Int J Adv Manuf Technol (2012) 59:815–828
DOI 10.1007/s00170-011-3527-8

ORIGINAL ARTICLE

Optimization of process planning with various flexibilities using an imperialist competitive algorithm

Kunlei Lian · Chaoyong Zhang · Xinyu Shao · Liang Gao

Received: 21 April 2011 / Accepted: 10 July 2011 / Published online: 7 August 2011

Abstract In this paper, we investigate the optimization of process planning in which various flexibilities are considered. The objective is to minimize total weighted sum of manufacturing costs. Various flexibilities, including process flexibility, sequence flexibility, machine flexibility, tool flexibility, and tool access direction (TAD) flexibility, generally exist in process planning and consideration of these flexibilities is essential for improving production efficiency and system flexibility. However, process planning is strongly NP-hard due to the existence of various flexibilities as well as complex machining precedence constraints. To tackle this problem, an imperialist competitive algorithm (ICA) is employed to find promising solutions with reasonable computational cost. ICA is a novel socio-politically motivated metaheuristic algorithm inspired by imperialist competition. It starts with an initial population and proceeds through assimilation, position exchange, imperialistic competition, and elimination. Computational experiments on three sets of process planning problem taken from literature are carried out, and comparisons with some existing algorithms developed for process planning are presented. The results show that the algorithm performs significantly better than existing algorithms like genetic algorithm (GA), simulated annealing (SA), tabu search (TS), and particle swarm optimization (PSO).

Keywords Process planning · Imperialist competitive algorithm · Genetic algorithm · Simulated annealing

K. Lian · C. Zhang (✉) · X. Shao · L. Gao
The State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China
e-mail: zcyhust@mail.hust.edu.cn

1 Introduction

Computer-aided process planning (CAPP) is a key component of computer integrated manufacturing system (CIMS). It links computer-aided design (CAD) and computer-aided manufacturing (CAM) by transforming a part design into a set of machining instructions. This paper addresses the process planning problem in generative CAPP systems. An important feature of the generative system is its high degree of flexibility to adapt itself to multi-variety and small-batch production environments. Therefore, optimization of process planning with various flexibilities is essential for improving production efficiency as well as system flexibility. Generally, process planning is concerned with conducting two activities, namely, operation selection and operation sequencing, simultaneously. Operation selection refers to the act of selecting necessary operations to achieve machining features of a given part and determining relevant manufacturing resources for each operation, while operation sequencing is the act of determining an optimized sequence of operations subject to the precedence constraints among operations. There exist a number of flexibilities with respect to these two activities of process planning: (1) During the phase of operation selection, a manufacturing feature can be achieved through different combinations of operations (process flexibility), and alternative machines, tools, and tool access directions (TAD) may exist for each operation (machine flexibility, tool flexibility, and TAD flexibility). (2) During the phase of operation sequencing, different sequences of operations result in different process plans for the same part (sequence flexibility). Although many studies, including Li et al. [1], Kim et al. [2], and Seok Shin et al. [3], dealt with some or part of these process planning flexibilities, to the best of the authors' knowledge, there is no existing study that comprehensively considered the impact of different flexibilities on the outputs of process planning. Therefore, in this paper, the process planning problem with various flexibilities is extensively studied.

Process planning can be modeled as a constraint-based traveling salesman problem (TSP) [4] and therefore is strongly NP-hard. Due to its intrinsic intractability of process planning, efficient algorithms are needed to solve this problem in reasonable computational time. In recent years, metaheuristic algorithms have shown their advantages in solving combinatorial optimization problems, and a number of metaheuristic approaches have been proposed for the optimization of process planning, for example, genetic algorithm (GA), simulated annealing (SA), particle swarm optimization (PSO), tabu search (TS), and ant colony optimization (ACO). This paper adopts a novel optimization algorithm named imperialist competitive algorithm (ICA) to address the process planning problem with various flexibilities. ICA is a population-based evolutionary algorithm motivated by imperialistic competition. It starts with an initial population and effectively searches the solution space through some specially designed operators to converge to optimal or near-optimal solutions. The superiority of ICA over some existing approaches developed for process planning is validated through extensive experiments using benchmark problems taken from literature. Computational results show that ICA is very effective and efficient in solving process planning problem with various flexibilities.

The rest of this paper is organized as follows. Section 2 gives a review of related work about process planning and ICA. Section 3 elaborates the process planning problem with various flexibilities. Details of the proposed ICA are presented in Section 4. Section 5 illustrates the efficiency of ICA through three experiments. Section 6 concludes the paper.

2 Literature review

Process planning has been extensively studied in the past 30 years, and numerous approaches have been proposed to obtain optimal or near-optimal solutions. For reviews on process planning, the reader is referred to Leo and Hongchao [5] and Marri et al. [6]. With respect to different flexibilities considered by researchers in the optimization of process planning, the existing approaches can be classified as the following three categories.

1. Process planning with machine flexibility, tool flexibility, TAD flexibility, and sequence flexibility. Zhang et al. [7] dealt with the process planning problem in a job shop manufacturing environment by considering multiple decision-making activities, i.e., operation selection, machine selection, setup selection, cutting tool selection, and operations sequencing, simultaneously. A genetic algorithm (GA) was employed to obtain near-optimal process plans. Qiao et al. [8] studied the operation sequencing problem for prismatic parts and four types of process planning rules, including precedence rules, clustering rules, adjacent order rules, and optimization rules, were considered. Ma et al. [9] proposed a simulated annealing (SA) algorithm for process planning optimization. The machine flexibility, tool flexibility, TAD flexibility, and sequence flexibility were considered in their work. Li et al. [1] developed a hybrid GA and SA algorithm to solve the process planning problem with constraints. GA was conducted in the first stage to generate some good process plans. SA was carried out in the second stage to search for optimal or near-optimal process plans based on the process plans obtained by GA. Li et al. [10] developed a constraint-handling method and a tabu search (TS) algorithm to address the process planning problem. Li et al. [11] studied the process planning problem in distributed manufacturing environment, and GA was applied to handle it. Guo et al. [12] proposed a particle swarm optimization (PSO) algorithm for the optimization of operation sequencing in process planning. Salehi and Tavakkoli-Moghaddam [13] divided the process planning into preliminary planning and secondary planning. The aim of preliminary planning was to generate some feasible operation sequences and secondary planning aimed at selecting optimized machining resources for each operation. Liu et al. [4] mapped the process planning to a constraint-based traveling salesman problem, and an ant colony optimization (ACO) algorithm was implemented to solve it.
2. Process planning with machine flexibility, process flexibility, and sequence flexibility. In recent years, integration of process planning and scheduling (IPPS) has drawn the attention of many researchers. Flexibilities considered in process planning of IPPS are usually different from that considered when conducting process planning alone. Kim et al. [2] addressed the integrated process planning and scheduling problem in job shop flexible manufacturing systems. Various flexibilities, including sequence flexibility, process flexibility, and machine flexibility, were considered in the process planning function, and a symbiotic evolutionary algorithm was proposed to obtain optimal solutions. Shao et al. [14] presented a new model to integrate the two functions of process planning and scheduling. A modified GA-based approach was developed to facilitate the integration and optimization of the two functions. Leung et al. [15] proposed an ACO algorithm in an agent-based system to integrate process planning and shopfloor scheduling. Artificial ants were implemented as software agents, and a graph-based solution method was suggested. Li et al. [16] formulated a mathematical model of integrated process

planning and scheduling, and an evolutionary algorithm-based approach was proposed.
3. Process planning with machine flexibility, tool flexibility, process flexibility, and sequence flexibility. Seok Shin et al. [3] proposed a symbiotic evolutionary algorithm to solve a multi-objective FMS process planning problem with various flexibilities. Four types of flexibilities related to machine, tool, sequence, and process were studied.

A review of related work reveals that the existing approaches suffer from at least one of the following drawbacks:

1. Process planning is conducted without consideration of various flexibilities, which limits the application of process planning to a small domain.
2. Process planning is considered only for prismatic parts.
3. Some approaches that are tested on relatively small number of problems may not adapt to realistic conditions in multi-variety production environment.

To the best of our knowledge, this paper is the first attempt to address the process planning with various flexibilities. Due to its combinatorial nature, it is reasonable to use metaheuristic algorithm to obtain near-optimal solution in acceptable computational time. ICA is a novel population-based evolutionary algorithm proposed by Atashpaz-Gargari and Lucas [17]. Similar to other metaheuristic algorithms mimicking different kinds of natural phenomena such as natural evolution, birds flocking, and fish schooling, ICA is inspired by the socio-political process of imperialistic competition. ICA has been successfully applied to solve many combinatorial optimization problems. Atashpaz-Gargari and Caro [18], Atashpaz-Gargari et al. [19], and Gargari et al. [20] firstly applied ICA to PID controller designing. Rajabioun et al. [21] used ICA as a tool to achieve Nash equilibrium point. Khabbazi et al. [22] utilized the ICA to minimize the error probability for binary phase shift keying modulation. Forouharfard and Zandieh [23] proposed an ICA to schedule receiving and shipping trucks in cross-docking systems. Kaveh and Talatahari [24] suggested an ICA application in optimal design of skeletal structures. Lucas et al. [25] applied ICA to design a linear induction motor. Nazari-Shirkouhi et al. [26] presented an ICA application in solving the integrated product mix-outsourcing problem. Sarayloo and Tavakkoli-Moghaddam [27] used ICA to address the dynamic cell formation problem with production planning. ICA had also been applied to wireless sensor network localization [28], scheduling of assembly flowshop problem [29], unit commitment problem [30], stochastic assembly balancing problem [31], and data clustering [32]. Recently, some researchers have incorporated chaos theory into ICA to develop more robust algorithms, such as adaptive ICA [33] and chaotic ICA [33, 35, 36]. Abdechiri et al. [34] proposed an adaptive ICA that utilized probabilistic model to balance the exploration and exploitation abilities of the ICA. Abdechiri et al. [34] and Bahrami et al. both [35] developed a chaotic ICA that used the chaos theory to adjust the movement angle of colonies towards the imperialists. Duan et al. [36] proposed a chaotic ICA that used chaotic sequences instead of random sequences to diversify the population and to improve ICA's performance in preventing premature convergence to local minimum. In addition, ICA has been hybridized with electromagnetic-like mechanism to solve group scheduling problem in flexible flow shops [37]. Therefore, ICA is adopted in this paper to obtain optimal or near-optimal solutions of the process planning problem. Computational results in Section 5 show that the proposed ICA outperforms some existing approaches developed for process planning.

3 The process planning problem

3.1 Process planning problem representation

Process planning is the act of selecting necessary operations needed to manufacture a part and determining the sequence of these selected operations subject to predefined precedence constraints. There generally exist a number of flexibilities in the optimization of process planning. In this paper, five flexibilities are considered, namely, process flexibility, machine flexibility, tool flexibility, TAD flexibility, and sequence flexibility. Process flexibility refers to the possibility of producing the same manufacturing feature with alternative operations or sequences of operations. Machine flexibility relates to the possibility of performing the same operation on different machines. Tool flexibility is decided by the possibility of performing the same operation with alternative cutting tools. TAD flexibility indicates the possibility of performing the same operation with different TADs. Sequence flexibility relates to the possibility of interchanging the sequence of operations which is required to manufacture a job.

In this paper, we utilize a network representation that is developed by Ho and Moodie [2, 3, 38] to describe the flexibilities explained previously. Figure 1 shows a network representation of a part. As is shown in Fig. 1, there are three types of nodes in the network, namely, starting node, intermediate node, and ending node. Both the starting node and the ending node are dummy ones that do not contain any operation information. They indicate the beginning and the ending of the manufacturing process of a part respectively. An intermediate node represents an operation. It contains alternative machines, cutting tools, and TADs that can perform the operation. The arrow that connects two nodes indicates the precedence relations between the nodes. A flexible process plan network must be acyclic to ensure that only a finite number of operations are traversed to form a process plan. In

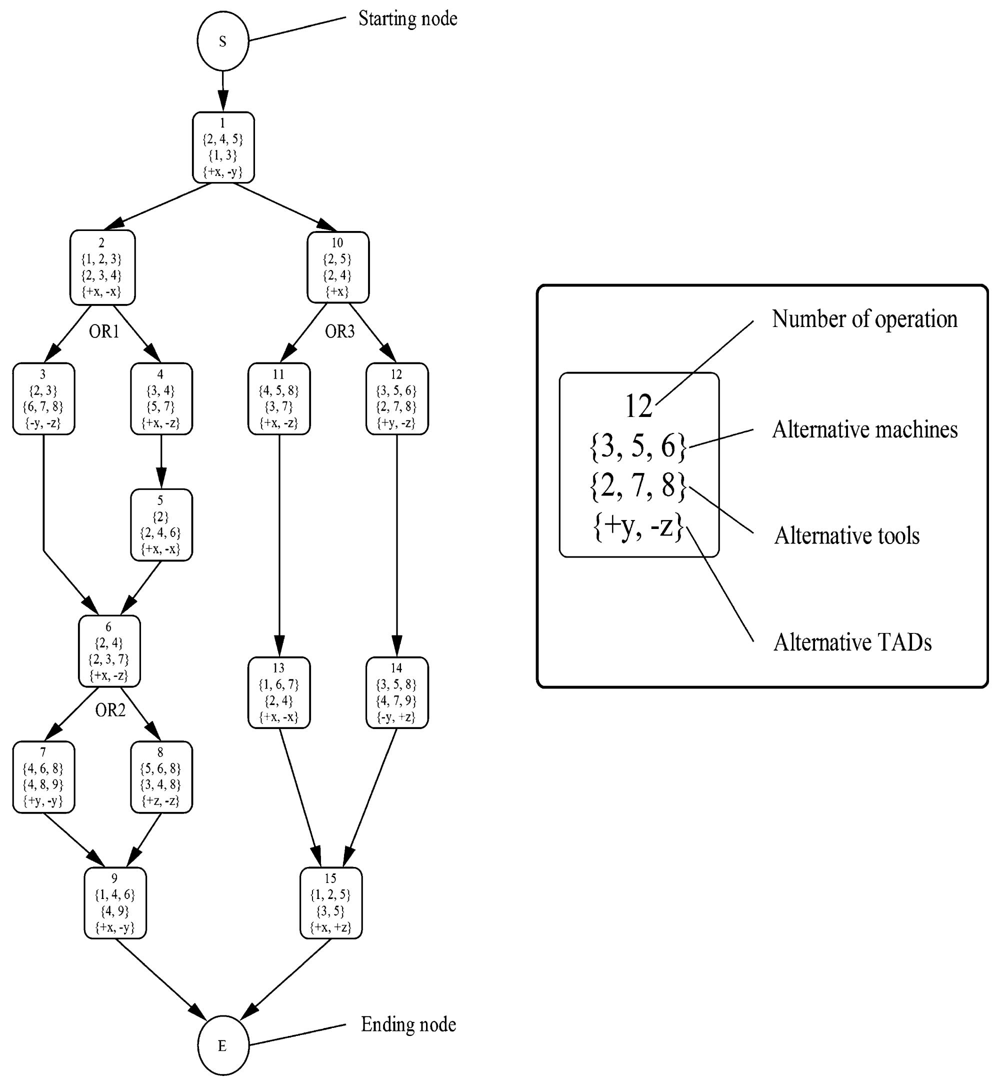
Starting node
S
1 {2, 4, 5} {1, 3} {+x, -y}
2 {1, 2, 3} {2, 3, 4} {+x, -x}
10 {2, 5} {2, 4} {+x}
OR1
OR3
3 {2, 3} {6, 7, 8} {-y, -z}
4 {3, 4} {5, 7} {+x, -z}
11 {4, 5, 8} {3, 7} {+x, -z}
12 {3, 5, 6} {2, 7, 8} {+y, -z}
5 {2} {2, 4, 6} {+x, -x}
6 {2, 4} {2, 3, 7} {+x, -z}
OR2
13 {1, 6, 7} {2, 4} {+x, -x}
14 {3, 5, 8} {4, 7, 9} {-y, +z}
7 {4, 6, 8} {4, 8, 9} {+y, -y}
8 {5, 6, 8} {3, 4, 8} {+z, -z}
9 {1, 4, 6} {4, 9} {+x, -y}
15 {1, 2, 5} {3, 5} {+x, +z}
E
Ending node
12 {3, 5, 6} {2, 7, 8} {+y, -z}
Number of operation
Alternative machines
Alternative tools
Alternative TADs

Fig. 1 A network representation of a part

order to describe the process flexibility, an 'OR' relationship is introduced. If the links that follows a node are connected by an 'OR' symbol, only one of them is needed to traverse.

3.2 Objectives

The objective considered in this paper is the total weighted cost (*TWC*) of machine cost (*MC*), tool cost (*TC*), machine change cost (*MCC*), tool change cost (*TCC*), and setup change cost (*SCC*). The definitions of these objectives are given as follows [1].

- Machine cost (*MC*)

$$MC = \sum_{i=1}^{n} MCI_i, \tag{1}$$

Where n is the number of operations and MCI_i is the predetermined machine cost index for using machine i.

- Tool cost (*TC)*

$$TC = \sum_{i=1}^{n} TCI_i, \tag{2}$$

Where TCI_i is the predetermined tool cost index for using tool i, a constant for a particular tool.

- Machine change cost (*MCC*): a machine change means two adjacent operations are performed on different machines.

$$MCC = MCCI \times \sum_{i=1}^{n} \Omega(M_{i+1} - M_i) \tag{3}$$

Where *MCCI* is the machine change cost index and M_i is the identity of the machine used for operation i.

$$\Omega(M_{i+1} - M_i) = \begin{cases} 1 \; if \; M_i \neq M_{i+1} \\ 0 \; if \; M_i \equiv M_{i+1} \end{cases} \tag{4}$$

- Set-up change cost (*SCC*): a set-up change is needed when two consecutive operations are not machined on the same machine with the same *TAD*. The number of set-up changes (*NSC*) can be computed as:

$$NSC = \sum_{i=1}^{n-1} \Omega_2(\Omega_1(M_i - M_{i+1}), \Omega_1(TAD_i - TAD_{i+1})) \tag{5}$$

The corresponding *SCC* can be computed as:

$$SCC = NSC \times SCCI, \tag{6}$$

Where

$$\Omega_1(X - Y) = \begin{cases} 1 & X \neq Y \\ 0 & X \equiv Y \end{cases} \tag{7}$$

$$\Omega_2(XY) = \begin{cases} 0 & X = Y = 0 \\ 1 & \text{otherwise,} \end{cases} \tag{8}$$

And *SCCI* is the set-up change cost index.

- Tool change cost (*TCC*): a tool change means that two consecutive operations are not executed on the same machine with the same tool. The number of tool changes (*NTC*) can be computed as:

$$NTC = \sum_{i=1}^{n-1} \Omega_2(\Omega_1(M_i, M_{i+1}), \Omega_1(T_i, T_{i+1})), \tag{9}$$

The corresponding *TCC* can be computed as:

$$TCC = \sum_{i=1}^{NTC} TCCI, \tag{10}$$

and the *TCCI* is the tool change cost index.

The total weighted cost (*TWC*) is defined as follows.

$$TWC = w_1{*}MC + w_2{*}TC + w_3{*}SC + w_4{*}MCC + w_5{*}TCC, \tag{11}$$

Where w_1–w_5 are the weights.

4 The proposed imperialist competitive algorithm

4.1 Frame work of imperialist competitive algorithm

ICA is a novel population-based optimization algorithm proposed by Atashpaz-Gargari and Lucas [17]. Each individual in the population is named as *country*, and the whole countries are classified into two categories: *imperialist* and *colony*. Imperialists are some of the best countries in the initial population, and the rest countries are colonies of them. Colonies of the initial population are distributed to initial imperialists. An imperialist and its corresponding colonies form an *empire*. The power of each country is used to indicate its fitness. During the iterations of the algorithm, empires compete to possess more colonies. Empires with more power have higher possibility to possess more colonies, and empires with less power will gradually lose their colonies. When all the colonies are possessed by a single imperialist, the algorithm terminates. The main steps of ICA are described as follows.

1. Assimilation. When a colony is possessed by an imperialist, it will move to the imperialist to achieve a new place in the solution space. This process is called assimilation. With respect to discrete optimization problems, assimilation of ICA is achieved by crossover operation and mutation operator [29].
2. Imperialistic competition. Since all empires try to possess more colonies to increase their power, imperialistic competition happens among empires to possess the weakest colony of the weakest empire. The possibility of an empire to possess a colony is determined by the power of the imperialist and its colonies.
3. Elimination. Powerless empire will lose its colonies during imperialist competition and elimination happens when an empire loses all its colonies.

The workflow of the ICA is given in Fig. 2.

The procedure of ICA is described as follows.

Step 1 Initialize parameters of ICA.
Step 2 Randomly generate N_{pop} number of countries. Choose N_{imp} number of best countries as imperialists and determine their colonies according to their power.
Step 3 If termination criterion is not met, repeat the following steps.
Step 4 Assimilation
Step 5 Imperialistic competition
Step 6 Revolution
Step 7 Elimination of powerless empires

4.2 Initial countries generation

4.2.1 Country representation

In this paper, a representation scheme that takes all the flexibilities into consideration is proposed. Figure 3 shows the encoding scheme of the part given in Fig. 1. As is shown in Fig. 3, each country that represents a potential solution to the process planning problem consists of two substrings: the head string and the tail string. The head

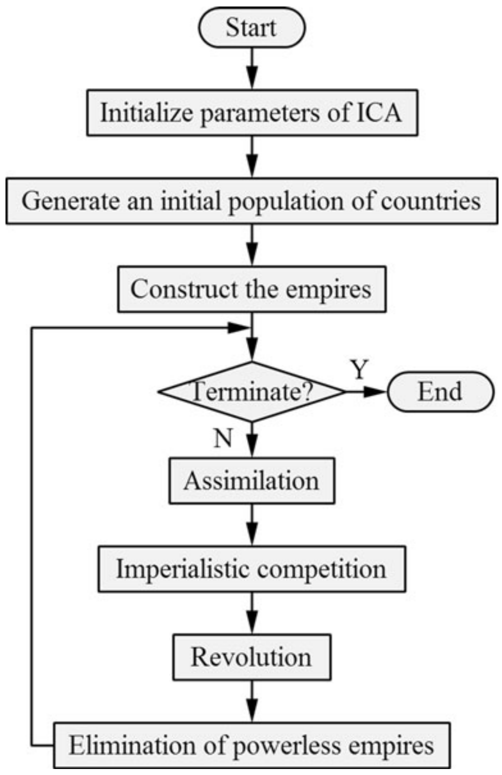
Start
Initialize parameters of ICA
Generate an initial population of countries
Construct the empires
Terminate?
Y
End
N
Assimilation
Imperialistic competition
Revolution
Elimination of powerless empires

Fig. 2 The workflow of the ICA

string contains manufacturing information about all the operations needed to produce a part. The head string is made up of operation sequence, machine sequence, tool sequence, and TAD sequence. The length of the head string equals the total number of operations in the network representation described above. Note that some of the operations may not be selected in the final process plan. Whether an operation is to be selected or not is decided by the tail string. The tail string consists of a sequence of binary number. The length of the tail string equals to the total number of ORs in the network representation of a part. A number of 0 indicates that a left OR path is chosen and a number of 1 indicates that a right OR path is chosen. In Fig. 3, the length of head string and tail string equals 15 and 3, respectively. Each column of the head string represents an operation. The first row indicates the operation sequence. The second row is the machine sequence which is used to specify with machine is selected to perform the operation in the corresponding position of operation sequence. The third row specifies the tool used for each operation and the fourth row indicates the TAD chosen for each operation. The head string contains sequence flexibility represented by operation sequence, machine flexibility by machine sequence, tool flexibility by tool sequence, and TAD flexibility by TAD sequence. And the tail string represents the process flexibility.

Since complex precedence constraints exist among operations, it's necessary to ensure the feasibility of operation sequence. In the proposed ICA algorithm, a constraints-handling algorithm is proposed to adjust infeasible solutions into feasible ones. The constraints-handling algorithm is described in the following section.

4.2.2 Constraints-handling algorithm

Since various precedence constraints exist among operations, initial process plans that are generated randomly or produced by crossover and mutation operator may be infeasible. In this paper, the constraints-handling algorithm proposed by Tseng [39] is adopted to adjust infeasible solutions into feasible ones. This algorithm firstly constructs a binary-tree structure based on the precedence relationships among operations. And the feasible solution is obtained by inorderly traversing the binary tree. Notations used in the algorithm are given as follows [39].

P	A process plan that is made up of operations sequence.
g_h	The operation in the hth position of the process plan P
r	Root node point
l	Leaf node point

The constraints adjusting algorithm is described as follow.

```
Step 1  Set $h$=2.
Step 2  Set $g_1$'s corresponding operation at root node point $R$.
Step 3  Set $g_h$'s corresponding operation at leaf node point $l$, and decide the precedence relationship of $r$ and $l$.
  1. If $p_{r,l}$=1, operation 1 should be machined before operation r.
     (a) If $r$'s left child node point is not empty, then set $r$'s left node point at the new root node point $r$ and repeat Step 3.
     (b) If $r$'s left child node point is empty, then insert $l$ at $r$'s left node point. Set $h$=$h$+1 and go to Step 4.
  2. If $p_{r,l}$=0, there is no precedence constraints between operation $r$ and $l$.
     (a) If $r$'s right node child node point is not empty, then set $r$'s right node point at the new root node point $r$ and go to Step 3.
     (b) If $r$'s right child node point is empty, then insert $l$ at $r$'s right node point. Set $h$=$h$+1, and go to Step 4.
Step 4  If $h$=$m$, go to Step 5; otherwise go to Step 2.
Step 5  List feasible solutions according to the inorder traversal rank and stop the algorithm.
```

The example given in Fig. 1 is used here to illustrate the constraints-handling algorithm. To apply the algorithm, the precedence matrix of operations C must be predetermined. As is shown in Fig. 4, a number of 1 at the ith row and jth

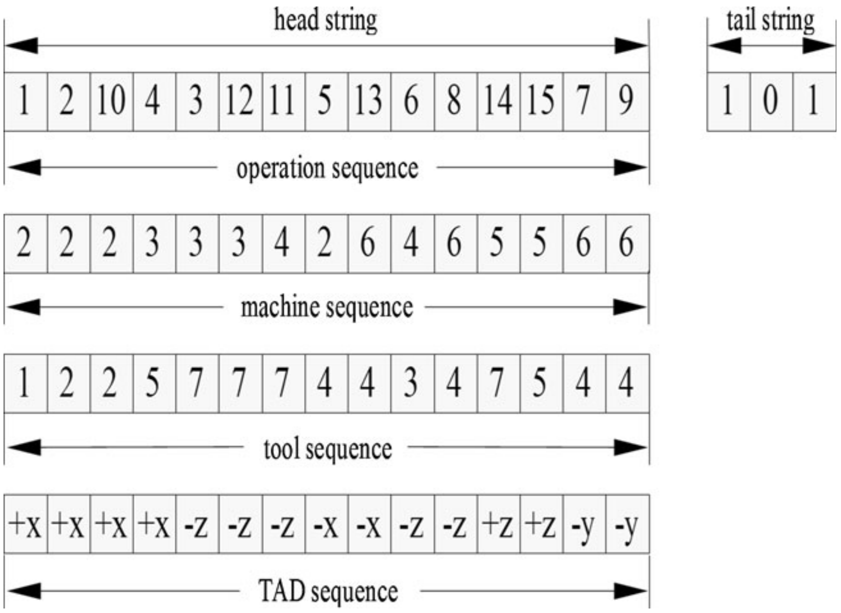

Fig. 3 The encoding scheme of the part given in Fig. 1

column indicates that operation *i* must be performed after operation *j*.

Suppose there is an infeasible operation sequence:13, 5, 14, 2, 4, 3, 12, 10, 6, 11, 15, 1, 9, 7, 8

Figure 5 shows the construction of the binary tree of the infeasible operation sequence.

Operation 13 is set as the root node first.

For operation 5, since $C_{5,13}=0$, which means that there exists no precedence constraints between operation 5 and operation 13, so operation 5 is placed at the right leaf of operation 13.

For operation 14, since $C_{13,14}=0$, operation 14 should be placed at the right leaf node of operation 13. Since there exists an operation 5 at the right leaf node operation 13, operation 5 becomes the new root node and the value of $C_{5,14}$ is checked. Since $C_{5,14}=0$, operation 14 is placed at the right leaf node of operation 5.

$$\begin{array}{c|ccccccccccccccc|}
 & O_1 & O_2 & O_3 & O_4 & O_5 & O_6 & O_7 & O_8 & O_9 & O_{10} & O_{11} & O_{12} & O_{13} & O_{14} & O_{15} \\
O_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_4 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_5 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_6 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_7 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_8 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_9 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_{10} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
O_{11} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
O_{12} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
O_{13} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
O_{14} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
O_{15} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0
\end{array}$$

Fig. 4 A number of 1 at the *i*th row and *j*th column

For operation 2, since $C_{13,2}=0$ and $C_{5,2}=1$, operation 2 should be placed at the left leaf node of operation 5.

Following the similar steps described above, a binary tree can be constructed. The feasible operation sequence is obtained by inorderly traversing the binary tree.

4.2.3 Objective evaluation

The example part given in Fig. 1 consists of five flexibilities, and the *TWC* equals to the weighted sum of *MC*, *TC*, *MCC*, *TCC*, and *SCC*.

4.3 Initial empires construction

Using the encoding scheme described above, a number of countries can be generated randomly and evaluated. To start the algorithm, some best countries are selected as the imperialists. Suppose the size of initial population is N_{pop}, the number of imperialists is N_{imp} and the number of colonies is $N_{\mathrm{col}}(N_{\mathrm{pop}} = N_{\mathrm{imp}} + N_{\mathrm{col}})$. To divide the colonies among the selected imperialists, the normalized cost of each imperialist must be computed using the following formula.

$$C_n = \max(c_i) - c_n \tag{12}$$

Where c_n is the cost (makespan) of *n*th imperialist, and C_n is its normalized cost. The colonies are distributed among imperialists based on their normalized power. The normalized power of each imperialist is defined by

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{\mathrm{imp}}} C_i} \right| \tag{13}$$

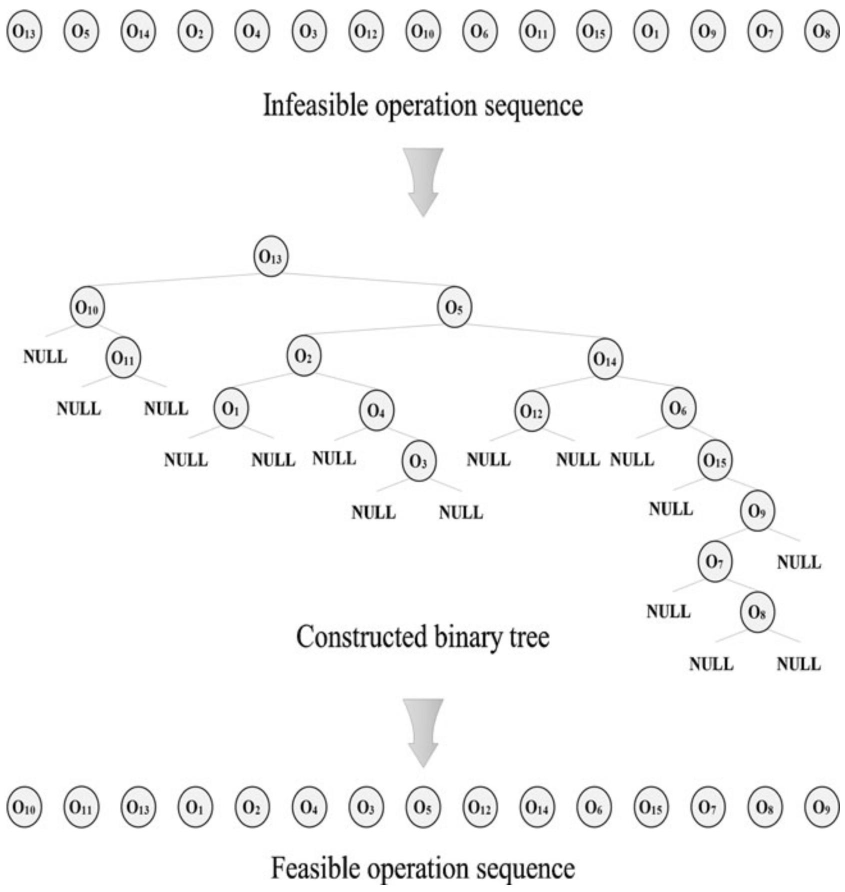

Fig. 5 The construction of the binary tree of the infeasible operation sequence

Then, the number of colonies of an empire will be

$$NC_n = \text{round}(p_n \cdot N_{\text{col}}) \qquad (14)$$

An imperialist with its corresponding colonies construct an empire.

4.4 Assimilation

Assimilation happens between an imperialist and its colonies and is accomplished by the following steps.

4.4.1 Moving colonies of an empire toward the imperialist

The movement of a colony to its imperialists is accomplished by crossover and mutation operator.

- Crossover

Two-point crossover operator is used in this paper and is illustrated in Fig. 6. The procedure of the crossover is described as follows.

Crossover of the head string:

Step 1 Select two crossover points randomly.
Step 2 Copy the elements after the first crossover points and before the second crossover points of the imperialist to the same positions of the new colony. Delete these elements from the colony.
Step 3 Copy the remaining elements of the colony to the undetermined positions of the new colony sequentially as they appear in the colony.

Crossover of the tail string:

Step 1 Select two crossover points randomly.
Step 2 Copy the elements after the first crossover points and before the second crossover points of the imperialist to the same positions of the new colony. Delete these elements from the colony.
Step 3 Copy the remaining elements of the colony to the undetermined positions of the new colony sequentially as they appear in the colony.

- Mutation

Swap and Insert are used as the mutation operator. As is shown in Fig. 7, the mutation operator consists of mutation of the head string and mutation of the tail string.

Mutation of the head string includes *swap*, *insert*, and operation mutation.

Swap	Choose two random elements of the head string and exchange their positions.

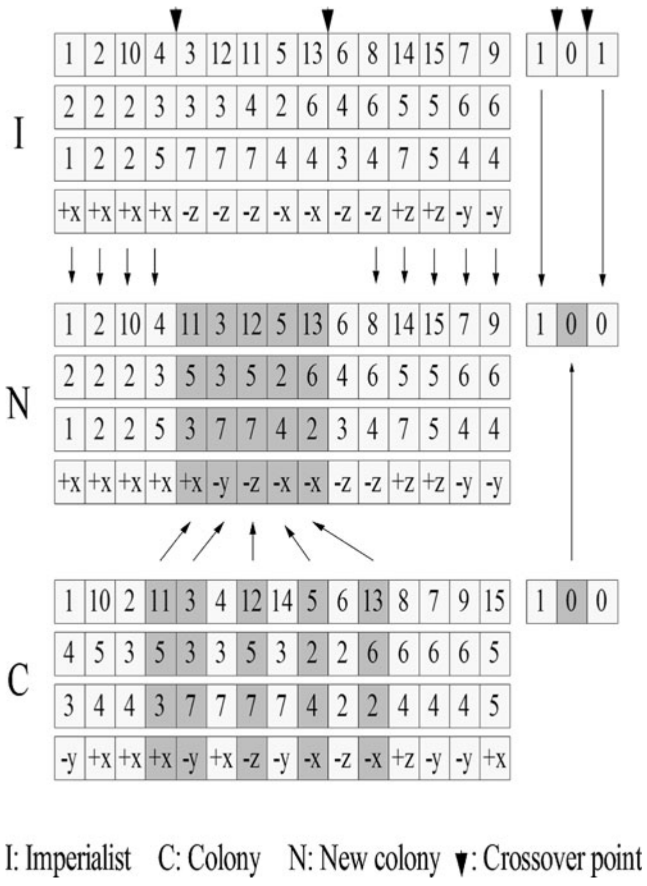

Fig. 6 Two-point crossover operator

Insert	Choose a random element of the head string and insert it to another randomly selected position.
Operation mutation	Randomly choose an element of the head string and replace its machine, tool, and TAD with alternative machine, tool, and TAD.

Note that mutation of the head string may result in infeasible solutions, and the constraints-handling algorithm is applied to the infeasible solutions to adjust them into feasible ones.

Mutation of the tail string indicates the act of changing a randomly determined element of the tail string to its opposite value.

4.4.2 Exchanging positions of an imperialist and a colony

After moving toward the imperialist, a colony may reach a position with lower cost than that of imperialist. In this case, the colony will become the imperialist in the current empire and vice versa. In the following iterations, colonies in the empire will move to the new imperialist.

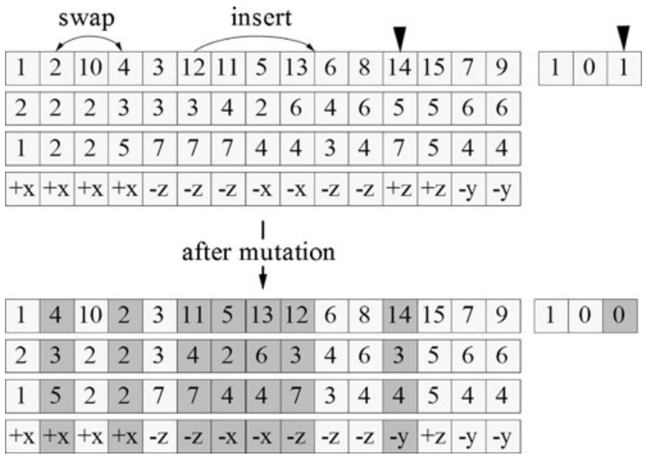

Fig. 7 The mutation operator consists of mutation of the head string and mutation of the tail string

4.5 Imperialistic competition

In ICA, all empires compete to take possession of more colonies besides their current colonies. The imperialistic competition gradually brings about a decrease in the power of weaker empires and an increase in the power of powerful ones. To model this competition among imperialists, the weakest colony of the weakest empire is freed from its current imperialist and waited to be possessed by all empires. During the competing process, each empire will have a likelihood of taking possession of the freed colony based on their total power, that is, empires with more total power will be more likely to possess it.

The total power of an empire is determined by the power of the imperialist and that of its colonies, that is, the equation of total cost is:

$$T.C._n = c_n(\text{imperialist}_n) + \alpha \cdot \text{mean}(c_n(\text{colonies of empire}_n)) \quad (15)$$

Where $T.C._n$ is the total cost of the nth empire and α is a positive number that is in the range of [0, 1]. Different value of α indicates the weight of the cost of the imperialist on the total cost of an empire.

The normalized total cost is computed by

$$N.T.C._n = \max(T.C._i) - T.C._n \quad (16)$$

Where$T.C._n$and$N.T.C._n$are total cost and normalized total cost of nth empire, respectively. Then the possession probability of each empire is given by

$$p_{p_n} = \left| \frac{N.T.C._n}{\sum_{i=1}^{N_{\text{imp}}} N.T.C._i} \right| \quad (17)$$

$$P = \left[p_{p_1}, p_{p_2}, \cdots, p_{p_{N_{\text{imp}}}}\right] \quad (18)$$

Then, a vector with the same size as P is created with each of its element uniformly distributed random numbers.

$$R = [r_1, r_2, \cdots, r_{N_{\mathrm{imp}}}] r_i \sim U(0,1) \tag{19}$$

Then, a vector D is formed by simply subtracting R from P.

$$D = P - R = [D_1, D_2, \cdots, D_{N_{\mathrm{imp}}}] \tag{20}$$

The empire whose relevant index in D is biggest will take possession of the freed colony.

4.6 Revolution

Revolution indicates that the weakest colony of the weakest empire is replaced by a random solution [29].

4.7 Elimination

When an empire loses all of its colonies, it will be eliminated from the population.

5 Experiments and discussions

5.1 Experimental design and parameter setting

In this section, performance of the proposed ICA in the optimization of process planning with various flexibilities is validated through three experiments using benchmark problems taken from literature. Benchmark problems of experiments 1 and 2 were proposed by Kim et al. [2] and Seok Shin et al. [3], respectively. Details of these problems are available in Kim [40] and Seok Shin et al. [41] and are omitted here. Experiment 3 consists of five extensively studied problems taken from literature. These three sets of experiments involve process planning problems with various flexibilities: Experiment 1 considers process planning with machine flexibility, process flexibility, and sequence flexibility; experiment 2 represents process planning with machine flexibility, tool flexibility, process flexibility, and sequence flexibility; and experiment 3 indicates process planning with machine flexibility, tool flexibility, TAD flexibility, and sequence flexibility.

For experiments 1 and 2, performance of ICA is evaluated and compared with that of GA and SA. We implemented the GA and SA by following the works of Li et al. [1] and Ma et al. [9]. Preliminary experiments have been conducted to determine the optimal values of the parameters of GA and SA. Parameters of GA were set as follows: population size N_{GA}=50, crossover rate P_c=0.8, mutation rate P_m=0.1, maximum number of generations N_{gen}=1,000 and roulette wheel selection was used as the selection scheme. For each generation of GA, the best individual is saved directly to the next generation. Parameters of SA were set as follows: Initial temperature T_{init}= 1000.0, ending temperature T_{end}=0.01, annealing rate R_a= 0.99, the number of iterations for each level of temperature N_{iter}=5. Note that both the GA and ICA used the same crossover and mutation operators described above. SA used the same mutation operator of ICA to generate neighborhood solutions.

Parameters of ICA include the total number of countries, N_{pop}, the total number of empires, N_{imp}, the weight α and the maximum number of iterations, $N_{\max}$. To determine the appropriate values of these parameters, preliminary turning experiments were conducted, and computational results showed that the following parameter setting could result in good balance between computational time and efficiency: N_{pop}=50, N_{imp}=4, α=0.6, $N_{\max}$=500. All the algorithms implemented in this paper were developed in C++ and run on a personal computer with a 2.0 GHz Intel Core2 Duo CPU.

5.2 Experiment 1

Experiment 1 consists of 18 parts with different number of operations. In this experiment, three flexibilities including machine flexibility, process flexibility, and sequence flexibility are taken into consideration. Computational results are shown in Table 1. For each problem, 20 independent runs are conducted, and the best, worst, and mean machining costs are reported. In the table, the first column denotes the problem number and the second the total number of operations (*TO*) for each problem. The *MCCI* in this experiment is set to 30. It can be seen from Table 1 that ICA outperformed GA and SA in all problems with respect to the best and mean machining cost.

5.3 Experiment 2

Experiment 2 consists of 18 parts with different number of operations. In this experiment, three flexibilities including machine flexibility, tool flexibility, process flexibility, and sequence flexibility are considered. The experimental results are shown in Table 2. The experiment is repeated 20 times for each problem, and the best, worst, and mean machining costs are reported. In the table, the first column denotes the problem number and the second the total number of operations (*TO*) for each problem. The *MCCI* and *TCCI* are set to 30 and 20, respectively. It can be seen from Table 2 that the proposed ICA gives more competitive results in all problems when compared with GA and SA. Note that, in this experiment,

Table 1 Computational results of experiment 1

No.	TO	GA			SA			ICA		
		Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean
1	8	197.0	197.0	197.0	197.0	197.0	197.0	197.0	197.0	197.0
2	14	187.0	192.0	187.4	187.0	187.0	187.0	187.0	187.0	187.0
3	19	198.0	321.0	237.6	163.0	231.0	194.0	163.0	193.0	178.6
4	16	269.0	293.0	270.2	269.0	269.0	269.0	269.0	269.0	269.0
5	18	124.0	159.0	142.6	124.0	154.0	139.2	124.0	124.0	124.0
6	20	220.0	280.0	241.8	190.0	250.0	220.7	190.0	190.0	190.0
7	21	206.0	236.0	210.8	206.0	236.0	213.5	206.0	206.0	206.0
8	20	148.0	148.0	148.0	148.0	148.0	148.0	148.0	148.0	148.0
9	20	164.0	227.0	189.4	140.0	198.0	166.4	140.0	164.0	142.5
10	11	137.0	137.0	137.0	137.0	137.0	137.0	137.0	137.0	137.0
11	9	119.0	179.0	141.5	119.0	149.0	125.0	119.0	119.0	119.0
12	18	307.0	367.0	333.5	307.0	337.0	323.5	307.0	307.0	307.0
13	18	133.0	163.0	137.7	133.0	163.0	140.5	133.0	133.0	133.0
14	13	127.0	157.0	131.5	127.0	128.0	127.1	127.0	127.0	127.0
15	15	156.0	216.0	180.0	156.0	186.0	159.0	156.0	156.0	156.0
16	21	147.0	164.0	149.5	147.0	164.0	148.7	147.0	147.0	147.0
17	22	167.0	197.0	173.2	167.0	222.0	169.7	167.0	167.0	167.0
18	17	182.0	242.0	209.3	182.0	212.0	197.5	182.0	182.0	182.0

another flexibility, that is, tool flexibility, is taken into consideration compared with experiment 1. Computational results in Table 2 confirm the proposed ICA's effectiveness in solving process planning with various flexibilities.

5.4 Experiment 3

Experiment 3 consists of five problems with various numbers of operations. In this experiment, the impact of

Table 2 Computational results of experiment 2

No.	TO	GA			SA			ICA		
		Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean
1	16	627	717	648.0	630.0	663.0	640.5	627	631	629.2
2	20	413	488	450.6	409.0	487.0	440.3	409	428	417.4
3	18	667	798	738.1	604	759	650.4	603	638	648.8
4	20	657	759	717.9	617	709	668.6	613	660	645.6
5	15	417	512	461.9	371	451	398.1	371	379	376.8
6	22	539	636	582.3	524	551	535.3	521	524	522.3
7	11	335	372	352.1	334	355	339.1	334	334	334.0
8	12	357	393	372.4	366	393	374.8	357	366	361.1
9	21	469	615	558.5	432	550	510.9	432	498	467.3
10	14	355	502	445.3	355	461	403.3	350	417	388.8
11	13	334	433	386.3	283	439	356.9	283	352	332.8
12	17	418	566	498.7	396	533	447.7	396	456	435.8
13	21	252	374	314.2	233	305	263.2	229	237	232.2
14	18	311	368	337.8	293	352	328.4	284	309	294.1
15	18	429	502	465.4	429	481	448.3	421	437	432.0
16	15	450	545	502.6	411	470	440.1	411	411	411.0
17	19	485	626	556.2	449	583	497.7	377	463	436.8
18	20	594	719	664.4	515	622	576.6	504	601	569.8

Table 3 Computational results of problem 1 in experiment 3

		ICA	TS[a]	SA[a]	GA[a]	PSO[b]
Codition (a)						
	Mean	2527.5	2609.6	2668.5	2796.0	2680.5
	Maximum	2530.0	2690.0	2829.0	2885.0	–
	Minimum	2525.0	2527.0	2535.0	2667.0	2535.0
Codition (b)						
	Mean	2090.0	2208.0	2287.0	2370.0	–
	Maximum	2090.0	2390.0	2380.0	2580.0	–
	Minimum	2090.0	2120.0	2120.0	2220.0	–

[a] Results are taken from [10]
[b] Results are taken from [12]

different objectives and different manufacturing conditions in actual shop floor on the results of process planning were studied. The performance of ICA was compared with some existing algorithms including GA, SA, TS, and PSO developed for the process planning problem.

5.4.1 Problem 1

Problem 1 is taken from Li et al. [1]. It has 20 operations, and the following two conditions are considered.

(a) All machines and tools are available, and w_1–w_5 in Eq. 11 are set as 1;
(b) All machines and tools are available, and w_2=w_5=0, w_1=w_3=w_4=1;

It can be seen from Table 3 that ICA obtained new better solutions under condition (a) and condition (b). The robustness of ICA under the two conditions is better than that of GA, SA, TS, and PSO.

5.4.2 Problem 2

The second part used by Li et al. [10] consists of 14 operations. Two conditions are considered for studies on this part.

Table 4 Computational results of problem 2 in experiment 3

		ICA	TS[a]	SA[a]	GA[a]
Codition (a)					
	Mean	1,328.0	1,342.0	1,373.5	1,611.0
	Maximum	1,328.0	1,378.0	1,518.0	1,778.0
	Minimum	1,328.0	1,328.0	1,328.0	1,478.0
Codition (b)					
	Mean	1,170.0	1,194.0	1,217.0	1,482.0
	Maximum	1,170.0	1,290.0	1,345.0	1,650.0
	Minimum	1,170.0	1,170.0	1,170.0	1,410.0

[a] Results are taken from [10]

Table 5 Computational results of problem 3 in experiment 3

		ICA	SA[a]
Condition (a)			
	Minimum	743.0	833.0
Condition (b)			
	Minimum	1,198.0	1,288.0

[a] Results are taken from [9]

(a) All machines and tools are available, and w_1–w_5 in Eq. 12 are set as 1;
(b) All machines and tools are available, and w_2=w_5=0, w_1=w_3=w_4=1;

Table 4 shows that the proposed ICA can provide competitive results on two conditions.

5.4.3 Problem 3

The third part presented by Ma et al. [9] consists of nine features and 13 operations. The following two conditions are considered for studies on this part.

1. All machines and tools are available;
2. Machine 2 is down;

Table 5 shows that ICA outperforms SA in both condition (a) and condition (b). And new better solutions are achieved.

5.4.4 Problem 4

The fourth part is presented by Guo et al. [12] to test the efficiency of PSO on process planning problem. This part consists of 11 features and 14 operations. Only the condition that all machining resources are available is considered. Table 6 shows that ICA outperforms GA, SA, and PSO in this problem.

5.4.5 Problem 5

This part is used by Zhang et al. [7] to test GA's capability and flexibility of handling process planning problems under different requirements. It contains 19 machining features and a total number of 23 operations. In this paper, the

Table 6 Computational results of problem 4 in experiment 3

	ICA	PSO[a]	SA[a]	GA[a]
Mean	1,364.1	1,430.0	1,447.4	1,459.4
Minimum	1,357.0	1,361.0	1,421.0	1,381.0

[a] Results are taken from [12]

following four conditions were considered to compare the performance of ICA and GA. Following the work of Zhang et al. [7], 60 trials were conducted for each condition.

1. Minimizing the total weighted cost.

   The average machining cost over 60 trials is 1,741; the minimum machine cost is 1,739, and the maximum machining cost is 1,749. The frequency of the machining cost (1,739) is much higher than that of other cost.
2. Minimizing the number of machine changes only.

   All the 60 trials find a process plan with zero machine changes.
3. Minimizing the number of setup changes only.

   All the 60 trials find a process plan with zero setup changes.
4. Minimizing the number of tool changes only.

   All the 60 trials find a process plan with zero tool changes.

6 Discussions

It can be seen from the computational results of the above three experiments that ICA has advantages over existing approaches in solving the process planning problems with various flexibilities. In both experiment 1 and experiment 2, ICA shows better efficiency and robustness compared with GA and SA. In experiment 3, the performance of ICA was compared with that of GA, SA, TS, and PSO; experiments on different problems indicate that ICA is superior to these existing algorithms in computational effectiveness and efficiency.

7 Conclusions

This paper discusses the process planning problem in which various flexibilities are considered. Optimization of process planning is a NP-hard problem and efficient heuristic algorithms should be proposed to obtain near-optimal solutions with reasonable computational cost. In this paper, we first concentrated the process planning problem with various flexibilities. Then, a novel metaheuristic algorithm, that is, ICA was utilized to find near-optimal solutions. The performance of the proposed ICA was validated over three experiments using benchmark problems taken from literature and also compared with many other algorithms developed for the optimization of process planning in the literature. Computational results show the efficiency of the algorithm. In future, the proposed ICA could be employed to solve some other industrial optimization problems. In addition, process planning problem with various flexibilities considered in this paper could be applied in industrial applications.

Acknowledgment This research is supported by the State Key Program of National Natural Science of China (Grant No. 51035001), National Natural Science Foundation of China (Grant NO. 50825503), and National Natural Science Foundation of China (Grant No. 50875101).

Reference

1. Li WD, Ong SK, Nee AYC (2002) Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts. Int J Prod Res 40(8):1899–1922. doi:10.1080/00207540110119991
2. 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
3. Seok Shin K, Park J-O, Keun Kim Y (2011) Multi-objective FMS process planning with various flexibilities using a symbiotic evolutionary algorithm. Comput Oper Res 38(3):702–712
4. Liu X-j, Yi H, Ni Z-h (2010) Application of ant colony optimization algorithm in process planning optimization. Journal of Intelligent Manufacturing (in press)
5. Leo A, Hongchao Z (1989) Computer-aided process planning: the state-of-the-art survey. Int J Prod Res 27(4):553
6. Marri HB, Gunasekaran A, Grieve RJ (1998) Computer-aided process planning: a state of art. Int J Adv Manuf Technol 14 (4):261–268. doi:10.1007/bf01199881
7. Zhang F, Zhang YF, Nee AYC (1997) Using genetic algorithms in process planning for job shop machining. Evol Comput IEEE Transac 1(4):278–289
8. Qiao L, Wang X-Y, Wang S-C (2000) A GA-based approach to machining operation sequencing for prismatic parts. Int J Prod Res 38(14):3283–3303
9. Ma GH, Zhang YF, Nee AYC (2000) A simulated annealing-based optimization algorithm for process planning. Int J Prod Res 38 (12):2671–2687
10. Li WD, Ong SK, Nee AYC (2004) Optimization of process plans using a constraint-based tabu search approach. Int J Prod Res 42 (10):1955–1985. doi:10.1080/00207540310001652897
11. Li L, Fuh JYH, Zhang YF, Nee AYC (2005) Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments. Robot Comput-Integr Manuf 21 (6):568–578
12. Guo Y, Mileham A, Owen G, Li W (2006) Operation sequencing optimization using a particle swarm optimization approach. Proc Inst Mech Eng, Part B: J Eng Manuf 220(12):1945–1958
13. Salehi M, Tavakkoli-Moghaddam R (2009) Application of genetic algorithm to computer-aided process planning in preliminary and detailed planning. Eng Appl Artif Intell 22(8):1179–1187
14. Shao X, Li X, Gao L, Zhang C (2009) Integration of process planning and scheduling—a modified genetic algorithm-based approach. Comput Oper Res 36(6):2082–2096
15. Leung CW, Wong TN, Mak KL, Fung RYK (2010) Integrated process planning and scheduling by an agent-based ant colony optimization. Comput Ind Eng 59(1):166–180
16. Li X, Gao L, Shao X, Zhang C, Wang C (2010) Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. Comput Oper Res 37(4):656–667
17. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Evolutionary Computation. CEC 2007. IEEE Congress on, p 2007. pp 4661–4667
18. Atashpaz-Gargari E, Caro L (2007) Designing an optimal PID controller using Colonial Competitive Algorithm. In: First Iranian Joint Congress on Intelligent and Fuzzy Systems

19. Atashpaz-Gargari E, Hashemzadeh F, Lucas C (2008) Designing MIMO PIID controller using colonial competitive algorithm: applied to distillation column process. In: Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on. pp 1929–1934
20. Gargari EA, Hashemzadeh F, Rajabioun R, Lucas C (2008) Colonial competitive algorithm: a novel approach for PID controller design in MIMO distillation column process. Int J Intell Comput Cybern 1 (3):337–355. doi:10.1108/17563780810893446
21. Rajabioun R, Atashpaz-Gargari E, Lucas C (2008) Colonial competitive algorithm as a tool for Nash equilibrium point achievement. In: Gervasi O, Murgante B, Laganà A, Taniar D, Mun Y, Gavrilova M (eds) Computational science and its applications—ICCSA 2008, vol 5073. Lecture Notes in Computer Science. Springer Berlin, Heidelberg, pp 680–695. doi:10.1007/978-3-540-69848-7_55
22. Khabbazi A, Gargari EA, Lucas C (2009) Imperialist competitive algorithm for minimum bit error rate beamforming. Int J Bio-Inspir Comput 1(1/2):125–133. doi:10.1504/IJBIC.2009.022781
23. Forouharfard S, Zandieh M (2010) An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems. Int J Adv Manuf Technol 51(9):1179–1193. doi:10.1007/s00170-010-2676-5
24. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. Comput Struct 88(21–22):1220–1229
25. Lucas C, Nasiri-Gheidari Z, Tootoonchian F (2010) Application of an imperialist competitive algorithm to the design of a linear induction motor. Energy Convers Manag 51(7):1407–1411
26. Nazari-Shirkouhi S, Eivazy H, Ghodsi R, Rezaie K, Atashpaz-Gargari E (2010) Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm. Expert Syst Appl 37(12):7615–7626
27. Sarayloo F, Tavakkoli-Moghaddam R (2010) Imperialistic competitive algorithm for solving a dynamic cell formation problem with production planning. In: Huang D-S, Zhao Z, Bevilacqua V, Figueroa J (eds) Advanced Intelligent Computing Theories and Applications, vol 6215. Lecture Notes in Computer Science. Springer Berlin, Heidelberg, pp 266–276. doi:10.1007/978-3-642-14922-1_34
28. Sayadnavard MH, Haghighat AT, Abdechiri M Wireless sensor network localization using imperialist competitive algorithm. In: Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, 2010. pp 818–822
29. Shokrollahpour E, Zandieh M, Dorri B (2010) A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. Int J Prod Res 49(11):3087–3103
30. Moghimi Hadji M, Vahidi B (2011) A solution to the unit commitment problem using imperialistic competition algorithm. Power Syst, IEEE Trans on PP 99:1–1
31. Bagher M, Zandieh M, Farsijani H (2010) Balancing of stochastic U-type assembly lines: an imperialist competitive algorithm. The International Journal of Advanced Manufacturing Technology: 1–15. doi:10.1007/s00170-010-2937-3
32. Niknam T, Taherian Fard E, Pourjafarian N, Rousta A (2011) An efficient hybrid algorithm based on modified imperialist competitive algorithm and K-means for data clustering. Eng Appl Artif Intell 24(2):306–317
33. Abdechiri M, Faez K, Bahrami H (2010a) Adaptive imperialist competitive algorithm (AICA). In: Cognitive Informatics (ICCI). 9th IEEE International Conference on, p 2010. pp 940–945
34. Abdechiri M, Faez K, Bahrami H (2010b) Neural network learning based on chaotic imperialist competitive algorithm. In: Intelligent Systems and Applications (ISA). 2nd International Workshop on, p 2010. pp 1–5
35. Bahrami H, Faez K, Abdechiri M (2010) Imperialist competitive algorithm using chaos theory for optimization (CICA). In: Computer Modelling and Simulation (UKSim). 12th International Conference on, p 2010. pp 98–103
36. Duan H, Xu C, Liu S, Shao S (2010) Template matching using chaotic imperialist competitive algorithm. Pattern Recognit Lett 31(13):1868–1875
37. Karimi N, Zandieh M, Najafi AA (2010) Group scheduling in flexible flow shops: a hybridised approach of imperialist competitive algorithm and electromagnetic-like mechanism. International Journal of Production Research (in press)
38. Ho YC, Moodie CL (1996) Solving cell formation problems in a manufacturing environment with flexible processing and routing capabilities. Int J Prod Res 34(10):2901–2923
39. 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
40. Kim YK (2003) A set of data for the integration of process planning and job shop scheduling. http://syslab.chonnam.ac.kr/links/data-pp&s.doc.
41. Test-bed problems for multi-objective FMS process planning using multi-objective symbiotic evolutionary algorithm. (2010) http://syslab.chonnam.ac.kr/links/MO_FMS_PP_MOSEA.doc.