Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 9628935, 13 pages
https://doi.org/10.1155/2017/9628935

Research Article

Combining Extended Imperialist Competitive Algorithm with a Genetic Algorithm to Solve the Distributed Integration of Process Planning and Scheduling Problem

Shuai Zhang, Yangbing Xu, Zhinan Yu, Wenyu Zhang, and Dejian Yu

School of Information, Zhejiang University of Finance and Economics, No. 18 Xueyuan Street, Xiasha, Hangzhou 310018, China

Correspondence should be addressed to Shuai Zhang; zs760914@sina.com

Received 30 April 2017; Accepted 1 November 2017; Published 20 November 2017

Academic Editor: Marco Mussetta

Distributed integration of process planning and scheduling (DIPPS) extends traditional integrated process planning and scheduling (IPPS) by considering the distributed features of manufacturing. In this study, we first establish a mathematical model which contains all constraints for the DIPPS problem. Then, the imperialist competitive algorithm (ICA) is extended to effectively solve the DIPPS problem by improving country structure, assimilation strategy, and adding resistance procedure. Next, the genetic algorithm (GA) is adapted to maintain the robustness of the plan and schedule after machine breakdown. Finally, we perform a two-stage experiment to prove the effectiveness and efficiency of extended ICA and GA in solving DIPPS problem with machine breakdown.

1. Introduction

Given the increasing fluctuations of the manufacturing environment and the constant need to arrange the process integrally and dynamically, schemes that separately formulate the plan and schedule no longer accord with the increasingly complex requirements. Under these environments, a model that settles the plan and schedule simultaneously, that is, integrated process planning and scheduling (IPPS), becomes necessary. In summary, the objective of IPPS is to determine an optimal schedule with machine selection for operation and operation sequence for the jobs [1]. Beyond the former strategy, IPPS considers the whole process while enhancing it in terms of dealing with the dynamic environment and making the theory consistent with reality.

Although IPPS makes a greater degree of progress than the former approaches in arranging and programming the manufacturing contents, the absence of distributed feature raises a dilemma for this model. Against the backdrop of distributed production, several requirements such as raw material availability and transportation concerns have urged manufacturing companies to adopt distributed strategies. Therefore, we aim to extend the IPPS and develop a more advanced model to handle distributed planning and scheduling, that is, distributed integration of process planning and scheduling (DIPPS).

The advantage of DIPPS lies in its feature of coordinating planning and scheduling in a distributed environment. Thus, the DIPPS is more suitable for the current manufacturing environment than IPPS. The DIPPS is generalized as follows [2]: given n jobs consisting of multiple alternative producing processes with different operations in u optional manufacturing units (MUs) with distinct assembly techniques and equipment, we must determine the plans and schedules including units, process plans, and machines for each job by considering the objectives and constraints.

On the other hand, the robustness of plans and schedules is of great concern through the manufacturing process. Once machine breakdown occurs during the manufacturing, the previous plan and schedule are bound to fall short of their anticipated objectives as a result of the changed context. In this case, an updated plan and schedule that aim to sustain the constraints and objectives should be structured for the remaining jobs and operations.

To deal with the DIPPS problem proposed in this study, we combine an extended imperialist competitive algorithm

(EICA) and a genetic algorithm (GA) together. Furthermore, the EICA is utilized to generate plans and schedules that are implemented before the emergency takes place, while GA is reserved to handle arrangements once a machine breakdown occurs. The traditional imperialist competitive algorithm (ICA) that simulates competition among empires has a strong global exploration capability in solving the NP-hard problems. In this study, we extend the ICA by improving its country structure, assimilation strategy, and adding a resistance procedure. The EICA has been proved as a more effective and efficient algorithm by comparing it with GA and traditional ICA in solving DIPPS problem, whereas for GA, as a mature and populated evolutionary algorithm, its capability in manufacturing practice has been studied and proved in our previous works [3, 4] and many other studies. Here, because of the advantage of its structural similarity with EICA and mature application in dealing with planning and scheduling, the GA can be easily formulated by implementing the structure from EICA; it therefore responds excellently to the breakdown emergency and alters the plan and schedule into satisfactory states.

The remainder of this paper is organized as follows. In Section 2, the works relating to ICA and robustness are briefly introduced. In Section 3, a mathematical model for the DIPPS problem is established. In Section 4, the imperialist competitive algorithm is extended to effectively solve the DIPPS problem. In Section 5, the GA is adapted to deal with machine breakdown. In Section 6, a two-stage experiment is presented to prove the effectiveness and efficiency of EICA and GA to solve DIPPS problem in the case of a machine breakdown. Section 7 presents our conclusions.

2. Related Work

2.1. Evolutionary Algorithms. GA is an algorithm to search for the optimal solution by simulating the natural evolution process. And there are some well-known evolutionary algorithms inspired by GA, such as biogeography based optimization [5, 6] and genetic swarm optimization [7]. In our previous work, we used GA to optimize the DIPPS in fuzzy environment [8]. Besides the wide application of GA, there are some other evolutionary algorithms that have been used for solving optimization problem. For example, Rahmat-Samii et al. [9] used PSO for antenna design optimization. Dorronsoro et al. [10] used evolutionary algorithm to model and solve minimization problems. Grimaccia et al. [11] used social network optimization to design generators for vehicle energy harvesting.

In addition, ICA is a metaheuristic algorithm inspired by sociopolitical ideology and first proposed by Atashpaz-Gargari and Lucas [12]. Generally, there are always competitions when numerous countries exist. By means of war and conquest, some powerful countries, called imperialists, conquer and colonize others, forming empires. As time goes by, the imperialists assimilate their colonies and conquer colonies belonging to other ones. In contrast, weaker empires gradually lose their colonies to more powerful ones and eventually face extinction. At the end of competition, there is an ideal state in which the most powerful empire conquers all lands. By simulating the competition above, the ICA innovatively structures its procedure to solve a variety of outstanding problems. Through the last several years, several significant works [13–15] have sought to strengthen the global exploration power in order to broaden its application.

Since being proposed, the ICA has gained popularity and achieved significant performance in solving manufacturing planning and scheduling problems. To settle the optimization of process planning with various flexibilities, Lian et al. [16] utilized the ICA to find promising solutions with reasonable computational cost under the objective of minimizing total weighted sum of manufacturing cost. Shokrollahpour et al. [17] and Seidgar et al. [18] both exploited the ICA in assembly flow shop problem while, respectively, using the Taguchi method and neural network as their own tools in regulating the parameters. Additionally, in the no-wait two-stage hybrid flow shop, Moradinasab et al. [19] introduced a new procedure called global war in ICA to avoid the local optima. This step helps to transfuse some new empires in a certain extent and achieves desirable performance in the experiment. In addition, in the work of Zhou et al. [20], ICA was adopted to deal with the assembly sequence planning. Compared with GA and PSO, the ICA performs better in the experiment and the quality of result is less related to the initial populations. Moreover, Madani-Isfahani et al. [21] presented an ICA to solve a biobjective unrelated parallel machine scheduling problem where setup times are sequence dependent.

Despite the achievements in the specific domain of manufacturing arrangement, infrequent work has been done to settle IPPS problems, let alone for DIPPS problems. To the best of our knowledge, only Lian et al. [22] have applied the ICA to solve IPPS while omitting the disposition of robustness. The scarce adoption of ICA in this area is not contrary to our expectations. Because IPPS and even DIPPS problem have far more variables and constraints to deal with, they inevitably contain a high magnitude of information to manage. The complexity of simultaneous planning and scheduling also predisposes the process of measure searching to be handled delicately.

2.2. Robustness. Under the constantly changing conditions of manufacturing, static and unchangeable plans and schedules are impractical. When the initial plans and schedules are put into effect, machine breakdown may take place and disturb the manufacturing procedure in an uncontrollable way that invalidates the former arrangement. To keep plans and schedules robust and flexible, some extra work is essential.

Among the methods applied for replanning and rescheduling, right-shifting is the most convenient way. This corresponds to waiting for the breakdown to be fixed and then carrying on with the work [23]. For instance, Liu et al. [24] used right-shift rescheduling to retain the same sequence of all remaining jobs as that of the predictive schedule. Although this method saves quite a lot of follow-up work, it loses the optimality of planning and scheduling at the same time. Therefore, other means have been figured out. Saygin and Kilic [25] adopted a step-by-step manner by dividing the whole scheduling period into shorter

periods and proceeding by overlapping the schedule of each period on the previous one to handle the effect of changes like breakdowns. Jensen [23] proposed a new way of creating robust and flexible solutions for job-shop scheduling problems by busing a robustness measure based on a neighborhood for schedules. Additionally, Hasan et al. [26] used shifted gap-reduction instead of right-shifting in order to minimize the effect of interruptions in job-shop scheduling problem.

When selecting methods for dealing with machine breakdown in scheduling and planning, the focal points should be targeted at convenience and optimization where the former point pays attention to the adjustment time of replanning and rescheduling while the latter one is concerned with the optimality of replanning and rescheduling. In this study, GA is associated with solving the machine breakdown. Because of the similarity of EICA and GA in representation, GA can be structured and put into work promptly once breakdown takes place, and GA is more effective than ICA in solving the problem with small solution space. In addition, with abundant verification preformed in previous works for IPPS problems, it has a positive reputation for strong and reliable performance.

3. The Mathematical Model for the DIPPS Problem

As defined in the Introduction, DIPPS aims to determine an appropriate manufacturing unit (MU) while selecting the process plans and schedules for jobs. The so-called MUs are some geographically dispersed units contained in an integral factory system that have the capability to operate independently. In the DIPPS problem on which we focus in this study, each MU can process all types of jobs that need to be treated. However, because of the multifarious assembly techniques and equipment necessitated by differences in construction year and purposes among MUs, the respective optional process plans and machines are totally different. Based on this situation, the arrangement should be settled cautiously. Furthermore, owing to geographical dispersion, the transportation time for the finished work from MUs to the central factory is also a significant concern in the course of scheming.

To examine the optimality and rationality of the plan and schedule, we formulate the relevant objectives and constraints; before formulating them, the corresponding indexes are expounded as follows:

U: the quantity of MU

N: the quantity of job

P^{nu}: the quantity of plans of the nth job in the uth MU

J_p^{nu}: the quantity of operations of the pth plan of nth job in the uth MU

M^u: the quantity of machines in the uth MU

o_{pjm}^{nu}: the jth operation of the pth plan of the nth job processed by the mth machine in the uth MU

it_{pjm}^{nu}: the initial time of o_{pjm}^{nu}

$\mathrm{it}_{p'j'm}^{nu}$: the initial time of any other operation processed on the same machine as o_{pjm}^{nu}

ot_{pjm}^{nu}: the operating time of o_{pjm}^{nu}, which contains the setup time

oct_{pjm}^{nu}: the completion time of o_{pjm}^{nu}

$j\mathrm{ct}^{nu}$: the completion time of the nth job in the uth MU

tt^{nu}: the transportation time of the nth job from the uth MU to central factory

X^{nu}: the binary variable, with 1 representing that the nth job assigned to the uth MU and 0 otherwise.

In order to evaluate the excellence of methods for the DIPPS, an objective is formulated.

Objective. Minimize the total makespan g_{tms}; that is, minimize the period of time from the very beginning of processing to the end of transportation:

$$g_{\mathrm{tms}} = \max_{u\in[1,U]} \left\{ \max_{n\in[1,N]} \left[X^{nu} \cdot \left(j\mathrm{ct}^{nu} + \mathrm{tt}^{nu} \right) \right] \right\}. \tag{1}$$

Meanwhile, the following constraints are required for the sake of rationality.

Constraint 1. Only one job can be assigned to a single MU:

$$\sum_{u=1}^{U} X^{nu} = 1 \quad n \in [1, N]. \tag{2}$$

Constraint 2. Once an operation is under processing, no other operations can cut in:

$$\begin{aligned} &\mathrm{oct}_{pjm}^{nu} = \mathrm{it}_{pjm}^{nu} + \mathrm{ot}_{pjm}^{nu} \\ &n \in [1, N],\ u \in [1, U],\ p \in \left[1, P^{nu}\right],\ j \in \left[1, J_p^{nu}\right],\ m \in \left[1, M^u\right]. \end{aligned} \tag{3}$$

Constraints 3. Parallel processing for more than one job is not allowed on any machine:

$$\mathrm{it}_{p'j'm}^{nu} \notin \left(\mathrm{it}_{pjm}^{nu}, \mathrm{oct}_{pjm}^{nu} \right) \quad n \in [1, N],\ u \in [1, U],\ p \& p' \in \left[1, P^{nu}\right],\ j \& j' \in \left[1, J_p^{nu}\right],\ m \in \left[1, M^u\right]. \tag{4}$$

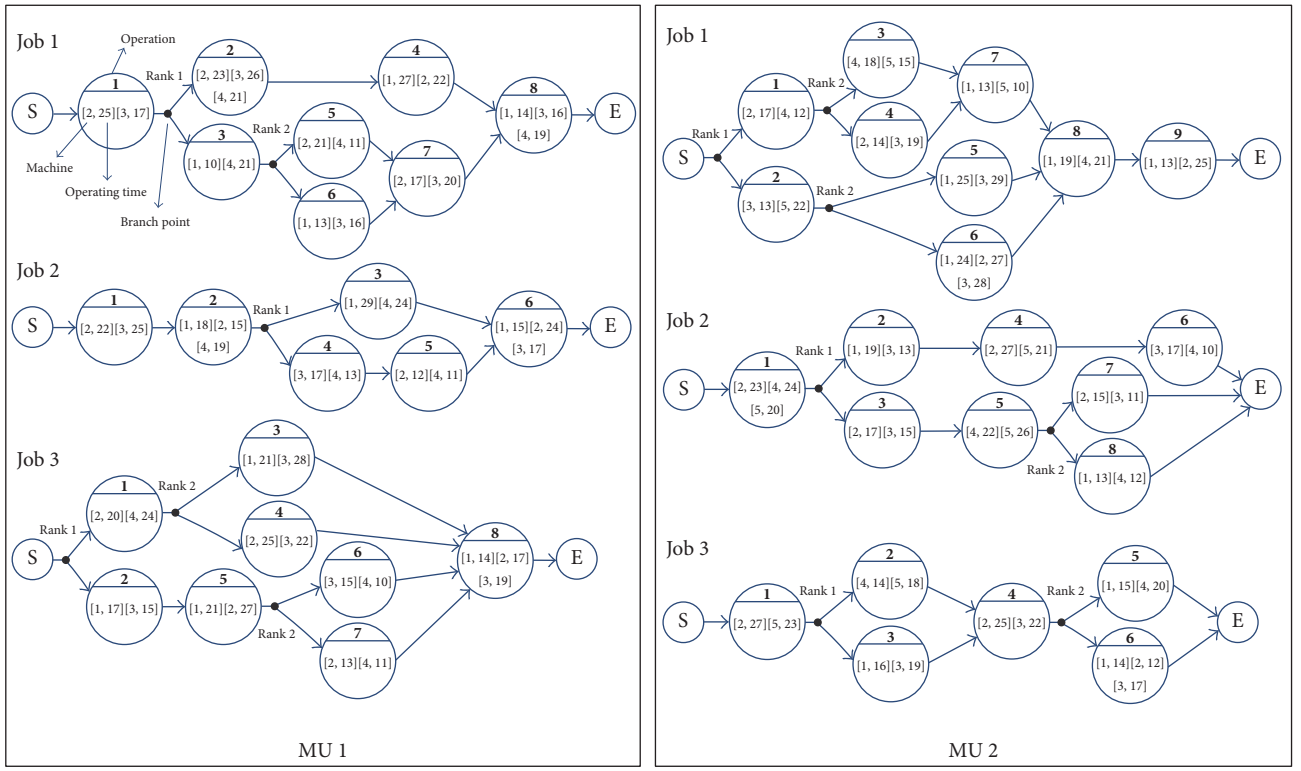

Figure 1: Representation of plans and schedules in MU 1 and MU 2.

Constraint 4. The operating sequence of operations of a specific job cannot be altered once determined:

$$\begin{aligned} &\mathrm{it}_{p(j+1)}^{nu} - \mathrm{oct}_{pj}^{nu} \geq 0 \\ &n \in [1, N],\ u \in [1, U],\ p \in \left[1, P^{nu}\right],\ j \in \left[1, J_p^{nu}\right]. \end{aligned} \tag{5}$$

In this study, with a goal of simplifying the resolving process, a simplified example containing two independent MUs and three jobs is structured. For the purpose of representing alternative plans and machines for different jobs and operations in diverse MUs, a directed acyclic graph (DAG) is adopted here. The traditional DAG contains vertices and directed edges and is applied throughout mathematics, computer science, and engineering with the capability to clearly represent processes. In this study, we extend the DAG with more features to illustrate the information visually and prepare for the subsequent calculation.

Specifically, Figure 1 exhibits alternative plans and schedules for different jobs through different MUs. Take the first DAG in Figure 1 which shows Job 1 alternative plans and schedules in MU 1; for example, each vertex and directed edge stands for the operation and the processing trend, respectively. The black dots named "branch points" are attached in the DAG to represent the branch information that will be used in the construction of EICA. To clarify the start and end of the plan, two virtual operations without any operating time, labeled "S" and "E," are set. With regard to valid operations, that is, the vertices except the starting and the ending ones, each of them represents the optional machine number and respective operating times by data sets. Tracking along with the directed edges from the "S" operation to the "E" operation without any backtracking, one specific plan with a certain sequence and corresponding operations is determined.

4. EICA for the DIPPS Problem

The traditional ICA is introduced in Section 2.1. Here, we illustrate how to extend the traditional ICA to solve the DIPPS problem effectively.

4.1. Country Structure and Initialization. The first step of EICA is to generate the initial population, referred to as countries of ICA. To clearly represent the information between MU, plan, and operation, we devise a three-segment country structure (Figure 2).

Generally, the first segment represents the MUs where each job is assigned. The second segment represents the "branch information." The "branch information" is the specific directed edge that a job selects in each branch point. Referring back to each DAG in Figure 1, we first rank the branch points individually and assign the former branch points higher rankings whereas equivalent-level branch points get the same rank. From left to right, the rank declines. The upper the branch chosen by the plan, the smaller the natural number assigned to the position. For example, in this

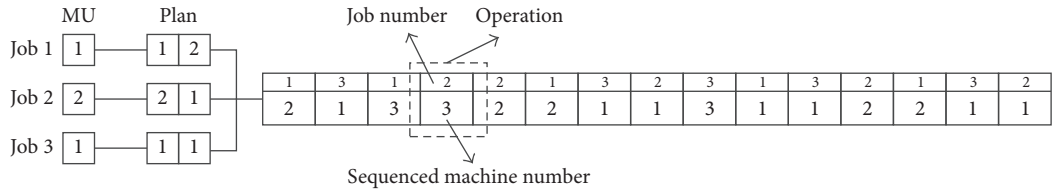

Figure 2: Three-segment country structure in EICA.

instance, there are up to two branches, so the position will be set as "1" if the upper branch is chosen; otherwise, it will be set as "2." By this means, when all the branch directions are decided, the plan is fully determined. When considering the rank number discrepancy of the same job in different MUs, the length of the second segment for a specific job is the maximum rank number among all MUs, and the invalid position will be omitted in the course of decoding. After the above steps, the second segment of the country structure where one position represents a branch of the same level is formulated. Thus, the first two segments form a complete plan for every job.

In the third segment, we blend the schedules of different jobs for convenience, and every column of this segment represents an operation with the corresponding job number and machine labeled inside. Meanwhile, the column of each job's operation in this segment manifests its order of precedence in processing and its proximity to the left indicates higher priority. The length of this segment is the product of multiplying job quantities by the operation quantities of the longest plan among all possible plans. For example, the example here has three jobs and the longest plan from Figure 1 consists of five operations; thus, the third segment here has 15 columns. Similar to the second segment, when the chosen plan has fewer operations than its own maximum one, the remaining position will be neglected in the decoding. To avoid illegal schedules in the course of algorithm application, candidate machine numbers of an operation are resequenced from the lowest one to "substituted number" sequenced from "1." For example, in Figure 1, Job 3 first operation in MU 1 can choose Machines 2 and 4. Therefore, the sequenced "substituted numbers" used in the position are 1 and 2.

For the purpose of decoding the structure to determine the specific schedules, three strategies explained by Bierwirth and Mattfeld [27], that is, active schedule, semiactive schedule, and nondelay schedule, are commonly used. In this study, we adopt the active schedule.

After the countries, that is, the populations, are initialized, the next step is to divide them into imperialists and colonies and then structure empires. Given F_{pop} countries, F_{imp} countries with the most power are selected as imperialists, and the remaining F_{col} countries are classified as colonies affiliated with imperialists. Here, we convert the power of vth country into cost c_v, that is, the total makespan g_{tms} of it (also known as the fitness value). The lower the cost for a country, the higher its potential to become an imperialist:

$$c_v = g_{\text{tms}}^v \quad v \in \left[1, F_{\text{pop}}\right]. \tag{6}$$

In the subsequent process of determining how many and which colonies each imperialist owns, the cost c_e of eth imperialist is transformed into the normalized cost C_e:

$$C_e = \max_{i \in [1, F_{\text{imp}}]} \{c_i\} - c_e \quad e \in \left[1, F_{\text{imp}}\right]. \tag{7}$$

The normalized power pwr_e of eth imperialist can be calculated as follows:

$$\text{pwr}_e = \frac{C_e}{\sum_{i=1}^{F_{\text{imp}}} C_i}. \tag{8}$$

Generally, the quantity of colonies each imperialist occupies is proportionate to its normalized power; that is, with greater power come more colonies. The initial number N.C._e of colonies belonging to the nth imperialist is

$$\text{N.C.}_e = \text{round}\left\{\text{pwr}_e \cdot F_{\text{col}}\right\}. \tag{9}$$

The colony number of every imperialist is thus worked out, and we randomly distribute the corresponding quantity of colonies to the imperialist. When the distribution is complete, the structure of our empires becomes explicit.

4.2. Assimilation. Once an imperialist occupies its colonies and composes its empire, it attempts to promote the power of its affiliated colonies by assimilating them. In this step, the colonies approach the imperialist in a certain form and have the chance to become more powerful. Particularly, we implement a novel assimilation strategy here to transmit a part of the structure of the imperialist to the colonies so as to not only achieve assimilation but also adjust the original plan and schedule to avoid local optima. The detailed process is listed as follows (Figure 3).

Step 1. Designate a colony for assimilation, and randomly select a job (e.g., Job 2) and ascertain its plan and schedule information in the imperialist.

Step 2. Pass down the MU and plan of the selected job in the imperialist, that is, the corresponding information of this job in the first two parts, to replace the former MU and plan of the same job in the colony (denoted by the bold line with an arrow).

Step 3. Randomly select a job in the colony. If the job is different from the formerly selected one (e.g., Job 3), first transmit its operations to the positions of the former selected job's operations in sequence (denoted by thin lines with arrows); otherwise, do nothing.

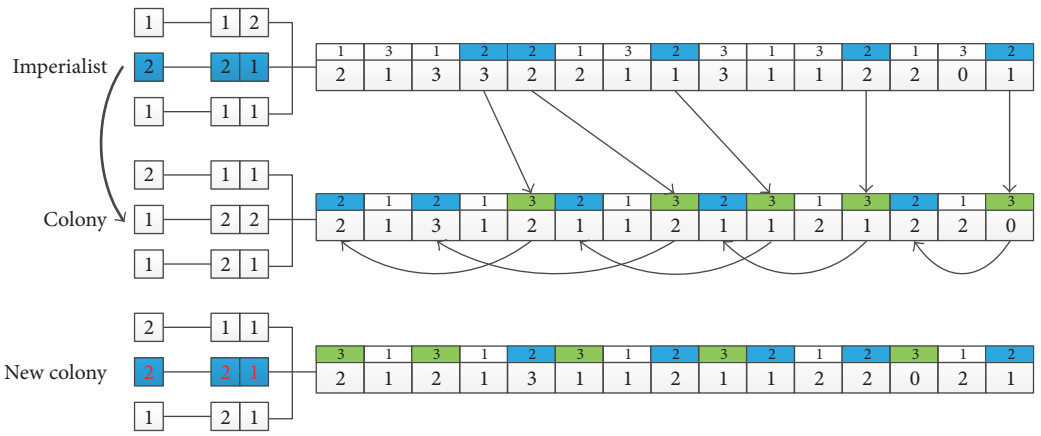

Figure 3: Assimilating the colony into imperialist.

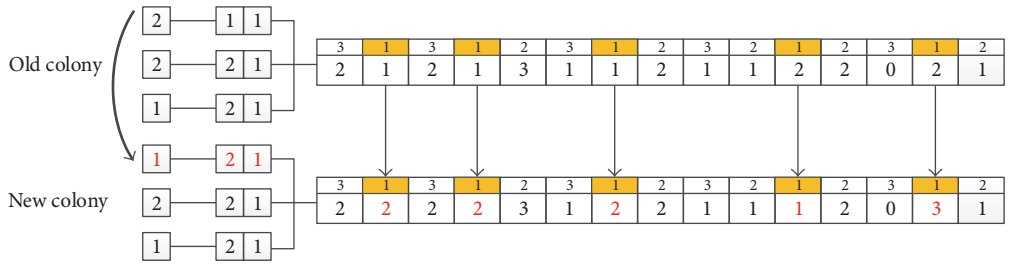

Figure 4: Resisting operation of EICA.

Step 4. Transmit the operations of the selected job in Step 1 of imperialist to the positions of the latter selected job in the third part of colony while keeping them in the same order as in the imperialist (denoted by dotted lines with arrows).

4.3. Resistance. In the real world, when weak countries are conquered as colonies of more powerful countries, some of them will make an attempt to resist the domination of their occupier by counteracting the assimilation of the imperialist. To broaden the search coverage and prevent a fall into local optima, resistance procedure is added to EICA. To allow for the turbulence of the resisting action, we limit the application of this step within one job. And, in this study, we set the resistance rate as 0.05 for each country. The concrete process of resistance in this study (Figure 4) is to randomly select a job and alter the corresponding three parts randomly.

4.4. Position Exchanges. After the action of the previous two parts, the costs of the colonies are altered. There also may exist a colony whose cost is lower than any other country in its empire (including the imperialist and colonies). When this situation occurs, the position between the lowest-cost colony and the imperialist exchanges, which means the lowest-cost colony becomes the new imperialist and the former one is relegated to its vanquished colony. In the next assimilation process, other colonies also begin to move towards the new imperialist.

4.5. Competition. The competition part indicates the struggle among imperialists for the colonies of weaker empires. In traditional ICA, the strongest empire will take over the weakest colony in the weakest empire. However, the selected colony will not always be the weakest in whole country. In this study, the strategy of elite replacement is introduced into the competition, which means the strongest imperialist will take over the weakest country as its colony in every generation. If there is only one empire, this process will be skipped.

4.6. Elimination and Stop Criterion. When an empire is weak enough to cross a certain threshold, it moves towards collapse. Its colonies will all be divided and occupied by other empires, and the whole empire is thus eliminated. In this study, the elimination condition is triggered when an empire loses every colony it owns.

In an ideal state, all other empires will be eliminated and only one will survive under the condition that the imperialist and its colonies all have the same cost after repeating the process of assimilation and elimination a certain number of times. However, this convergence is too rare to expect under the practical circumstances. Therefore, the stop criterion

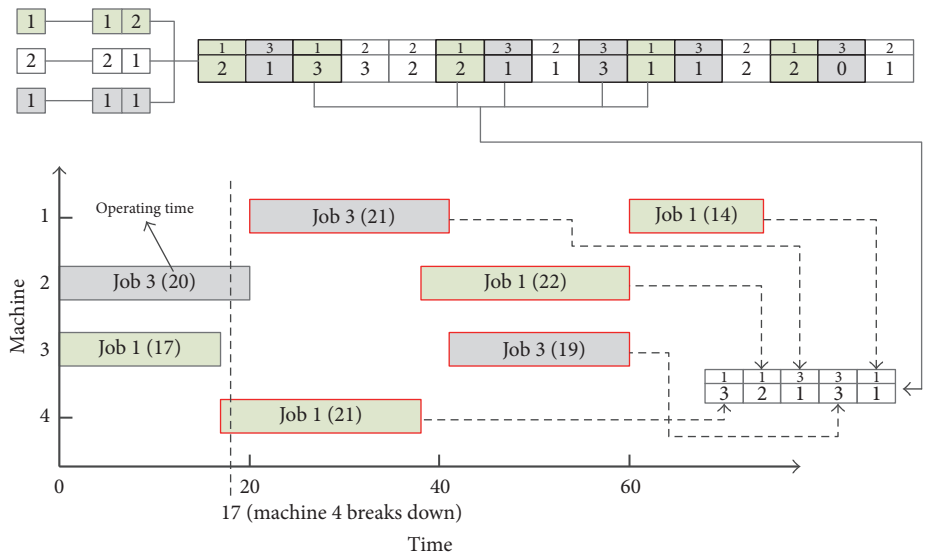

FIGURE 5: Gantt chart and the chromosomal structure for rescheduling.

usually adopts preset iterations or stabilization of the average fitness values among several generations.

5. GA for Machine Breakdown

Machine breakdown occurs occasionally in the course of a production process and can send the plan and schedule into disarray due to the time and cost required for repair. In this study, we adapt the GA to reschedule the undone operations when breakdown happens. Benefiting from the similar structure of EICA and GA and the powerful global exploration capability of the latter algorithm, the GA can be easily formulated by utilizing the structure from EICA; it therefore responds to the breakdown emergency excellently and gives consideration to the reschedule with both efficiency and effectiveness. Before elaborating upon the application of GA, the following assumptions and prerequisites are posited.

(1) When machine breakdown occurs, the remaining operating time of the operation that is processing on the broken machine at that time will consequently be discrepant. If the operation waits on the machine to be repaired and resumes immediately after the recovery, then it only needs to process the undone part. However, if the operation is transferred to another machine or it stays on the same machine but is not the first processed operation after the breakdown, then the process begins fully anew.

(2) The machine breakdown will not disturb the current processing operations on other machines.

(3) Considering the transportation time and cost, the undone operations cannot be rescheduled to any other MU.

(4) If the breakdown occurs in the MU where the remaining operations belong to only one job, then GA is of no use. The rescheduling procedure for this exceptional case is to find the shortest-time-costing machine for each remaining operation and then compare the changed plan with the right-shifting plan and choose the better one of the two.

5.1. Initialization. The remarkably similar structure between the country and the chromosome makes it possible to form the initial structure on the basis of the third segment of the country (Figure 5). The length of the chromosome, that is, the number of genes, is in accordance with the quantity of incomplete operations. To preserve the excellence and keep in constancy with the former plan and schedule, one of the initial chromosomes is directly abstracted from the best country in the last generation, while the rest are randomly generated.

5.2. Fitness Function and Reproduction. Because the rescheduled operations are part of the whole DIPPS, to calculate the fitness value of the chromosome and verify the excellence of the revised schedule, the chromosome should be brought back to the structure of the lowest-cost country and we should measure the change in cost. In other words, the fitness function adopted here is the cost of the country.

In this study, we employ a tournament selection scheme for the reproduction of the chromosome. In tournament selection, a number of individuals are selected randomly (dependent on the tournament size, typically between 2 and 7) from the population and the individual with the best fitness is chosen for reproduction [28].

5.3. Crossover and Mutation. In the crossover operation, we adapt the following strategy (Figure 6).

Step 1. Randomly select two parents from the generation and initialize two empty offspring.

Step 2. Randomly select several jobs (for an even number of jobs, it is $N/2$; and for an odd number of jobs, it is $((N-1)/2)$) from jobs where the unfinished operations belong; then duplicate those jobs' operations from P1 and P2 to the same position of O1 and O2, respectively.

Table 1: The transportation time from MU to central factory.

MU	Job			
	Job 1	Job 2	Job 3	Job 4
MU 1	34	27	36	40
MU 2	43	29	25	43

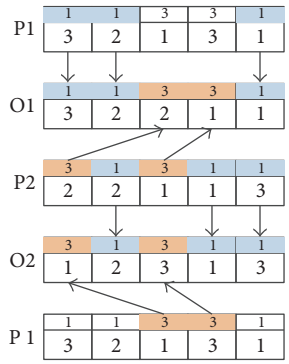

Figure 6: Crossover operation.

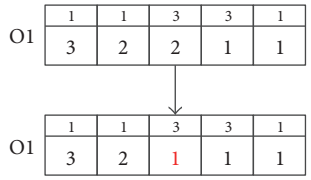

Figure 7: Mutation operation.

Step 3. The remaining genes of P1 and P2 are copied to the same positions in O2 and O1.

To mutate the chromosome, the rapid one-point mutation is applied here by randomly selecting an operation and then changing its sequenced machine number (Figure 7).

6. Experiment

In this section, a case study that is simulated based on the real DIPPS environment is implemented to prove the effectiveness of EICA and GA in solving the DIPPS problem with machine breakdown. The experimental software is developed in the C# programming language and implemented on a personal computer with Windows 7 64-bits, an Intel (R) Core 2.40 GHz, and 4 GB RAM.

To conduct the experiment, we construct a case that contains four jobs and two independent MUs. The representations of transportation time and plans and schedules in different MUs are illustrated in Table 1 and Figures 8 and 9.

6.1. EICA for DIPPS. In this section, the EICA is applied to solve the specific DIPPS problem we preestablished. As we explained in Section 4.6, the ideal convergence where all other empires move towards elimination and only one survives rarely happens in practical circumstances. Consequently, we set another two stop criteria here; the iteration stops once either of them is satisfied: (1) achieve the maximal generation, which is set at 300; (2) the relative difference of average costs between two contiguous generations is no more than 0.005 within four consecutive generations.

Table 2: The partial evolutionary results of EICA.

Rank	Minimum total makespan	Average total makespan
(1)	116	141.57
(2)	118	148.91
(3)	119	141.69
(4)	119	148.23
(5)	120	140.62

Because of the stochasticity of evolutionary algorithms, we run EICA 30 times independently and rank them according to their minimum total makespan values. Because of the limitation of space, Table 2 exhibits the partial evolutionary results. Figure 10 exhibits the evolutionary trajectories of EICA for the best one. The blue line and the red line represent the evolution trajectories of average total makespan and minimum total makespan, respectively.

The best solution after the adoption of EICA is shown in Figure 11; the total makespan of this plan and schedule is 116.

We run ICA and GA 30 times and rank them according to their minimum total makespan value, respectively. Table 3 exhibits the partial evolutionary results. Figure 12 exhibits the evolutionary trajectories of EICA, GA, and traditional ICA for the best one, respectively. The red line, black line, and purple line represent the evolution trajectories of minimum total makespan of EICA, traditional ICA, and GA, respectively.

6.2. GA for Machine Breakdown. Through the former stage of the experiment, the applicable solution is found by EICA and shown in Figure 11. In the solution, Job 1 and Job 4 are processed in MU 1 while Job 2 and Job 3 are processed in MU 2. To test and verify the capability of GA to reschedule the undone operations when machine breakdown occurs, we artificially construct two random machine breakdown scenarios for each MU. Here, in order to measure the influence of machine breakdown for single MU, the criterion is set as makespan f_{ms}^{u}, that is, the completion time of the last operation in the corresponding MU:

$$f_{\mathrm{ms}}^{u} = \max\left(X^{nu} \cdot j\mathrm{ct}^{nu}\right) \quad n \in [1, N], \; u \in [1, U]. \tag{10}$$

In this case, every generation has 20 chromosomes while the crossover and mutation rates are set as 0.85 and 0.05, respectively. Because the search coverage is smaller than the former part, the GA will terminate when the algorithm achieves the maximal generation. And the maximal generation is set at 30.

Table 4 shows the information of the breakdown machines. In this experiment, we compare GA, ICA, and

Table 3: The partial evolutionary results of ICA and GA.

	ICA			GA	
Rank	Minimum total makespan	Average total makespan	Rank	Minimum total makespan	Average total makespan
(1)	120	152.28	(1)	117	123.13
(2)	122	140.72	(2)	121	130.98
(3)	128	145.70	(3)	123	127.28
(4)	131	147.88	(4)	123	125.31
(5)	132	144.67	(5)	124	131.27

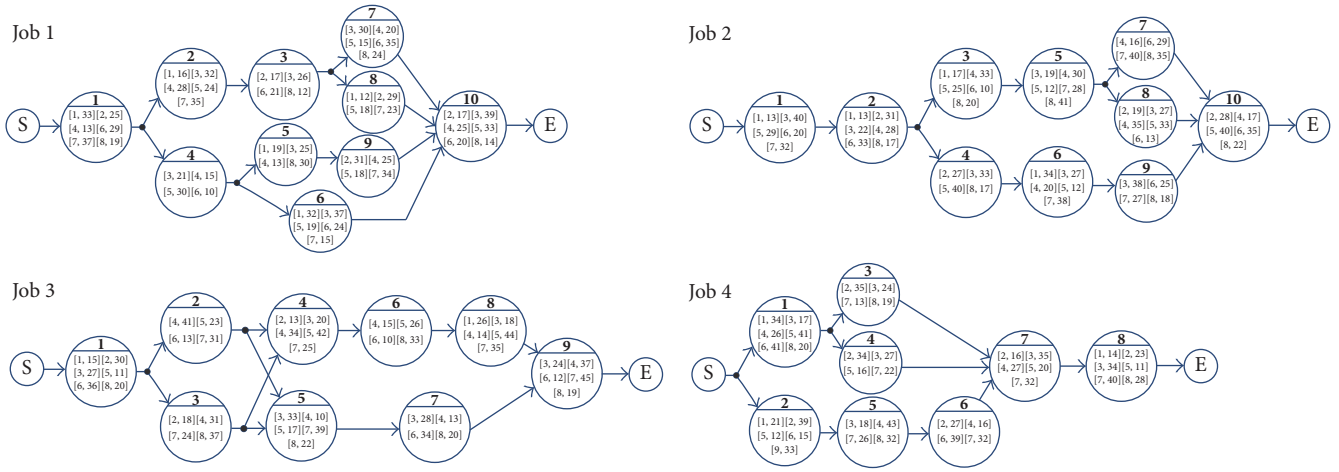

Figure 8: The representation of four jobs' plans and schedules in MU 1.

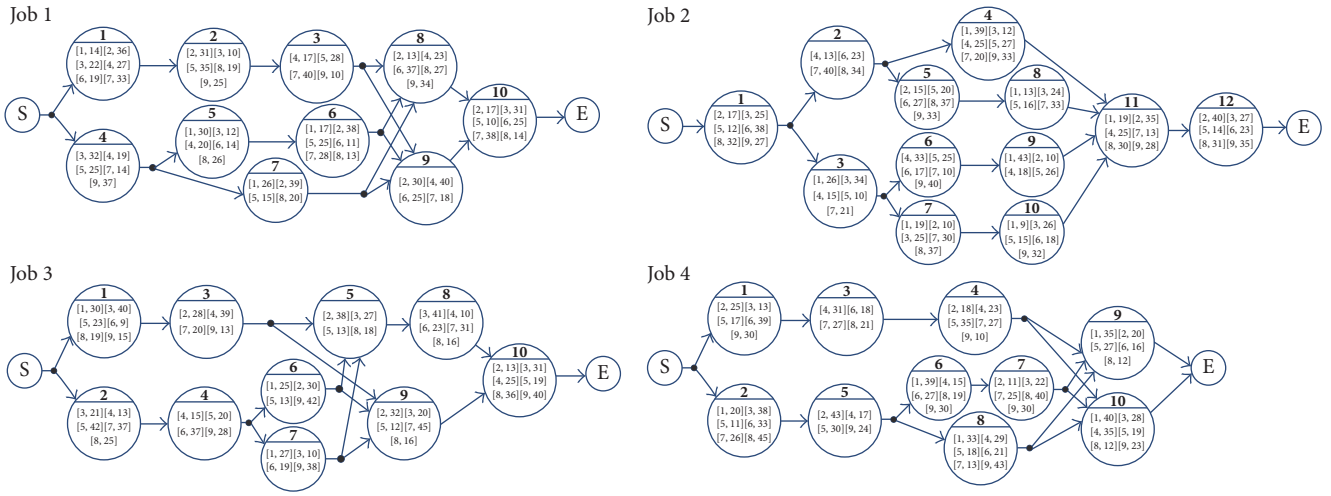

Figure 9: The representation of four jobs' plans and schedules in MU 2.

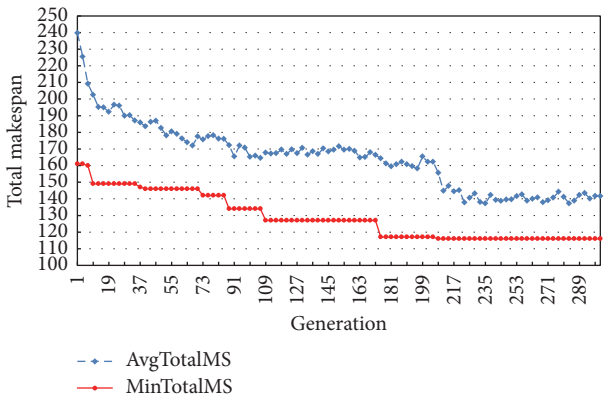

Figure 10: Evolutionary trajectories of EICA.

right-shifting method to get the rescheduling results. We run GA and ICA 30 times, independently, and rank them according to their minimum total makespan value, respectively. Table 5 shows the partial evolutionary results. Figures 13(a–d) exhibit the minimum makespan evolutionary trajectories of GA and ICA for the best one in four cases, respectively. The black line and green line represent the evolution of minimum makespan of GA and ICA, respectively. The corresponding results are shown in Table 6. In all four cases the GA generates much better reschedules than the ICA and the right-shifting plan.

6.3. Robustness. Because there exist no benchmarks of DIPPS, we simulate the real DIPPS environment to assess the optimization capability of EICA. In this study, we assume that each operation can randomly select the machine, and the maximum number of machines in the MU is six. We also assume that the operation time of each machine in an operation ranges from one to fifty and the transportation time from MU to central factory ranges from twenty to fifty. We do the independent experiments five times, and the average minimum makespan is obtained through GA

Table 4: Machine breakdown information.

ID	Information			
	MU	Breakdown machine	The moment of breakdown	Time for repair
(a)	1	4	11	27
(b)	1	7	17	30
(c)	2	2	15	28
(d)	2	4	15	35

Table 5: The partial evolutionary results of GA and ICA.

ID	GA		ICA	
	Minimum total makespan	Average total makespan	Minimum total makespan	Average total makespan
(a)	69	69.55	70	81.75
	69	71.05	72	78.95
(b)	63	63.00	66	66.85
	63	63.45	69	69.00
(c)	64	65.40	75	79.50
	72	72.00	87	87.00
(d)	79	79.00	85	88.80
	79	79.15	89	94.80

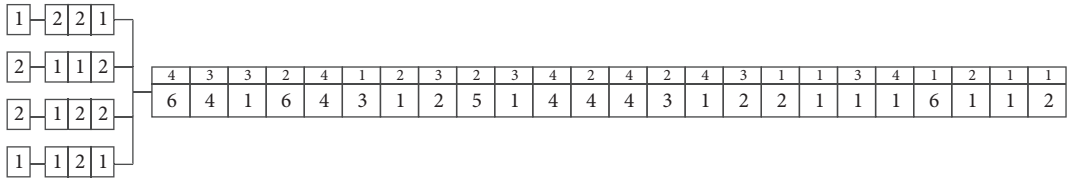

Figure 11: Best solution found by EICA.

Table 6: Rescheduling results of three methods.

Method	ID			
	(a)	(b)	(c)	(d)
Right-shifting	101	100	98	110
ICA	70	66	69	85
GA	**69**	**63**	**64**	**79**

Table 7: The average minimum makespan obtained by GA and EICA.

Experiments	GA	EICA
Experiment a	130.23	127.00
Experiment b	144.43	139.43
Experiment c	132.40	125.33
Experiment d	116.63	113.73
Experiment e	114.90	118.57

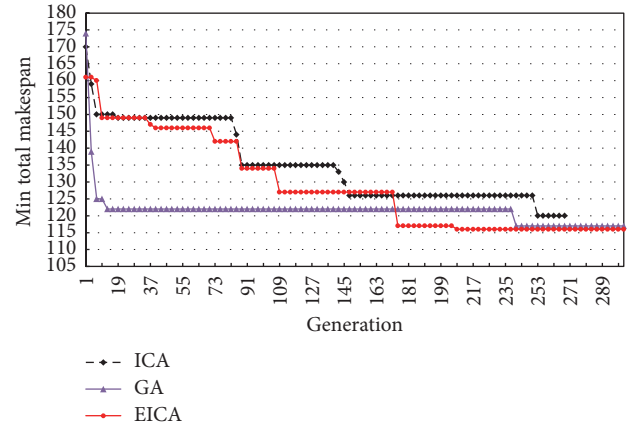

Figure 12: Comparisons of evolutionary trajectories of minimum makespan using three algorithms.

and EICA, respectively. GA and EICA are run 30 times in each experiment, respectively, and the average minimum makespans are exhibited in Table 7. According to Table 7, the proposed EICA performs better than GA in four out of five experiments, which strongly prove the EICA is an effective algorithm for DIPPS.

7. Conclusion

In this study, the DIPPS model that aims to determine process plans and schedules while selecting the appropriate MU for jobs is first constructed. In contrast with the IPPS, the DIPPS discussed here considers the distributed environment of manufacturing, which increases the area of the search

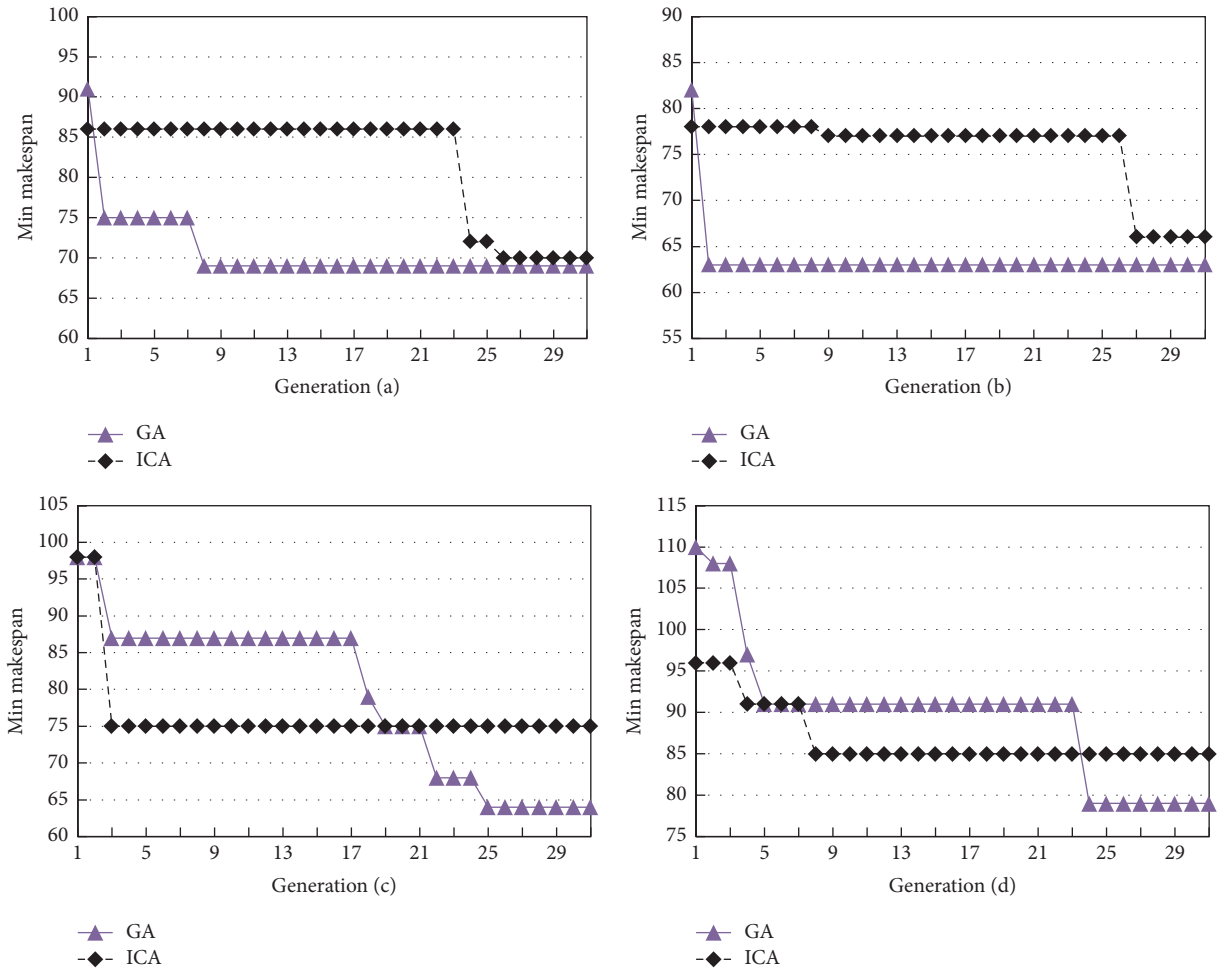

Figure 13: Comparisons of evolutionary trajectories of minimum makespan using GA and ICA.

domain. To effectively solve the DIPPS problem and find an optimal or near-optimal plan and schedule, the ICA is extended by improving country structure, assimilation strategy, and adding resistance procedure. Additionally, because of the inevitability of machine breakdown in manufacturing processes, the GA that not only possesses strong global exploration capability but also has the chromosome whose structure can be easily extracted from the country structure in EICA is adapted to maintain robustness.

To verify the ability of EICA and GA, we conduct a two-stage experiment. In the first stage, the EICA without considering machine breakdown is applied in a case with four jobs and two MUs; in the second stage, the GA is tested through four machine breakdown cases. The results of this two-stage experiment demonstrate the effectiveness and efficiency of extended ICA and GA in solving DIPPS problem with machine breakdown.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

The work has been supported by National Natural Science Foundation of China (no. 51475410, no. 51375429) and Zhejiang Natural Science Foundation of China (no. LY17E050010).

References

[1] C. Moon, J. Kim, and S. Hur, "Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain," *Computers and Industrial Engineering*, vol. 43, no. 1-2, pp. 331–349, 2002.

[2] S. Zhang, Z. N. Yu, W. Y. Zhang, D. J. Yu, and D. P. Zhang, "Distributed integration of process planning and scheduling using an enhanced genetic algorithm," *International Journal of Innovative Computing, Information & Control*, vol. 11, no. 5, pp. 1587–1602, 2015.

[3] W. Y. Zhang, S. Zhang, M. Cai, and J. X. Huang, "A new manufacturing resource allocation method for supply chain optimization using extended genetic algorithm," *International Journal of Advanced Manufacturing Technology*, vol. 53, no. 9-12, pp. 1247–1260, 2011.

[4] J. Wu, W. Y. Zhang, S. Zhang, Y. N. Liu, and X. H. Meng, "A matrix-based Bayesian approach for manufacturing resource allocation planning in supply chain management," *International Journal of Production Research*, vol. 51, no. 5, pp. 1451–1463, 2013.

[5] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[6] H. P. Ma, D. Simon, P. Siarry, and M. R. Fei, "Biogeography-based optimization: a 10-year review," *IEEE Transactions on Emerging Topic in Computational Intelligence*, vol. 1, no. 5, pp. 391–407, 2017.

[7] F. Grimaccia, M. Mussetta, P. Pirinoli, and R. R. Zich, "Genetical swarm optimization (GSO): a class of population-based algorithms for antenna design," in *Proceedings of the 1st International Conference on Communications and Electronics*, pp. 467–471, Hanoi, Vietnam, 2006.

[8] S. Zhang, Z. N. Yu, W. Y. Zhang, D. J. Yu, and Y. B. Xu, "An extended genetic algorithm for distributed integration of fuzzy process planning and scheduling," *Mathematical Problems in Engineering*, vol. 2016, no. 3, pp. 1–13, 2016.

[9] Y. Rahmat-Samii, D. Gies, and J. Robinson, "Particle swarm optimization (PSO): a novel paradigm for antenna designs," *Ursi Radio Science Bulletin*, vol. 76, no. 3, pp. 14–22, 2017.

[10] B. Dorronsoro, P. Ruiz, G. Danoy, Y. Pinge, and P. Bouvry, *Evolutionary algorithms for mobile ad hoc networks*, Wiley Publishing, 2014.

[11] F. Grimaccia, G. Gruosso, M. Mussetta, A. Niccolai, and R. E. Zich, "Design of tubular permanent magnet generators for vehicle energy harvesting by means of social network optimization," *IEEE Transactions on Industrial Electronics*, no. 99, p. 1, 2017.

[12] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," *IEEE Congress on Evolutionary Computation*, pp. 4661–4667, 2007.

[13] H. Bahrami, M. Abdechiri, and M. R. Meybodi, "Imperialist competitive algorithm with adaptive colonies movement," *International Journal of Intelligent Systems & Applications*, vol. 4, no. 2, pp. 49–57, 2012.

[14] J. L. Lin, H. C. Chuan, Y. H. Tsai, and C. W. Cho, "Improving imperialist competitive algorithm with local search for global optimization," in *Proceedings of the Asia Modelling Symposium*, pp. 61–64, 2013.

[15] A. Marto, M. Hajihassani, D. Jahed Armaghani, E. Tonnizam Mohamad, and A. M. Makhtar, "A novel approach for blast-induced flyrock prediction based on imperialist competitive algorithm and artificial neural network," *Scientific World Journal*, vol. 2014, Article ID 643715, 11 pages, 2014.

[16] K. Lian, C. Zhang, X. Shao, and L. Gao, "Optimization of process planning with various flexibilities using an imperialist competitive algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 59, no. 5-8, pp. 815–828, 2012.

[17] E. Shokrollahpour, M. Zandieh, and B. Dorri, "A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem," *International Journal of Production Research*, vol. 49, no. 11, pp. 3087–3103, 2011.

[18] H. Seidgar, M. Kiani, M. Abedi, and H. Fazlollahtabar, "An efficient imperialist competitive algorithm for scheduling in the two-stage assembly flow shop problem," *International Journal of Production Research*, vol. 52, no. 4, pp. 1240–1256, 2014.

[19] N. Moradinasab, R. Shafaei, M. Rabiee, and P. Ramezani, "No-wait two stage hybrid flow shop scheduling with genetic and adaptive imperialist competitive algorithms," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 2, pp. 207–225, 2013.

[20] W. Zhou, J. Yan, Y. Li, C. Xia, and J. Zheng, "Imperialist competitive algorithm for assembly sequence planning," *International Journal of Advanced Manufacturing Technology*, vol. 67, no. 9-12, pp. 2207–2216, 2013.

[21] M. Madani-Isfahani, E. Ghobadian, H. Iranitekmehdash, R. Tavakkoli-Moghaddam, and M. Naderi-Beni, "An imperialist competitive algorithm for a bi-objective parallel machine scheduling problem with load balancing consideration," *International Journal of Industrial Engineering Computations*, vol. 4, no. 2, pp. 191–202, 2013.

[22] K. L. Lian, C. Y. Zhang, L. Gao, and X. Y. Li, "Integrated process planning and scheduling using an imperialist competitive algorithm," *International Journal of Production Research*, vol. 50, no. 15, pp. 4326–4343, 2012.

[23] M. T. Jensen, "Generating robust and flexible job shop schedules using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 275–288, 2003.

[24] L. Liu, H. Y. Gu, and Y. G. Xi, "Robust and stable scheduling of a single machine with random machine breakdowns," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 7-8, pp. 645–654, 2007.

[25] C. Saygin and S. E. Kilic, "Integrating flexible process plans with scheduling in flexible manufacturing systems," *The International Journal of Advanced Manufacturing Technology*, vol. 15, no. 4, pp. 268–280, 1999.

[26] S. M. K. Hasan, R. Sarker, and D. Essam, "Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns," *International Journal of Production Research*, vol. 49, no. 16, pp. 4999–5015, 2011.

[27] C. Bierwirth and D. C. Mattfeld, "Production scheduling and rescheduling with genetic algorithms," *Evolutionary computation*, vol. 7, no. 1, pp. 1–18, 1999.

[28] X. Y. Li, X. Y. Shao, L. Gao, and W. R. Qian, "An effective hybrid algorithm for integrated process planning and scheduling," *International Journal of Production Economics*, vol. 126, no. 2, pp. 289–298, 2010.