposed algorithm, the crossover function is also embedded in the synthesis process. The synthesis reaction is realized as follows: firstly, randomly select two molecules from the current population; secondly, apply the crossover operator on the two selected molecules; thirdly, select the better one between the two child molecules as the new molecule.

4.5. TS-based local search

The TS-based local search is introduced in the proposed algorithm to enhance the convergence ability of HCRO. The detailed steps of the TS-based local search are given as follows:

Step 1. Let the current solution be S_c.
Step 2. If the stop criterion satisfies, then stop the algorithm; otherwise, perform steps 3 to 7.
Step 3. Let $m=0$
Step 4. For the current solution S_c, generate S_n neighboring solutions to construct a neighbor set. Evaluate each neighboring solution, and sequence each neighboring solution according to the fitness value in a non-decreasing order.
Step 5. Select the new current solution $S_{c'}$ by the following rules: (1) the first neighboring solution which is not tabooed; (2) if all solutions are tabooed, select the first solution which satisfies the aspiration rule.
Step 6. If $S_{c'}$ is better than S_c, then replace S_c by $S_{c'}$, let $m=0$; otherwise, let $m=m+1$.
Step 7. If m is greater than I_{iter}, then go back to step 2; otherwise go back to step 4.

4.6. The framework of the HCRO algorithm

The detailed steps of the proposed HCRO algorithm are as follows:

Step 1: Initialization phase
  Step 1.1 Set the population size P_{size};
  Step 1.2 Initialize the population, let the central energy *buffer* equals 0.
Step 2: Evaluate the *PE* value of each molecule in the population.
Step 3: Perform the first loop body.
  Step 3.1 If the stopping criterion is satisfied, output the best solution; otherwise, perform sub-steps 3.2 to 3.6.
  Step 3.2 Get r randomly in interval [0, 1], if $r > MoleColl$, then perform sub-step 3.3, otherwise, perform sub-step 3.4.
  Step 3.3 Select a molecule w from the population randomly, and apply the on-wall ineffective collision on it.
  Step 3.4 Randomly select two molecules from the population, and apply the inter-molecular ineffective collision on them.
  Step 3.5 Evaluate the new generated molecules, memorize the local best for each molecule and the global best for the current population, and perform the TS-based local search function on the global best molecule.
  Step 3.6 Evaluate the new generated molecules, if the best solution has been updated in the latest G_{max} generations, then go back to sub-step 3.1, otherwise, go to step 4.
Step 4**:** Perform the second loop body.
  Step 4.1 If the stopping criterion is satisfied, output the best solution; otherwise, perform sub-steps 4.2 to 4.6.
  Step 4.2 Get r randomly in interval [0, 1], if $r > MoleColl$, then perform sub-step 4.3, otherwise, perform sub-step 4.4.
  Step 4.3 Select a molecule w from the population randomly, and apply the on-wall ineffective collision or decomposition on it based on the decomposition condition.
  Step 4.4 Randomly select two molecules from the population, and apply the inter-molecular ineffective collision or synthesis on them based on the synthesis condition.
  Step 4.5 Evaluate the new generated molecules, memorize the local best for each molecule and the global best for the current population, and perform the TS-based local search function on the global best molecule.

Step 4.6 If the best solution has been updated in the latest $G_{\max}$ generations, then go back to sub-step 4.1, otherwise, go back to step 3.

5. Numerical results

This section discusses the computational experiments used to evaluate the performance of the proposed algorithm. Our algorithm was implemented in C++ on an Intel Core i5 3.3 GHz PC with 4 GB memory. The compared algorithms include SMGA (Sakawa and Mori, 1999), GPSO (Niu et al., 2008), and RKGA (Zheng et al., 2010). Two sets of benchmarks are tested, i.e., the four 10 jobs-10 machines problems (Sakawa and Mori, 1999; Sakawa and Kubota, 2000), the extension version of 12 10 jobs-10 machines problems (ABZ5-6, ORB01-05, and LA16-20), and four 15 jobs-10 machines problems (LA21-24). For the first problem set with fixed PM tasks, we compared with the experimental results from Zheng et al. (2010) and Lei (2010b). We coded SMGA, GPSO, and RKGA for all benchmarks with flexible PM tasks. For SMGA, GPSO, and RKGA, we adopt the parameter settings proposed by Sakawa and Mori (1999); Niu et al. (2008) and Zheng et al. (2010), respectively, except the computational times for each instance is set 100 s. The best and average results of experiments for the above 20 problems from 20 independent runs were collected for performance comparisons.

5.1. Experimental parameters

The parameter values are given as follows:

- Initial population size P_{size}: 50;
- Maximum generations without updating the best solution G_{max}: 1000;
- *KELossRate*: 0.2;
- *MoleColl*: 0.5;
- Initial *KE* for each molecule: 100,000;
- Crossover possibility: $q_1=q_2=0.5$;
- The parameters for the TS based local search are as follows: (1) tabu list size: $n\times m/2$; (2) tabu tenure: $n\times m/2$; (3) the maximum generations without improvement to stop the local search I_{iter}: $n\times m$; (4) $S_n=10$;
- Stop condition: the maximum computational time exceeds 100 s.

5.2. Experimental results on the four benchmarks from Sakawa and Mori (1999); Sakawa and Kubota (2000)

Four benchmarks with 10 jobs-10 machines scale (Sakawa and Mori, 1999; Sakawa and Kubota, 2000) are conducted by the proposed algorithm. Table 3 gives the fixed PM tasks for each machine in the above four benchmarks. In Table 3, each PM task is represented by a pair of integer numbers, which tells the start time and the end time of the PM task. For example, for the instance Case 1 in Table 3, the PM task on M_1 is indicated by (20,25), which tells that the PM task will start at time point 20, while end at time point 25.

5.2.1. Experimental results on the four benchmarks with fixed PM tasks

Table 4 gives the comparisons of the experimental results on the four benchmarks under non-resumable situation. In Table 4, there are eight columns. The first column tells the benchmark name, while the second column gives the name of the compared algorithm. The following column reports the average values of the fuzzy makespan after 20 independent runs. Then, the fourth column illustrates the $c_1(.)$ objective value of the average fuzzy makespan for each corresponding algorithm, where $c_1(.)$ is computed as Section 2.1.2. The following two columns report the optimal fuzzy makespan and its $c_1(.)$ objective value, respectively. The last two columns display the worst fuzzy makespan and its $c_1(.)$ objective value, respectively.

Table 3
Preventive maintenance activities for the four 10 jobs-10 machine benchmarks.

Case	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	(20,25)	(10,15)	(23,27)	(16,21)	(0,0)	(13,17)	(34,39)	(0,0)	(28,32)	(0,0)
2	(44,50)	(33,41)	(75,83)	(12,19)	(63,69)	(10,18)	(31,39)	(52,59)	(23,29)	(59,69)
3	(42,52)	(21,34)	(70,81)	(12,20)	(63,73)	(57,69)	(31,42)	(55,64)	(85,96)	(25,33)
4	(39,52)	(40,49)	(23,31)	(75,82)	(63,71)	(54,65)	(31,39)	(87,95)	(19,30)	(50,60)

Table 4
Comparisons of experimental results on the four fixed PM benchmarks (non-resumable).

Case	Algorithm	*avg*	c_1	*opt*	c_1	*wor*	c_1
1	SMGA	50.8, 58.9, 65.8	58.60	49,58,66	57.75	51,59,66	58.75
	GPSO	48.9, 58.5, 65.9	57.95	47,57,66	56.75	**51,59,65**	**58.50**
	RKGA	47.5, 57.7, 66.2	57.28	**41,56,70**	**55.75**	**51,59,65**	**58.50**
	HCRO	**47.3**, **57.6**, **65.9**	**57.10**	47,57,66	56.75	**51,59,65**	**58.50**
2	SMGA	128.8,152.8,177.3	152.93	128,152,177	152.25	132,154,176	154.00
	GPSO	127.2,152.1,175.8	151.80	131,151,169	150.50	122,153,183	152.75
	RKGA	129.5,151.5,173.4	151.48	131,151,169	150.50	128,152,177	152.25
	HCRO	**129.8,150.7,171.0**	**150.55**	**129,150,171**	**150.00**	**131,151,171**	**151.00**
3	SMGA	130.4,152.8,173.9	152.48	135,151,164	150.25	138,156,169	154.75
	GPSO	129.6,152.4,173.8	152.05	131,150,166	149.25	135,157,174	155.75
	RKGA	128.2,150.2,172.1	150.18	**116,146,179**	**146.75**	134,151,176	153.00
	HCRO	**132.2,151.0,164.7**	**149.73**	132,149,158	147.00	**136,153,163**	**151.25**
4	SMGA	118.2,138.4,158.4	138.35	115,139,157	137.50	123,139,155	139.00
	GPSO	116.5,136.9,159.5	137.45	116,135,160	136.50	123,139,155	139.00
	RKGA	**117.1,136.5,157.0**	**136.78**	**124,135,148**	**135.50**	126,139,150	138.50
	HCRO	123.1,135.4,154.1	137.00	**124,135,148**	**135.50**	**124,135,158**	**138.00**

From Table 4, we can see that: (1) the proposed HCRO algorithm is better than the other three algorithms for solving Case 2; (2) the optimal results obtained by HCRO is slightly worse than RKGA for solving Case 1, while with the same worst results with RKGA and GPSO; (3) for solving Case 4, our algorithm obtains the same optimal result with RKGA, which is better than the other two algorithms, i.e., SMGA and GPSO. In addition, the worst result due to our algorithm is the best among the four algorithms; (4) the worst results obtained by our algorithm for the four benchmarks are the best among the four algorithms, which verify the efficiency of HCRO; (5) the average experimental results of our algorithm are optimal or near-optimal, which guarantee the robustness of HCRO.

The comparisons of the experimental results on the four benchmarks under resumable situation are given in Table 5. It can be seen from Table 5 that: (1) the proposed HCRO algorithm shows the best performance in average, optimal, and the worst case for solving the four given benchmarks; (2) the average values obtained by HCRO are obviously better than the other three algorithms for solving the three benchmarks, i.e., Case 2, Case 3, and Case 4, which verify the robustness of the proposed algorithm; (3) the optimal solutions obtain by HCRO for solving Case 2, Case 3, and Case 4 are also obviously better than SMGA, GPSO, and RKGA, which guarantee the exploitation capability of the proposed algorithm; (4) the worst results obtained by our algorithm for the last three benchmarks are greatly better than the other three algorithms, which shows the efficiency of HCRO.

5.2.2. Experimental results on the four benchmarks with flexible PM tasks

To make the problem more close to the reality, we realized the proposed algorithm to solve the FJSSP with flexible PM activities. The same four benchmarks with 10 jobs-10 machines scale are conducted by the proposed algorithm. The flexibility is realized as follows. Firstly, denote the time window of the each PM task as $[s_i, e_i]$, where i represents the PM task for M_i; secondly, let s_i and e_i be the same value as given in the fixed benchmark. For example, in Table 3, the time window for the PM task on M_1 is [20,25]; thirdly, expand the time window by the following steps:

Step 1. Let $pt=(e_i-s_i)$

Step 2. Let $\gamma=pt/2+\omega$

Step 3. Let $s_i=\begin{cases} s_i-\gamma, & \text{if } (s_i-\gamma)>0 \\ 0, & \text{otherwise} \end{cases}$

Step 4. Let $e_i=e_i+\gamma$

where ω is a random number ranges [0, 5].

Table 6 gives the comparisons of the experimental results on the four benchmarks with flexible PM activities under non-resumable

Table 5
Comparisons of experimental results on the four fixed PM benchmarks (resumable).

Case	Algorithm	*avg*	c_1	*opt*	c_1	*wor*	c_1
1	SMGA	47.5, 56.6, 64.6	56.33	46, 56, 66	56.00	50,57,63	56.75
	GPSO	46.4, 55.9, 64.7	55.73	33, 54, 74	53.75	50,57,62	56.50
	RKGA	45.1, 55.3, 65.3	55.25	33, 54, 74	53.75	50,57,62	56.50
	HCRO	**47.3**, **54.9**, **63.3**	**55.10**	**45**, **52**, **64**	**53.25**	**50**,**56**,**63**	**56.25**
2	SMGA	126.5, 149.6, 172.5	149.55	121, 147, 175	147.50	128, 151, 171	150.25
	GPSO	124.7, 148.8, 171.2	148.38	117, 147, 171	145.50	128, 151, 171	150.25
	RKGA	126.1, 149.3, 170.6	148.83	120, 144, 170	144.50	128, 151, 171	150.25
	HCRO	**121.5**, **141.3**, **165.1**	**142.30**	**123**, **141**, **161**	**141.50**	**120**, **142**, **168**	**143.00**
3	SMGA	126.5, 148.8, 172.1	149.05	116, 146, 176	146.00	134, 152, 163	150.25
	GPSO	130.4, 148.3, 164.7	147.93	129, 145, 156	143.75	136, 155, 170	154.00
	RKGA	126.8, 146.5, 164.6	146.10	129, 145, 156	143.75	132, 149, 164	148.50
	HCRO	**127.2**, **145.0**, **159.1**	**144.08**	**122**, **143**, **164**	**143.00**	**128**, **147**, **158**	**145.00**
4	SMGA	118.3, 135.4, 155.5	136.15	108, 133, 160	133.50	125, 139, 161	141.00
	GPSO	113.2, 134.9, 157.1	135.03	109, 134, 153	132.50	125, 139, 158	140.25
	RKGA	113.6, 135.2, 157.6	135.40	109, 135, 156	133.75	125, 139, 161	141.00
	HCRO	**112.6**, **130.0**, **151.9**	**131.13**	**109**, **129**, **155**	**130.50**	**113**, **130**, **153**	**131.50**

Table 6
Comparisons of experimental results on the four flexible PM benchmarks (non-resumable).

Case	Algorithm	*avg*	c_1	*opt*	c_1	*wor*	c_1
1	SMGA	48.4, 56.3, 62.6	55.90	48, 55, 60	54.50	52, 59, 65	58.75
	GPSO	47.2, 54.8, 61.8	54.65	45, 54, 63	54.00	50, 57, 63	56.75
	RKGA	46.9, 54.6, 61.2	54.33	45, 54, 63	54.00	45, 55, 63	54.50
	HCRO	**45.3**, **54.0**, **62.6**	**53.98**	**46**, **54**, **61**	**53.75**	**45**, **54**, **63**	**54.00**
2	SMGA	129.0, 150.0, 170.0	149.75	124, 147, 167	146.25	136, 156, 178	156.50
	GPSO	127.2, 146.8, 165.5	146.58	128, 147, 164	146.50	128, 148, 167	147.75
	RKGA	126.5, 147.1, 167.6	147.08	127, 146, 167	146.50	126, 149, 166	147.50
	HCRO	**124.6**, **146.6**, **167.0**	**146.20**	**125**, **146**, **167**	**146.00**	**124**, **147**, **167**	**146.25**
3	SMGA	132.9, 152.0, 164.8	150.43	127, 143, 155	142.00	143, 163, 174	160.75
	GPSO	128.5, 146.4, 159.7	145.25	120, 142, 161	141.25	135, 156, 168	153.75
	RKGA	127.6, 144.3, 155.8	143.00	124, 142, 156	141.00	132, 150, 164	149.00
	HCRO	**125.6**, **143.2**, **156.1**	**142.03**	**126**, **142**, **151**	**140.25**	**126**, **144**, **158**	**143.00**
4	SMGA	123.7, 136.7, 151.2	137.08	117, 132, 150	132.75	130, 142, 158	143.00
	GPSO	120.2, 132.8, 147.0	133.20	119, 131, 143	131.00	127, 138, 151	138.50
	RKGA	119.5, 132.2, 146.8	132.68	116, 131, 145	130.75	123, 136, 147	135.50
	HCRO	**114.9**, **130.3**, **144.1**	**129.90**	**114**, **130**, **144**	**129.50**	**118**, **132**, **146**	**132.00**

situation. From Table 6, we can conclude that: (1) the proposed HCRO algorithm is better than the other three algorithms for the four benchmarks in finding optimal solutions, which shows the effectiveness of the proposed algorithm; (2) HCRO can obtain better average values than RKGA, GPSO and SMGA, for solving all the four benchmarks, which verifies the efficiency of the proposed algorithm; (3) in comparison of the worst results of the four algorithms, HCRO shows the best performance.

The comparisons of the experimental results on the four benchmarks with flexible PM activities under resumable situation are given in Table 7. It can be seen from Table 7 that: (1) the proposed HCRO algorithm shows the best performance in average, optimal, and the worst case for solving the two benchmarks, i.e., Case 2 and Case 3; (2) the optimal results for the four benchmarks due to our algorithms are the best, which shows the searching ability of HCRO; (3) the average results and the worst results obtained by HCRO are also the best among the four algorithms, which verify the robustness of the proposed algorithm.

5.3. Experimental results on the 16 benchmarks with flexible PM activities

In this section, 16 instances are used, which are the extension version the classical JSSP instances, i.e., 10 jobs-10 machines problems (ABZ5-6, ORB01-05, and LA16-20), and 15 jobs-10 machines problems (LA21-24). To make the given 16 benchmarks be fuzzy JSSP instances, similar to Lei (2011), for each operation O_{ij} of each extended instance, $p_{ij}=(p_{ij}-\alpha_{ij}, p_{ij}, p_{ij}+\beta_{ij})$, where p_{ij} is the corresponding deterministic processing time of the operation o_{ij}, integer value $\alpha_{ij}\in[0.06p_{ij},0.15p_{ij}]$, $\beta_{ij}\in[0.1p_{ij},0.19p_{ij}]$. If $\beta_{ij}<1$, then $\beta_{ij}\in[1,2]$. For simplicity, the extended instances are called ABZ5-PM, ABZ6-PM, ORB01-PM, and so on.

The fixed preventive maintenance activities for the 16 benchmarks are given in Table 8. The flexible PM for these instances are realized through the steps listed in Section 5.2.2. We coded SMGA, GPSO, and RKGA for comparison. In order to verify the performance of the proposed algorithm, we also coded the canonical CRO (denoted by CRO-I), and the proposed HCRO without TS-based local search (denoted by CRO-II). For SMGA, GPSO, and RKGA, we adopt the parameter settings proposed by Sakawa and Mori (1999), Niu et al. (2008), and Zheng et al. (2010), respectively, except the computational times for each instance is set 100 s.

To make detailed comparisons among the six algorithms, we define three distances as follows. Denote D_{bt} as the distance to the c_1 objective value of the best fuzzy makespan, D_{wt} as the distance to the c_1 objective value of the minimum worst fuzzy makespan, and D_{avg} as the distance to the c_1 objective value of the minimum average fuzzy makespan. The best, the minimum worst value, and the minimum average fuzzy makespan are collected by the experimental results of the six compared algorithms after 20 independent runs, respectively.

Table 7
Comparisons of experimental results on the four flexible PM benchmarks (resumable).

Case	Algorithm	*avg*	c_1	*opt*	c_1	*wor*	c_1
1	SMGA	46.7, 55.3, 63.1	55.10	46, 54, 61	53.75	50, 57, 63	56.75
	GPSO	45.8, 54.1, 63.2	54.30	46, 54, 61	53.75	45, 55, 68	55.75
	RKGA	45.5, 54.1, 61.9	53.90	**44**, **54**, **62**	**53.50**	48, 55, 60	54.50
	HCRO	**45.6**, **54.0**, **61.9**	**53.88**	**44**, **54**, **62**	**53.50**	**46, 54, 62**	**54.00**
2	SMGA	122.3, 145.9, 168.9	145.75	118, 141, 166	141.50	130, 152, 171	151.25
	GPSO	121.1, 141.3, 161.3	141.25	**123**, **138**, **154**	**138.25**	127, 147, 163	146.00
	RKGA	120.6, 141.2, 161.6	141.15	**123**, **138**, **154**	**138.25**	127, 147, 163	146.00
	HCRO	**124.0**, **139.2**, **154.2**	**139.15**	**123**, **138**, **154**	**138.25**	**124**, **140**, **155**	**139.75**
3	SMGA	129.9, 149.1, 167.3	148.85	126, 144, 157	142.75	137, 158, 176	157.25
	GPSO	125.3, 144.4, 159.0	143.28	117, 144, 154	139.75	128, 147, 162	146.00
	RKGA	123.2, 143.2, 157.8	141.85	121, 140, 156	139.25	122, 144, 173	145.75
	HCRO	**124.2**, **141.9**, **156.5**	**141.13**	**122**, **136**, **158**	**138.00**	**123**, **142**, **163**	**142.50**
4	SMGA	119.8, 136.3, 153.3	136.43	109, 130, 150	129.75	133, 148, 161	147.50
	GPSO	116.2, 131.8, 148.4	132.05	114, 130, 144	129.50	119, 136, 164	138.75
	RKGA	119.0, 132.1, 149.3	133.13	114, 130, 144	129.50	125, 136, 149	136.50
	HCRO	**112.8**, **129.9**, **146.3**	**129.73**	**117**, **129**, **141**	**129.00**	**113**, **130**, **148**	**130.25**

Table 8
Preventive maintenance activities for the 16 benchmarks.

Benchmark	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
ABZ5-PM	(231, 270)	(115, 161)	(365, 405)	(657, 680)	(372, 418)	(564, 589)	(345, 390)	(749, 795)	(220, 258)	(698, 730)
ABZ6-PM	(115, 161)	(231, 270)	(365, 405)	(757, 780)	(172, 218)	(564, 589)	(345, 390)	(49, 95)	(220, 258)	(698, 730)
ORB01-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(49, 95)	(220, 258)	(398, 430)
ORB02-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(49, 95)	(220, 258)	(398, 430)
ORB03-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(49, 95)	(220, 258)	(398, 430)
ORB04-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(49, 95)	(220, 258)	(398, 430)
ORB05-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(49, 95)	(220, 258)	(398, 430)
LA16-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA17-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA18-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA19-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA20-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA21-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA22-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA23-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)
LA24-PM	(415, 461)	(231, 270)	(365, 405)	(557, 580)	(172, 218)	(564, 589)	(345, 390)	(149, 195)	(220, 258)	(398, 430)

The formula for D_{bt}, D_{wt}, and D_{avg}, are as follows:

$$D_{bt}=\frac{C_c^b-C_b^b}{C_b^b}\times 100\%,\quad D_{wt}=\frac{C_c^w-C_b^w}{C_b^w}\times 100\%,\quad D_{avg}=\frac{C_c^a-C_b^a}{C_b^a}\times 100\% \quad (4)$$

where, C_c^b, C_c^w, and C_c^a represent the best, the minimum worst, and the average c_1 objective value obtained by the compared algorithm, respectively, while C_b^b, C_b^w, and C_b^a represent the best, the minimum worst, and the minimum average c_1 objective value collected by the six compared algorithms.

Tables 9–11 give the comparisons of the experimental results on the 16 extended benchmarks with flexible PM activities under resumable situation. The compared algorithms are SMGA, GPSO, RKGA, CRO-I, CRO-II, and HCRO. In Table 9, the first column displays the benchmark name; the second column tells the best fuzzy makespan collected by the six algorithms after 20 independent runs; the following column illustrates the c_1 objective value of the obtained best fuzzy makespan for the corresponding benchmark; the remaining six columns list D_{bt} values obtained by SMGA, GPSO, RKGA, CRO-II, CRO-I, and HCRO, respectively. Table 10 gives the comparison results on D_{wt}, while Table 11 displays the comparison results on D_{avg}.

From Table 9, we can conclude that: (1) for solving the 16 extended benchmarks with flexible PM activities, on average, the proposed HCRO shows better performance in finding optimal solutions than the other five compared algorithms, i.e., SMGA, GPSO, RKGA, CRO-I, and CRO-II; (2) HCRO obtained 11 optimal results out of 16 benchmarks, RKGA found three best solutions, the canonical CRO got two best values, and the proposed CRO-II collected four optimal results; (3) for comparison on the average performance in finding best values, on average, HCRO is the best among the six compared algorithms, which almost converges to the best value for each benchmark; CRO-II shows slightly worse performance than HCRO, which is better than the other four algorithms; then, the following algorithms are RKGA, GPSO, CRO-I, and SMGA, respectively; (4) the canonical CRO (CRO-I) obtained optimal solutions for two benchmarks, i.e., ORB02-PM and LA19-PM, while the distance D_{bt} obtained by CRO-I, for solving LA21-PM, is slightly worse. The above analysis tells that CRO-I holds exploitation capability for several benchmarks, but the robustness should be improved; (5) HCRO shows better performance than the canonical CRO for solving 12 benchmarks, except ABZ6-PM, ORB02-PM, ORB04-PM, and LA19-PM, which verify the advantage of the proposed algorithm.

The minimum worst fuzzy makespan obtained by all compared algorithm is collected for each benchmark. The distance D_{wt} for each compared algorithm is listed in Table 10 for each corresponding benchmark. It can be concluded from Table 10 that: (1) HCRO obtained 12 minimum worst values out of 16 benchmarks, while the numbers for other compared algorithms are 2, 1, 1, 0 and 0, for RKGA, CRO-I, GPSO, CRO-II, and SMGA,

Table 9
Comparisons of D_{bt} on the 16 flexible PM benchmarks (resumable).

Benchmark	Best fuzzy makespan	c_1	SMGA	GPSO	RKGA	CRO-I	CRO-II	HCRO
ABZ5-PM	1153,1279,1418	1282.25	1.83	0.58	0.92	0.94	0.51	0.00
ABZ6-PM	892,972,1077	978.25	1.48	0.59	0.00	0.26	0.79	0.69
ORB01-PM	1010,1128,1269	1133.75	1.76	0.66	1.79	2.09	0.60	0.00
ORB02-PM	840,940,1040	940.00	4.81	0.40	1.38	0.00	0.11	0.11
ORB03-PM	1044,1096,1167	1100.75	6.38	2.38	1.66	2.95	0.84	0.00
ORB04-PM	994,1046,1164	1062.50	3.67	1.22	0.00	0.24	0.89	0.89
ORB05-PM	883,924,1023	938.50	6.18	4.18	3.14	2.82	0.00	0.00
LA16-PM	879,989,1113	992.50	1.21	0.65	1.16	1.23	0.71	0.00
LA17-PM	726,803,902	808.50	3.28	2.01	1.48	2.57	0.00	0.00
LA18-PM	837,874,917	875.50	2.46	0.20	0.00	0.57	0.31	0.31
LA19-PM	793,879,974	881.25	2.52	1.11	1.82	0.00	2.13	1.45
LA20-PM	879,936,1018	942.25	2.55	0.74	1.22	1.96	1.27	0.00
LA21-PM	1009,1120,1241	1122.50	6.21	0.73	1.05	4.50	1.02	0.00
LA22-PM	878,979,1106	985.50	3.75	3.04	0.86	4.06	0.00	0.00
LA23-PM	973,1076,1195	1080.00	3.73	0.35	0.12	3.13	0.00	0.00
LA24-PM	889,992,1099	993.00	1.81	1.31	0.13	4.96	0.76	0.00
Average performance			3.35	1.26	1.04	2.02	0.62	0.22

Table 10
Comparisons of D_{wt} on the 16 flexible PM benchmarks (resumable).

Benchmark	Minimum worst	c_1	SMGA	GPSO	RKGA	CRO-I	CRO-II	HCRO
ABZ5-PM	1187,1308,1464	1316.75	3.04	0.89	0.49	2.26	0.04	0.00
ABZ6-PM	955,997,1054	1000.75	1.60	1.25	0.95	0.20	1.22	0.00
ORB01-PM	1138,1199,1288	1206.00	1.08	0.50	0.00	0.60	0.37	0.62
ORB02-PM	903,961,1038	965.75	5.02	5.05	2.17	4.71	1.24	0.00
ORB03-PM	1099,1167,1249	1170.50	1.35	1.52	0.36	0.88	0.83	0.00
ORB04-PM	1022,1096,1194	1102.00	4.06	1.45	1.70	3.36	0.43	0.00
ORB05-PM	964,1009,1070	1013.00	1.97	0.00	1.46	0.30	0.17	0.17
LA16-PM	964,1025,1126	1035.00	2.97	0.85	1.01	1.23	0.77	0.00
LA17-PM	802,844,903	848.25	1.33	0.50	0.00	1.27	1.21	0.18
LA18-PM	828,907,1008	912.50	3.04	0.03	1.29	0.00	0.47	0.19
LA19-PM	828,907,1009	912.75	1.48	0.85	1.86	0.88	0.03	0.00
LA20-PM	911,967,1025	967.50	1.27	1.96	0.67	0.13	0.31	0.00
LA21-PM	1091,1150,1226	1154.25	8.19	2.71	0.37	6.54	0.17	0.00
LA22-PM	947,1013,1099	1018.00	6.83	1.77	0.42	6.36	0.37	0.00
LA23-PM	1050,1111,1172	1111.00	3.24	0.11	0.11	7.63	0.23	0.00
LA24-PM	976,1028,1086	1029.50	7.97	2.82	2.11	10.85	0.68	0.00
Average performance			3.40	1.39	0.94	2.95	0.53	0.07

Table 11
Comparisons of D_{avg} on the 16 flexible PM benchmarks (resumable).

Benchmark	Best average fuzzy makespan	c_1	SMGA	GPSO	RKGA	CRO-I	CRO-II	HCRO
ABZ5-PM	1185.4,1294.1,1428.1	1300.43	2.37	0.51	0.89	1.37	0.00	0.12
ABZ6-PM	904.1,984,1090.8	990.73	1.02	0.21	0.00	0.09	0.33	0.13
ORB01-PM	1084.7,1165.9,1265.6	1170.53	2.32	0.46	0.72	0.67	0.15	0.00
ORB02-PM	878.4,953.2,1044.9	957.43	4.65	0.91	0.78	2.20	0.72	0.00
ORB03-PM	1066.4,1140.5,1231	1144.60	3.07	1.03	0.00	1.86	0.30	0.00
ORB04-PM	997.8,1081.1,1185.6	1086.40	2.44	0.98	0.35	1.76	0.92	0.00
ORB05-PM	899.8,975.6,1069.4	980.10	3.16	1.54	0.61	1.08	0.00	0.00
LA16-PM	927,1002.3,1106	1009.40	2.48	0.72	1.59	0.86	1.05	0.00
LA17-PM	757,825.2,909.8	829.30	2.17	0.98	0.33	1.68	0.13	0.00
LA18-PM	815.9,890.3,987.7	896.05	2.45	0.00	0.40	0.30	0.09	0.08
LA19-PM	820.3,896.8,994	901.98	1.24	0.28	0.79	0.58	0.20	0.00
LA20-PM	883.2,952.7,1036.9	956.38	1.67	0.54	0.62	0.82	0.57	0.00
LA21-PM	1037.9,1130.3,1243.4	1135.48	8.17	1.91	1.20	6.50	0.85	0.00
LA22-PM	919.2,998.9,1098.5	1003.88	7.14	1.82	0.49	5.05	0.00	0.17
LA23-PM	1003.9,1085.6,1184.4	1089.88	4.19	0.32	0.25	4.62	0.06	0.00
LA24-PM	928,1007.7,1103.7	1011.78	8.57	1.89	0.62	6.76	0.12	0.00
Average performance			3.57	0.88	0.60	2.26	0.34	0.03
Average computational time (sec)			69.23	49.51	48.52	72.28	47.49	49.52

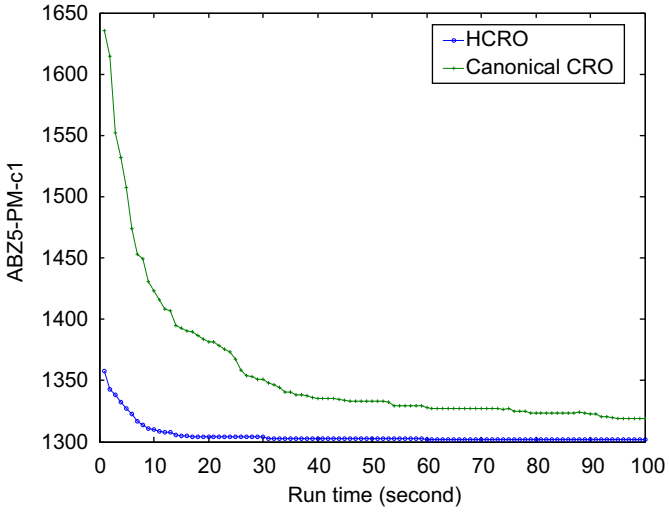

Fig. 9. Comparison of convergence curve of c_1 objective value for ABZ5-PM.

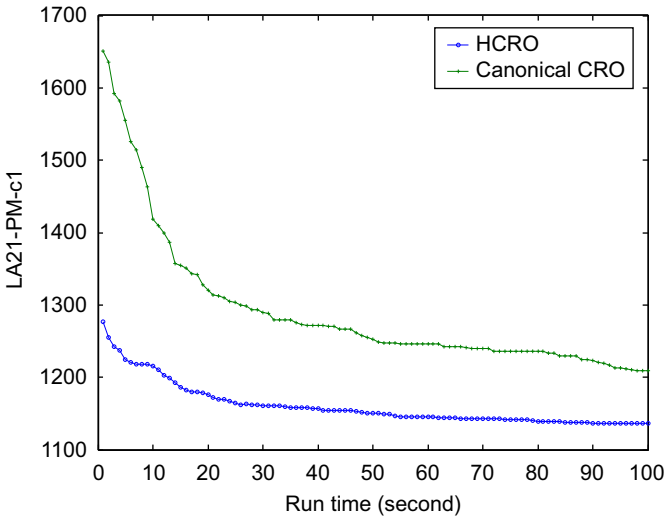

Fig. 10. Comparison of convergence curve of c_1 objective value for LA21-PM.

respectively; (2) on average, considering the worst fuzzy makespan after 20 independently runs, HCRO is the best among the six algorithms, the following algorithms are CRO-II, RKGA, GPSO, CRO-I, and SMGA.

Table 11 tells the comparison on the experimental results of the average performance for solving the given 16 benchmarks with flexible PM activities. It can be seen from Table 11 that: (1) HCRO obtained the best average values for solving 12 instances, except ABZ5-PM, ABZ6-PM, LA18-PM, and LA22-PM, while the other algorithms can only obtain best values for at most three instances; (2) HCRO holds the best average performance among the six algorithms, the following algorithms are CRO-II, RKGA, GPSO, CRO-I, and SMGA; (3) the average computational time for each algorithm is given in the last row in Table 11. HCRO consumed about 50 s, which can be acceptable in realistic production horizon. However, the average computational time of the canonical CRO is obviously larger than HCRO, which again verify the convergence ability of the proposed algorithm.

To make detailed comparisons between the proposed HCRO and the canonical CRO, Figs. 9 and 10 give the comparison of the average convergence ability between HCRO and the canonical CRO (CRO-I), for solving the two instances with different problem scales, i.e., 10 jobs-10 machines (ABZ5-PM), and 15 jobs-10 machines (LA21-PM), respectively. From the two figures, we can conclude that the proposed HCRO algorithm has better convergence ability than the canonical CRO for solving both the medium scale 10 jobs-10 machines problem, and the relative large scale 15 jobs-10 machines problem.

6. Conclusion

In this study, a hybrid algorithm combing the chemical-reaction optimization and the tabu search algorithm is proposed for solving the fuzzy job-shop scheduling problem with flexible preventive maintenance activities. The detailed encoding and decoding mechanism is developed for the problem. The main contributions of the proposed HCRO are as follows:

(1) To enhance the exploitation capability of the proposed algorithm, TS-based local search is embedded in HCRO;
(2) In the canonical CRO, decomposition and synthesis reactions are used to produce molecules with very different structures. That is, the above two reactions complete the exploration task in the evolution phase. To make the two reactions perform exploration

functions while maintaining convergence capability, a well-designed crossover operator is developed in HCRO;
(3) HCRO divides the evolution phase into two loop bodies: the first loop body contains on-wall ineffective collision, inter-molecular ineffective collision; the second loop body includes all the four elementary reactions. The evolution phase begins with the first loop body, that is, the exploitation task is firstly been performed. When the exploitation cannot be continued after certain number of generations, the second loop body started to perform both exploration and exploitation function. Then, the two loop bodies run alternatively.

Two sets of benchmarks with flexible PM activities are tested to make a detailed comparison between HCRO and other efficient algorithms from the literature. The benchmarks range from 10 jobs-10 machines to 15 jobs-10 machines. Experimental results show the robustness and efficiency of the proposed algorithm. The future work is as follows:

(1) To improve the four elementary reactions and enhance the balance between the exploitation and exploration, and to increase the simulation result reliability;
(2) To apply the proposed algorithm for solving other potential applications, such as the fuzzy flexible job-shop scheduling problem with flexible PM activities, and the flexible flow shop scheduling problem with flexible PM activities;
(3) To apply HCRO for solving other kinds of job-shop scheduling problems, such as multi-objective JSSP, JSSP with setup-time, and no-wait JSSP. The proposed HCRO is suitable to be applied to JSSP in general, after completing the following issues: (a) problem adaptive encoding and decoding mechanism; (b) improved elementary reactions; (c) neighborhood structure considering the problem structure.
(4) To apply the design of experiment (DoE) method to determine parameters for the experiments.

References

Armentano, V.A., Scrich, C.R., 2000. Tabu search for minimizing total tardiness in a job shop. International Journal of Production Economics 63, 131–140.

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

Hu, Y.M., Yin, M.H., Li, X.T., 2011. A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm. International Journal of Advanced Manufacturing Technology 56, 1125–1138.

Inés, G.R., Camino, R.V., Jorge, P., 2010. A genetic solution based on lexicographical goal programming for a multiobjective job shop with uncertainty. Journal of Intelligent Manufacturing 21, 65–73.

Kuroda, M., Wang, Z., 1996. Fuzzy job shop scheduling. International Journal of Production Economics 44, 45–51.

Lam, A.Y.S., Xu, J., Li, V.O.K., 2010a. Chemical Reaction Optimization for Population Transition in Peer-to-peer Live Streaming. In: Proceeding of the IEEE Congress on Evolutionary Computation (CEC 2010) Barcelona, Spain, July, 2010, pp. 1429–1436.

Lam, A.Y.S., Li, V.O.K., 2010b. Chemical-reaction-inspired metaheuristic for optimization. IEEE Transactions on Evolutionary Computation 14, 381–399.

Lam, A.Y.S., Li, V.O.K., Yu, J.J.Q., 2012a. Real-coded chemical reaction optimization. IEEE Transactions on Evolutionary Computation 16, 339–353.

Lam, A.Y.S., Li, V.O.K., 2012b. Chemical reaction optimization: a tutorial. Memetic Computing 4, 3–17.

Lei, D.M., 2007. Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems. International Journal of Advanced Manufacturing Technology 37, 157–165.

Lei, D.M., 2010a. Solving fuzzy job shop scheduling problems using random key genetic algorithm. International Journal of Advanced Manufacturing Technology 49, 253–262.

Lei, D.M., 2010b. Fuzzy job shop scheduling problem with availability constraints. Computers and Industrial Engineering 58, 610–617.

Lei, D.M., 2011. Scheduling fuzzy job shop with preventive maintenance through swarm-based neighborhood search. International Journal of Advanced Manufacturing Technology 54, 1121–1128.

Li, J.Q., Pan, Q.K., Liang, Y.C., 2010. An effective hybrid tabu search algorithm for multi-objective flexible job shop scheduling problems. Computers and Industrial Engineering 59, 647–662.

Ma, Y., Chu, C.B., Zuo, C.R., 2010. A survey of scheduling with deterministic machine availability constraints. Computers and Industrial Engineering 58, 199–211.

Niu, Q., Jiao, B., Gu, X.S., 2008. Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. Applied Mathematics and Computation 205, 148–158.

Sakawa, M., Mori, T., 1999. An efficient genetic algorithm for job shop scheduling problems with fuzzy processing time and fuzzy due date. Computers and Industrial Engineering 36, 325–341.

Sakawa, M., Kubota, R., 2000. Fuzzy programming for multi-objective job shop scheduling with fuzzy processing time and fuzzy due date through genetic algorithm. European Journal of Operational Research 120, 393–407.

Schmidt, G., 2000. Scheduling with limited machine availability. European Journal of Operational Research 121, 1–15.

Song, X.Y., Zhu, Y.L., Yin, C.W., Li, F.M., 2006. Study on the Combination of Genetic Algorithms and Ant Colony Algorithms for Solving Fuzzy Job Shop Scheduling. In: Proceedings of the IMACS Multi-conferences on Computational Engineering in Systems Application, 2006, Beijing, pp. 1904–1909.

Wang, L., 2003. Shop Scheduling with Genetic Algorithms. Tsinghua University and Springer Press, Beijing 2003.

Wang, S.J., Yu, J.B., 2010. An effective heuristic for flexible job-shop scheduling problem with maintenance activities. Computers and Industrial Engineering 59, 436–447.

Wu, C.S., Li, D.C., Tsai, T.I., 2006. Applying the fuzzy ranking method to the shifting bottleneck procedure to solve scheduling problems of uncertainty. International Journal of Advanced Manufacturing Technology 31, 98–106.

Xu, J., Lam, A.Y.S., Li, V.O.K., 2010. Chemical Reaction Optimization for the Grid Scheduling Problem. In: Proceeding of the IEEE International Conference on Communications Cape Town, South Africa, May, 2010, pp. 1–5.

Xu, J., Lam, A.Y.S., Li, V.O.K., 2011. Chemical reaction optimization for task scheduling in grid computing. IEEE Transactions on Parallel and Distributed Systems 22, 1624–1631.

Zheng, Y.L., Li, Y.X., Lei, D.M., 2010. Scheduling Jobs and Preventive Maintenance on Fuzzy Job Shop Using Genetic Algorithm. In: Proceedings of the Machine Learning and Cybernetics, 2010, Qingdao, pp. 1583–1589.

Zribi, N., Kamel, A.E., Borne, P., 2008. Minimizing the makespan for the MPM job-shop with availability constraints. International Journal of Production Economics 112, 151–160.