Int J Adv Manuf Technol (2016) 85:1513–1528
DOI 10.1007/s00170-015-8037-7

ORIGINAL ARTICLE

A multi-objective memetic algorithm for integrated process planning and scheduling

Liangliang Jin[1,2] · Chaoyong Zhang[1,2] · Xinyu Shao[1,2] · Xudong Yang[3] · Guangdong Tian[4]

Received: 24 June 2015 / Accepted: 25 October 2015 / Published online: 10 November 2015

Abstract Process planning and scheduling are two crucial components in a manufacturing system. The integration of the two functions has an important significance on improving the performance of the manufacturing system. However, integrated process planning and scheduling is an intractable non-deterministic polynomial-time (NP)-hard problem, and the multiple objectives requirement widely exists in real-world production situations. In this paper, a multi-objective mathematical model of integrated process planning and scheduling is set up with three different objectives: the overall finishing time (makespan), the maximum machine workload (MMW), and the total workload of machines (TWM). A multi-objective memetic algorithm (MOMA) is proposed to solve this problem. In MOMA, all the possible schedules are improved by a problem-specific multi-objective local search method, which combines a variable neighborhood search (VNS) procedure and an effective objective-specific intensification search method. Moreover, we adopt the TOPSIS method to select a satisfactory schedule scheme from the optimal Pareto front. The proposed MOMA is tested on typical benchmark instances and the experimental results are compared with those obtained by the well-known NSGA-II. Computational results show that MOMA is a promising and very effective method for the multi-objective IPPS problem.

Keywords Integrated process planning and scheduling (IPPS) · Multi-objective optimization · Multi-objective memetic algorithm · Variable neighborhood search · Intensification search

✉ Chaoyong Zhang
zcyhust@hust.edu.cn

[1] School of Mechanical Science and Engineering, Huazhong University of Science & Technology, Wuhan 430074, China

[2] The State Key Laboratory of Digital Manufacturing Equipment & Technology, Huazhong University of Science & Technology, 1037 Luoyu Road, Wuhan, China

[3] School of Mechanical Engineering, Guizhou University, Guiyang 550025, China

[4] Transportation College, Northeast Forestry University, Harbin 150040, China

1 Introduction and motivation

Process planning and scheduling are two key components in a manufacturing system [1–4]. Process planning translates the design data into the best method to manufacture a part [2, 3, 5–8]. It determines the optimal process plan (operations and their sequence within the precedence relationship constraints) and other parameters [9, 10], and bridges the gap between product designing and manufacturing [11]. In general, a job can have more than one possible process plans. Scheduling attempts to assign manufacturing resources optimally to produce qualified parts with certain criteria. In a schedule, all the operations should follow a certain precedence relationship specified by the process plan. Obviously, the two functions are interrelated but not overlapped: process planning gives the technological constraints for the scheduling process. Although there is a close relationship between process planning and scheduling, these two functions are usually performed sequentially [12–16]: the input of scheduling is the output of process planning. Limitations of this sequential paradigm have been identified and well discussed in [13, 17, 18]. However, the critical failing of the sequential paradigm appears: a predetermined process plan may not be the best one for the scheduling.

The investigation in this area starts from deep researches and applications of the mono-objective integrated process planning and scheduling (IPPS) problem with makespan criterion. These solution methodologies for the mono-objective IPPS problem can be classified into two categories: mathematical programming approaches and approximation approaches, such as meta-heuristic algorithms and artificial intelligence methods. Özgüven et al. proposed mixed integer linear programming (MILP) models [19]. However, their MILP models are suitable for small-scale instances only. For a complex instance, one cannot put up with the excessive CPU time. The other excellent MILP model was presented in [20]. The authors of that paper transformed the nonlinear model into a MILP model. However, their model is also suitable for small scale instances only. As with other non-deterministic polynomial-time (NP)-hard problems, meta-heuristic based algorithms have garnered tremendous attentions and have been adopted widely for the mono-objective IPPS problem due to their impressive features, such as short computational time, without problem self-related information, and strong versatility. Kim et al. are among the pioneers who study the IPPS problem in depth. In particular, they devised a set of representative benchmark instances (Kim's benchmark) which are further solved by their developed symbiotic evolutionary algorithm (SEA) considering the makespan objective [14]. In their paper, they also reported that the suggested symbiotic evolutionary algorithm (SEA) is better than the cooperative co-evolutionary genetic algorithm (CCGA). Later, Shao et al. suggested a modified GA-based approach for solving small-scale IPPS instances [21]; in their model, process planning and scheduling functions were carried out simultaneously. Lian et al. adopted a novel meta-heuristic algorithm, the imperialist competitive algorithm (ICA), to settle this problem [13] although their results were not quite promising for Kim's benchmark. Li et al. developed an effective GA based hybrid algorithm [12] for the IPPS problem; a tabu search (TS) with an effective neighborhood structure [22] was employed to improve the performance of the hybrid algorithm. Later, they introduced an active learning effect into GA and a novel active learning genetic algorithm (ALGA) was developed [23]. Qiao and Lv suggested an improved genetic algorithm (IGA) for this problem, where new genetic representations and genetic operators were developed [24]. Recently, Zhang and Wong [25] reported an object-coding genetic algorithm for the IPPS problem, and very promising results have been obtained. Except for the algorithms already mentioned, other approaches adopted for the IPPS problem include the grammatical optimization approach [26], the combined Genetic Algorithm (GA) and Fuzzy Neural Network (FNN) approach [27], the simulated annealing-based optimization approach [28], the modified particle swarm optimization (PSO) algorithm [29], the honey bee mating optimization (HBMO) algorithm [30], scenario based meta-heuristic approach [31], and agent-based methods [32].

It is worth mentioning that the above research papers concentrate on mono-objective algorithms. Usually, only the makespan criterion is not sufficient to capture a desirable scheduling scheme. Due to the multiple objectives requirement in real-world production situations, it is highly necessary to consider the multi-objective IPPS problem. To deal with a multi-objective optimization problem, a primary manner is to aggregate various objectives into a weighted sum. A sophisticated approach is to treat multiple objectives in a Pareto manner because it can make trade-offs between conflicting objectives. Moreover, the resultant Pareto front brings convenience for decision makers (DM). Recently, Mohapatra et al. reported a controlled elitist non-dominated sorting genetic algorithm [9]; their algorithm was developed from the well-known NSGA-II [33] and the number of individuals to be selected as new parent in the currently best Pareto front was restricted. They show that their algorithm is computationally more efficient than NSGA-II. Another novel algorithm to deal with the multi-objective IPPS problem is suggested in [18]; the Nash equilibrium in game theory was introduced to deal with the multiple objectives. However, both the two papers address small- or medium-scale IPPS problems only, and promising solutions for large-scale instances have not been reported.

Unfortunately, as one shortcoming of the previous algorithms, these algorithms so easily get stuck into local optima that high quality solutions are hardly acquired. Meanwhile, it comes to our mind that existing approaches regarding the multi-objective IPPS problem do not take effective local search into account: although delicately designed, they cannot capture promising solutions (Pareto fronts) especially for large-scale instances. Moreover, objective-specific intensification search methods have not been developed to take every criterion in to account. For example, Li et al. employed a tabu search (TS) in [18] for improving the makespan criterion only, and they did not develop effective local search methods for other two objectives.

To overcome these drawbacks, in this paper, the integration of process planning and scheduling is achieved through a multi-objective memetic algorithm (MOMA). We consider three objectives in capturing the optimal Pareto front: the overall finishing time (makespan), the maximum machine workload (MMW), and the total workload of machines (TWM). For the makespan criterion, individuals are improved by VNS while the developed intensification search method aims to improve the MMW and the TWM criteria. Benefitting from the objective-specific intensification search method and with the assistance of the effective VNS local search method, the MOMA can achieve a high solution quality. In addition, the TOPSIS decision supporting method [34] is introduced to obtain the most satisfactory scheduling scheme.

The remainder of this paper is organized as follows. "Preliminaries" section gives the definition of the multi-objective IPPS problem with a MILP model. The details of all the components of MOMA are described in "The multi-objective memetic algorithm (MOMA)" section. "Computational results with discussions" section provides computational results with discussions. Finally, conclusions and directions for future study will be given in "Conclusions" section.

2 Preliminaries

The IPPS problem can be defined as [29]: Given a set of n parts (jobs) to be processed on m machines with operations including alternative manufacturing resources, select the suitable manufacturing resources and sequence the operations so as to determine a schedule in which the precedence constraints among operations can be satisfied and the corresponding objectives can be achieved.

Jobs to be processed in the IPPS problem are represented by network graphs [11] as illustrated in Fig. 1a. The starting node and the ending node are dummy ones and, respectively, indicate the start and the completion of the manufacturing process of a job. Other nodes are operation nodes (marked with numbers) and OR nodes (also OR connectors, marked with "OR"). An operation node contains the alternative machines that can perform the operation and processing times required for the operation according to the machines. For operation 6 in Fig. 1a, it can be processed by machine 1 or 5 with processing times 42 or 38. An arrow coming from node A to node B implies that operation B should follow operation A directly or indirectly. An operation path that begins at an OR node and ends as it merges with other paths is called an OR link path. Only one of the OR link path out of the two is needed to be traversed. For the network of the job in Fig. 1a, paths (9-10-11) and (12-13) are two OR link paths. Based on the OR link paths, two (operation) combinations can be generated as presented in Fig. 1b and c. Possible process plans, e.g., $(7\rightarrow1\rightarrow8\rightarrow4\rightarrow2\rightarrow12\rightarrow3\rightarrow5\rightarrow13\rightarrow6\rightarrow14)$ and $(1\rightarrow2\rightarrow3\rightarrow4\rightarrow5\rightarrow6\rightarrow7\rightarrow8\rightarrow9\rightarrow10\rightarrow11\rightarrow14)$, can be obtained according to the two combinations as long as the operation precedence relationships in process plans are satisfied.

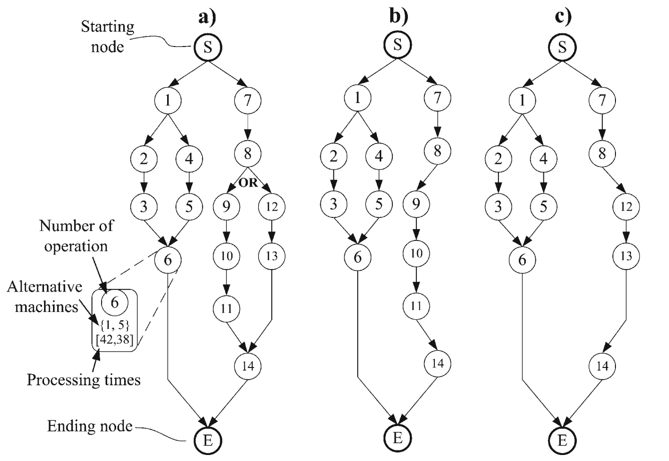

Fig. 1 Network representations

Traditional job shop scheduling problems generally assume that there is a single feasible process plan for each job, and each job has a fixed operation sequence (e.g., $O_1\rightarrow O_2\rightarrow O_3\rightarrow\ldots$) [35]. This implies that operation O_{i+1} should be scheduled after O_i and before O_{i+2} in any case. However, the IPPS problem is a more flexible problem because most jobs may have a large number of feasible process plans that rest on different operation permutations. Specifically, the precedence relationship of some operations is not fixed (e.g., operations 2 and 4 in Fig. 1a). Besides, some operations may not be selected in a process plan if an OR link path is not selected. The IPPS problem can be regarded as the flexible job shop scheduling problem (FJSP) plus various routings, and hence, the problem is a strong non-deterministic polynomial-time (NP)-hard problem. The characteristics of the IPPS problems are presented in the comprehensive review in [36]. For clarity, we provide here a mathematic model of the multi-objective IPPS problem since a mathematic model can more or less express the nature and important characteristics of the problem. Further discussion on this problem is based on the following assumptions:

1. Jobs and machine tools are independent and available at time zero.
2. Job preemption is not allowed, and each machine can handle only one job at a time. Different operations of one job cannot be processed simultaneously.
3. Once a job is finished, it will be immediately transferred to another machine with transportation time neglected.
4. Setup time of an operation is included in the processing time.

Subscripts, notations, and sets:

i,i'	jobs, $1\leq i\leq\|n\|$
j,j'	operations
k,k'	machines, $1\leq k\leq\|M\|$
l,l'	process plans
O_{ijl}	the j-th operation of job i using the l-th process plan of the job
n	the set of jobs
M	the set of all the machines
T_i	the set of process plans of job i
NO_{il}	the set of operations for l-th process plan of job i
R_{ijl}	the set of available machines of O_{ijl}

Parameters:

p_{ijkl}	the processing time of O_{ijl} on machine k
A	a very large positive integer

Variables:

$C_{\max}$	makespan
TWM	the total workload of machines
MMW	the maximum machine workload

$$X_{il} = \begin{cases} 1, \text{ the } l\text{-th process plan of job } i \text{ is selected} \\ 0, \text{ otherwise} \end{cases},$$

$$Z_{ijkl} = \begin{cases} 1, \text{ if } O_{ijl} \text{ is processed on machine } k \\ 0, \text{ otherwise} \end{cases},$$

$$Y_{ijli'j'l'} = \begin{cases} 1, \text{ if } O_{ijl} \text{ is processed directly or indirectly after } O_{i'j'l'} \\ 0, \text{ otherwise} \end{cases},$$

C_{ij} the completion time of the j-th operation of job i

Objective:

$$\min C_{\max} \tag{1}$$

$$\min TWM = \sum_{i\in n}\sum_{l\in T_i}\sum_{j\in NO_{il}}\sum_{k\in R_{ijl}} Z_{ijkl}p_{ijk} \tag{2}$$

$$\min MMW = \max_{k\in M}\left(\sum_{i\in n}\sum_{l\in T_i}\sum_{j\in NO_{il}} Z_{ijkl}p_{ijk}\right) \tag{3}$$

s.t.

$$\sum_{l\in T} X_{il} = 1, \quad \forall i\in n \tag{4}$$

$$\sum_{k\in R_{ijl}} Z_{ijkl} + (1-X_{il}) = 1, \quad \forall i\in n,\ l\in T_i,\ j\in NO_{il} \tag{5}$$

$$C_{i0} = 0, \quad \forall i\in n \tag{6}$$

$$C_{ij} \ge C_{ij-1} + \sum_{k\in R_{ijl}} p_{ijkl}Z_{ijkl}, \quad \forall i\in n,\ l\in T_i,\ j\in NO_{il},\ j\ge 1 \tag{7}$$

$$\begin{aligned} &C_{ij} \ge C_{i'j'} + p_{ijkl} - A\left(3-Y_{ijli'j'l'}-Z_{ijkl}-Z_{i'j'kl'}\right), \\ &\forall i\in n,\ i<|n|,\ i'>i,\ l\in T_i,\ l'\in T_{i'},\ j\in NO_{il},\ j'\in NO_{i'l'},\ k\in R_{ijl}\cap R_{i'j'l'} \end{aligned} \tag{8}$$

$$\begin{aligned} &C_{i'j'} \ge C_{ij} + p_{i'j'kl'} - A\left(2+Y_{ijli'j'l'}-Z_{ijkl}-Z_{i'j'kl'}\right) \\ &\forall i\in n,\ i<|n|,\ i'>i,\ l\in T_i,\ l'\in T_{i'},\ j\in NO_{il},\ j'\in NO_{i'l'},\ k\in R_{ijl}\cap R_{i'j'l'} \end{aligned} \tag{9}$$

$$C_{\max} \ge C_{ij}, \quad \forall i,\ j\in NO_{il} \tag{10}$$

In the model, the objectives are formulated by Eqs. (1–3). According to constraint set (4), one and only one process plan is selected for each job. For each operation that has been selected, it should be assigned to a machine. This point is reflected by constraint set (5). Constraint sets (6) and (7) are used to determine the starting time of each operation of a job. Constraint set (7) ensures that an operation of a job should be processed after its job predecessor is completed. The index j in constraint set (6) starts at 0 to accommodate the term C_{ij-1} in constraint set (7). Thus, constraint sets (6) and (7) also mean that every job can only be processed at time 0 or later. Constraint sets (8) and (9) schedule different operations on the same machine. Note that constraint sets (4) and (5) are very important because they prevent unnecessary variables from taking value 1. Thus, there are only two subscripts in the variable C_{ij}. Constraint set (10) is used for capturing the value of the makespan criterion. The mono-objective version of this model with makespan criterion has been used to solve small-scale instances successfully using GAMS/Cplex.

3 The multi-objective memetic algorithm (MOMA)

Different with other algorithms for the IPPS problem, problem-specific multi-objective local search methods are considered in MOMA where the three objectives are tactfully covered by these local search methods (to be discussed in "Problem-specific multi-objective local search" section). In this way, all the objectives can be optimized at the same time. In the following, basic components of MOMA, e.g., coding and decoding schemes, are described first.

3.1 Encoding and decoding

Figure 2 gives a pictorial presentation of the chromosome structure. A chromosome consists of a scheduling string, some process plan strings, and corresponding operation strings. The scheduling string adopts an operation-based coding scheme [37]. This coding scheme brings convenience to the decoding process. The scheduling string is a permutation of job numbers with each job number appearing exactly $|NO_{il}|$ times, where NO_{il} denotes the set of operations that belong to the l-th process plan of job i. The length of the scheduling string equals the sum of the maximum possible number of operations of each job ($\sum|NO_{il}|_{\max}$). If the number of the operations of a process plan is less than the maximum possible one of that job ($|NO_{il}|<|NO_{il}|_{\max}$), ($\sum(|NO_{il}|_{\max}-|NO_{il}|)$) randomly selected positions in the scheduling string should be filled with 0 s to avoid any infeasibility. In Fig. 2, there are a total of eight operations to be scheduled and the actual numbers of operations for the three jobs are three, two, and three, respectively. Assume that each job has three operations at most; the last position should be filled with a “0.” The second string contains the information of the process plan of a job. In a chromosome, there are a total of $|n|$ operation strings with each string containing $|NO_{il}|$ operations (positions). As illustrated in Fig. 2, every position in an operation string represents an operation. Two numbers in a pair of parentheses in each position represent the operation number and the selected machine, respectively. The permutation of operations in an operation string should satisfy the precedence relationships as indicated in the network graph; a feasible operation permutation of a job can be obtained by the binary tree method [38]. The actual number of the operations of a job in an individual is determined by the selected combination (also the selected OR link path). For instance, the job in Fig. 1a may have 12 or 11 operations dependent on the selected combination as shown in Fig. 1b and c. This information is represented by the number in the process plan string. The number in a process plan string specifies which operation combination is selected. For the case in Fig. 1a, it has two operation combinations. Consequently, the number in the process plan string is either 1 or 2, which means that one of the two operation combination is selected for this job.

The active scheduling procedure [39] is adopted in the decoding process. The whole procedure is summarized as follows.

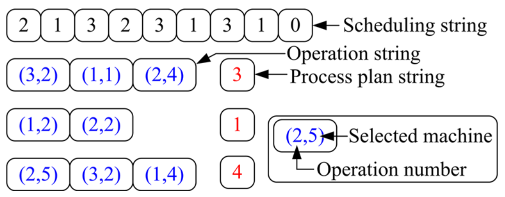

Fig. 2 The coding scheme

Nomenclature

O_{ij}	the j-th operation of the i-th job
k	the alternative machine for O_{ij}
t_{ijk}	the processing time of operation O_{ij} on machine k
c_{ij}	the completion time of O_{ij}
t_s	the starting time of a time slot
t_e	the ending time of a time slot
t_k	the last operation’s completion time on machine k at present.

Step 1: For each number in the scheduling string, determine the corresponding job and operation if the number is not equal to 0: if the number i ($i\neq0$) in the scheduling string appears exactly l times, it means the operation in the l-th position of the operation string of job i is selected. For convenience, we denote this operation as O_{ij}.

Step 2: Find out the operation O_{ij} by locating this operation in the operation string.

Step 3: Assign this operation on the predefined machine m_k.

- Step 3.1: Check whether there is a time slot on m_k.
- Step 3.2: If there is a time slot and $\max\{c_{i,j-1},t_s\}+t_{ijk}\leq t_e$, insert the operation in this time slot and go to step 1 for next operation. Otherwise, check the next slot on the same machine till all the time slots are checked.
- Step 3.3: If there is no available time slot, append the operation at the end of the machine with the starting time $\max\{c_{i,j-1},t_k\}$.

Step 4: Go to step 1 for next operation till the operations are all scheduled.

3.2 Genetic operators

Position-based crossover method [40], commonly used in scheduling problems, is adopted in MOMA as the crossover procedure. Randomly select some jobs, and exchange the operation strings together with corresponding process plan strings of the selected jobs in two individuals; other operation strings and process plan strings in both individuals are kept still. Then, two empty scheduling strings O1, O2 are initialized. The genes in parent 1, which reflect the unselected jobs, are passed on to the same positions as in the offspring 1 (O1). These genes are removed from parent 2 (P2), and the remaining elements are copied into the undetermined positions in O1 in the same order as they appear in P2. A repairing procedure is adopted to resolve the illegitimacy; it deletes redundant symbols or adds necessary symbols after check the actual number of operations of each selected job (Fig. 3a) gives an

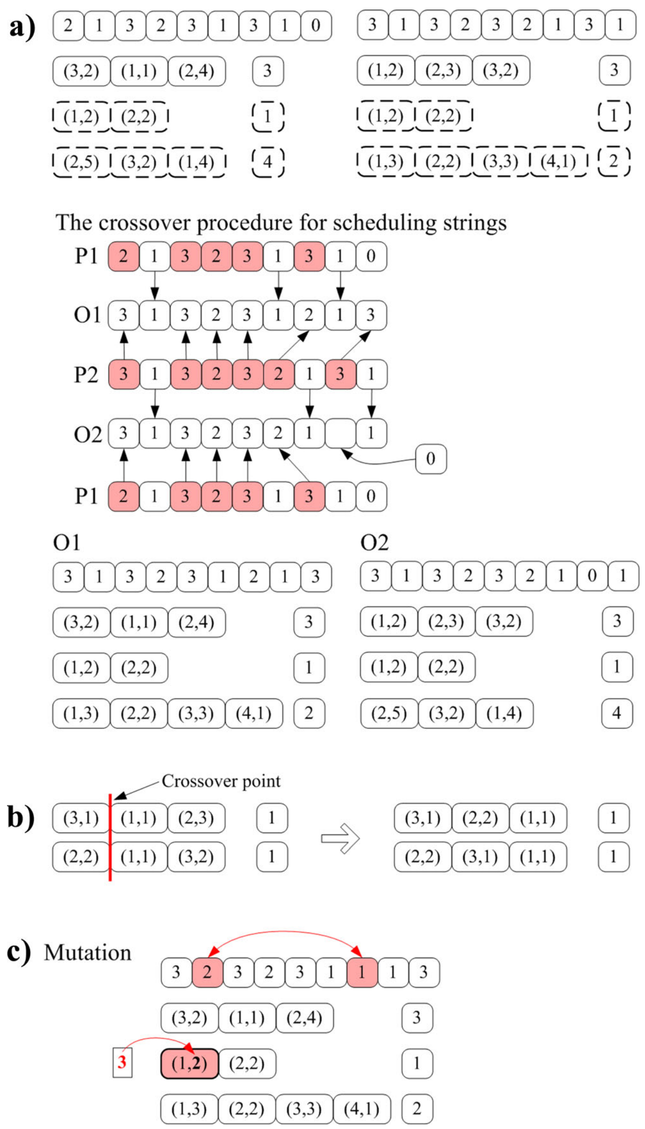

Fig. 3 Crossover and mutation operators

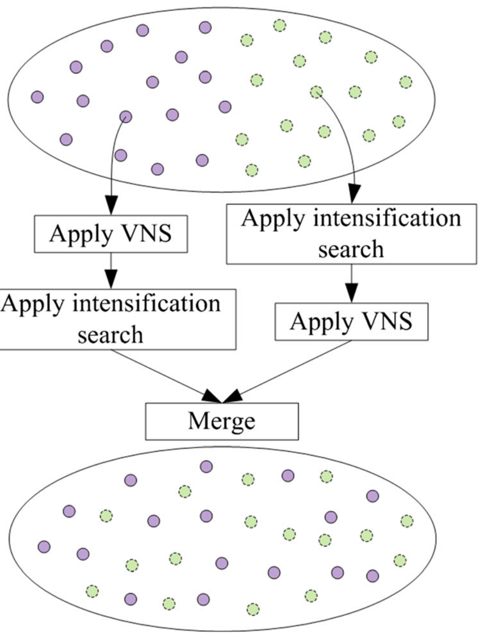

Fig. 5 The problem-specific multi-objective search procedure

example. Last two jobs are selected, and the selected operation strings with corresponding process plan strings (in dashed boxes) in the first individual are interchanged with the ones in the second individual. A "0" is filled in a position in the scheduling string of offspring 2 (O2) to avoid infeasibility. Two resultant individuals are given at the bottom of Fig. 3a.

Recall that the sequencing flexibility (SF) is an important feature of the IPPS problem. SF enables different feasible permutations with the same operations (combination). Different with the chromosome structures adopted by other reported algorithms that seldom consider SF, our chromosome structure allows MOMA to take an advantage of SF by performing the second crossover operator. This time, an order crossover (OX)-based one point crossover [41] is performed between two operation strings for the same job; the two jobs in both individuals should have the same number in the process plan

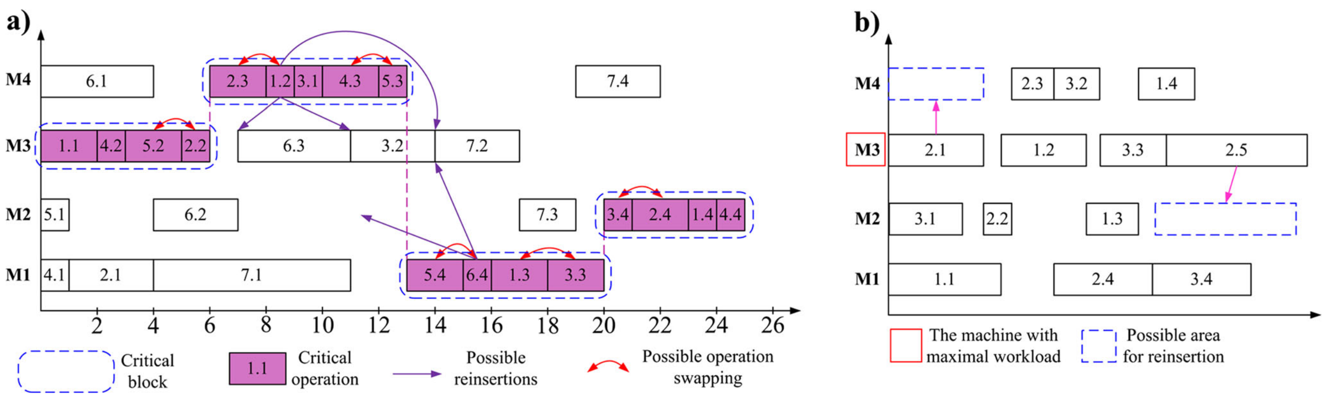
a)
M4
M3
M2
M1
Critical block
Critical operation
Possible reinsertions
Possible operation swapping
b)
The machine with maximal workload
Possible area for reinsertion

Fig. 4 Two local search methods

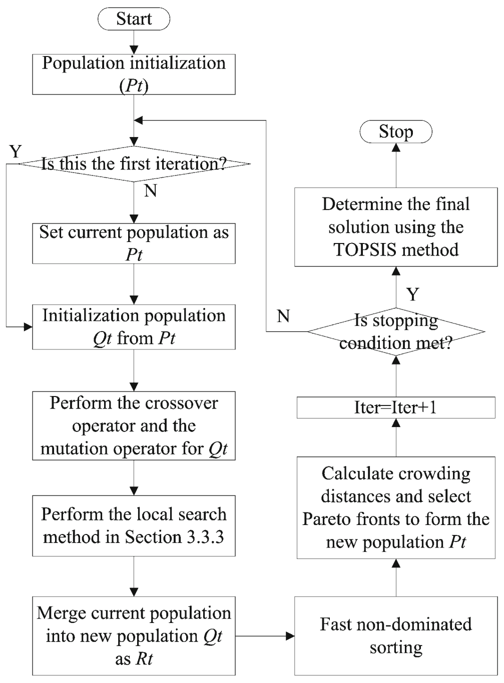
Start
Population initialization (*Pt*)
Is this the first iteration?
Y
N
Set current population as *Pt*
Initialization population *Qt* from *Pt*
Perform the crossover operator and the mutation operator for *Qt*
Perform the local search method in Section 3.3.3
Merge current population into new population *Qt* as *Rt*
Fast non-dominated sorting
Calculate crowding distances and select Pareto fronts to form the new population *Pt*
Iter=Iter+1
Is stopping condition met?
N
Y
Determine the final solution using the TOPSIS method
Stop

Fig. 6 Work flow of MOMA

strings (they have the same operation combination). As illustrated in Fig. 3b, there are two operations taken from two different individuals, and two new feasible operation stings are generated after the crossover procedure. To our knowledge, this crossover operator is often neglected in other approaches for this problem.

The mutation procedure is performed by just changing a machine for an operation or randomly swapping two numbers in the scheduling string. As illustrated in Fig. 3c, machine 3 will process the first operation of the second job after mutation. Meanwhile, numbers in two randomly selected positions in the scheduling string have been interchanged.

3.3 Problem-specific multi-objective local search

3.3.1 VNS for the makespan criterion

In our algorithm, we employ the VNS algorithm to improve the makespan criterion. In VNS, two neighborhood structures are adopted. A neighborhood structure combines the special knowledge of a certain problem to facilitate an effective and slight perturbation in an individual. These two adopted neighborhood structures have been successfully applied to the

Table 1 Test-bed instances

Number	Jobs	Job number	Operations
1	6	1, 2, 3, 10, 11, 12	79
2	6	4, 5, 6, 13, 14, 15	100
3	6	7, 8, 9, 16, 17, 18	121
4	6	1, 4, 7, 10, 13, 16	95
5	6	2, 5, 8, 11, 14, 17	96
6	6	3, 6, 9, 12, 15, 18	109
7	6	1, 4, 8, 12, 15, 17	99
8	6	2, 6, 7, 10, 14, 18	96
9	6	3, 5, 9, 11, 13, 16	105
10	9	1, 2, 3, 5, 6, 10, 11, 12, 15	132
11	9	4, 7, 8, 9, 13, 14, 16, 17, 18	168
12	9	1, 4, 5, 7, 8, 10, 13, 14, 16	146
13	9	2, 3, 6, 9, 11, 12, 15, 17, 18	154
14	9	1, 2, 4, 7, 8, 12, 15, 17, 18	151
15	9	3, 5, 6, 9, 10, 11, 13, 14, 16	149
16	12	1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15	179
17	12	4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18	221
18	12	1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17	191
19	12	2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18	205
20	12	1, 2, 4, 6, 7, 8, 10, 12, 14, 15, 17, 18	195
21	12	2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 18	201
22	15	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18	256
23	15	1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18	256
24	18	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18	300

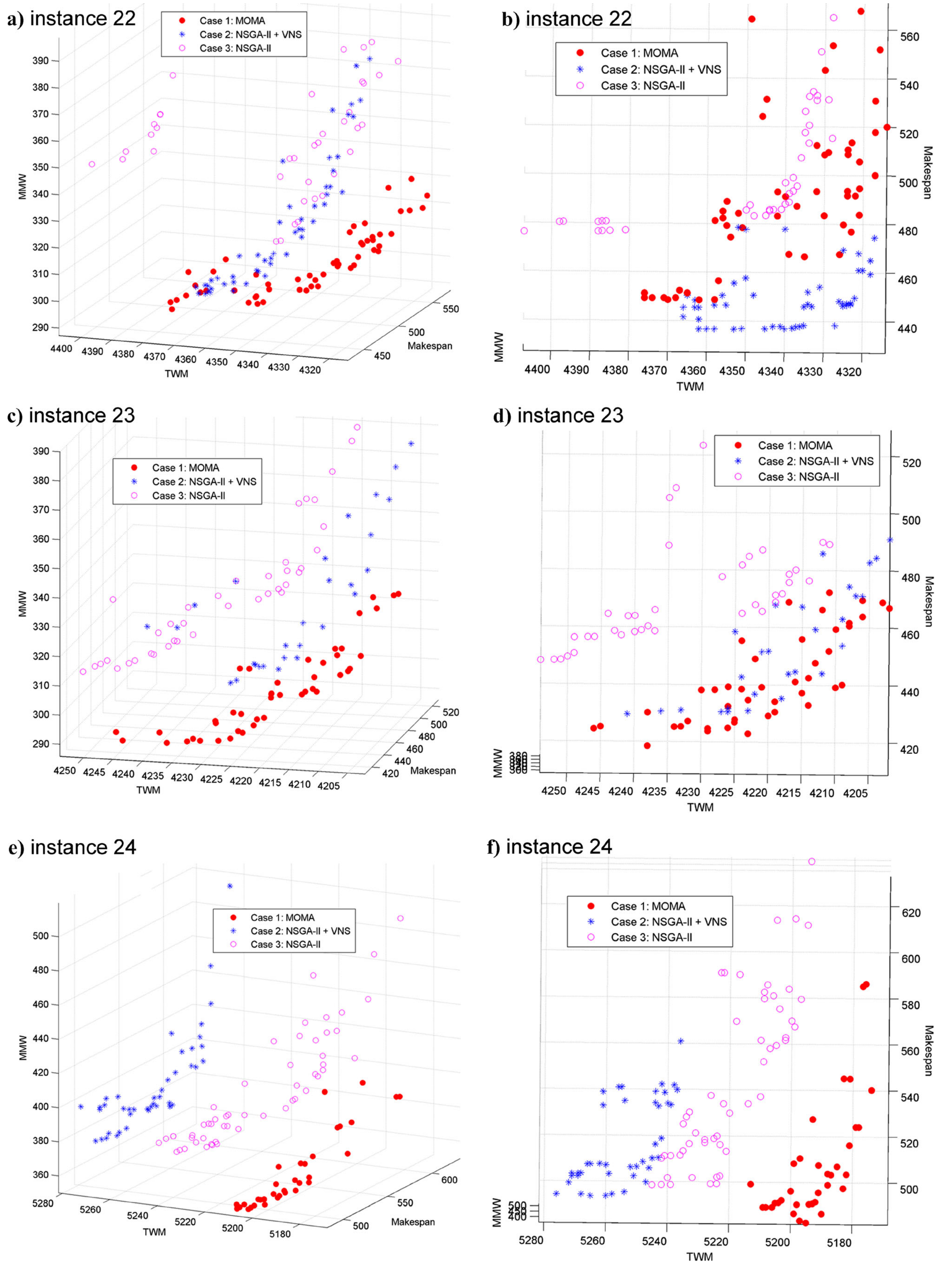

Fig. 7 Optimal Pareto fronts

(flexible) job shop scheduling problem. The first one is the N5 neighborhood structure [22]. It is famous for its high cost performance because only swapping the first and (or) last two operations in a critical block is performed. The second one is proposed by Mastrolilli and Gambardella [42]. It tries to remove every critical operation in a critical path from the current machine to break the critical path. Figure 4a gives the illustration of the two neighborhood structures. VNS tries to perform a slight perturbation in the scheduling string and operation string(s) using one neighborhood structure continuously till there is no further improvement. Then, it jumps to the second neighborhood. If the individual is improved, it jumps back to the first neighborhood to repeat the above process. Otherwise, the VNS procedure is terminated. The detailed VNS procedure is given in Algorithm 1.

Algorithm 1

```
IsContinue ← true;    //The N5 neighborhood is first applied
while IsContinue=true
    StopN5 ← false;
    while StopN5 =false do
        EndBlk ← false;
        while EndBlk =false do
            Find out a critical block;
            if the block contains more than one operation then
                if it is the first block then swap only the last two operations in the block end if
                if it is the last block then swap only the first two operations in the block end if
                if it is neither the first block nor the last block then swap the first two and the
                last two operations in the block end if
            end if
            if the makespan after a swap<current makespan then
                Accept the new schedule;
                StopN5 ← false;  EndBlk ← true;    //Break this while-loop
            else
                StopN5 ← true;  EndBlk ← false;
                //keep the original schedule and go for the next swap
            end if
            if all the blocks have been checked and there is no improvement then
                EndBlk ← true;  StopN5 ← true;  IsContinue ← false;
            end if
        end while
    end while
    //the algorithm jumps to the Nopt neighborhood
    Determine a critical path and the number of critical operations (NCO);
    Determine a right justified schedule to obtain the latest starting time of an operation;
    for i ← 1 to NCO do
        if $NM_i$ machines are able to process the ith critical operation then
            //The algorithm tries to move a critical operation to other $NM_i$-1machines.
            Generate available neighbors;  NeiborCnt ← 1;    //record the number of neighbors
            //Suppose the ith critical operation can be inserted in $N_{ij}$ positions.
            //Put each neighbor in the set NeiborSet.
            for j ← 1 to $NM_i$-1 do
                for k ← 1 to $N_{ij}$ do
                    NeiborSet[NeiborCnt] ← a neighbor;  NeiborCnt++;
                end for
            end for
        end if
    end for
    for i ← 1 to NeiborCnt do  Evaluate NeiborSet[i];  Find the best one: BestNeibor; end for
    Apply BestNeibor;
    if the makespan is improved then  Accept BestNeibor;  IsContinue ← true;
        //Jump back to the N5 neighborhood
    else
        Keep the previous schedule;  IsContinue ← false;    //Terminate the VNS algorithm
    end if
endwhile
```

3.3.2 The intensification algorithm for TWM and MMW criteria

The presence of multiple objectives encourages a family of non-dominated or non-inferior solutions (the optimal Pareto front). In fact, local search methods for all the considered objectives have seldom been facilitated in existing algorithms for the multi-objective IPPS problem. We developed an intensification search method for the TWM and the MMW criteria in MOMA. Figure 4b gives the principle of this procedure. It tries to remove each operation to be processed by the machine whose workload is largest to other available machines that have less workload. During the optimization process, the TWM and MMW criteria are considered at the same time. In Algorithm 2, the intensification search procedure is outlined.

Algorithm 2

```
1:  Step1: Identify the machine that has the maximum workload. Put the operations on this machine to the set MO.
2:  Step2: Record the objective values as: P_makespan, P_TWM, and P_TWM
3:  Step3: IsContinue ← false;
4:  for i ← |MO| to 1
5:    for j ← 1 to N(MO_i)    //MO_i denotes the ith operation in MO;  N(MO_i) the number of available machines of MO_i
6:      if (M_j != current machine)  then    //M_j is an available machine for operation MO_i
7:        Insert MO_i to this machine. Evaluate and record the objective values as N_makespan, N_TWM, and N_MMW
8:      end if
9:      //Judge if accept the machine assignment for the operation MO_i:
10:       If (N_MMW<P_MMW) and (N_TWM≤P_TWM)  then
11:         Accept this machine assignment and go to Step1.
12:       else if (N_MMW<P_MMW) and (P_MMW - N_MMW)>(N_TWM - P_TWM) then
13:         Accept this machine assignment and go to Step1.
14:       else if (N_TWM<P_TWM) and (N_MMW≤P_MMW)  then
15:         Accept this machine assignment and go to Step1.
16:       else if (N_TWM<P_TWM) and (P_TWM - N_TWM)>(N_MMW - P_MMW) then
17:         Accept this machine assignment and go to Step1.
18:       else
19:         Abandon this machine assignment.
20:       end if
21:    end for
22: end for
23: Step 4: if (! IsContinue)  then  Stop searching and terminate the algorithm.    end if
```

Both the TWM and the MMW criteria are considered in this intensification search: the MMW criterion is considered in lines 10–13 in Algorithm 2 while the procedure in lines 14–17 deals with the TWM criterion. More importantly, influenced by the Pareto optimality concept, this intensification search emphasizes trade-offs between different objectives as described in line 12 and line 16 in Algorithm 2: if there are more improvements of the MMW (or TWM) criterion than those of the TWM (or MMW) criterion, the machine assignment can still be accepted.

3.3.3 The problem-specific multi-objective search procedure

The multi-objective local search procedure is depicted in Fig. 5. It contains both VNS and the intensification search. $N/2$ (N is the population size) individuals are randomly selected (marked in green in Fig. 5). For the selected individuals, the intensification search algorithm (Algorithm 2) are first applied to improve the TWM and the MMW criteria. VNS (Algorithm 1) is then applied to optimize the makespan criterion. For the remaining $N/2$ individuals, makespan-oriented VNS is applied before the intensification search. After this, the two parts are merged and the local search procedure is finished.

3.4 The Pareto-based algorithm

The proposed MOMA maintains a population that is improved over iterations. The optimization process is sensed in terms of the three objectives. Just as the case in NSGA-II, the population is updated over iterations by selection, crossover, and mutation operators. First, an offspring population Qt (of size N) is created from the parent population Pt. Then, population Qt is combined with Pt to form an intermediate population, Rt, of size $2N$. After evaluating all the individuals in each generation, all the Pareto fronts in Rt can be identified. The new population Pt is obtained from the merged population Rt by taking the Pareto solutions. In MOMA, we adopt the novel sorting method proposed in [43]. This fast non-dominated sorting method, reduces the time complexity from $O(rN^2)$ in NSGA-II to $O(rN\log N)$ (r is the number of objectives).

3.5 The TOPSIS method

This decision supporting strategy is developed based on the concept that the chosen alternative should have the shortest distance from the ideal solution and the farthest from the negative-ideal solution [34]. After a given number of iterations, MOMA terminates with the resultant optimal Pareto front. In the final step of MOMA, a decision should be made in the presence of multiple criteria to determine the most reasonable individual from the optimal Pareto front. During the decision process, we assume that all the three criteria are equally important, and a set of weights w=[1/3, 1/3, 1/3] for the three objectives is accommodated to the decision matrix of TOPSIS method in MOMA. The whole algorithm is depicted in Fig. 6.

4 Computational results with discussions

To evaluate the performance of MOMA developed for the multi-objective IPPS problem, the typical benchmark instances reported in [14] are considered in our experiment. This data is the most popular set of instances tested in existing works. We first show that better optimal Pareto fronts can be captured by MOMA. Then, we compare MOMA with the well-known NSGA-II algorithm using two metrics. More importantly, satisfactory solutions are obtained by the TOPSIS method from the Pareto fronts captured by MOMA and NSGA-II, respectively; we compare the resultant satisfactory solutions to demonstrate the advantage of MOMA over NSGA-II. Finally, we present a visual comparison to demonstrate the necessity of multi-objective optimization for the IPPS problem.

The algorithm is programmed in C++ language and implemented on a computer with an Intel i5 3470 3.2GHz CPU and 4 GB of memory. The parameters are set as follows: the population size is set to 400; the crossover probability and the mutation probability are set to 0.7 and 0.08, respectively; and the algorithm terminates after 800 iterations. To accommodate MOMA to the three criteria, half of the individuals in MOMA are generated randomly while other individuals are assigned with shortest process plans and the machines with shortest processing times [24].

In our experiment, the 24 test-bed instances in Kim's benchmark [14] are constructed with 18 jobs. In each instance, there are a total of 15 machines to process the jobs. The number of operations of these instances varies from 79 to 300. The details of these instances are listed in Table 1.

To demonstrate the excellent performance of MOMA, visual comparisons are presented as depicted in Fig. 7. Three most complicated instances in Kim's benchmark (instances 22–24) are considered, and optimal Pareto fronts taken from the last iteration of three cases are depicted in Fig. 7. In case 1, we apply MOMA directly to obtain the optimal Pareto front. In case 2, the proposed intensification search is not employed. Thus, case 2 can be deemed as the NSGA-II algorithm combined with VNS. For the last case, both VNS and the intensification search are not considered, and only NSGA-II is applied. In all the three cases, parameters remain unchanged.

As can be seen in Fig. 7a and b, the Pareto front of case 1 is more competitive than those of other two cases for the MMW and TWM criteria. For the makespan criterion, we can find that the optimal Pareto front of case 2 is better according to the top view in Fig. 7b: it locates much closer to the coordinate axis of the TWM criterion. For the MMW and the TWM criteria, however, the Pareto front of case 2 exhibits its weakness. This is due mainly to the fact that the VNS procedure is makespan-oriented and only VNS is not enough to cope with the three criteria in case 2. Clearly, the Pareto front of case 1 can make a reasonable tradeoff between the three objectives. In other words, the non-dominated solutions of case 1 in Fig. 7b are distributed more equally than the situation of case 2. Further, we can easily find out that the Pareto front yielded by MOMA (case 1) is more competitive than the one obtained by NSGA-II (case 3) according to Fig. 7a and b. The above comparisons between MOMA and the other two algorithms (NSGA-II+VNS and NSGA-II) indicate that the intensification search method plays an important role in improving the MMW and the TWM criteria. For instance 23, MOMA yields competitive non-dominated solutions for the three criteria from observation of Fig. 7c and d. Figure 7d gives a top view of the optimal Pareto fronts yielded by the three algorithms. In Fig. 7d, non-dominated solutions captured by MOMA give a better distribution along the axis of the TWM criterion. For the last instance, MOMA yields an excellent Pareto front again. According to Fig. 7e, the Pareto front captured by MOMA totally dominated other two Pareto fronts. As indicated in Fig. 7e and its top view (Fig. 7f), the MMW and the TWM criteria receive less improvements again in case 2 and case 3.

Table 2 Comparisons between MOMA and NSGA-II

Number	AR		SP	
	MOMA	NSGA-II	MOMA	NSGA-II
1	1*	1	7.32*	54.24
2	1*	0	33.66*	79.31
3	0.92*	0.21	25.11*	28.58
4	1*	0	213.86	27.87
5	0.96*	0.91	7.98*	85.92
6	1*	0.29	104.03	69.26
7	0.77*	0.5	109.172	76.56
8	1*	0.42	36.87	15.44
9	0.97*	0.23	5.84*	28.36
10	0.88*	0.84	28.96*	64.32
11	1*	0	22.5	17.81
12	1*	0.05	12.83*	108.35
13	0.95*	0.65	20.69*	172.31
14	0.65	1	24.32*	144.8
15	0.9*	0.55	12.53*	204.40
16	1*	0.56	78.03*	127.56
17	1*	0	13.92*	38.78
18	1*	0	71.60	38.03
19	1*	0.18	15.62*	105.66
20	1*	0.28	46.25*	64.45
21	0.24	1	72.75*	88.81
22	1*	0	46.69	35.91
23	1*	0	14.80	24.39
24	1*	0	64.44*	101.66

*The case where MOMA performs better than NSGA-II

Table 3 Non-dominated solutions

Number	Makespan	TWM	MMW
1	428	1843	129
2	428	1828	145
3	427	1822	150
4	428	1820	149
5	428	1826	147
6	436	1840	131
7	431	1839	132
8	432	1831	134
9	429	1816	154
10	428	1834	143
11	427	1841	133
12	428	1835	139
13	427	1842	132
14	427	1838	134
15	427	1835	143
16	427	1831	147
17	427	1824	149

Based on the observations of Fig. 7 and the above discussion, it is very intuitive to find out that MOMA is the winner: it can generate more satisfactory non-dominated solutions than NSGA-II. MOMA is much better than the traditional NSGA-II for the multi-objective IPPS problem because applying problem-specific local search methods in MOMA makes it able to find a better spread of solutions than other algorithms in terms of the quality of solutions.

To demonstrate the fact that MOMA gains an advantage over other algorithms, we compare the performance of MOMA in Table 2 with that of the well-known NSGA-II algorithm using Kim's benchmark. Two metrics, the average ratio (AR) and the spacing metric (SP), are introduced to evaluate the optimal Pareto solutions of MOMA and NSGA-II. For the AR metric, it can be calculated in the following manner. Let *P1* and *P2* be the optimal Pareto solutions gained from MOMA and NSGA-II, respectively. The set *P* can be obtained by merging *P1* and *P2*, e.g., *P*=*P1*∪*P2*. Thus, the AR value of an algorithm is calculated as the ratio of the optimal Pareto solutions in *P1* that are not dominated by any other solutions in *P*, as illustrated in Eq. (11).

$$AR_{(P_i)} = \frac{\left|P_i - \left\{x \in P_i \middle| \exists y \in P : y \succ x\right\}\right|}{|P_i|} \tag{11}$$

In Eq. (11), $y \succ x$ means solution *y* dominates *x*. The higher the ratio AR is, the better the solution set P_i is. The SP metric, which is usually adopted for the optimization problem with two objectives, are designed to evaluate the distribution of the solutions in the optimal Pareto front. As described in Eqs. (12) and (13), we improve this metric so that it can adapt to the triple-objective IPPS problem studied in this work.

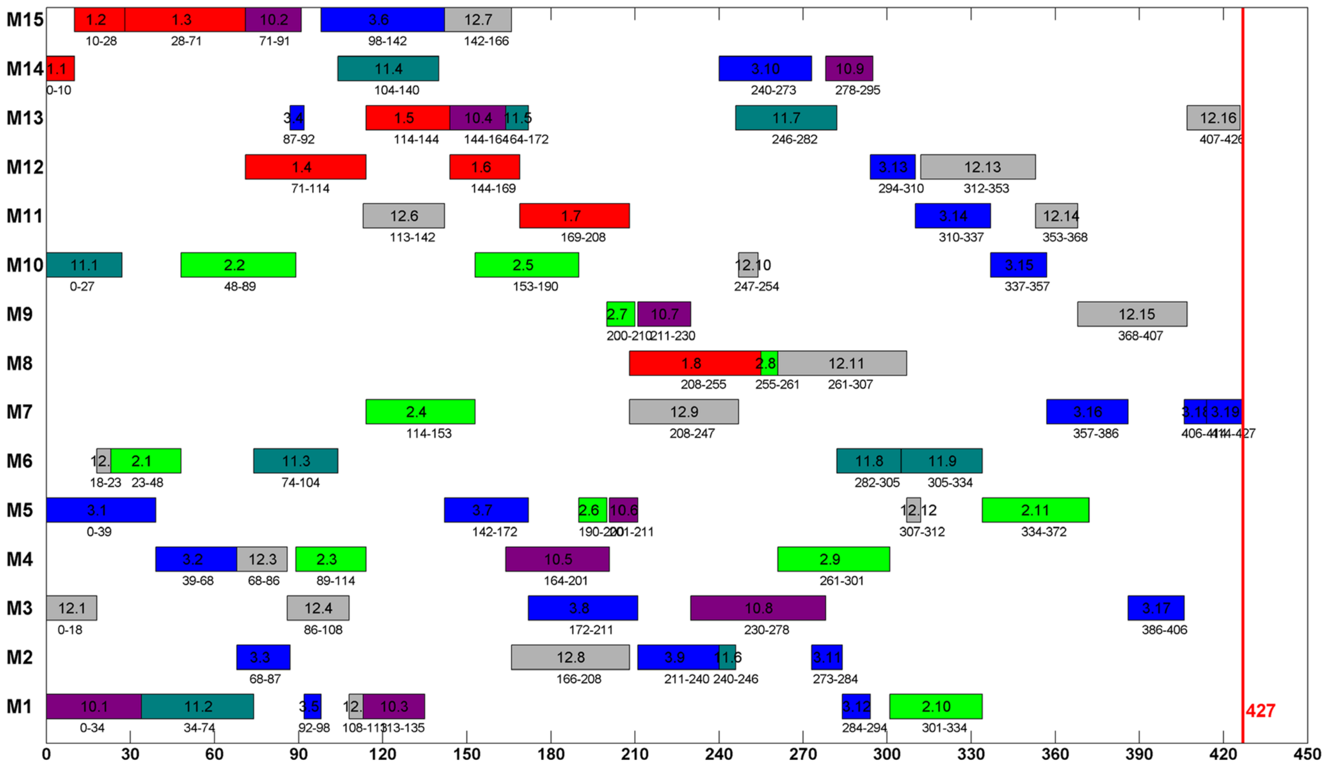

Fig. 8 The Gantt chart of the first instance

$$d_i = \min_j \left(\left| f_1(x_i) - f_1(x_j) \right| + \left| f_2(x_i) - f_2(x_j) \right| + \left| f_3(x_i) - f_3(x_j) \right| \right),$$
$$i, j = 1, 2, \ldots, n_P \qquad (12)$$

$$SP = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n_P} \left(\overline{d} - d_i \right)^2} \qquad (13)$$

In Eqs. (12) and (13), n_P is the total number of non-dominated solutions in the optimal Pareto front, $f_k(x_i)$ is the value of the k-th criterion of solution x_i, and $\overline{d}$ is the mean value of d_i. A less SP value is more acceptable.

Most of the AR values of MOMA in Table 2 take value 1 or the value very close to 1 while the AR values of NSGA-II fluctuates wildly between 0 and 1. Based on the observation of Table 2, MOMA outperforms NSGA-II for 22 instances. For the SP metric, the high performance of MOMA is observed again. Small SP values of MOMA in Table 2 mean the distances between the non-dominated solutions captured by MOMA are distributed more equally. It can be seen that the novel algorithm MOMA outperforms NSGA-II for 16 instances. We believe that the high performance of MOMA stems from the VNS procedure and the proposed intensification search method because most of the non-dominated solutions obtained by NSGA-II are dominated by the ones obtained by MOMA where VNS and intensification search method are applied.

Because MOMA and NSGA-II each generate a set of non-dominated solutions after a certain number of iterations, the most acceptable schedule of each instance can be determined by the TOPSIS decision method from the optimal Pareto front. Table 3 gives all the non-dominated solutions yielded by MOMA in the optimal Pareto front of instance 1. By using the TOPSIS method, the third solution in Table 3 is determined as the most acceptable solution, and the corresponding Gantt chart is depicted in Fig. 8.

In Table 4, we list the objective values of the resultant satisfactory schedules obtained by the TOPSIS method. Competitive objective values in Table 4 are marked with asterisks.

From observation of Table 4, it can be seen that MOMA outperforms NSGA-II: eight resultant schedules of MOMA are better than those of NSGA-II for all the three criteria while only one resultant schedule of NSGA-II is better than that obtained by MOMA. Although the resultant solutions of MOMA and NSGA-II of other 15 instances do not dominate each other, superiorities of the novel algorithm MOMA can still be observed: for the 12 instances (instances 1, 3, 5–6, 8–9, 12–13, 15–16, 19, and 21) of the remaining 15 instances, the resultant objective values of the MMW criterion of MOMA gain enormous advantages over the ones of NSGA-II according to Table 4.

Table 4 Objective values of resultant schedules

Number	MOMA			NSGA-II		
	Makespan	TWM	MMW	Makespan	TWM	MMW
1	427	1822	150	427	1812	193
2	343*	1623*	174*	343	1624	206
3	347	1713	166	347	1710	181
4	306*	1433*	148*	307	1443	148
5	319	1588	159	319	1581	248
6	427	2134	175	427	2128	239
7	372	1826	189	372*	1826*	179*
8	343	1686	148	343	1676	237
9	427	1641	169	427	1637	198
10	428	2727	237	427	2728	244
11	348*	2449*	205*	348	2481	280
12	320	2231	175	327	2225	212
13	427	2936	245	427	2935	262
14	375	2749	210	384	2728	253
15	427	2430	215	427	2422	244
16	427	3451	251	427	3448	282
17	359*	3358*	254*	388	3391	302
18	329*	3043*	229*	347	3089	252
19	440	3733	270	439	3739	324
20	400	3558	262	421	3529	276
21	427	3336	268	427	3318	309
22	448*	4358*	317*	475	4381	379
23	418*	4238*	294*	448	4249	332
24	482*	5195*	362*	497	5224	397

*The case where MOMA performs better than NSGA-II

Finally, we give an example to demonstrate that it is highly necessary to consider the multiple objectives requirement arising in a real-world production environment. Figure 8 gives the Gantt chart of the first instance obtained by MOMA while Fig. 9 depicts the schedule of the same instance that is yielded by a mono-objective algorithm with makespan criterion only. Although these two schedules have the same overall finishing time (427), the scheduling scheme obtained by MOMA presented in Fig. 8 is more satisfactory than the other one after judging the MMW values in Fig. 10 which gives the workload of each machine presented in Figs. 8 and 9. The maximum machine workload of the schedule in Fig. 8 is 150 while the one in Fig. 9 is 254. Clearly, the workload of the scheduling scheme in Fig. 8 is distributed more equally on 15 machines than the case in Fig. 9. This means the proposed algorithm can truly reduce the maximum machine workload of a schedule by allocating excessive operations to other available machines. For the TWM criterion, MOMA shows its talent again: the total workload of machines in Fig. 8 is 1822, and this value is

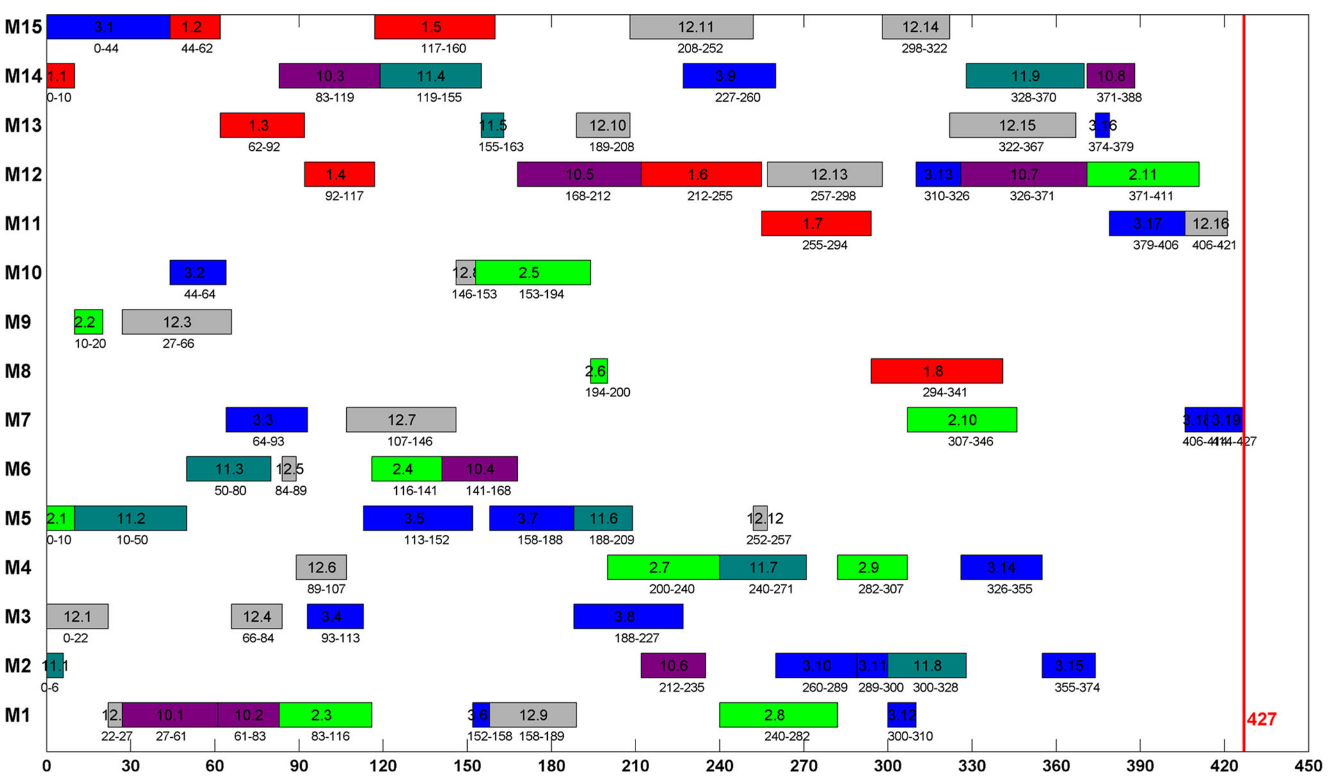
Fig. 9 The Gantt chart of the first instance with makespan criterion only

less than 1860 which is the total workload of machines in Fig. 9. Clearly, with the same makespan value, the schedule obtained by a mono-objective algorithm in Fig. 9 is not very promising due to the disadvantages on the other two criteria. Thus, choosing the schedule in Fig. 8 can meet the real-life situation better. Based on the analysis and comparisons in this section, we can conclude that MOMA outperforms NSGA-II in addressing the multi-objective IPPS problem.

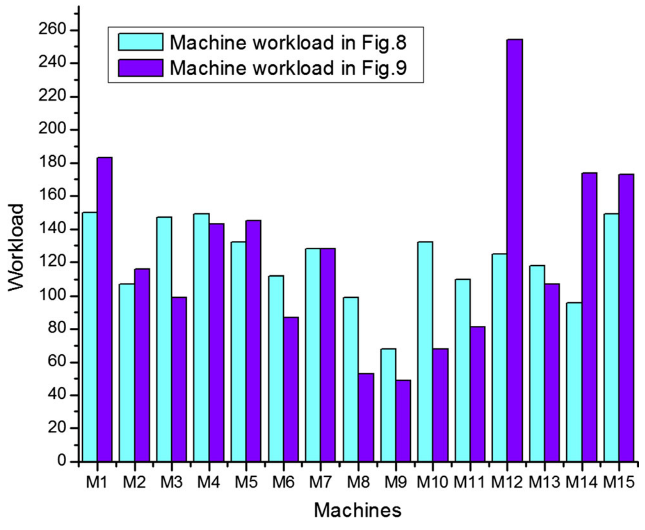

Fig. 10 Workload of each machine

5 Conclusions

In this study, the IPPS problem is addressed in a Pareto manner and solved by the novel algorithm MOMA. Compared with existing multi-objective meta-heuristic algorithms, the proposed algorithm adopts an objective-oriented multi-objective local search method, where makespan-oriented variable neighborhood search algorithm and machine workload-specific intensification search are adopted to improve all the three objectives. This is the distinct feature of MOMA. A novel fast non-dominated sorting method is also employed in MOMA to reduce the computational complexity. After the optimal Pareto front is obtained, the TOPSIS method is adopted in determining the most satisfactory schedule among a set of non-dominated solutions. To evaluate the proposed algorithm, MOMA was tested on typical benchmark instances and compared with the well-known NSGA-II algorithm. Computational results reflected the following conclusions:

1. MOMA can effectively improve all the three objectives.
2. Compared with other multi-objective algorithms, e.g., the well-known NSGA-II, MOMA is able to achieve more competitive non-dominated solutions.
3. It is quite necessary to employ local search methods for all the objectives in multi-objective optimization algorithms.

The focus of this work is mainly on deterministic constraints of jobs. However, a manufacturing system in the real world is often subject to many sources of uncertainty. One source of uncertainty lies in the processing time of an operation which may fluctuate within a certain range. Thus, a predefined schedule may usually be affected. In future research, uncertain processing times should be included, and a multi-objective algorithm for the IPPS problem with uncertain processing time can be considered.

Acknowledgments This work was supported by the Funds for International Cooperation and Exchange of the National Natural Science Foundation of China (no. 51561125002), the National Natural Science Foundation of China (nos. 51275190 and 51575211), and the Fundamental Research Funds for the Central Universities (HUST:2014TS038). The authors thank the anonymous referees for their insightful and valuable suggestions and comments to improve this paper.

Compliance with ethical standards

Conflict of interest The authors declare that they have no competing interests.

References

1. Mohammadi G, Karampourhaghghi A, Samaei F (2012) A multi-objective optimisation model to integrating flexible process planning and scheduling based on hybrid multi-objective simulated annealing. Int J Prod Res 50(18):5063–5076. doi:10.1080/00207543.2011.631602
2. X-y W, X-y L, Gao L, H-y S (2014) Honey bees mating optimization algorithm for process planning problem. J Intell Manuf 25(3):459–472. doi:10.1007/s10845-012-0696-8
3. Lian KL, Zhang CY, Shao XY, Gao L (2012) Optimization of process planning with various flexibilities using an imperialist competitive algorithm. Int J Adv Manuf Tech 59(5-8):815–828. doi:10.1007/s00170-011-3527-8
4. Jain A, Jain PK, Singh IP (2006) An integrated scheme for process planning and scheduling in FMS. Int J Adv Manuf Tech 30(11-12):1111–1118. doi:10.1007/s00170-005-0142-6
5. Salehi M, Bahreininejad A (2011) Optimization process planning using hybrid genetic algorithm and intelligent search for job shop machining. J Intell Manuf 22(4):643–652. doi:10.1007/s10845-010-0382-7
6. Li X, Gao L, Wen X (2013) Application of an efficient modified particle swarm optimization algorithm for process planning. Int J Adv Manuf Tech 67(5-8):1355–1369. doi:10.1007/s00170-012-4572-7
7. Lv S, Qiao L (2013) A cross-entropy-based approach for the optimization of flexible process planning. Int J Adv Manuf Tech 68(9-12):2099–2110. doi:10.1007/s00170-013-4815-2
8. Gopala Krishna A, Mallikarjuna Rao K (2006) Optimisation of operations sequence in CAPP using an ant colony algorithm. Int J Adv Manuf Tech 29(1-2):159–164. doi:10.1007/s00170-004-2491-y
9. Mohapatra P, Nayak A, Kumar SK, Tiwari MK (2014) Multi-objective process planning and scheduling using controlled elitist non-dominated sorting genetic algorithm. Int J Prod Res 1–24. doi:10.1080/00207543.2014.957872
10. Zhang WJ, Xie SQ (2007) Agent technology for collaborative process planning: a review. Int J Adv Manuf Tech 32(3-4):315–325. doi:10.1007/s00170-005-0345-x
11. Li XY, Shao XY, Gao L (2008) Optimization of flexible process planning by genetic programming. Int J Adv Manuf Tech 38(1-2):143–153. doi:10.1007/s00170-007-1069-x
12. Li XY, Shao XY, Gao L, Qian WR (2010) An effective hybrid algorithm for integrated process planning and scheduling. Int J Prod Econ 126(2):289–298. doi:10.1016/j.ijpe.2010.04.001
13. Lian KL, Zhang CY, Gao L, Li XY (2012) Integrated process planning and scheduling using an imperialist competitive algorithm. Int J Prod Res 50(15):4326–4343. doi:10.1080/00207543.2011.622310
14. Kim YK, Park K, Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30(8):1151–1171. doi:10.1016/s0305-0548(02)00063-1
15. Kumar M, Rajotia S (2006) Integration of process planning and scheduling in a job shop environment. Int J Adv Manuf Tech 28(1-2):109–116. doi:10.1007/s00170-004-2317-y
16. Lee H, Kim SS (2001) Integration of process planning and scheduling using simulation based genetic algorithms. Int J Adv Manuf Tech 18(8):586–590. doi:10.1007/s001700170035
17. Saygin C, Kilic SE (1999) Integrating flexible process plans with scheduling in flexible manufacturing systems. Int J Adv Manuf Tech 15(4):268–280. doi:10.1007/s001700050066
18. Li XY, Gao L, Li WD (2012) Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. Expert Syst Appl 39(1):288–297. doi:10.1016/j.eswa.2011.07.019
19. Özgüven C, Özbakır L, Yavuz Y (2010) Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 34(6):1539–1548. doi:10.1016/j.apm.2009.09.002
20. Tan W, Khoshnevis B (2004) A linearized polynomial mixed integer programming model for the integration of process planning and scheduling. J Intell Manuf 15(5):593–605. doi:10.1023/B:JIMS.0000037710.80847.b6
21. Shao XY, Li XY, Gao L, Zhang CY (2009) Integration of process planning and scheduling—a modified genetic algorithm-based approach. Comput Oper Res 36(6):2082–2096. doi:10.1016/j.cor.2008.07.006
22. Nowicki E, Smutnicki C (1996) A fast taboo search algorithm for the job shop problem. Manage Sci 42(6):797–813. doi:10.1287/mnsc.42.6.797
23. Li XY, Gao L, Shao XY (2012) An active learning genetic algorithm for integrated process planning and scheduling. Expert Syst Appl 39(8):6683–6691. doi:10.1016/j.eswa.2011.11.074
24. Lihong Q, Shengping L (2012) An improved genetic algorithm for integrated process planning and scheduling. Int J Adv Manuf Tech 58(5-8):727–740. doi:10.1007/s00170-011-3409-0
25. Zhang L, Wong TN (2015) An object-coding genetic algorithm for integrated process planning and scheduling. Eur J Oper Res 244(2):434–444. doi:10.1016/j.ejor.2015.01.032
26. Baykasoglu A, Ozbakir L (2009) A grammatical optimization approach for integrated process planning and scheduling. J Intell Manuf 20(2):211–221. doi:10.1007/s10845-008-0223-0
27. Seker A, Erol S, Botsali R (2013) A neuro-fuzzy model for a new hybrid integrated Process Planning and Scheduling system. Expert Syst Appl 40(13):5341–5351. doi:10.1016/j.eswa.2013.03.043
28. Li WD, McMahon CA (2007) A simulated annealing-based optimization approach for integrated process planning and scheduling. Int J Comp Integ M 20(1):80–95. doi:10.1080/09511920600667366
29. Guo YW, Li WD, Mileham AR, Owen GW (2009) Applications of particle swarm optimisation in integrated process planning and

scheduling. Robot Cim-Int Manuf 25(2):280–288. doi:10.1016/j.rcim.2007.12.002

30. Jin L, Zhang C, Shao X (2015) An effective hybrid honey bee mating optimization algorithm for integrated process planning and scheduling problems. Int J Adv Manuf Tech 80(5-8):1253–1264. doi:10.1007/s00170-015-7069-3
31. Haddadzade M, Razfar MR, Zarandi MHF (2014) Integration of process planning and job shop scheduling with stochastic processing time. Int J Adv Manuf Tech 71(1-4):241–252. doi:10.1007/s00170-013-5469-9
32. Shukla SK, Tiwari MK, Son YJ (2008) Bidding-based multi-agent system for integrated process planning and scheduling: a data-mining and hybrid tabu-SA algorithm-oriented approach. Int J Adv Manuf Tech 38(1-2):163–175. doi:10.1007/s00170-007-1087-8
33. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE T Evolut Comput 6(2):182–197. doi:10.1109/4235.996017
34. Hwang C-L, Yoon K (1981) Multiple attribute decision making: methods and applications a state-of-the-art survey. Springer-Verlag
35. Qiu X, Lau HK (2014) An AIS-based hybrid algorithm for static job shop scheduling problem. J Intell Manuf 25(3):489–503. doi:10.1007/s10845-012-0701-2
36. Gen M, Lin L (2014) Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. J Intell Manuf 25(5):849–866. doi:10.1007/s10845-013-0804-4
37. Bierwirth C (1995) A generalized permutation approach to job shop scheduling with genetic algorithms. Oper Res Spektrum 17(2-3):87–92. doi:10.1007/BF01719250
38. Tseng HE (2006) Guided genetic algorithms for solving a larger constraint assembly problem. Int J Prod Res 44(3):601–625. doi:10.1080/00207540500270513
39. Bierwirth C, Mattfeld DC (1999) Production scheduling and rescheduling with genetic algorithms. Evol Comput 7(1):1–17. doi:10.1162/evco.1999.7.1.1
40. Cheng R, Gen M, Tsujimura Y (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. Comput Ind Eng 30(4):983–997. doi:10.1016/0360-8352(96)00047-2
41. Zhang F, Zhang YF, Nee AYC (1997) Using genetic algorithms in process planning for job shop machining. IEEE T Evolut Comput 1(4):278–289. doi:10.1109/4235.687888
42. Mastrolilli M, Gambardella L (1996) Effective neighborhood functions for the flexible job shop problem. J Sched 3(3):3–20
43. Zheng J, Ling C, Shi Z, Xue J, Li X (2004) A multi-objective genetic algorithm based on quick sort. In: Tawfik A, Goodwin S (eds) Advances in artificial intelligence, vol 3060, Lecture Notes in Computer Science. Springer, Berlin, pp 175–186. doi:10.1007/978-3-540-24840-8_13