SCIENCE CHINA
Information Sciences

July 2019, Vol. 62 070202:1–070202:18
https://doi.org/10.1007/s11432-018-9693-2

• RESEARCH PAPER •
Special Focus on Pigeon-Inspired Optimization

A multi-objective pigeon inspired optimization algorithm for fuzzy production scheduling problem considering mould maintenance

Xiaoyue FU[1], Felix T.S. CHAN[1], Ben NIU[2*], Nick S.H. CHUNG[1] & Ting QU[3]

[1]*Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China;*
[2]*College of Management, Shenzhen University, Shenzhen* 518060*, China;*
[3]*School of Electrical and Information Engineering, Jinan University, Guangzhou* 519070*, China*

Received 6 August 2018/Revised 21 October 2018/Accepted 30 November 2018/Published online 9 May 2019

Abstract The fuzzy production scheduling problem considering mould maintenance (FPSP-MM) is studied. The processing time and the maintenance time are represented by triangular fuzzy numbers. When tasks are executed based on the sequence provided by the fuzzy schedule, the real duration of each task needs to be known so the posteriori solution with deterministic processing times can be obtained. Therefore, the concept of the schedule robustness needs to be considered for the fuzzy problem. The robustness is considered as the optimization objective except for the fuzzy makespan in this research. To optimize these two objective functions, a multi-objective pigeon inspired optimization (MOPIO) algorithm is developed. To extend the pigeon inspired optimization (PIO) algorithm from the single-objective case to the multi-objective case, non-dominated solutions are used as candidates for the leader pigeon designation and a special crowding distance is used to ensure a good distribution of solutions in both the objective space and the corresponding decision space. Furthermore, an index-based ring topology is used to manage the convergence speed. Numerical experiments on a variety of simulated scenarios show the excellent efficiency and effectiveness of the proposed MOPIO algorithm by comparing it with other algorithms.

Keywords fuzzy, production scheduling, mould maintenance, pigeon inspired optimization, multi-objective

Citation Fu X Y, Chan F T S, Niu B, et al. A multi-objective pigeon inspired optimization algorithm for fuzzy production scheduling problem considering mould maintenance. Sci China Inf Sci, 2019, 62(7): 070202, https://doi.org/10.1007/s11432-018-9693-2

1 Introduction

Production scheduling involves assigning certain tasks to limited resources so that all the constraints are satisfied, and all the objectives are achieved. There are different types of production scheduling problems, however, in most of these production scheduling problems, only the allocation of machines is considered. In some industries, such as the plastic products industry and the die stamping industry, the mould is an important resource that needs to be considered. Traditionally, resources are assumed to be always available in the whole production planning stage. However, in real situations, some machines may be unavailable because of stochastic failures [1]. So, the maintenance of resources needs to be considered when production plans are made. The system productivity is improved when resource maintenance planning and production scheduling are integrated [2]. To integrate production scheduling with multi-resource maintenance, researchers have made great efforts. To minimize the overall makespan, a joint scheduling

* Corresponding author (email: drniuben@gmail.com)

strategy is proposed to solve the integrated production scheduling with mould maintenance problem [3]. Besides, a more complex integrated problem that contains multiple resources and maintenance tasks was considered [4]. The results showed that the makespan was reduced significantly by the proposed jointly scheduling method. In addition, another integrated problem that each job includes multiple operations with multiple moulds was studied [5]. Moreover, to minimize the makespan and unavailability of the machine and the mould, Wang and Liu [6] proposed a multi-objective integrated optimization method with NSGA-II adaption to solve the multi-objective parallel machine scheduling problem with flexible preventive maintenance on the machine and mould. Further, setup time and mould maintenance were considered [7] to show the influence of the mould maintenance on production scheduling.

In the existing research on the integrated production scheduling problem with mould maintenance, all the information such as the processing time and the maintenance time is determinate, however, this information is undetermined because of some human-related factors in most real-world manufacturing environments. Because of the uncertainty, the solutions built with the evaluated data may become antiquated during the implementation. Many studies model the uncertainty processing time as triangular fuzzy numbers and different algorithms were proposed to obtain good fuzzy schedules [8–10]. Moreover, in most of the research on the fuzzy problems, only one objective is considered. However, in many practical cases, only one criterion is not enough. Several objectives should be considered simultaneously to improve the industrial advantages in the fierce market competition. In traditional research on the optimization of the scheduling problem with single objective, the objectives, such as minimization of the makespan (maximum completion time), minimization of the total completion time, minimization of the total tardiness are considered. When the uncertainty is considered, the traditional objective is not sufficient, and robustness need to be considered as a significant optimization objective. The exact starting times for each task are not obtained through the fuzzy schedule and solutions to the fuzzy problem should be treated as priori solutions. When tasks are executed according to the sequence provided by the fuzzy schedule, the real duration of each task needs to be known and a real executed schedule (the posteriori solution with deterministic data) is obtained. Therefore, except for the fuzzy makespan, which is an important evaluation criterion of the industry competitiveness and the most often used objective in the optimization problem [11, 12], the concept of schedule robustness needs to be considered for the fuzzy problem.

The pigeon inspired optimization (PIO) algorithm was firstly proposed by Duan and Qiao [13], and is a novel bio-inspired swarm intelligence optimizer mimicking the homing characteristics of pigeons. There are two main operators of the PIO algorithm: the map and compass operator, and the landmark operator. Since the PIO algorithm was first proposed, it has been used in many real-world applications and many new variants based on it have been put forward. To achieve the target detection task for unmanned aerial vehicles (UAVs) at low altitude, a hybrid model of edge potential function (EPF) and the simulated annealing pigeon inspired optimization (SAPIO) algorithm were proposed by Li and Duan [14]. The robustness and effectiveness of the SAPIO algorithm were shown by a number of comparative experiments with other algorithms. Moreover, a novel predator-prey pigeon-inspired optimization (PPPIO) [15] was proposed to solve the uninhabited combat aerial vehicle (UCAV) three-dimension path planning problem in the dynamic environment. The comparative simulation results show that the proposed PPPIO algorithm is more efficient than other algorithms for solving the problem. In addition, an orthogonal PIO algorithm [16] was suggested and employed in the training process of the echo state network (ESN) to obtain the desired parameters. The superiority of the orthogonal PIO algorithm is shown by comparing with several existing bio-inspired optimization algorithms. Furthermore, an improved PIO algorithm [17] was utilized by converting the parameter design problem for the automatic carrier landing system to an optimization problem. A series of experiments were conducted to demonstrate the feasibility and effectiveness of the proposed method. Comparative results indicated that the proposed method is much better than other methods. In addition, the Gaussian pigeon inspired optimization (GPIO) algorithm [18] was proposed for solving the optimal formation reconfiguration problems of multiple orbital spacecraft. The feasibility and effectiveness of the proposed GPIO algorithm in solving orbital spacecraft formation reconfiguration problems were verified by the comparative experiments with the basic PIO and the particle

swarm optimization (PSO).

Furthermore, the PIO is also extended to solve multi-objective problems. Qiu and Duan [19] proposed the multi-objective pigeon inspired optimization (MPIO) to solve the multi-objective optimization problems in designing the parameters of brushless direct current motors. Comparative experimental results with the modified non-dominated sorting genetic algorithm are given to show the feasibility, validity, and superiority of the proposed algorithm. Moreover, the MPIO was modified based on the hierarchical learning behavior in pigeon flocks and an UAV distributed flocking control algorithm based on the modified MPIO was proposed to coordinate UAVs to fly in a stable formation under complex environments. Comparison experiments with the basic MPIO and the modified non-dominated sorting genetic algorithm (NSGA-II) were carried out to show the feasibility, validity, and superiority of the proposed algorithm [20].

Although there have been a few studies on the production scheduling problem considering mould maintenance, none of them considered the indeterminate processing time and maintenance time, and the uncertainty needs to be considered in practical application. Furthermore, when the fuzziness is considered in the integrated problem, the robustness should also be considered because that some differences may exist between the fuzzy schedule and the actually executed schedule. Moreover, since the integrated production scheduling with the maintenance problem is NP-hard, meta-heuristics are usually used to solve these problems. So far, most of these integrated problems are solved by the genetic algorithm (GA) approach. However, because of the unguided mutation of the GA, the convergence rate of the GA is slow. Furthermore, the performance of the GA depends on the diversity mechanism. If the diversity mechanism does not work properly, it is easy for the GA to converge into local optima prematurely. Except for GA, other algorithms such as PSO are also used for these integrated problems. The main advantage of PSO is its rapid convergence rate but it is susceptible to premature convergence, especially when the dimensions or decision variables are large. To handle the stagnation and speed up the convergence, many strategies need to be proposed, such as improving the distribution of initial solutions, changing the communication mechanism, and adjusting the parameters. Since these algorithms have limitations, more innovative algorithms need to be proposed to solve the integrated problems. As a new swarm intelligent algorithm, the PIO algorithm has not yet been applied in the production scheduling problem, and more efficient variants of PIO need to be explored to solve real-world problems.

Based on the research gap, this study proposes a multi-objective pigeon inspired optimization (MOPIO) algorithm for fuzzy production scheduling problem considering mould maintenance (FPSP-MM). The uncertainty is restricted to the fuzzy processing time and fuzzy maintenance time, which means that the processing time and the maintenance time can be represented by triangular fuzzy numbers. There are two objectives in this optimization problem, the fuzzy makespan, and the robustness. To extend the PIO algorithm from the single-objective to the multi-objective case, non-dominated solutions are used as candidates for the leader pigeon and a special crowding distance is used to ensure a good distribution of solutions in the decision space and in the corresponding objective space. Furthermore, an index-based ring topology is used to manage the convergence speed. To evaluate the results, the hypervolume (HV) and the cover rate (CR) are used as two performance indicators. In the experiments, some instances are generated by fuzzifying the benchmarks from the deterministic problem and some instances are generated randomly. To show the advantages of the proposed MOPIO algorithm, multi-objective particle swarm optimization (MOPSO) and NSGA-II which are the most popular algorithms to solve multi-objective problems, are used as the comparison algorithms.

The reminder of this paper is organized as follows. Section 2 describes the FPSP-MM problem. Section 3 proposes the MOPIO algorithm. Section 4 presents the experimental design. Section 5 shows the computational results acquired and some discussion is made to show the superiority of the MOPIO algorithm. Section 6 provides the conclusion and suggestions for further research.

2 Problem description

The FPSP-MM problem can be described as follows: P jobs are allocated on Q injection machines and N injection moulds. Each problem is defined as $P \times Q \times N$. At time zero, all jobs are well prepared, and all the machines and moulds are accessible. Each job must use the mould which is given in advance. Each job can choose the machines it uses, but not all the machines are available for all jobs. Different jobs may be performed by the same mould. Each job cannot be performed by more than one machine in a given time slot, and each machine cannot deal with more than one job in a given time slot. Each mould cannot carry out more than one job at a given time slot. The batch size and the unit fuzzy operation time of each job are given. The total operation time of a job is the product of the unit fuzzy operation time and the batch size. The batch size of each job cannot be split and there is no interruption during the production process of a job. The unit fuzzy operation time of a job depends on the mould it uses. The unit fuzzy operation time is represented by a triangular fuzzy number. The maintenance time of the resources depends on the time that the maintenance begins. The maintenance time is shorter when the maintenance is conducted earlier. The maintenance time of a resource depends on the longest accumulated working time of a resource (the accumulated working time of a resource is a triangular fuzzy number, the longest of which means the third number of the triangular fuzzy number). The maintenance time may be fuzzy or crisp. Only perfect maintenance is considered, which means that after the maintenance, the condition of the resource is as good as new. In this paper, the preventive maintenance is assumed to be able to prevent all the random breakdowns. Moreover, the set-up time and the quality issue are not considered in this research. The objective is to find good production scheduling and machine maintenance planning aiming at minimizing the makespan and maximizing the robustness. To extend the determinate single-objective problem to a fuzzy multi-objective problem, four issues need to be solved: the definition of the arithmetic operations on triangular fuzzy numbers, the fuzzy maintenance time, the robustness and the Pareto dominance relationship. These problems are given in Subsections 2.1–2.4.

2.1 Arithmetic operations on triangular fuzzy numbers

The unit fuzzy processing time is a triangular fuzzy number (TFN), denoted as $A = (a^1, a^2, a^3)$, where a^1 is the best processing time, a^3 is the worst processing time and a^2 is the most possible processing time. When the original deterministic model is extended to a model with uncertainty, two difficulties need to be solved. First, the arithmetic operations of addition and maximum need to be given when deterministic numbers are changed into TFNs. Second, the definition of minimal makespan also need to be given when the makespan is a triangular fuzzy number. According to [21], for two TFNs, $A = (a^1, a^2, a^3)$ and $B = (b^1, b^2, b^3)$. Their addition is defined as

$$A + B = ((a^1 + b^1), (a^2 + b^2), (a^3 + b^3)). \tag{1}$$

The maximum operation is defined as

$$\max(A, B) = (\max(a^1, b^1), \max(a^2, b^2), \max(a^3, b^3)). \tag{2}$$

To find the minimal makespan, three ranking criteria are given as follows:

$$C_1(A) = \frac{a^1 + 2 \times a^2 + a^3}{4}, \tag{3}$$

$$C_2(A) = a^2, \tag{4}$$

$$C_3(A) = a^3 - a^1. \tag{5}$$

To compare two TFNs, the values of C_1 are compared. If the values of C_1 are the same for two TFNs, then the values of C_2 are compared. If the values of C_1 and C_2 are the same for two TFNs, then the values of C_3 are compared.

Table 1 Maintenance time based on machine/mould age

Machine age	Maintenance time	Mould age	Maintenance time
$0 < a^3 \leqslant 180$	(150, 150, 150)	$0 < b^3 \leqslant 120$	(150, 150, 150)
$180 < a^3 \leqslant 420$	(94, 94, 94)+$(a^1, a^2, a^3)/3$	$120 < b^3 \leqslant 280$	(94, 94, 94)+$(b^1, b^2, b^3)/2$
$420 < a^3 \leqslant 600$	(160, 160, 160)+$(a^1, a^2, a^3)/3$	$280 < b^3 \leqslant 400$	(160, 160, 160)+$(b^1, b^2, b^3)/2$
$600 < a^3$	(720, 720, 720)	$400 < b^3$	(720, 720, 720)

2.2 Uncertain maintenance time

In the determinate model proposed by Wong et al. [3], the accumulated working time of a resource is defined as the resource age (the idle time is not included). A piecewise linear function is used to describe the relationship between maintenance time and resource (machine or mould) age. In practice, a mould has a higher possibility of breakdown than a machine. So, the maximum age of the machine (MA) is longer than the maximum age of the mould (NA). Maintenance has to be conducted after completion of the current job once a resource reaches its maximum age. In the uncertainty model, since the processing time is a triangular fuzzy number, the resource age is also a triangular fuzzy number. The maintenance time is decided by the worst value in the triangular fuzzy number of the resource age. The age of the machine is represented by $A = (a^1, a^2, a^3)$. The age of the mould is represented by $B = (b^1, b^2, b^3)$. The relationship between the maintenance time and the resource age is shown in Table 1.

2.3 Objective measure

There are two objectives in this problem. The first is the fuzzy makespan and the second is the robustness. According to Palacios et al. [11], the objective related to the fuzzy makespan C is defined as the expected value of the triangle fuzzy number

$$E(C) = \frac{C^1 + 2 \times C^2 + C^3}{4}. \tag{6}$$

The ranking method based on the expected value is shown to be convenient and it is proven that the ranking result is similar to other ranking methods. It is obvious that the smaller the expected value, the better the objective value. The robustness is defined as

$$\mathrm{Rob}(C) = \max\{(C^2 - C^1), (C^3 - C^2)\}, \tag{7}$$

and it measures the maximum possible difference between the makespan of the real execution and the most likely estimated makespan. It is a priori measure and the smaller the value, the better the robustness. So the objectives of this problem are shown as follows:

$$\min\ E(C), \tag{8}$$

$$\min\ \mathrm{Rob}(C). \tag{9}$$

2.4 Pareto domination relationship

For the multi-objective optimization problems with two or more objectives to be optimized,

$$\min\ f(x) = (f_1(x), f_2(x), \ldots, f_m(x)), \tag{10}$$

where $x = (x_1, x_2, \ldots, x_n)$ is an n-dimensional decision vector, $f(x)$ is an m-dimensional objective vector. The n-dimensional space, comprised of all the possible values of the decision vector x, is known as the decision space, and the m-dimensional space consisting of all the possible values of the objective vector $f(x)$ is the objective space. There are many different solutions for multi-objective optimization problems and these solutions can be compared based on the Pareto dominance relationship. Given two feasible solutions x and y, solution x is said to dominate solution y if $f_i(x) \leqslant f_i(y)$, $i = 1, 2, \ldots, m$ and there exists at least one $j \in 1, 2, \ldots, m$ so that $f_i(x) < f_i(y)$. A solution is said to be non-dominated if it is

not dominated by any other solutions. The set of all the non-dominated solutions in the decision space is called the Pareto-optimal set (PS). The Pareto front (PF) is the set of all the vectors in the objective space that corresponds to the PS.

3 Optimization methodology

3.1 Basic pigeon inspired optimization

PIO simulates the homing behavior of pigeons. There are two main operators in the algorithm.

(1) Map and compass operator. Through shaping the map in their brains by magnetoreception, pigeons are able to sense the earth's magnetic field. Moreover, pigeons adjust their directions based on the altitude of the sun, which has the same function as a compass. When pigeons fly to their destination, they rely less and less on the sun and magnetic particles. Each pigeon has a position X_i and a velocity V_i in a D-dimension search space. Both the positions and the velocities of the pigeons are updated in each iteration. The new position X_i and velocity V_i of pigeon i at the t-th iteration can be calculated by the following equations:

$$V_i(t) = V_i(t-1) \times e^{-Rt} + \text{rand} \times (X_g - X_i(t-1)), \tag{11}$$

$$X_i(t) = X_i(t-1) + V_i(t), \tag{12}$$

where $V_i(t-1)$ and $V_i(t)$ are the velocities of the pigeon i at $(t-1)$-th and t-th iteration. $X_i(t-1)$ and $X_i(t)$ are the positions of the pigeon i at $(t-1)$-th and t-th iteration. R is the map and compass factor, rand is a random number and X_g is the current global best position, which can be obtained by comparing all the positions among all the pigeons.

(2) Landmark operator. The pigeons depend on landmarks that they are near when the pigeons are close to their destination. They will fly directly to the destination if they are familiar with the landmarks. They will follow pigeons who are familiar with the landmarks if they are far from the destination and unfamiliar with the landmarks. In the landmark operator, half the number of pigeons is decreased in every generation, which means pigeons that are far from the destination and unfamiliar with the landmarks will follow the pigeons that are familiar with the landmarks. Then, the pigeons close to their destination will fly to their destination quickly, which is represented by $X_C(t)$ (the center of some pigeons positions at the t-th iteration). The position updating rule for pigeon i at the t-th iteration can be given by

$$N_p(t) = \frac{N_p(t-1)}{2}, \tag{13}$$

$$X_C(t) = \frac{\sum X_i(t) \times \text{fitness}(X_i(t))}{N_p \times \sum \text{fitness}(X_i(t))}, \tag{14}$$

$$X_i(t) = X_i(t-1) + \text{rand} \times (X_C(t) - X_i(t-1)), \tag{15}$$

where $\text{fitness}(X_i(t))$ is the quality of the pigeon i at the t-th iteration, and for minimum optimization problems, it is usually chosen as $\text{fitness}(X_i(t)) = \frac{1}{f_{\min}(X_i(t))+\xi}$. For maximum optimization problems, it is usually chosen as $\text{fitness}(X_i(t)) = f_{\max}(X_i(t))$. For each individual pigeon, the optimal position of the Nc-th iteration can be denoted by X_p, and $X_p = \min(X_i(1), X_i(2), \ldots, X_i(\text{Nc}))$.

3.2 Encoding and decoding of the pigeon

In MOPIO, each pigeon has information on the job sequence (J), the corresponding machine sequence (M), machine maintenance (AM) and information on the mould maintenance (OM). In the evolution process of MOPIO, the positions values of these pigeons always fluctuate in the space of a real number. Random key representation [22] and the smallest position value (SPV) rule [23] are applied to decode the positions of these pigeons into a suitable scheduling solution for this problem. After decoding, the values of the J parameters are integers between 1 and P (P is the number of jobs); the values of the M parameters are integers between 1 and Q (Q is the number of machines); the AM parameter is the

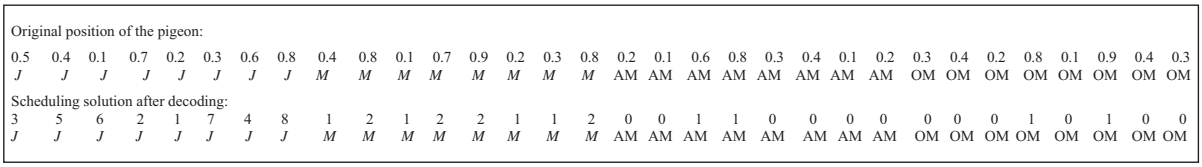

Figure 1 Encoding and decoding of the example pigeon.

maintenance decision on the machine, with value 0 or 1; the OM parameter is the maintenance decision on the mould, with value 0 or 1; if the relevant AM or OM is 1, the corresponding resource is maintained after finishing the job, otherwise they are not maintained. An example pigeon before and after decoding is shown in Figure 1. In this example, there are 8 jobs, 2 machines, and 2 moulds. Jobs 1, 2, 3, 4 can only be produced by mould 1 and jobs 5, 6, 7, 8 can only be produced by mould 2. From Figure 1, it can be seen that the value of J in the original position of the pigeon is (0.5 0.4 0.1 0.7 0.2 0.3 0.6 0.8), and it is transferred into (3 5 6 2 1 7 4 8). The sequence (0.5 0.4 0.1 0.7 0.2 0.3 0.6 0.8) is ranked according to the ascending order and given (0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8). Since the number 0.1 is in the third position of the original sequence, it is decoded into 3. Since the number 0.2 is in the fifth position of the original sequence, it is decoded into 5. Using this method, the original positions can be decoded into a suitable scheduling solution (3 5 6 2 1 7 4 8). The value of the corresponding M in the original position of the pigeon is (0.4 0.8 0.1 0.7 0.9 0.2 0.3 0.8). The interval [0.1 0.9] (0.1 is the minimum and 0.9 is the maximum among all the numbers) is divided into 2 intervals, [0.1 0.5) and [0.5 0.9] (there are 2 machines in this example). Since 0.4, 0.1, 0.2, and 0.3 are in the first interval, after decoding, the value in the relevant position is 1. Since 0.8, 0.7, and 0.9 are in the second interval after decoding, the value in the relevant position is 2. So, the corresponding M can be transferred into (1 2 1 2 2 1 1 2). The value of the corresponding machine AM in the original position of the pigeon is (0.2 0.1 0.6 0.8 0.3 0.4 0.1 0.2) and the interval [0.1 0.8] (0.1 is the minimum and 0.8 is the maximum among all the numbers) is divided into 2 intervals, [0.1 0.45) and [0.45 0.8]. Since 0.1, 0.2, 0.3 and 0.4 are in the interval [0.1 0.45), the value in the relevant position is decoded into 0. Since 0.6 and 0.8 are in the interval [0.45 0.8], the value in the relevant position is decoded into 1. So, the corresponding machine AM is decoded into (0 0 1 1 0 0 0 0). A similar decoding method can be applied to OM. After decoding, it is known that job 3 is distributed on machine 1 and machine 1 will not be maintained after job 3, and the injection mould on machine 1 will not be maintained either. Job 2 is allocated to machine 2, but machine 2 will be maintained after job 2 since the corresponding AM parameter is 1 and the injection mould on machine 2 will also be maintained because the corresponding OM parameter is 1.

Furthermore, we give the fuzzy scheduling charts of the example pigeon (3 5 6 2 1 7 4 8 1 2 1 2 2 1 1 2 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0). The machine scheduling chart is shown in Figure 2 and the mould scheduling chart is shown in Figure 3. In this example, the unit fuzzy processing time of mould 1 is (8 10 11). The corresponding batch sizes of jobs 1, 2, 3, 4 are 2, 3, 1, 2. The unit fuzzy processing time of mould 2 is (9 11 13). The corresponding batch sizes of jobs 5, 6, 7, 8 are 3, 2, 2, 1. The fuzzy maintenance times on machine 1 and machine 2 are (2 5 7) and (3 4 6) respectively, and the fuzzy maintenance times on mould 1 and mould 2 are (2 3 6) and (4 6 8) respectively. In Figures 2 and 3, different jobs are represented by different colours. The fuzzy number under the line is the start time of each job and the fuzzy number above the line is the end time of each job. The maintenance on the machine and the mould is represented in black. From Figures 2 and 3, the fuzzy makespan is (86, 107, 122).

Moreover, to illustrate the influence of the fuzziness on the problem, we give the normal scheduling without fuzziness. For fair comparison, the example pigeon (3 5 6 2 1 7 4 8 1 2 1 2 2 1 1 2 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0) is used. The batch sizes of the jobs are not changed. We only modify the fuzzy properties of the processing time and maintenance time into the normal properties. The unit processing time of mould 1 is 10, which is the most possible value in the triangular fuzzy number (8 10 11). The unit processing time of mould 2 is 11, which represents the most possible value in the triangular fuzzy number (9 11 13). The maintenance times on machine 1 and machine 2 are 5 and 4, respectively. The maintenance times on mould 1 and mould 2 are 3 and 6, respectively. The Gantt charts of the example

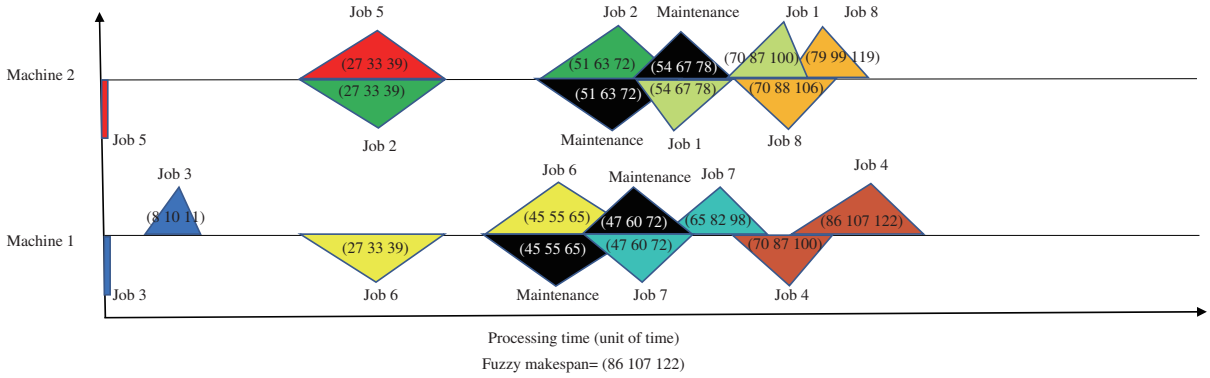

Figure 2 (Color online) Fuzzy machine scheduling of the example pigeon.

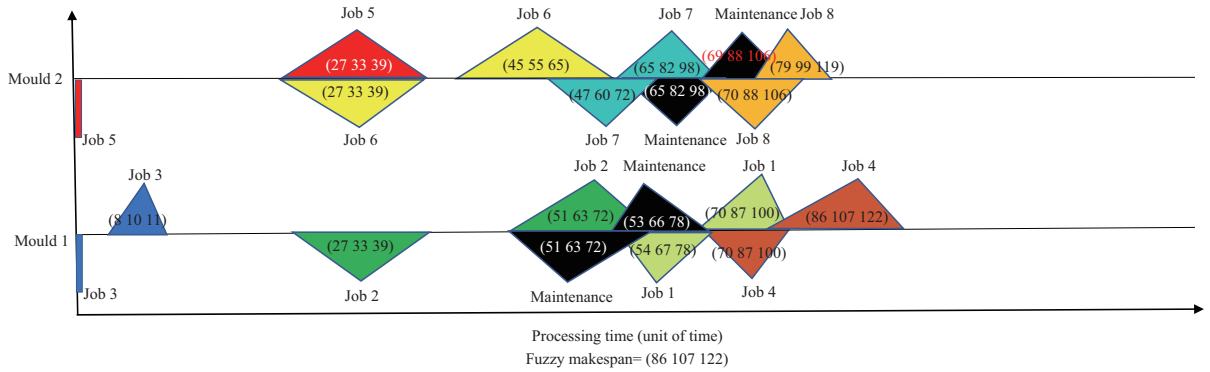

Figure 3 (Color online) Fuzzy mould scheduling of the example pigeon.

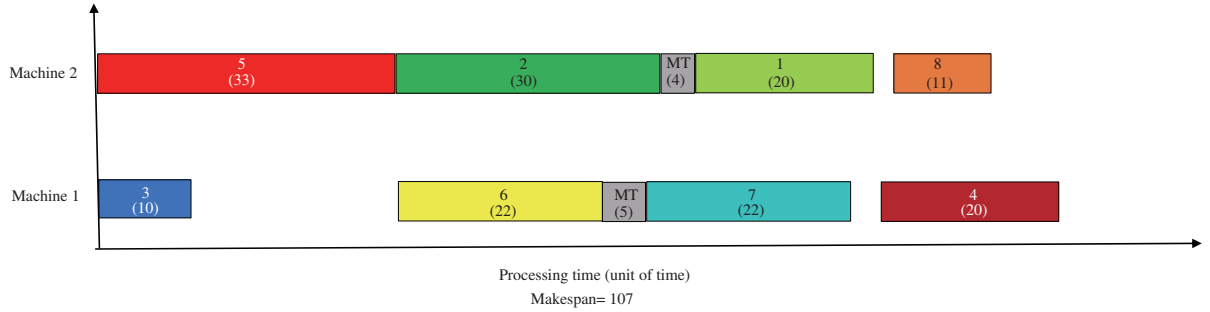

Figure 4 (Color online) Machine scheduling of the example pigeon without fuzziness.

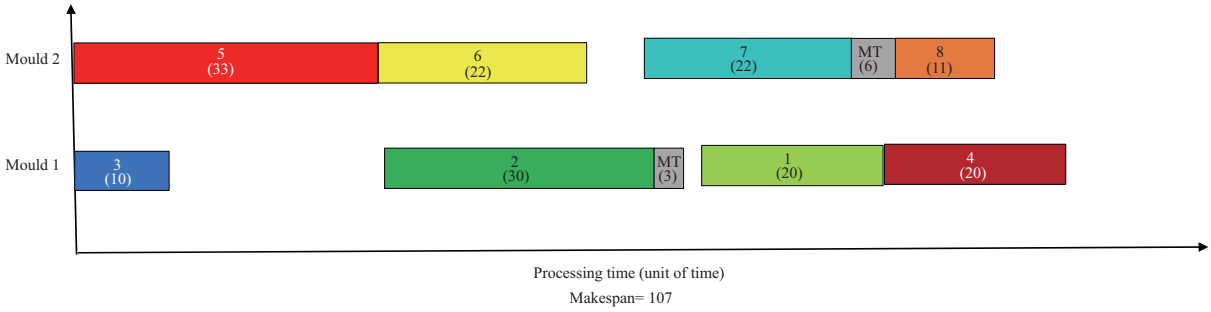

Figure 5 (Color online) Mould scheduling of the example pigeon without the fuzziness.

are shown in Figures 4 and 5, and the makespan is 107. It represents the most possible value in the triangular fuzzy makespan (86, 107, 122). From the comparative results, the solution is still applicable when the fuzzy problem is modified into a normal case, and can illustrate that the rules to calculate

the makespan for the fuzzy problem are appropriate. The solutions obtained by the proposed MOPIO algorithm can also be used for normal cases. However, when the fuzziness is considered in this model, more factors and objectives need to be considered to make it closer to reality.

3.3 Multi-objective pigeon inspired optimization

Inspired by the multi-objective particle swarm optimizer using ring topology proposed by Yue et al. [24], an MOPIO algorithm is proposed using ring topology and a special non-dominated sorting method.

In the map and compass operation of MOPIO, we use the best pigeon in the neighborhood of each pigeon instead of the global best pigeon in order to avoid the population convergencing to a single point. So Eq. (11) is modified into

$$V_i(t) = V_i(t-1)e^{-Rt} + \text{rand}(X_{\text{nbest}_i} - X_i(t-1)), \tag{16}$$

where X_{nbest_i} is the best pigeon in the neighborhood of the i-th pigeon. Other symbols have the same meanings as the symbols in (11). Two archives are established: the personal best archive (PBA) and neighborhood best archive (NBA). The personal best pigeon and the neighborhood best for each pigeon are chosen from the corresponding PBA and NBA. For the NBA, NBA$\{i\}$ denotes the best position within the i-th particle neighborhood. Each neighborhood includes three particles, the i-th particle and its immediate neighbors on its right and left. Moreover, an index-based ring topology is used to build the neighborhood, and pigeons in different neighborhoods cannot interact with each other directly. The use of the NBA promotes the formation of multiple niches by restricting the information transmission through the population. Furthermore, a special sorting scheme, named the non-dominated-scd-sort algorithm, is used to rank the pigeons.

In the landmark operator, the number of the pigeons is chosen as the numbers of pigeons in PBA, and the fitness is defined as $\text{fitness}(X_i(t))=\frac{1}{\text{obj1}_{\min}(X_i(t))+\text{obj2}_{\min}(X_i(t))+\xi}$, where $\text{obj1}_{\min}(X_i(t))+\text{obj2}_{\min}(X_i(t))$ is the sum of two objective values. The positions of all pigeons are updated according to (13)–(15), and pigeons are ranked based on the non-dominated-scd-sort algorithm. When all the iterations end, the best pigeons from the PBA represent the final optimization solutions.

The non-dominated-scd-sort algorithm was proposed by Yue et al. [24], and involves two steps. In the first step, the pigeons are sorted according to the non-dominated sorting scheme [25]. In the second step, the special crowding distances of the non-dominated pigeons are calculated, which also involves two steps. In the first step, the crowding distance (CD), for each pigeon in the decision space and the corresponding image in the objective space are calculated. In the second step, the CDs from the first step are used to assign a special crowding distance (SCD) for each pigeon. The SCD concept involves a max or min selection step that involves crowding metrics from the decision and objective spaces. The diversity in the solution and objective spaces are promoted simultaneously by this methodology. Finally, the non-dominated solutions are ranked in descending order according to their special crowding distances. After sorting, the first particle is the non-dominated solution with the largest SCD.

The flowchart of the proposed MOPIO is shown in Figure 6 and the details of each step are given as follows.

Step 1. Input the parameters of the fuzzy production problem with mould maintenance, including the numbers of jobs, machines, and moulds, the batch size of each job, the corresponding mould of each job, the available machines for each job, the unit fuzzy operation time of each job.

Step 2. Initialize the parameters of MOPIO, including the dimensions of the solution space, the size of the population, NUM, map and compass factor R, the number of iteration Nc_1max and the number of iteration Nc_2max for two operators.

Step 3. Each pigeon is randomly allocated a position and a velocity. Calculate the objective values of all the pigeons. $\text{POP}_i(t)$ represents the pigeon i at the t-th iteration. Initialize the number of elements in the PBA and the NBA. PBA$\{i\}$ saves the best position for pigeon i and NBA$\{i\}$ saves the best position in the neighborhood of the pigeon i. PBA$\{i\} = \text{POP}_i(0)$, NBA$\{i\} = $ PBA$\{i\}$.

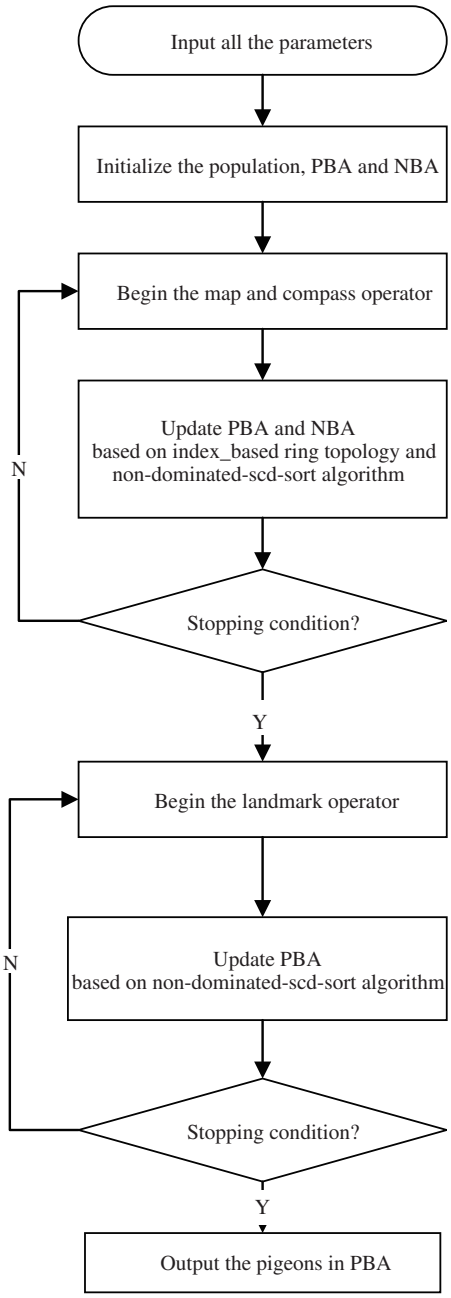

Figure 6 Flowchart of the MOPIO.

Step 4. Begin the map and compass operator. Set $t_1 = 1$. When the iteration t_1 is smaller than $\mathrm{Nc}_1\mathrm{max}$, for all the NUM pigeons, sort the pigeons in PBA$\{i\}$ and NBA$\{i\}$ based on the non-dominated-scd-sort algorithm. Select the first pigeon of the sorted NBA$\{i\}$ as the nbest_i. Update the $\mathrm{POP}_i(t_1)$ according to (16) and (12). Calculate the objective values and the SCD of $\mathrm{POP}_i(t_1 + 1)$.

Step 5. Update PBA. For all the NUM pigeons, put $\mathrm{POP}_i(t_1 + 1)$ into PBA$\{i\}$ and remove all the pigeons dominated by $\mathrm{POP}_i(t_1 + 1)$.

Step 6. Update NBA. Use the index-based ring topology to decide the neighborhood of pigeon i, which includes three pigeon groups, the PBA$\{i\}$, PBA$\{i - 1\}$ and PBA$\{i + 1\}$. Particularly, if $i = 1$, the neighborhood of the pigeon i is defined as PBA$\{\mathrm{NUM}\}$, PBA$\{1\}$ and PBA$\{2\}$; if $i = \mathrm{NUM}$, the neighborhood is defined as PBA$\{\mathrm{NUM} - 1\}$, PBA$\{\mathrm{NUM}\}$ and PBA$\{1\}$. Then obtain the non-dominated pigeons based on the non-dominated-scd-sort algorithm in the neighborhood and put them into the NBA$\{i\}$.

Step 7. Set $t_1 = t_1 + 1$. If the iteration t_1 is smaller than $\mathrm{Nc}_1\mathrm{max}$, return to step 4. If the iteration t_1 is bigger than $\mathrm{Nc}_1\mathrm{max}$, go to step 8.

Step 8. Begin with the landmark operator. Set $t_2 = 1$. When the iteration t_2 is smaller than the $\mathrm{Nc}_2\mathrm{max}$, the landmark operator is activated. The size $N_p(1)$ is chosen as the number of pigeons in the archive PBA at the last iteration of the map and compass operation. The size $N_p(t_2)$ is chosen as the number of pigeons in the archive PBA at iteration t_2. Then the size N_p is decreased by half in every iteration according to (13), which means that half of the pigeons will follow the other half of the pigeons that are familiar with the landmark. The fitness is set as $\mathrm{fitness}(X_i(t)) = \frac{1}{\mathrm{obj1}_{\min}(X_i(t))+\mathrm{obj2}_{\min}(X_i(t))+\xi}$, where $\mathrm{obj1}_{\min}(X_i(t)) + \mathrm{obj2}_{\min}(X_i(t))$ is the sum of two objective values. The center of the pigeons $X_C(t)$ will be calculated according to (14), and the positions of the pigeons will be updated according to (15).

Step 9. Update PBA. Calculate the objective values and the SCD of all the pigeons $\mathrm{POP}_i(t_2+1)$. Put the new pigeons $\mathrm{POP}_i(t_2+1)$ in the PBA and delete the pigeons dominated by $\mathrm{POP}_i(t_2+1)$ based on the non-dominated-scd-sort algorithm.

Step 10. Set $t_2 = t_2 + 1$. If the iteration t_2 is smaller than $\mathrm{Nc}_2\mathrm{max}$, return to step 8. If the iteration t_2 is bigger than $\mathrm{Nc}_2\mathrm{max}$, these pigeons in PBA are taken as the final optimization results.

4 Experimental design

The main objective of the numerical experiments is to test the optimization performance of the proposed MOPIO algorithm. Since the problem is an extension of the problem proposed by Wong et al. [3], the benchmark datasets used in this paper are generated by randomly fuzzifying the crisp benchmarks from Wong et al. [3]. To fuzzy a crisp benchmark dataset and generate a triangular fuzzy processing time $P_{ij} = (P_{ij}^1, P_{ij}^2, P_{ij}^3)$, according to Lei [26], the most plausible value P_{ij}^2 of the fuzzy processing time is equal to the value of the crisp processing time P_{ij} in the crisp datasets. The values of P_{ij}^1 and P_{ij}^2 are randomly generated from $[0.85P_{ij}, 0.95P_{ij}]$ and $[1.1P_{ij}, 1.19P_{ij}]$. The sizes of these three problems are $(30 \times 3 \times 5)$, $(40 \times 6 \times 10)$ and $(60 \times 9 \times 15)$. The quality of the solutions produced by the proposed MOPIO algorithm are verified by comparing the results obtained by adapted NSGA-II [25] and MOPSO [27]. Furthermore, three more randomly generated datasets of sizes $(20 \times 2 \times 4)$, $(35 \times 4 \times 6)$ and $(65 \times 8 \times 10)$ are used as benchmarks to further illustrate the performance of the proposed MOPIO algorithm.

Numerical experiments are implemented in the Matlab environment on a personal computer with Intel (R) Core (TM) i7-6700 CPU 3.40 GHz CPU.

4.1 Performance indicator

To evaluate an algorithm, quantitative analysis is as important as qualitative analysis. Except listing the non-dominated solutions found over a certain number of runs by different algorithms, this study uses two performance metrics to measure the performance of different algorithms: the HV [28] and the CR which is a modification from the cover rate in [24].

The HV metric is used to measure the volume of hypercube enclosed by PF A and a reference vector $r_{\mathrm{ref}} = (r_1, r_2, \ldots, r_n)$ with a larger value representing better performance is calculated as follows:

$$\mathrm{HV}(A) = \bigcup_{a \in A} \mathrm{vol}(a), \tag{17}$$

where $\mathrm{vol}(a)$ is the hypercube volume enclosed by the solution a in the PF A and the reference vector $r_{\mathrm{ref}} = (r_1, r_2, \ldots, r_n)$. The bigger the value, the better the algorithm.

The CR represents the overlap ratio between different PFs obtained by different algorithms. The definition of the $\mathrm{CR}(A, B)$ is as follows:

$$\mathrm{CR}(A, B) = \left(\prod_{l=1}^{n} \delta_l \right)^{1/2n}, \tag{18}$$

Table 2 Levels of different parameters

Level	Swarm size	Max iteration of each operation	R
1	20	100	0.01
2	50	200	0.2
3	80	400	0.4

Table 3 Different parameters combinations and its influence on results

No.	Swarm size	Max iteration of each operation	R	Avg(HV)	Sd(HV)
1	1	1	1	184570	25677
2	1	2	2	183710	52467
3	1	3	3	224810	95112
4	2	1	2	244600	34749
5	2	2	3	240460	54875
6	2	3	1	304520	34394
7	3	1	3	237050	42451
8	3	2	1	264770	24049
9	3	3	2	262280	15366

$$\delta_l = \begin{cases} 1, & F_l^{\max} = F_l^{\min}, \\ 0, & f_l^{\min} \geqslant F_l^{\min} \parallel f_l^{\max} \leqslant F_l^{\min}, \\ \left(\dfrac{\min(f_l^{\max}, F_l^{\max}) - \max(f_l^{\min}, F_l^{\min})}{F_l^{\max} - F_l^{\min}} \right)^2, & \text{otherwise}, \end{cases} \tag{19}$$

where n is the dimensionality of the objective space; $f_l^{\max}$ and $f_l^{\min}$ are respectively the maximum and minimum of the l-th objective value obtained by algorithm A. $F_l^{\max}$ and $F_l^{\min}$ are the maximum and minimum of the l-th objective value obtained by algorithm B. If $\mathrm{CR}(A, B)$ is bigger than $\mathrm{CR}(B, A)$, it means that the scope of the PF obtained by algorithm A is larger than the scope of the PF obtained by algorithm B, which means that algorithm A is better than algorithm B in terms of the CR.

4.2 Parameter tunings

Since the parameters of the algorithm have a considerable influence on the results of the solution, we test different combinations of parameters on a medium size dataset ($40 \times 6 \times 10$) to decide the final parameters for this algorithm. There are three main parameters in the proposed MOPIO: swarm size, the max iteration (the iteration of each operation is half of the overall iteration) and the map and compass factor R. For each parameter, three levels are selected to find an appropriate parameter combination. The details of the parameter sets are shown in Table 2. The Taguchi method is used to reduce the number of experiments. Each parameter combination run ten times. The orthogonal arrays based on Taguchi methods, the average values and the standard deviation of the HV for each parameter combination are shown in Table 3. Based on [29], the impact trend of the different parameters with different levels is shown in Figure 7, where the performance of the proposed algorithm is better when the swarm size is at level 2, the max iteration is at level 3, and the R is at level 1. The larger the max iteration, the better the result, however, a larger swarm size does not mean better results. The smaller the R, the better the result. The swarm size is related to the search ability but it does not mean that a larger size is the better. If the swarm size is too large, there may be a higher repetitive rate of the swarm which decreases the search efficiency. The max iteration can improve the performance under some given conditions and it may increase the search time. The increment of the max iteration, is not useful when the solution reaches its extremum. The map and compass factor R decides the influence of the previous velocity on the present velocity. The smaller the R, the bigger the influence of the previous velocity on the present velocity. Based on the above analysis, the parameter combination {swarm size = 50, iteration = 400 and $R = 0.01$} is chosen as the parameter used for the following tests.

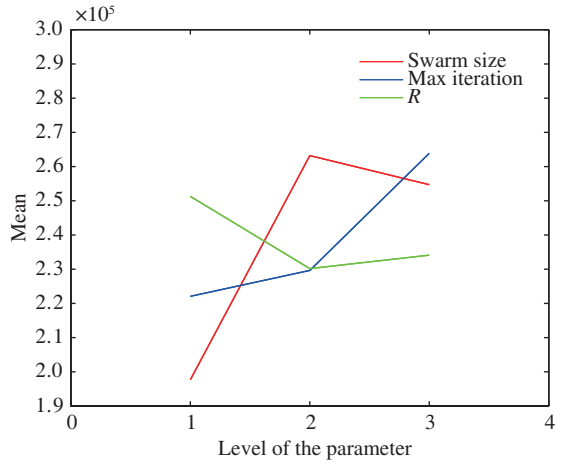

Figure 7 (Color online) Factor level trend.

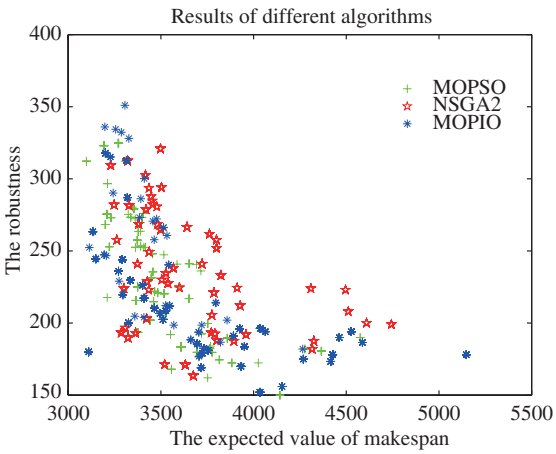

Figure 8 (Color online) Pareto fronts of the fuzzed benchmark ($30 \times 3 \times 5$).

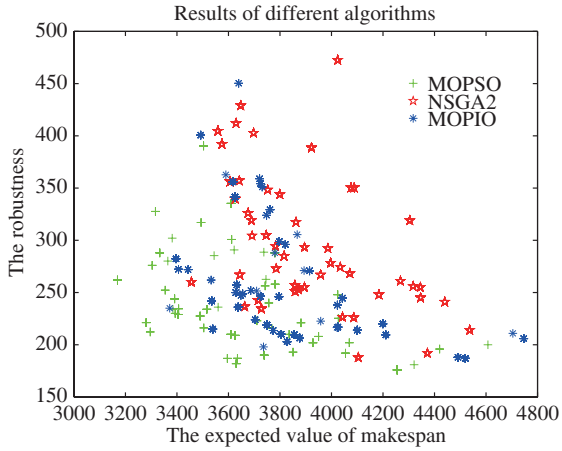

Figure 9 (Color online) Pareto fronts of the fuzzed benchmark ($40 \times 6 \times 10$).

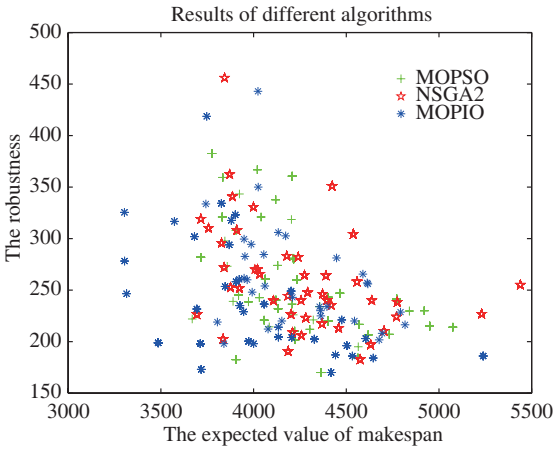

Figure 10 (Color online) Pareto fronts of the fuzzed benchmark ($60 \times 9 \times 15$).

4.3 Comparison with other multi-objective algorithms

To test the performance of the proposed MOPIO, this research compares it with other two well-known multi-objective algorithms: adapted NSGA-II [25] and MOPSO [27]. For unbiased comparison, the swarm size is set as 50, and the max iteration is set as 400 for these three algorithms. The parameters related to the MOPSO and NSGA-II are the same as the parameters in the original literature. The R is set as 0.01 for MOPIO. The reference vector for the calculation of HV is set as (5200, 400), and each algorithm is run ten times. For the fuzzed benchmark datasets, the PFs of three algorithms are shown in Figures 8–10. The comparison results of the performance indicators (the HV and the CR) are shown in Tables 4 and 5. Avg(HV) is the average value of the HV for the 10 runs, SD(HV) is the standard deviation of the HV for the 10 runs and Avg(CR) is the average value of the CR for the 10 runs. SD(CR) is the standard deviation of the CR for the 10 runs.

Furthermore, three more instances are produced randomly to better illustrate the performance of the proposed MOPIO algorithm. The size of these three problems are $(20\times2\times4)$, $(35\times4\times6)$ and $(65\times8\times10)$. The most plausible value P_{ij}^2 of the fuzzy processing time is produced randomly between 30 and 55 units of time. The values of P_{ij}^1 and P_{ij}^3 are randomly generated from $[0.85P_{ij}^2, 0.95P_{ij}^2]$ and $[1.1P_{ij}^2, 1.19P_{ij}^2]$, and the batch size of the jobs is produced randomly between 2 and 6 units. The comparison results of

Table 4 HV comparison results of the fuzzed benchmarks

		MOPIO	NSGA-II	MOPSO
$30 \times 3 \times 5$	Avg(HV)	410140	369320	394800
	Sd(HV)	35771	38432	34513
$40 \times 6 \times 10$	Avg(HV)	296680	227000	340670
	Sd(HV)	33643	36029	42257
$60 \times 9 \times 15$	Avg(HV)	260000	198240	225760
	Sd(HV)	66773	60707	45271

Table 5 CR comparison results of the fuzzed benchmarks

		MOPIO vs. NSGA-II	NSGA-II vs. MOPIO	MOPIO vs. MOPSO	MOPSO vs. MOPIO
$30 \times 3 \times 5$	Avg(CR)	0.7019	0.3904	0.7762	0.6362
	Sd(CR)	0.3260	0.2850	0.3016	0.2784
$40 \times 6 \times 10$	Avg(CR)	0.5584	0.4338	0.5456	0.5219
	Sd(CR)	0.2912	0.2627	0.2530	0.3083
$60 \times 9 \times 15$	Avg(CR)	0.5519	0.4986	0.71	0.5342
	Sd(CR)	0.334	0.3163	0.2967	0.34

Table 6 HV comparison results of the random benchmarks

		MOPIO	NSGA-II	MOPSO
$20 \times 2 \times 4$	Avg(HV)	370120	259960	420310
	Sd(HV)	42687	52709	44402
$35 \times 4 \times 6$	Avg(HV)	257290	115370	117340
	Sd(HV)	15086	26714	23554
$65 \times 8 \times 10$	Avg(HV)	357070	210300	337900
	Sd(HV)	24190	38107	36947

Table 7 CR comparison results of the random benchmarks

		MOPIO vs. NSGA-II	NSGA-II vs. MOPIO	MOPIO vs. MOPSO	MOPSO vs. MOPIO
$20 \times 2 \times 4$	Avg(CR)	0.6842	0.4734	0.7957	0.7018
	Sd(CR)	0.3189	0.2486	0.1846	0.2225
$35 \times 4 \times 6$	Avg(CR)	0.8664	0.2704	0.5145	0.5269
	Sd(CR)	0.1158	0.2374	0.4783	0.4566
$65 \times 8 \times 10$	Avg(CR)	0.7290	0.3524	0.5578	0.5319
	Sd(CR)	0.3821	0.3453	0.3433	0.3339

the performance indicators are shown in Tables 6 and 7.

In the proposed MOPIO algorithm, the neighborhood best is used instead of the global best in the original PIO algorithm when the velocity is updated. To explain the influence of this modification, we compare the proposed MOPIO algorithm which uses the neighbourhood best with another version of MOPIO named MOPIO-GBA which uses the global best. In the MOPIO-GBA, the global best archive (GBA) is built. In each iteration of the map and compass operation, we add the first pigeon in every PBA$\{i\}$ based on the ranking by the non-dominated-scd-sort algorithm into the GBA. Then, all the pigeons in the GBA are ranked based on the non-dominated-scd-sort algorithm and the global best is selected as the first pigeon in the GBA. Except for the velocity updating formula, other steps in the MOPIO-GBA are the same as the proposed MOPIO. The above six instances are used to test the performance of MOPIO-GBA and MOPIO. The comparison results of the performance indicators are shown in Tables 8 and 9.

Moreover, a solution of the instance $20 \times 2 \times 4$ is shown in Figures 11 and 12. For the instance $20 \times 2 \times 4$, the batch sizes of these 20 jobs are (5 3 4 6 2 5 6 6 5 4 3 4 3 6 5 4 3 4 5 4) and the moulds used by these jobs are (2 3 4 2 3 4 1 4 4 2 4 1 3 2 4 3 2 4 3 1). The unit fuzzy processing times by different moulds are shown in Table 10. The maintenance time is based on the relationship in Table 1. The numbers in

Table 8 HV comparison results of MOPIO and MOPIO-GBA

		MOPIO	MOPIO-GBA
$20 \times 2 \times 4$	Avg(HV)	370120	174130
	Sd(HV)	42687	11377
$30 \times 3 \times 5$	Avg(HV)	410140	160680
	Sd(HV)	35771	11802
$35 \times 4 \times 6$	Avg(HV)	257290	126860
	Sd(HV)	42687	59960
$40 \times 6 \times 10$	Avg(HV)	296680	157030
	Sd(HV)	33643	48312
$60 \times 9 \times 15$	Avg(HV)	260000	103150
	Sd(HV)	66773	51170
$65 \times 8 \times 10$	Avg(HV)	357070	276271
	Sd(HV)	24190	25374

Table 9 CR comparison results of MOPIO and MOPIO-GBA

		MOPIO vs. MOPIO-GBA	MOPIO-GBA vs. MOPIO
$20 \times 2 \times 4$	Avg(CR)	0.6571	0.3122
	Sd(CR)	0.4645	0.1836
$30 \times 3 \times 5$	Avg(CR)	0.3667	0.0989
	Sd(CR)	0.5507	0.1713
$35 \times 4 \times 6$	Avg(CR)	0.3490	0.1724
	Sd(CR)	0.5643	0.2529
$40 \times 6 \times 10$	Avg(CR)	0.4713	0.4111
	Sd(CR)	0.4417	0.4070
$60 \times 9 \times 15$	Avg(CR)	0.5360	0.4029
	Sd(CR)	0.3725	0.3071
$65 \times 8 \times 10$	Avg(CR)	0.8996	0.3242
	Sd(CR)	0.1273	0.0963

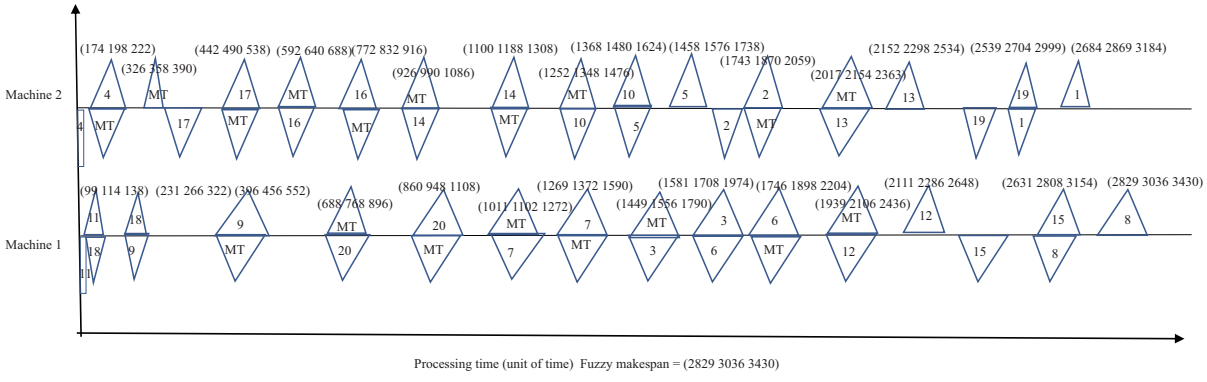

Figure 11 (Color online) The machine scheduling for the instance $20 \times 2 \times 4$.

the triangles are the job numbers, and the triangles under the line represent the starting time of each job. The triangles above the line represent the ending time of each job. MT means maintenance on the machine or the mould. After decoding, the solution is (4 17 16 11 14 10 18 9 5 20 7 3 6 2 13 12 15 19 8 1 2 2 2 1 2 2 1 1 2 1 1 1 1 2 2 1 1 2 1 2 1 1 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1 0 1 1 1). To make the figures clearer, only the ending time of each job is shown.

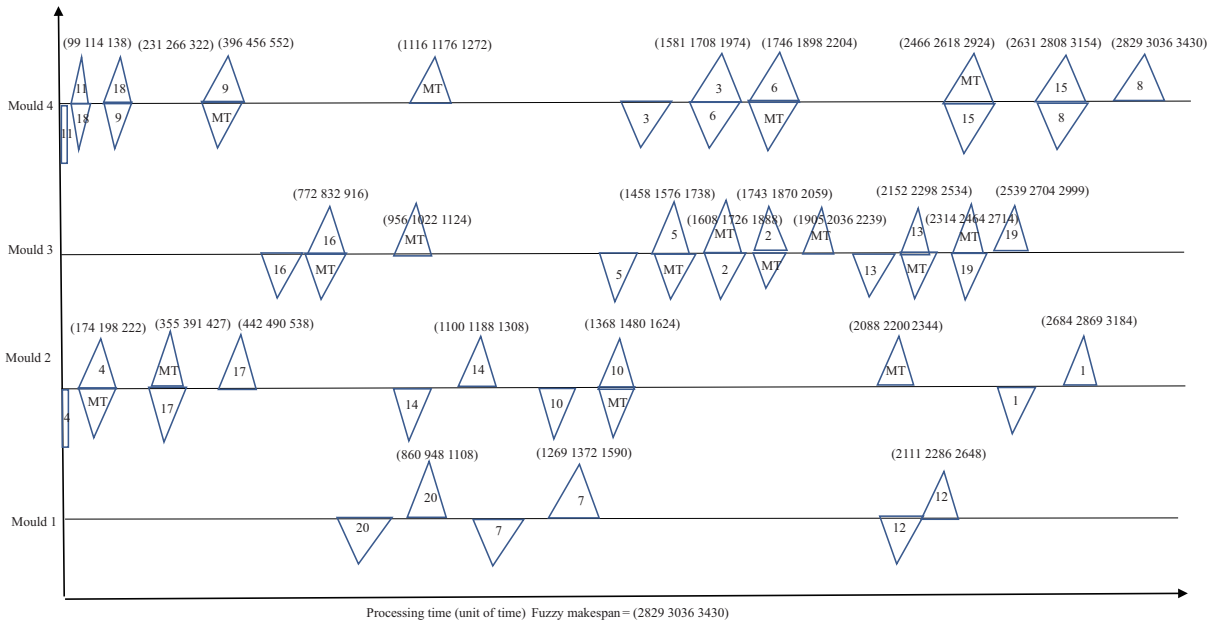

Figure 12 (Color online) The mould scheduling for the instance $20 \times 2 \times 4$.

Table 10 The unit fuzzy processing time of different moulds for instance $20 \times 2 \times 4$

	Machine 1	Machine 2
Mould 1	(43 45 53)	(43 45 53)
Mould 2	(0 0 0)	(29 33 37)
Mould 3	(0 0 0)	(45 48 57)
Mould 4	(33 38 46)	(33 38 46)

5 Results and discussion

From Figure 8, the PF of the dataset $(30\times3\times5)$ by MOPIO is in the left bottom of the coordinate system compared with the PFs by MOPSO and NSGA-II. From the Table 4, the dataset $(30\times3\times5)$, the value of Avg(HV) by MOPIO, is bigger than the values of Avg(HV) by MOPSO and NSGA-II. Furthermore, the boundary of the PF by MOPIO is larger than the PFs by MOPSO and NSGA-II. From Table 5, the value of the Avg(CR(MOPIO, NSGA-II)) is bigger than the value of the Avg(CR(NSGA-II, MOPIO)) and the value of the Avg(CR(MOPIO, MOPSO)) is bigger than the value of the Avg(CR(MOPSO, MOPIO)). It can be concluded that for the dataset $(30 \times 3 \times 5)$, the quality of the PF by MOPIO is better than the PFs by MOPSO and NSGA-II. For the dataset $(60 \times 9 \times 15)$, by comparing the location of PFs in the Figure 10 and the values for dataset $(60 \times 9 \times 15)$ in Tables 4 and 5, it can be concluded that for the dataset $(60\times9\times15)$, the quality of the PF by MOPIO is better than the PFs by MOPSO and NSGA-II based on the two performance indicators.

From Figure 9, although the PF by MOPSO is in the left bottom of the coordinate system compared with the PFs by MOPIO and NSGA-II, the boundary of the PF by MOPIO is larger than the PFs by MOPSO and NSGA-II. Turning back to the values in Tables 4 and 5, it can be seen that for the dataset $(40\times6\times10)$, the value of Avg(HV) of the PF by MOPSO is bigger than the values of the PFs by MOPIO and NSGA-II. However, the value of the Avg(CR(MOPIO, MOPSO)) is bigger than the value of the Avg(CR(MOPSO, MOPIO)) and the value of the Avg(CR(MOPIO, NSGA-II)) is bigger than the value of the Avg(CR(NSGA-II, MOPIO)).

From Tables 6 and 7, it can be known that the value of the Avg(HV) for MOPIO is better than the values of Avg(HV) for MOPSO and NSGA-II for the bigger datasets which are randomly generated. However, for the small dataset, the value of the Avg(HV) for MOPSO is better than the value of the Avg(HV) for MOPIO and NSGA-II. For these three datasets, the value of the Avg(CR(MOPIO, MOPSO)) is bigger

than the value of the Avg(CR(MOPSO, MOPIO)) and the value of the Avg(CR(MOPIO, NSGA-II)) is bigger than the value of the Avg(CR(NSGA-II, MOPIO)).

Furthermore, from the Tables 8 and 9, it can be seen that for all these six instances, the proposed MOPIO using the neighbourhood best best for each pigeon always has a better performance than the MOPIO using the global best pigeon in the map and compass operation, in terms of the CR and the HV.

After analyzing the PFs of the different algorithms in different aspects, we can conclude that the PFs obtained by MOPIO always have a better CR compared with MOPSO and NSGA-II. For most of the datasets, the HV of the solutions obtained by MOPIO is better than MOPSO and NSGA-II. Moreover, it can be seen that the modification of the velocity updating equation in the map and compass operation is effective after comparing the proposed MOPIO with neighbourhood best with MOPIO with the global best. Because of the mechanism of the index-based ring topology used in the MOPIO, more Pareto-optimal solutions are located compared with using the global best pigeon for all the pigeons. Furthermore, since stable niches are induced by the ring topology in decision space, each pigeon is able to advance in its own niche. In each niche, a pioneer is selected and if these pioneers have a good distribution, there is a high possibility that more Pareto-optimal solutions can be located. Moreover, the non-dominated-scd-sort algorithm applied in the MOPIO for sorting the pigeons helps in choosing the pigeons with better performance and distribution in every iteration. The solutions which are less crowded have more opportunities to survive. At the same time, if solutions are near each other in the objective space but not crowded in the decision space, they also have opportunities to be reserved, based on the ranking algorithm. The diversity of the population is improved by the ring topology and the non-dominated-scd-sort algorithm used in the proposed algorithm [24], which makes it have a better performance when dealing with the multi-objective problem.

6 Conclusion

In this research, the FPSP-MM is studied. The processing time and the maintenance time are represented by triangular fuzzy numbers. Two objectives are optimized, the fuzzy makespan and the robustness. An MOPIO algorithm is proposed to solve this multi-objective fuzzy problem. To extend the basic PIO algorithm from the single-objective case to the multi-objective case, a special non-dominated sorting method is used to obtain solutions that are used as candidates for the leader pigeon, and a good distribution of solutions in the objective space and in the corresponding decision space is guaranteed. Moreover, we make each pigeon exchange information with its closed neighbors, instead of the global best pigeon, with the help of index-based ring topology. The diversity of the population is improved by forming more niches. Furthermore, a series of experiments on the fuzzified benchmarks from existing literature and some randomly generated instances show the efficiency and effectiveness of the proposed MOPIO algorithm by comparing it with other algorithms.

In this research, all the jobs were well prepared at the beginning time, however, in real cases, some new jobs may arrive during the production process. When the new tasks are considered, the original schedule needs to be adjusted to obtain better solutions. In the future, a new algorithm based on PIO should be proposed to solve this problem. Furthermore, more mechanisms should be found to improve the effectiveness of the proposed MOPIO algorithm.

Acknowledgements This work was supported by Research Grants Council of the Hong Kong Special Administrative Region, China (Grant No. PolyU 15201414), National Natural Science Foundation of China (Grant Nos. 71471158, 71571120, 71271140), Research Committee of the Hong Kong Polytechnic University under Student Account Code RUKH, Project Supported by Guangdong Province Higher Vocational Colleges and Schools Pearl River Scholar Funded Scheme 2016, and Project of Innovation and Entrepreneurship Education Research Center for University Student of Guangdong Province (Grant No. 2018A073825).

References

1 Rajkumar M, Asokan P, Vamsikrishna V. A GRASP algorithm for flexible job-shop scheduling with maintenance constraints. Int J Prod Res, 2010, 48: 6821–6836
2 Berrichi A, Yalaoui F, Amodeo L, et al. Bi-objective ant colony optimization approach to optimize production and maintenance scheduling. Comput Oper Res, 2010, 37: 1584–1596
3 Wong C S, Chan F T S, Chung S H. A genetic algorithm approach for production scheduling with mould maintenance consideration. Int J Prod Res, 2012, 50: 5683–5697
4 Wong C S, Chan F T S, Chung S H. A joint production scheduling approach considering multiple resources and preventive maintenance tasks. Int J Prod Res, 2013, 51: 883–896
5 Wong C S, Chan F T S, Chung S H. Decision-making on multi-mould maintenance in production scheduling. Int J Prod Res, 2014, 52: 5640–5655
6 Wang S J, Liu M. Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. J Manuf Syst, 2015, 37: 182–192
7 Shen L, Yang H B, Gao S, et al. Production scheduling with mould maintenance in flow shop. In: Proceedings of the 4th International Conference on Sensors, Mechatronics and Automation, Zhuhai, 2016. 730–733
8 Sakawa M, Mori T. An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate. Comput Ind Eng, 1999, 36: 325–341
9 Arik O A, Toksari M D. Multi-objective fuzzy parallel machine scheduling problems under fuzzy job deterioration and learning effects. Int J Prod Res, 2018, 56: 2488–2505
10 Jamrus T, Chien C F, Gen M, et al. Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing. IEEE Trans Semicond Manuf, 2018, 31: 32–41
11 Palacios J J, González-Rodríguez I, Vela C R, et al. Robust multiobjective optimisation for fuzzy job shop problems. Appl Soft Comput, 2017, 56: 604–616
12 Xiong J, Xing L N, Chen Y W. Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. Int J Prod Econ, 2013, 141: 112–126
13 Duan H B, Qiao P X. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. Int J Intel Comput Cyber, 2014, 7: 24–37
14 Li C, Duan H B. Target detection approach for UAVs via improved pigeon-inspired optimization and edge potential function. Aerosp Sci Tech, 2014, 39: 352–360
15 Zhang B, Duan H B. Three-dimensional path planning for uninhabited combat aerial vehicle based on predator-prey pigeon-inspired optimization in dynamic environment. IEEE/ACM Trans Comput Biol Bioinf, 2017, 14: 97–107
16 Duan H B, Wang X H. Echo state networks with orthogonal pigeon-inspired optimization for image restoration. IEEE Trans Neural Netw Learn Syst, 2016, 27: 2413–2425
17 Deng Y M, Duan H B. Control parameter design for automatic carrier landing system via pigeon-inspired optimization. Nonlinear Dyn, 2016, 85: 97–106
18 Zhang S J, Duan H B. Gaussian pigeon-inspired optimization approach to orbital spacecraft formation reconfiguration. Chinese J Aeronaut, 2015, 28: 200–205
19 Qiu H X, Duan H B. Multi-objective pigeon-inspired optimization for brushless direct current motor parameter design. Sci China Technol Sci, 2015, 58: 1915–1923
20 Qiu H X, Duan H B. A multi-objective pigeon-inspired optimization approach to UAV distributed flocking among obstacles. Inf Sci, 2018, doi: 10.1016/j.ins.2018.06.061
21 Fortemps P. Jobshop scheduling with imprecise durations: a fuzzy approach. IEEE Trans Fuzzy Syst, 1997, 5: 557–569
22 Bean J C. Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput, 1994, 6: 154–160
23 Tasgetiren M F, Liang Y C, Sevkli M, et al. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res, 2007, 177: 1930–1947
24 Yue C T, Qu B Y, Liang J. A multi-objective particle swarm optimizer using ring topology for solving multimodal multi-objective problems. IEEE Trans Evol Comput, 2018, 22: 805–817
25 Deb K, Pratap A, Agarwal S, et al. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput, 2002, 6: 182–197
26 Lei D M. Scheduling fuzzy job shop with preventive maintenance through swarm-based neighborhood search. Int J Adv Manuf Technol, 2011, 54: 1121–1128
27 Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput, 2004, 8: 256–279
28 Zitzler E, Thiele L, Laumanns M, et al. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput, 2003, 7: 117–132
29 Li J Q, Pan Q K, Mao K, et al. Solving the steelmaking casting problem using an effective fruit fly optimisation algorithm. Knowledge-Based Syst, 2014, 72: 28–36