International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Integrated process planning and scheduling using an imperialist competitive algorithm

Kunlei Lian [a] , Chaoyong Zhang [a] , Liang Gao [a] & Xinyu Li [a]
[a] State Key Laboratory of Digital Manufacturing Equipment & Technology, Huazhong University of Science and Technology, Wuhan, Hubei, China

Available online: 02 Nov 2011

To cite this article: Kunlei Lian, Chaoyong Zhang, Liang Gao & Xinyu Li (2011): Integrated process planning and scheduling using an imperialist competitive algorithm, International Journal of Production Research, DOI:10.1080/00207543.2011.622310

To link to this article: http://dx.doi.org/10.1080/00207543.2011.622310

International Journal of Production Research
2011, 1–18, iFirst

Taylor & Francis
Taylor & Francis Group

Integrated process planning and scheduling using an imperialist competitive algorithm

Kunlei Lian, Chaoyong Zhang*, Liang Gao and Xinyu Li

State Key Laboratory of Digital Manufacturing Equipment & Technology, Huazhong University of Science and Technology, Wuhan, Hubei, China

(*Received 9 January 2011; final version received 5 August 2011*)

Effective performance of modern manufacturing systems requires integrating process planning and scheduling more tightly, which is consistently challenged by the intrinsic interrelation and intractability of these two problems. Traditionally, these two problems are treated sequentially or separately. Integration of process planning and scheduling (IPPS) provides a valuable approach to improve system performance. However, IPPS is more complex than job shop scheduling or process planning. IPPS is strongly NP-hard in that, compared to an NP-hard job shop scheduling problem with a determined process plan, the process plan for each job in IPPS is also to be optimised. So, an imperialist competitive algorithm (ICA) is proposed to address the IPPS problem with an objective of makespan minimisation. An extended operation-based representation scheme is presented to include information on various flexibilities of process planning with respect to determined job shop scheduling. The main steps of the proposed ICA, including empires construction, assimilation, imperialistic competition, revolution and elimination, are elaborated using an illustrative example. Performance of the proposed ICA was evaluated on four sets of experiments taken from the literature. Computational results of the ICA were compared with that of some existing algorithms developed for IPPS, which validates the efficiency and effectiveness of the ICA in solving the IPPS problem.

Keywords: imperialist competitive algorithm; integrated process planning and scheduling; scheduling

1. Introduction

Process planning and scheduling are two important functions in manufacturing systems. Both of them involve assignment of resources. Process planning links computer aided design and computer aided manufacturing by specifying resources needed to produce a part and determining detailed instructions for transforming raw materials into the final product. Scheduling is the act of assigning operations of all the jobs on available machines with precedence relationships among operations satisfied to optimise some predefined objectives. In traditional approaches, process planning and scheduling are conducted sequentially or separately, where scheduling of jobs is performed after the process plan for each job has been generated. These approaches suffer from the following problems (Li *et al*. 2010a).

(1) Traditional job shop scheduling with predetermined process plans limits the flexibility of the manufacturing system to adapt to dynamic changes of shop status.
(2) Process planning without consideration of actual job shop status may lead the process planners to choose desirable machining resources for each job, which usually results in unbalanced resource loads.
(3) Process plans generated separately may become infeasible in scheduling processes because of dynamic changes of shop floor status.
(4) Optimisation objectives of process planning and scheduling are usually different. Conflicting problems may occur if there is no appropriate consideration and coordination.

To overcome these problems, it is necessary to integrate process planning and scheduling more tightly. In recent years, integrated process planning and scheduling (IPPS) has attracted more and more attention, and numerous efforts have been made to research applications of the IPPS system. Among the approaches proposed in the past few years, the three best known approaches (Baykasoğlu and Özbakır 2009) are stated below.

(1) *Nonlinear process planning (NLPP)*. In NLPP, all possible process plans for a part are generated and ranked according to certain criteria. Then, the scheduling function repeatedly selects a process plan until a suitable plan is found for each part.

*Corresponding author. Email: zcyhust@mail.hust.edu.cn

ISSN 0020–7543 print/ISSN 1366–588X online
© 2011 Taylor & Francis
http://dx.doi.org/10.1080/00207543.2011.622310
http://www.tandfonline.com

(2) *Closed loop process planning (CLPP)*. CLPP differs from NLPP in that it introduces a dynamic feedback process. In CLPP, scheduling process gives information of availability of machine resources to process planning processes based on the status of shop floor.
(3) *Distributed process planning (DPP)*. Process planning in DPP is divided into two phases. In the first phase, machining features and their relationships are analysed, and corresponding manufacturing processes are determined. In the second phase, final process plans are generated based on the negotiation of required job operations and available manufacturing resources. The detailed process plans and scheduling plans are obtained simultaneously.

Due to their advantages in solving combinatorial optimisation problems, metaheuristic algorithms, including the genetic algorithm (GA), the simulated annealing (SA), particle swarm optimisation (PSO), ant colony optimisation (ACO) and the evolutionary algorithm (EA), have been applied to the IPPS problem in the past few years. Tan and Khoshnevis (2000), Li *et al*. (2010b) and Phanden *et al*. (2011) have separately presented a review on IPPS. Morad and Zalzala (1999) suggested a GA approach by simultaneously considering processing capabilities of machines, processing costs with the scheduling of jobs. Lee and Kim (2001) implemented a simulation-based GA where a simulation module was integrated to compute performance measures. Zhang *et al*. (2003) reported a unique method for IPPS in a batch-manufacturing environment. Li and McMahon (2007) presented an adaptive SA algorithm to optimise makespan, machine utilisation, job tardiness and manufacturing costs in an IPPS system. Guo *et al*. (2009a) proposed a PSO application to the optimisation of IPPS problems. New operators were incorporated into the basic PSO approach to optimise multiple criteria, including makespan, total job tardiness and balanced level of machine utilisation. Girish and Jawahar (2009) addressed the job shop scheduling problem (JSP) associated with multiple job routings and two metaheuristic algorithms, namely, the GA and ACO, were employed to find the optimal allocation of operations to the machines for minimum makespan criterion. Wang *et al*. (2009) presented a dynamic approach to reduce tardy jobs through the IPPS in a batch-manufacturing environment. Cai *et al*. (2009) proposed a cross-machine setup planning approach using GAs for machines with different configurations. Leung *et al*. (2010) utilised an ACO algorithm in an agent-based system to integrate process planning and shop floor scheduling. In their proposed approach, artificial ants were implemented as software agents, and a graph-based solution method was proposed. Shao *et al*. (2009) modified the GA to facilitate the integration and optimisation of process planning and scheduling. Li *et al*. (2010a) presented a mathematical model and an EA-based approach for the IPPS problem. Li *et al*. (2010c) proposed a hybrid algorithm to address the IPPS problem where tabu search (TS) was used as a local search strategy to improve the GA's search capability. Agent-based approaches were also proposed by some researchers to solve the IPPS problem (Wong *et al*. 2006a, 2006b, Zattar *et al*. 2010, Nejad *et al*. 2011). Weiming *et al*. (2006) presented a review that was focused on agent-based approaches to process planning and scheduling integration.

Examination of existing researches on the IPPS problem reveals that there mainly exist three optimisation strategies in the literature, and brief introductions, as well as comments, are given below.

(1) The first optimisation strategy is based on the hypothesis that several parallel searches for different pieces of the solution are more efficient than a single search for the entire solution (Moriarty and Miikkulainen 1997). The symbiotic evolutionary algorithm (SEA) proposed by Kim *et al*. (2003) belongs to this approach. It divides IPPS into several subproblems, including process planning for each part and scheduling of all parts. An entire solution is constructed by combining all partial solutions to subproblems. This optimisation strategy makes full use of the capability of parallel searches for different pieces of IPPS solution, and can obtain satisfactory results in a reasonable time. However, results achieved by the representative SEA algorithm of this optimisation strategy are inferior when compared to that of some other algorithms like the modified GA (Shao *et al*. 2009), EA (Li *et al*. 2010a) and hybrid algorithm (Li *et al*. 2010c).
(2) The second optimisation strategy consists of two stages. In the first stage, a number of optimal or near-optimal process plans for each part are determined. In the second stage, a process plan is selected for each part, and the overall schedule is optimised. Based on this approach, a modified GA (Shao *et al*. 2009), EA (Li *et al*. 2010a) and hybrid algorithm (Li *et al*. 2010c) have been proposed to address the IPPS problem. This optimisation strategy reduces the complexity of the IPPS problem and therefore saves computational time at the expense of raising the risk of missing the optimal solutions because only some near-optimal process plans for each part are considered.

(3) The third optimisation strategy is based on agents. Agent-based technology has been employed by many researchers to deal with the IPPS problem (Wong *et al.* 2006a, 2006b, Li *et al.* 2010d). It provides a valuable approach to obtain promising solutions.

In this paper, a new optimisation strategy, which conducts the generation of a schedule plan and the determination of a process plan for each part simultaneously, is proposed. Compared to the existing strategies described above, the optimisation strategy proposed in this paper integrates process planning and scheduling more tightly, and therefore increases the chances to obtain optimal solutions. On the other hand, efficient algorithms are needed to explore the vast solution space in a reasonable computational time. In this paper, a novel metaheuristic algorithm named the imperialist competitive algorithm (ICA) is employed to tackle the IPPS problem and, to the best of the authors' knowledge, there exists no published work on employing the ICA to address IPPS. So, this paper focuses on the application of the ICA on the IPPS, and its performance is validated through four sets of experiments. Comparisons with some existing algorithms are also provided.

Similar to other metaheuristic algorithms mimicking various kinds of natural phenomena such as natural evolution, birds flocking, fish schooling and the foraging behaviour of ants, the ICA is inspired by imperialistic competition. The ICA was proposed by Atashpaz-Gargari and Lucas (2007), and has shown its capability in dealing with various optimisation problems (Abdechiri *et al.* 2010, Bahrami *et al.* 2010, Nazari-Shirkouhi *et al.* 2010, Shokrollahpour *et al.* 2010, Karimi *et al.* 2011). The ICA was originally developed for solving continuous optimisation problems, and we modify it here to solve the IPPS problem with makespan criterion. Computational experiments and comparisons show that the proposed ICA approach outperforms some existing approaches for solving the IPPS problems.

The rest of the paper is organised as follows. Section 2 formulates the IPPS problem, Section 3 describes the implementation details of the ICA to the IPPS problem, computational experiments are given in Section 4 and Section 5 concludes the paper.

2. IPPS

2.1 *Flexible process plans and representation*

Process planning is the act of selecting necessary operations required to manufacture a part and determining the sequence of these selected operations subject to predefined precedence constraints. Three types of flexibility are usually considered in process planning, namely, operation flexibility, sequencing flexibility and processing flexibility (Kim *et al.* 2003). Operation flexibility refers to the possibility of performing an operation on alternative machines, with possibly distinct processing times and costs. This type of flexibility is also called routing flexibility. Sequencing flexibility relates to the possibility of interchanging the sequence in which manufacturing operations required for the completion of a part are performed. Process flexibility refers to the possibility of producing the same manufacturing feature with alternative operations, or alternative processes. Here, the process means a set of one or more operations. Consideration of these flexibilities can produce better performance in the optimisation of scheduling.

These flexibilities can be described using a network representation proposed by Ho and Moodie (1996). As shown in Figure 1, there are three node types in the network, namely, starting node, ending node and intermediate node. The starting node and the ending node are dummy ones and indicate the start and the completion of the manufacturing process of a part, respectively. An intermediate node indicates an operation with alternative machines that can perform the operation and corresponding machining times required for the operation according to the machines. Precedence relationships among operations are denoted using the arrows connecting them. The 'OR' symbol (named the OR-connector) is used to describe processing flexibility that the same manufacturing feature can be completed with different operation procedures. The links following a node and connected by an OR-connector are called OR-links, and for each OR-connector, only one of the links will be traversed. An OR-link path denotes an operation path that begins and ends with the same OR-connector. For the links that are not connected by OR-connectors, all of them will be traversed. In the network of job 2 in Figure 1, paths (3, 5) and (4) are two OR-links with the same OR-connector (OR2), and only one of them will be visited. Note that the OR-link path of one OR-connector can be contained by the OR-link path of another OR-connector, e.g. path (3, 5) and path (4) of OR2 are contained by path (2, 3, 4, 5, 6) of OR1. Operation flexibility is achieved by choosing different alternative machines. Sequence flexibility is represented by the precedence relations of the network diagram producing various operation sequences. An alternative process plan is one path from the starting node to the ending node.

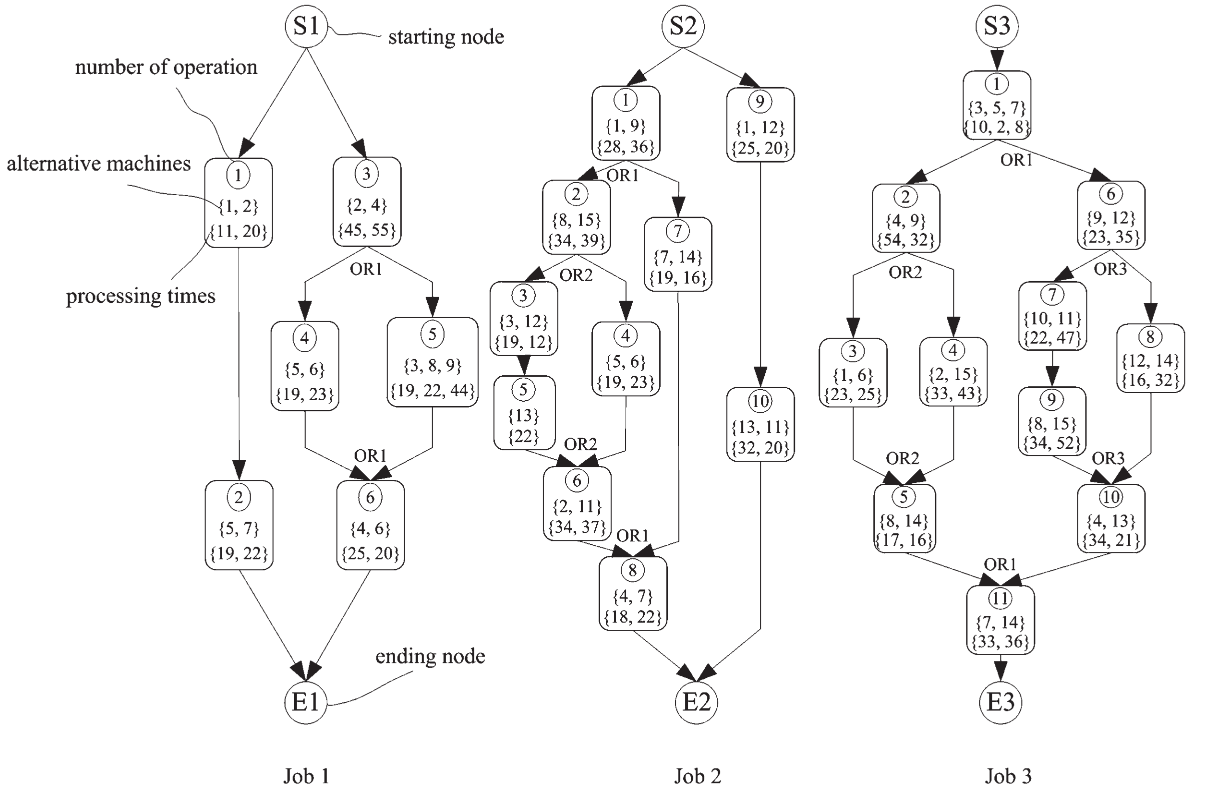

Figure 1. Network representation of the illustrative example.

2.2 *Job shop scheduling*

The conventional JSP problem is stated as follows (Zhang *et al.* 2008). Given n jobs that have to be processed on m machines, each job consists of a predetermined sequence of operations; each operation needs to be performed on a given machine for a given machining time without interruption. Job preemption is not allowed, and each machine can handle only one job at a time. Operations of the same job cannot be processed simultaneously, and each job must visit each machine exactly once. In this paper, transportation time of a job between two consecutive machines is ignored and setup time for the operations on the machines is included in the processing time. Assignment of operations to a time slot on a machine results in a schedule.

Gantt charts are utilised in this paper to represent a schedule. The X-axis of the Gantt chart indicates time. Each row on the Y-axis indicates a machine and the detailed arrangement for the operations of the jobs on this machine.

2.3 *IPPS*

The IPPS problem can be stated as follows (Guo *et al.* 2009b). Given a set of n parts that are to be processed on machines with operations including alternative manufacturing resources, select suitable manufacturing resources and sequence the operations to determine a schedule in which the precedence constraints among operations can be satisfied and the corresponding objectives can be achieved.

Conventional JSP assumes that the process plan for each part is determined before the scheduling functions, that is, process planning and scheduling are conducted sequentially or separately. IPPS integrates these two interrelated functions tightly and would probably provide better performance of manufacturing systems. On the other hand, IPPS enlarges the solution space of the determined JSP in that various flexibilities of process planning are taken into consideration. Since both process planning and JSP are NP-hard, the resulting IPPS falls into the category of an NP-hard problem too (Khoshnevis and Chen 1991). Concerning the intractability of IPPS, in this paper, a novel metaheuristic algorithm named the ICA is employed to tackle the problem.

3. The ICA for the IPPS problem

The ICA is a new population-based EA proposed by Atashpaz-Gargari and Lucas (2007). In the ICA, solutions to an optimisation problem are named as *country* – the equivalent of *chromosome* in the GA, *particle* in PSO or *ant* in ACO. Each *country* has a *power* to indicate its fitness and for the IPPS problem considered in this paper, *power* is inversely proportional to the makespan (also called *cost* in optimisation terminology). In the initial population of *countries*, some of the best ones (countries with least makespan or cost) are selected as *imperialists*, and the rest of the countries constitute the *colonies* of these imperialists. All the colonies will be distributed to these imperialists. An *imperialist* and its corresponding *colonies* construct an *empire*. The total power of an empire is composed of the power of the imperialist and that of all the colonies in the empire. The underlying motivation of the ICA is that all imperialists try to assimilate their corresponding colonies and compete for taking possession of colonies from each other.

There are four main steps in the ICA:

(1) *Assimilation*. Assimilation is the process of moving all the colonies to their relevant imperialists. If the resulting colony is better than its imperialist, it will become the new imperialist and vice versa.
(2) *Imperialistic competition*. Imperialistic competition indicates that all empires try to possess more colonies and increase its power. This procedure is realised by releasing the weakest colony of the weakest empire and making a competition among all empires to possess this colony.
(3) *Revolution*. Shokrollahpour *et al.* (2010) applied the ICA to bi-criteria scheduling of the assembly flowshop problem. A new step named revolution was added to the original ICA. Revolution means that some of the weakest colonies are selected and replaced with new randomly generated ones in each generation of the ICA. This revolution strategy is also adopted in this paper.
(4) *Elimination*. After imperialistic competition, a powerless empire will lose its colonies gradually, and elimination happens when an imperialist has no colonies.

The original ICA terminates when there exists only one empire, that is, all the colonies have the same position and power as the imperialist. Preliminary experiments have shown that the probability of all the countries having the same position and power was quite low. In this paper, the maximum number of iterations (*MaxIter*) is used as the termination criterion.

Although there exist some similarities between the ICA and GA in usage of some GA operators like reproduction, crossover and mutation (described in the following sections), they are significantly different. Firstly, although both the ICA and GA are population-based algorithms, they utilise different 'reproduction' strategy. In the GA, individuals (solutions) that are not selected to reproduce offspring are eliminated from the current population, while all the countries (solutions) in the ICA are kept throughout the optimisation process. Secondly, crossover in the GA is applied to two selected parent individuals based on their fitness, while crossover in the ICA happens between an imperialist and its colonies. In other words, crossover in the ICA aims to force colonies moving towards their imperialists. Other moving strategies besides the crossover concept borrowed from the GA could also be employed to accomplish the same assimilation purpose. Thirdly, mutation in the GA is applied to each individual with a certain predefined probability to enhance population diversification, while the mutation in the ICA is always applied to colonies after movement towards their imperialists. In addition, in contrast with the GA, which is inspired from natural evolution, the ICA is a socio-politically motivated global search algorithm. The underlying optimisation strategies of the GA and ICA are quite different. In the GA, better individuals are more likely to pass on their genes to the next generation. In the ICA, all the colonies are guided to move towards some imperialists that represent local optima of an optimisation problem at hand. The ICA converges when all countries reach the same position.

The implementation details of the ICA for IPPS are described in the following subsections.

3.1 *Initial country generation and evaluation*

3.1.1 *Solution representation*

In this paper, each *country* (including *imperialist* and *colony*) that represents a solution to the IPPS problem in the ICA population consists of two components: a scheduling plan and a process plan for each job. Figure 2 shows the proposed representation scheme of the illustrative example given in Figure 1.

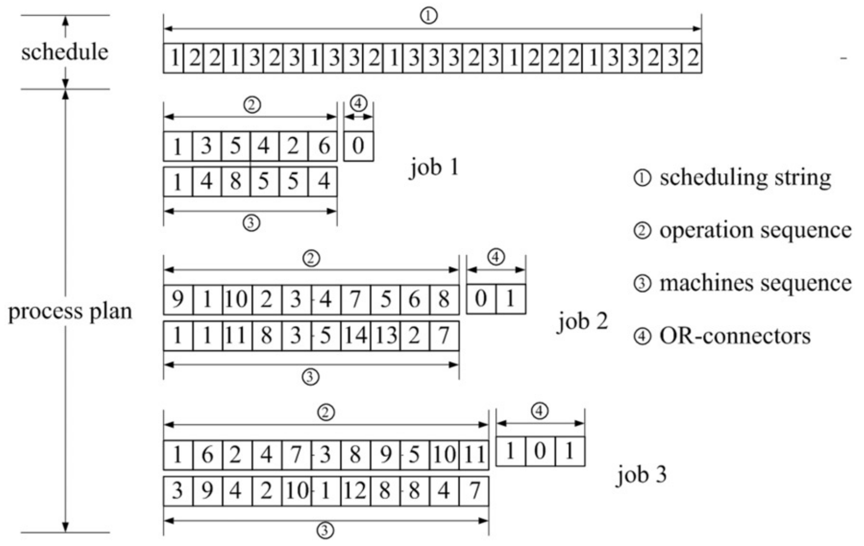

Figure 2. Illustration of proposed encoding scheme using the example given in Figure 1.

The first component is the scheduling plan string that adopts the operation-based representation scheme (Bierwirth 1995). For an IPPS problem made up of n jobs, let l_i denote the total number of operations of job i. The scheduling plan string is a permutation of job numbers, and each job number appears l_i times in the string. The nth appearance of a job number refers to the nth operation in the process plan of this job. This representation scheme has a superior feature that any permutation of the string can be decoded to a feasible solution. The length of the scheduling plan string (the total number of operations of all jobs) equals $\sum l_i$. In the illustrative example of Figure 1, there are three jobs to be scheduled and there exist 6, 10 and 11 operations for each job, respectively, so n is equal to 3, and $l_1 = 6$, $l_2 = 10$, $l_3 = 11$. $\sum l_i$ is equal to 27. Therefore, the length of the scheduling string is 27.

The second component contains the process plan for each job. Each process plan is made up of an operation sequence, machine sequences and OR-connectors. The operation sequence involves the machining sequence of operations for the completion of a job, and the machine sequence represents each operation's corresponding machine. The length of the operation sequence and machine sequence is equal to the total number of operations of a job. Note that some operations may not be conducted in scheduling. To determine which operations in the operation sequence are selected for scheduling, OR-connectors are responsible for forming the final process plan from all the operations. The value of 0 means selecting a left OR-link path and the value of 1, a right OR-link path. Figure 2 shows the process plan for each job given in Figure 1. Take the process plan of job 2 for example, the total number of operations is 10, so the length of the operation sequence and machine sequence equals 10. The third element of the operation sequence is operation 10, and its corresponding machine is machine 11. There exist two OR-connectors in the network of job 2, so the length of the OR-connectors of the process plan for job 2 equals 2. Note that the value of the second OR-connector is 1, so the right OR-link path (4) instead of OR-link path (3, 5) is selected.

3.1.2 *Makespan evaluation*

The schedule plan string can be decoded into semi-active, active, non-delay and hybrid schedules. This paper adopts the active schedule. In the decoding stage, the actual process plan for each job is first determined based on the values of the OR-connectors, followed by the determination of the schedule plan. Note that not all the operations shown in Figure 2 are selected in the actual process plans. Figure 3 shows the actual schedule plan and process plan for each job, which are used to calculate makespan.

The notations used to explain the procedure are as follows.

M	the total number of machines;
o_{ij}	the jth operation of the ith jobs;
k	the alternative machine corresponding to o_{ij};

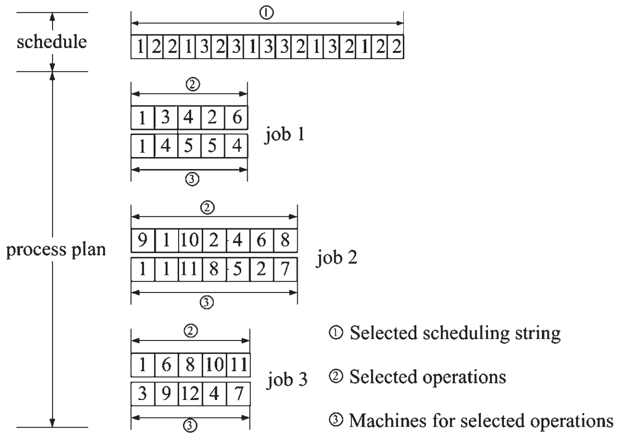

Figure 3. Determined schedule and process plans for makespan evaluation.

t_{ijk}	the processing time of operation o_{ij} on machine k, $t_{ijk} > 0$;
as_{ij}	the allowable starting time of operation o_{ij};
s_{ij}	the earliest starting time of operation o_{ij};
e_{ij}	the earliest ending time of operation o_{ij}, i.e. $e_{ij} = s_{ij} + t_{ijk}$.

The decoding procedure is given as follows (Li *et al.* 2010c).

Step 1: Determine the machine for each element in the scheduling plan string based on the machine sequence of each job's process plan.

Step 2: For each operation, $as_{ij} = c_{i(j-1)}$, $c_{i(j-1)}$ is the completion time of the operation that precedes o_{ij} of the same job.

Step 3: Check the idle time of the machine of o_{ij}, which results in a set of idle periods $[t_s, t_e]$, and examine these this periods sequentially. For the current period, if $\max(as_{ij}, t_s) + t_{ijk} < t_e$, the earliest starting time s_{ij} equals $\max(as_{ij}, t_s)$; otherwise, examine the next idle period. If there is no idle period satisfying this condition, then $s_{ij} = \max(as_{ij}, c(o_{ij} - 1))$, $c(o_{ij} - 1)$ is the completion time of the operation preceding o_{ij} on the same machine.

Step 4: The completion time for every operation $c_{ij} = s_{ij} + t_{ijk}$.

Step 5: Calculate the starting time and ending time for every operation of every job.

In the above procedure, the starting time and completion time of each operation can be obtained. Makespan c is the completion time of the last performed operation.

3.2 *Empire construction*

After the generation of the initial population of countries, some of the best countries are selected as the imperialists. Suppose the size of the initial population is N_{pop}, the number of imperialists is N_{imp} and the number of colonies is N_{col}. To divide the colonies among imperialists, the normalised cost (makespan) of each imperialist must be computed using the following formula:

$$C_n = \max(c_i) - c_n, \tag{1}$$

where c_n is the cost of the nth imperialist and C_n is its normalised cost. The colonies are distributed among imperialists based on their normalised power. The normalised power of each imperialist is defined by

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{\text{imp}}} C_i} \right| \tag{2}$$

Then, the number of colonies of an empire will be

$$NC_n = \text{round}(p_n \cdot N_{\text{col}}). \tag{3}$$

An imperialist with its corresponding colonies construct an empire.

3.3 *Assimilation*

3.3.1 *Moving colonies of an empire toward the imperialist*

According to the basic ICA, imperialists try to assimilate their colonies and make them similar to themselves, which is achieved by moving their colonies toward themselves. As to the IPPS problem considered in this paper, a crossover operator is defined to force colonies to move to their imperialists. In order to increase the ability of searching different areas around the imperialist, a mutation operator is applied to each colony after movement.

- Crossover

Considering the encoding scheme proposed in this paper, the crossover operator involves crossover of the schedule plan string and process plan string respectively, as is shown in Figures 4 and 5.

Based on the operation-based representation scheme, the crossover operator developed by Wang and Zheng (2001) is applied to the scheduling string in this paper. The crossover of the schedule plan string is described as follows.

Firstly, the set of job numbers is divided into two exclusive subsets S_1 and S_2 randomly, and each subset contains at least one job. Then, the schedule strings of the imperialist and the colony are scanned to generate the schedule string of the new colony. Elements of the schedule in the imperialist that belongs to S_1 are copied to the new colony directly, then the elements of the schedule in the colony that belongs to S_2 are copied to the remaining positions of the new colony. As Figure 4 illustrates, S_1 contains job 1 and S_2, job 2 and job 3. All the job numbers '1' in I are copied directly to the same positions to N, and all the job numbers '2' and '3' in C are copied to the remaining positions of N.

Crossover of the process plan string is made up of (1) crossover of operation sequence and machine sequence and (2) crossover of OR-connectors. Both of these crossover operators adopt the single point crossover.

Crossover of the operation sequence and machine sequence is given as follows.

Step 1: Initialise an empty operation sequence and machine sequence of the process plan for the new colony.

Step 2: Randomly determine a crossover point.

Step 3: Copy the elements of the operation sequence and machine sequence of the process plan of I that are before the crossover point to the same positions of N. Delete these elements from the operation sequence and machine sequence of C from left to right.

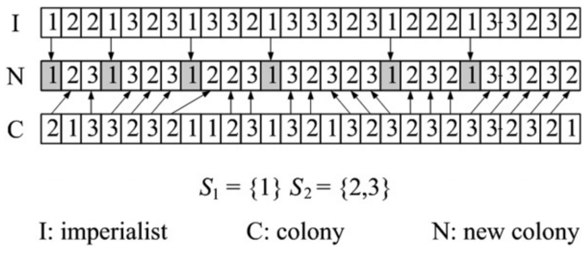
I 1 2 2 1 3 2 3 1 3 3 2 1 3 3 3 2 3 1 2 2 2 1 3 3 2 3 2
N 1 2 3 1 3 2 3 1 2 2 3 1 3 2 3 2 3 1 2 3 2 1 3 3 2 3 2
C 2 1 3 3 2 3 2 1 1 2 3 1 3 2 1 3 2 3 2 3 2 3 3 2 3 2 1
$S_1 = \{1\}$ $S_2 = \{2,3\}$
I: imperialist C: colony N: new colony

Figure 4. Crossover of a schedule plan.

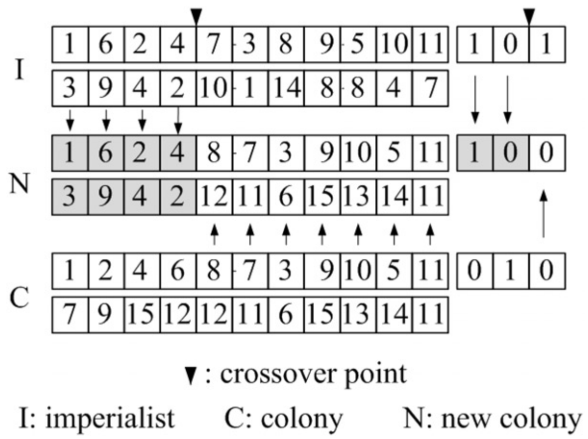

Figure 5. Crossover of a process plan.

Step 4: Copy the remaining elements of the operation sequence and machine sequence of C to the remaining positions of the operation sequence and machine sequence of N sequentially.

Crossover of the OR-connectors is described as follows.

Step 1: Initialise empty OR-connectors of the new colony.

Step 2: Randomly determine a crossover point.

Step 3: Copy the elements of OR-connectors of the process plan of I that are before the crossover point to the same positions of N.

Step 4: Copy the elements of OR-connectors of the process plan of C that are after the crossover point to the same positions of N.

- Mutation

The mutation operator consists of both schedule plan mutation and process plan mutation, as is shown in Figures 6 and 7.

The schedule plan mutation consists of two widely used neighbourhood strategies, namely, *swap* and *insert*. *Insert* is the act of randomly selecting a job number and inserting it to another random position of the current scheduling string. *Swap* is to exchange the positions of two randomly selected different job numbers in schedule plan string.

The process plan mutation involves mutation of the operation sequence, machine sequence and OR-connectors. Mutation of the operation sequence and machine sequence adopts the same *insert* and *swap* operators described above. Mutation of OR-connectors is to randomly select an OR value and change it to its opposite value.

3.3.2 *Exchanging positions of an imperialist and a colony*

After moved towards the imperialist, a colony may reach a position with lower cost than that of the 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.

3.4 *Imperialistic competition*

In the 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 waits 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 all its colonies, that is, the equation of total cost is:

$$T.C._n = c_n\ (\text{imperialist}_n) + \alpha \cdot \text{mean}\ (c_n(\text{colonies of empire}_n)), \tag{4}$$

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 values of α indicate the weight of the cost of the imperialist on the total cost of an empire.

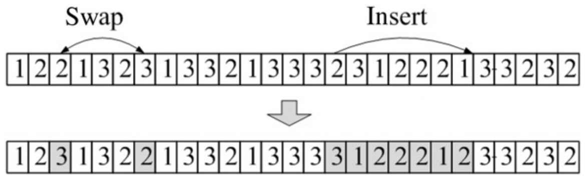

Figure 6. Mutation of the schedule plan.

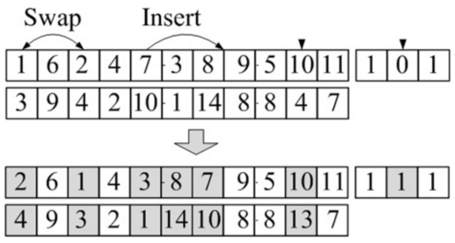

Figure 7. Mutation of the process plan.

The normalised total cost is computed by

$$N.T.C._n = \max(T.C._i) - T.C._n, \tag{5}$$

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

$$p_{p_n} = \left| \frac{N.T.C._n}{\sum_{i=1}^{N_{\text{imp}}} N.T.C._i} \right| \tag{6}$$

and

$$P = \left[p_{p_1}, p_{p_2}, \ldots, p_{p_{N_{\text{imp}}}}\right] \tag{7}$$

Then, a vector R with the same size as P is created, and its elements are uniformly distributed random numbers:

$$R = \left[r_1, r_2, \ldots, r_{N_{\text{imp}}}\right]. \tag{8}$$

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

$$D = P - R = \left[D_1, D_2, \ldots, D_{N_{\text{imp}}}\right]. \tag{9}$$

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

3.5 *Revolution*

Revolution indicates that the weakest colony of the weakest empire is replaced by a randomly generated solution (Shokrollahpour *et al.* 2010).

3.6 *Elimination of powerless imperialists*

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

3.7 *The proposed ICA procedure for IPPS*

Based on the above discussions, the computational procedure of the proposed ICA is described as follows.

Step 1: Initialise parameters of the ICA: N_{pop}, N_{imp}, α and $MaxIter$.

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 the 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.

The flowchart of the ICA is depicted in Figure 8.

4. Experimental studies

Performance of the ICA has been tested on several IPPS problems in the literature. Four groups of experiments are considered. The first group of experiments contains small size problems. In the second and third groups of experiments, medium size problems are tested. The fourth group of experiments is presented by solving a benchmark

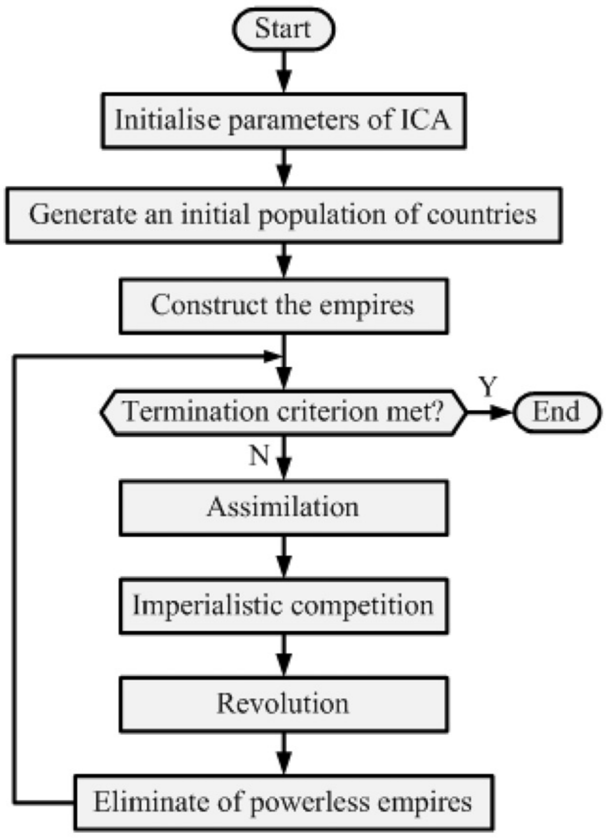
Start
Initialise parameters of ICA
Generate an initial population of countries
Construct the empires
Termination criterion met?
Y
End
N
Assimilation
Imperialistic competition
Revolution
Eliminate of powerless empires

Figure 8. Workflow of the ICA.

problem to confirm the effectiveness of the proposed ICA for a large-scale problem. In order to validate the performance, results of the problems from the presented ICA are compared with those of some other algorithms in the literature.

The algorithm implemented in this paper was coded in C++ and run on a personal computer with a 2.0 GHz Intel Core2 Duo CPU. Parameters of the ICA include total number of countries N_{pop}, total number of empires N_{imp}, the weight α and maximum number of iterations *MaxIter*. Preliminary experiments were conducted to determine values of these parameters. Specifically, we chose alternatives values for N_{pop}: [50, 100, 200], N_{imp}: [7, 10, 15] and α: [0.2, 0.4, 0.6], which resulted in a total number of 27 combinations of parameters. We tested the performance of each parameter combination on problem 1 in the first experiment. Computational results showed that the following parameter values could obtain satisfactory results: $N_{pop} = 100$, $N_{imp} = 7$, $\alpha = 0.6$ and $MaxIter = 1000$. For computational comparisons, each experiment was repeated five times for every test-bed problem, and the best solution obtained at each run was taken. The computational times listed in the various tables are the mean times of all five independent runs recorded.

4.1 *Experiment 1*

In this experiment, six problems are taken from the literature to compare the performance of the proposed ICA with that of other methods. The makespan values for these problems obtained from the proposed ICA and well-known methods in the literature are listed in Table 1. The values set in bold type are used to emphasise the better computational results our proposed method achieved.

Table 1 shows that the proposed ICA succeeded in finding all the best-known results in less computational time for all the small size problems, and a better solution is found for problem 4. The Gantt chart of the best schedule obtained by the ICA of problem 4 is given in Figure 9.

Table 1. Computational results of experiment 1 (*n* and *m* represent the total number of jobs and the total number of machines, respectively. CPU time of the results marked by * are not provided in the referenced paper).

			Best known algorithms		ICA
No.	Reference	$n \times m$	Algorithm	Makespan (CPU time/s)	Makespan (CPU time/s)
1	(Li *et al.* 2010a)	5×5	EA, SA, GA	33*	**33** (1.65)
2	(Moon *et al.* 2008)	5×5	Modified GA	14*	**14** (0.03)
3	(Li *et al.* 2010a)	6×5	EA	27 (3.20)	**27** (0.30)
4	(Li *et al.* 2010a)	6×5	EA	92 (3.23)	**90** (0.50)
5	(Li *et al.* 2010c)	8×5	HA	24*	**24** (0.17)
6	(Nasr and Elsayed 1990)	4×6	EA	17*	**17** (0.05)

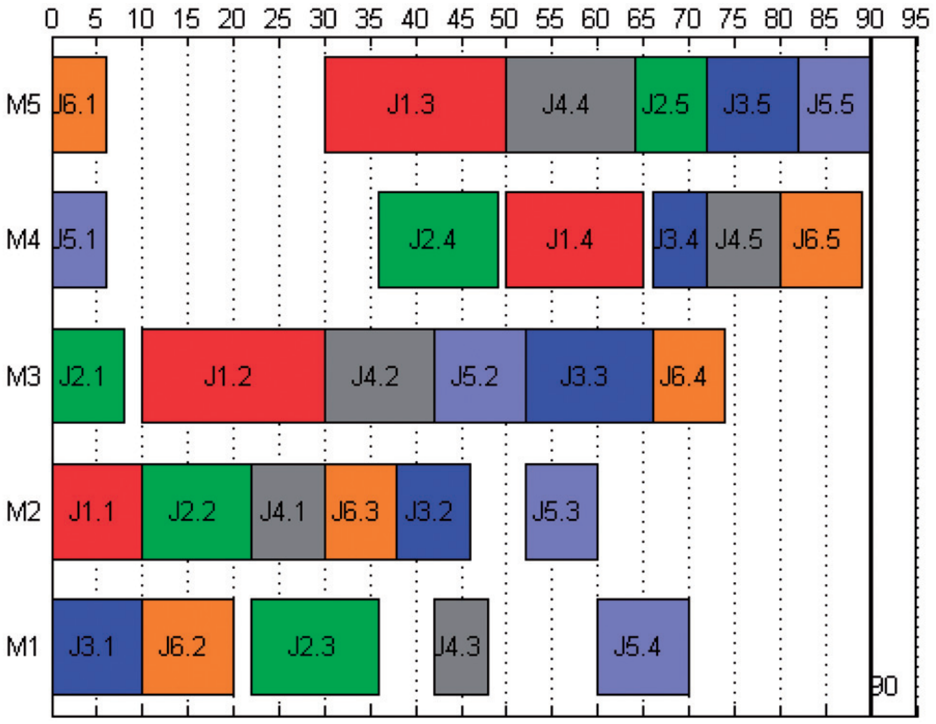

Figure 9. Gantt chart of the obtained solution to problem 4 in experiment 1.

4.2 *Experiment 2*

The testing data in experiment 2 was taken from Jain *et al.* (2006). In this experiment, 18 jobs and four machines are used to construct six experiments with the number of jobs varying from eight to 18. Table 2 shows the experimental results, and Figure 10 illustrates the Gantt chart of the optimal solution found for the first problem. It can be seen from Table 2 that the ICA outperforms the EA on all six problems. In addition, the ICA needs less computational time to obtain better results.

4.3 *Experiment 3*

The data of experiment 3 were taken from Kim *et al.* (2003) and Kim (2003). In this experiment, 24 problems are constructed with 18 jobs with various combinations of flexibility levels and 15 machines. The number of operations of these problems varies from 79 to 300.The number of jobs, the number of operations and the jobs in each problem are listed in Table 3.

Table 2. Computational results of experiment 2 (the results marked by * are adopted from Li *et al.* (2010a)).

		Makespan (CPU time/s)		
No.	Jobs	No integration*	EA*	ICA
1	8	615 (3.42)	520 (3.28)	**499** (2.98)
2	10	831 (3.72)	621 (3.38)	**586** (3.26)
3	12	934 (3.91)	724 (3.67)	**679** (2.73)
4	14	1004 (4.14)	809 (3.69)	**803** (2.50)
5	16	1189 (4.39)	921 (3.73)	**900** (2.49)
6	18	1249 (4.69)	994 (4.09)	**976** (3.87)

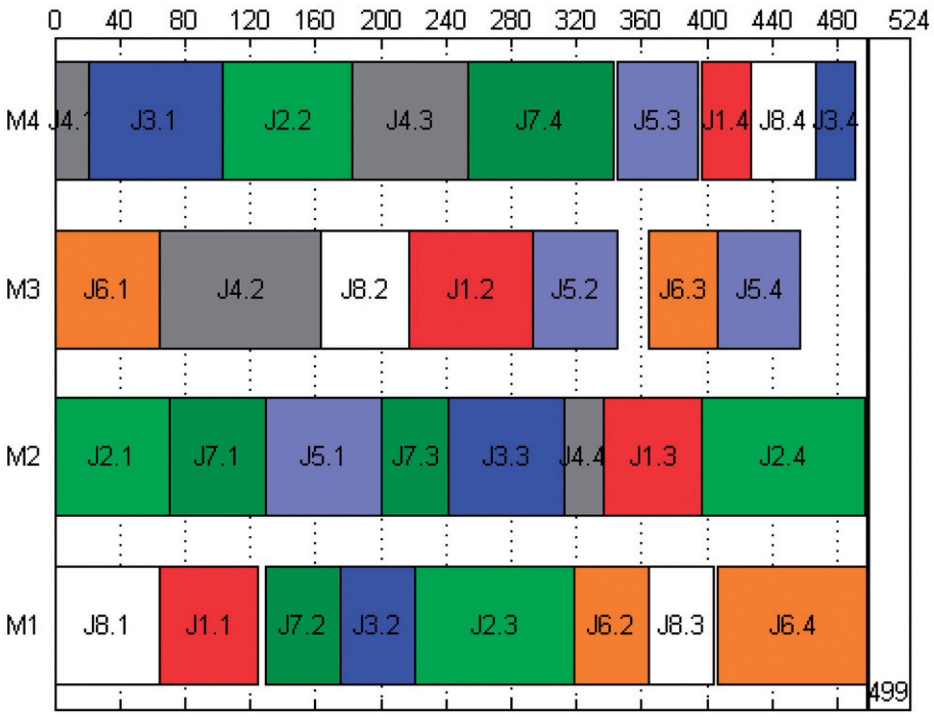

Figure 10. Gantt chart of the obtained solution to problem 1 in experiment 2.

From observation of Table 4, we can conclude that the ICA outperforms the hierarchical approach (HA) and cooperative coevolutionary genetic algorithm (CCGA) for all 24 test problems. Compared to the SEA, the same or better solutions have been found for 18 out of the 24 test problems in less computational time. Figure 11 gives the Gantt chart of the optimal solution found for problem 24.

The computational results of experiment 2 and 3 indicate that the ICA is able to obtain optimal or near-optimal solutions for medium size IPPS problems in a reasonable computational time. For most of the test problems, the ICA found better solutions.

4.4 *Experiment 4*

This problem is taken from Chan *et al.* (2005). In this experiment, the problem is constructed with 100 jobs and 10 machines. Table 5 shows the experimental results. Figure 12 illustrates the Gantt chart of the optimal solution obtained for the problem. Table 5 indicates that the ICA is very effective in solving a large-scale IPPS problem as compared to the single GA and the GA with dominated genes (GADG) (Chan *et al.* 2005).

Table 3. Test-bed problems of experiment 3.

No.	Jobs	Operations	Job number
1	6	79	1-2-3-10-11-12
2	6	100	4-5-6-13-14-15
3	6	121	7-8-9-16-17-18
4	6	95	1-4-7-10-13-16
5	6	96	2-5-8-11-14-17
6	6	109	3-6-9-12-15-18
7	6	99	1-4-8-12-15-17
8	6	96	2-6-7-10-14-18
9	6	105	3-5-9-11-13-16
10	9	132	1-2-3-5-6-10-11-12-15
11	9	168	4-7-8-9-13-14-16-17-18
12	9	146	1-4-5-7-8-10-13-14-16
13	9	154	2-3-6-9-11-12-15-17-18
14	9	151	1-2-4-7-8-12-15-17-18
15	9	149	3-5-6-9-10-11-13-14-16
16	12	179	1-2-3-4-5-6-10-11-12-13-14-15
17	12	221	4-5-6-7-8-9-13-14-15-16-17-18
18	12	191	1-2-4-5-7-8-10-11-13-14-16-17
19	12	205	2-3-5-6-8-9-11-12-14-15-17-18
20	12	195	1-2-4-6-7-8-10-12-14-15-17-18
21	12	201	2-3-5-6-7-9-10-11-13-14-16-18
22	15	256	2-3-4-5-6-8-9-10-11-12-13-14-16-17-18
23	15	256	1-4-5-6-7-8-9-11-12-13-14-15-16-17-18-19
24	18	300	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18

Table 4. Computational results of experiment 3 (the results marked by * are adopted from Kim *et al.* (2003) with no CPU time of HA and CCGA provided).

No.	HA*	CCGA*	SEA (CPU time/s)*	ICA (CPU time/s)
1	483	458	428 (60.5)	**427** (19.9)
2	383	363	343 (68.9)	**343** (52.2)
3	386	366	347 (81.7)	**345** (70.4)
4	328	312	306 (65.6)	**306** (40.3)
5	348	327	319 (63.5)	**319** (60.0)
6	506	476	438 (73.3)	**435** (65.6)
7	386	378	372 (69.0)	**372** (65.8)
8	376	363	343 (67.3)	**343** (57.4)
9	507	464	428 (73.2)	**427** (63.3)
10	504	476	443 (136.0)	**440** (112.5)
11	413	410	369 (165.8)	**367** (150.8)
12	361	360	328 (143.4)	**327** (134.5)
13	505	498	452 (161.2)	457 (182.6)
14	423	420	381 (150.8)	390 (172.3)
15	496	482	434 (156.0)	**432** (153.2)
16	521	512	454 (333.6)	466 (411.3)
17	474	466	431 (435.2)	443 (457.6)
18	417	396	379 (357.0)	384 (380.2)
19	550	535	490 (417.8)	**490** (402.5)
20	473	450	447 (384.0)	**440** (356.8)
21	525	501	477 (392.4)	**466** (354.6)
22	560	567	534 (1033.3)	**529** (980.4)
23	533	531	498 (1016.6)	**495** (993.2)
24	607	611	587 (1622.7)	**577** (1478.5)

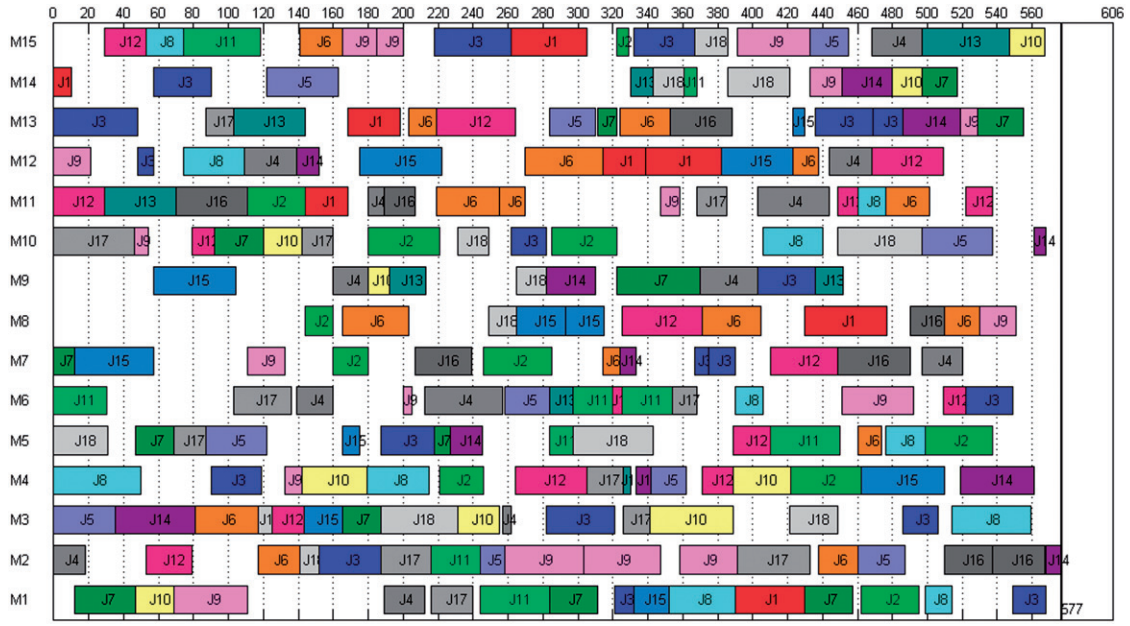
Figure 11. Gantt chart of the obtained solution to problem 24 in experiment 3.

Table 5. Computational results of experiment 4 (the results marked by * are adopted from Chan *et al.* (2005)).

Algorithm	GA*	GADG*	ICA (CPU time/s)
Makespan	267	229	169 (104.6)

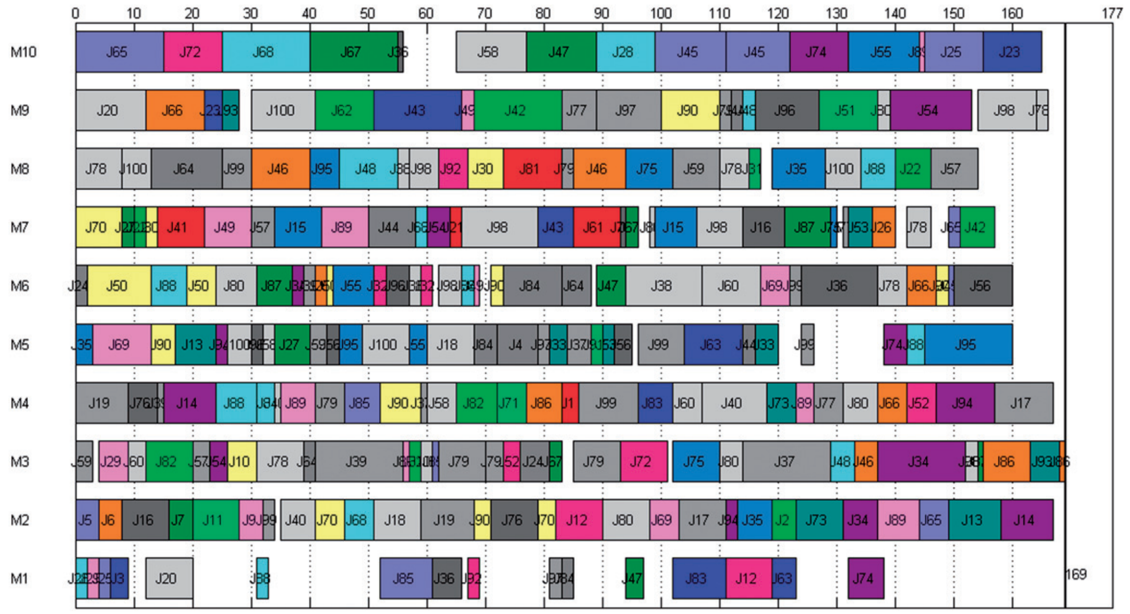
Figure 12. Gantt chart of the obtained solution to the problem in experiment 4.

4.5 *Discussion*

The experimental results for all test problems indicate that the proposed ICA is effective to solve IPPS problems of various sizes. Some distinguishing features of the ICA that contribute to its success are summarised as follows. Since imperialists in the ICA represent the better solutions among all the countries, assimilation forces the colonies to move to the optimal or near-optimal solutions in the search space of the IPPS problem. The imperialistic competition avoids the ICA being trapped into local optima. In the proposed ICA, revolution introduces a new randomly generated solution, which helps to enhance country diversity. Furthermore, the elimination of powerless empires in the proposed ICA enhances the convergence speed.

Future improvements of ICA could be conducted through the following aspects.

(1) Due to the adoption of normalised cost of each imperialist to distribute initial colonies, the *empire construction* phase of the original ICA would always result in an empire with no colonies. Specifically, the normalised cost of the worst imperialist, the one with the biggest makespan in the present IPPS problem, would be zero, in other words, this empire will most likely be eliminated in the very first iteration of ICA. The same problem exists in the imperialistic competition phase where the weakest empire would have the probability of zero to possess a freed colony.
(2) The ICA in this paper utilised the concept of crossover and mutation borrowed from the GA to accomplish the assimilation process. Some other assimilation strategies could be developed to better accommodate the assimilation purpose.

5. Conclusions

In this paper, the ICA is presented to minimise makespan for the IPPS problem, in which operation flexibility, sequencing flexibility and processing flexibility of each job are taken into account simultaneously. This algorithm starts from a population of countries and proceeds through assimilation, imperialistic competition, revolution and elimination of powerless empires. The ICA has been tested on four sets of IPPS instances varying from small sizes to large sizes, and compared with previous algorithms, such as the HA, EA, CCGA and SEA. Experimental results show that the ICA can obtain better optimal or near-optimal solutions than the other algorithms, which manifests that the ICA is very effective in solving the IPPS problem. Contributions of the present work are summarised as follows.

(1) A new optimisation strategy for the IPPS problem is introduced. Compared to existing approaches in the literature, the optimisation strategy presented in this paper is able to integrate process planning and scheduling more tightly and more promising solutions can be obtained. Computational results of four sets of experiments validate the effectiveness of the present strategy.
(2) A new optimisation algorithm named the ICA is utilised to address the IPPS problem. To the best of the authors' knowledge, this is the first application of the ICA to IPPS problems. Results show that the ICA can obtain very promising results in a reasonable computational time. In future work, the ICA could be applied to solve other difficult scheduling optimisation problems.

In future research, the IPPS problem under dynamic environments, such as new job arrival, machine breakdown and multi-objective IPPS problems, could be considered.

Acknowledgements

The authors thank the anonymous referees whose comments helped to much improve this paper. This research work is supported by the National High-Tech Research and Development Program of China 863 Program (grant no. 2007AA04Z107) and the State Key Program of National Natural Science of China (grant no. 51035001).

References

Abdechiri, M., Faez, K., and Bahrami, H., 2010. Neural network learning based on chaotic imperialist competitive algorithm. *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop*, 1–5.

Atashpaz-Gargari, E. and Lucas, C., 2007. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress*, 4661–4667.

Bahrami, H., Faez, K., and Abdechiri, M., 2010. Imperialist competitive algorithm using chaos theory for optimization. *Computer Modelling and Simulation (UKSim), 2010 12th International Conference*, 98–103.

Baykasoğlu, A. and Özbakır, L., 2009. A grammatical optimization approach for integrated process planning and scheduling. *Journal of Intelligent Manufacturing*, 20 (2), 211–221.

Bierwirth, C., 1995. A generalized permutation approach to job shop scheduling with genetic algorithms. *OR Spectrum*, 17 (2), 87–92.

Cai, N., Wang, L., and Feng, H.Y., 2009. GA-based adaptive setup planning toward process planning and scheduling integration. *International Journal of Production Research*, 47 (10), 2745–2766.

Chan, F.T.S., Chung, S.H., and Chan, P.L.Y., 2005. An adaptive genetic algorithm with dominated genes for distributed scheduling problems. *Expert Systems with Applications*, 29 (2), 364–371.

Girish, B.S. and Jawahar, N., 2009. Scheduling job shop associated with multiple routings with genetic and ant colony heuristics. *International Journal of Production Research*, 47 (14), 3891–3917.

Guo, Y.W., *et al.*, 2009a. Applications of particle swarm optimisation in integrated process planning and scheduling. *Robotics and Computer-Integrated Manufacturing*, 25 (2), 280–288.

Guo, Y.W., *et al.*, 2009b. Optimisation of integrated process planning and scheduling using a particle swarm optimisation approach. *International Journal of Production Research*, 47 (14), 3775–3796.

Ho, Y.C. and Moodie, C.L., 1996. Solving cell formation problems in a manufacturing environment with flexible processing and routeing capabilities. *International Journal of Production Research*, 34 (10), 2901–2923.

Jain, A., Jain, P., and Singh, I., 2006. An integrated scheme for process planning and scheduling in FMS. *The International Journal of Advanced Manufacturing Technology*, 30 (11), 1111–1118.

Karimi, N., Zandieh, M., and Najafi, A.A., 2011. Group scheduling in flexible flow shops: a hybridised approach of imperialist competitive algorithm and electromagnetic-like mechanism. *International Journal of Production Research*, 49 (16), 4965–4977.

Khoshnevis, B. and Chen, Q.M., 1991. Integration of process planning and scheduling functions. *Journal of Intelligent Manufacturing*, 2 (3), 165–175.

Kim, Y.K., 2003. A set of data for the integration of process planning and job shop scheduling [online]. Available from: http://syslab.chonnam.ac.kr/links/data-pp&s.doc.

Kim, Y.K., Park, K., and Ko, J., 2003. A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers & Operations Research*, 30 (8), 1151–1171.

Lee, H. and Kim, S.-S., 2001. Integration of process planning and scheduling using simulation based genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 18 (8), 586–590.

Leung, C.W., *et al.*, 2010. Integrated process planning and scheduling by an agent-based ant colony optimization. *Computers & Industrial Engineering*, 59 (1), 166–180.

Li, W.D. and McMahon, C.A., 2007. A simulated annealing-based optimization approach for integrated process planning and scheduling. *International Journal of Computer Integrated Manufacturing*, 20 (1), 80–95.

Li, X., *et al.*, 2010a. Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. *Computers & Operations Research*, 37 (4), 656–667.

Li, X., *et al.*, 2010b. A review on integrated process planning and scheduling. *International Journal of Manufacturing Research*, 5 (2), 161–180.

Li, X., *et al.*, 2010c. An effective hybrid algorithm for integrated process planning and scheduling. *International Journal of Production Economics*, 126 (2), 289–298.

Li, X., *et al.*, 2010d. An agent-based approach for integrated process planning and scheduling. *Expert Systems with Applications*, 37 (2), 1256–1264.

Moon, C., *et al.*, 2008. Integrated process planning and scheduling in a supply chain. *Computers & Industrial Engineering*, 54 (4), 1048–1061.

Morad, N. and Zalzala, A.M.S., 1999. Genetic algorithms in integrated process planning and scheduling. *Journal of Intelligent Manufacturing*, 10 (2), 169–179.

Moriarty, D.E. and Miikkulainen, R., 1997. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5 (4), 373.

Nasr, N. and Elsayed, E.A., 1990. Job shop scheduling with alternative machines. *International Journal of Production Research*, 28 (9), 1595–1609.

Nazari-Shirkouhi, S., *et al.*, 2010. Solving the integrated product mix-outsourcing problem using the Imperialist Competitive Algorithm. *Expert Systems with Applications*, 37 (12), 7615–7626.

Nejad, H.T.N., Sugimura, N., and Iwamura, K., 2011. Agent-based dynamic integrated process planning and scheduling in flexible manufacturing systems. *International Journal of Production Research*, 49 (5), 1373–1389.

Phanden, R.K., Jain, A., and Verma, R., 2011. Integration of process planning and scheduling: a state-of-the-art review. *International Journal of Computer Integrated Manufacturing*, 24 (6), 517–534.

Shao, X., *et al.*, 2009. Integration of process planning and scheduling – a modified genetic algorithm-based approach. *Computers & Operations Research*, 36 (6), 2082–2096.

Shokrollahpour, E., Zandieh, M., and Dorri, B., 2010. A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. *International Journal of Production Research*, 49 (11), 3087–3103.

Tan, W. and Khoshnevis, B., 2000. Integration of process planning and scheduling—a review. *Journal of Intelligent Manufacturing*, 11 (1), 51–63.

Wang, L. and Zheng, D.-Z., 2001. An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research*, 28 (6), 585–596.

Wang, J., *et al.*, 2009. Reducing tardy jobs by integrating process planning and scheduling functions. *International Journal of Production Research*, 47 (21), 6069–6084.

Weiming, S., Lihui, W., and Qi, H., 2006. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36 (4), 563–577.

Wong, T.N., *et al.*, 2006a. An agent-based negotiation approach to integrate process planning and scheduling. *International Journal of Production Research*, 44 (7), 1331–1351.

Wong, T.N., *et al.*, 2006b. Integrated process planning and scheduling/rescheduling—an agent-based approach. *International Journal of Production Research*, 44 (18/19), 3627–3655.

Zattar, I.C., *et al.*, 2010. A multi-agent system for the integration of process planning and scheduling using operation-based time-extended negotiation protocols. *International Journal of Computer Integrated Manufacturing*, 23 (5), 441–452.

Zhang, Y.F., Saravanan, A.N., and Fuh, J.Y.H., 2003. Integration of process planning and scheduling by exploring the flexibility of process planning. *International Journal of Production Research*, 41 (3), 611–628.

Zhang, C.Y., *et al.*, 2008. A very fast TS/SA algorithm for the job shop scheduling problem. *Computers & Operations Research*, 35 (1), 282–294.