*International Journal of Innovative Computing, Information and Control*
Volume **11**, Number **5**, October **2015**

ICIC International ©2015 ISSN 1349-4198
pp. 1587–1602

# DISTRIBUTED INTEGRATION OF PROCESS PLANNING AND SCHEDULING USING AN ENHANCED GENETIC ALGORITHM

Shuai Zhang[1], Zhinan Yu[1,*], Wenyu Zhang[1], Dejian Yu[1] and Dongping Zhang[2]

[1]School of Information
Zhejiang University of Finance and Economics
No. 18, Xueyuan Street, Xiasha District, Hangzhou 310018, P. R. China
zs760914@sina.com; wyzhang@e.ntu.edu.sg; yudejian62@126.com
*Corresponding author: yznyzd@gmail.com

[2]College of Information Engineering
China Jiliang University
No. 258, Xueyuan Street, Xiasha District, Hangzhou 310018, P. R. China
06a0303103@cjlu.edu.cn

Received February 2015; revised June 2015

Abstract. *The distributed integration of process planning and scheduling (DIPPS) not only integrates manufacturing planning and scheduling parts together but also arranges them in the distributed environment to make the manufacturing system more dynamic and efficient. In this paper, we first propose a new DIPPS model to deal with the integrated manufacturing process in the distributed context. Then, an enhanced genetic algorithm (EGA) is constructed to solve the mathematical model. In particular, the EGA features three-segmental and two-dimensional encoding method, improved crossover and double-layer mask mutation scheme to deal with the big solution space in DIPPS. Also, a clear and fast decoding strategy is proposed to decode the chromosome into practical schedules conveniently. In the experiment, the extraordinary capability of EGA is verified by a case study and a comparison with the conventional genetic algorithm.*

**Keywords:** The distributed integration of planning and scheduling, Integrated manufacturing, Genetic algorithm

## 1. Introduction.

In manufacturing activities, both process planning and scheduling are indispensable parts to solve resource conflicts and improve productivity. Specifically, process planning is a function that establishes the technological requirements necessary to convert a part from initial material to a finished form, while scheduling is another manufacturing function that aims to assign manufacturing resources to the operations indicated in the process plans [1]. Meanwhile, with the wide application of distributed manufacturing and the accompanying challenges, process planning and scheduling are of greater importance in manufacturing systems. In early researches, most efforts merely strived to arrange the scheduling part by solving the job-shop scheduling problem (JSP). Although there are modeling and technical breakthroughs in JSP research, this independent research without considering the planning part and the distribution feature of manufacturing may cause unexpected consequences in the dynamic manufacturing environment today, such as job conflicts in raw material and manufacturing equipment.

On the one hand, integrated process planning and scheduling (IPPS) was proposed in view of the urgent need for comprehensive research of process planning and scheduling in manufacturing systems. In general, IPPS features the simultaneous arrangement of process planning and scheduling, and can be viewed as an extension of the JSP in the dimension of the planning part.

On the other hand, many scholars focused on the distributed job-shop problem (DJSP) that deals with the JSP in the distributed environment. A typical scenario of DJSP consists of multiple independent manufacturing cells (MCs) located in one factory where each MC is capable of handling one or more services based on the availability of manufacturing resources.

In this paper, we combine the integration feature of the IPPS with the distribution feature of the DJSP and propose the distributed integration of process planning and scheduling (DIPPS). This creative model takes process planning and scheduling into consideration simultaneously and sets them in a distributed environment. With these improvements, the manufacturing process can be more efficient even under distributed manufacturing environments. Overall, the DIPPS problem can be summarized as follows. Given $I$ jobs consisting of multiple alternative producing processes with different operations in $N$ optional (MCs) with distinct assembly techniques and equipment, we must determine the plans and schedules including cells, process plans and machines for each job by considering the objectives and constraints. Figure 1 shows the distributed environment structure of DIPPS.

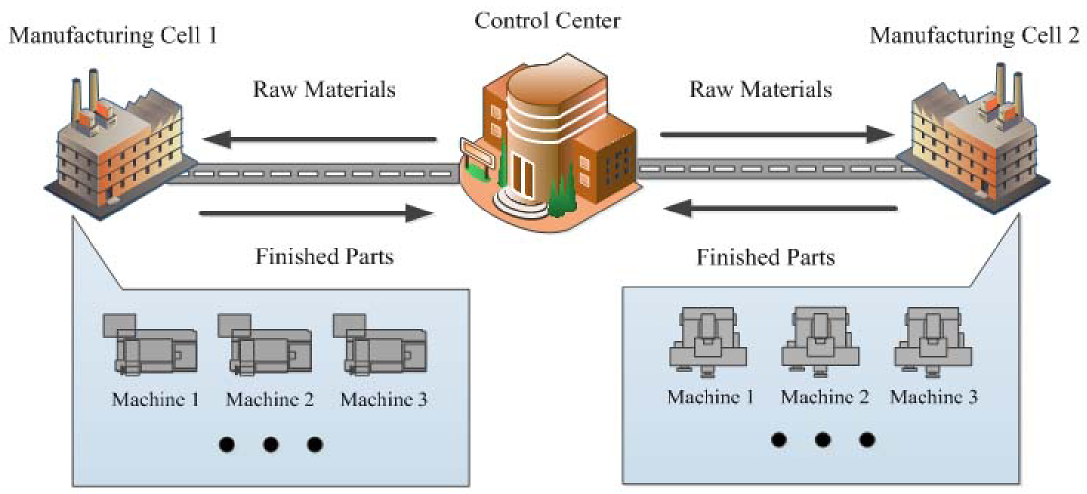

FIGURE 1. The distributed manufacturing environment

To tackle the massive information and constraints in DIPPS, we propose an enhanced genetic algorithm (EGA) in this paper. The EGA has creative encoding and decoding methods, and improves the crossover and mutation operations of the conventional GA. It is competent to solve the DIPPS problem with efficiency and effectiveness.

The remainder of this paper is organized as follows. In Section 2, we review literature related to IPPS and DJSP. In Section 3, the DIPPS problem and its solution, the EGA, are detailed and discussed. In Section 4, experiments with case study and comparison are proposed to show the superiority of the EGA in solving the DIPPS problem. In Section 5, we will draw the conclusion.

## 2. **Related Work.**

The DIPPS discussed in this paper basically depends on the previous studies in IPPS and DJSP. Therefore, we mainly review the literature related to these two areas in this section.

2.1. **The work in IPPS.** A large amount of research has been conducted to tackle IPPS. Among the various methods used to solve this problem, GA is the one with great popularity and convenience. Morad and Zalzala [2] described a GA-based approach that considers the processing capabilities of machines, including processing costs, as well as the number of rejects produced by alternative machines simultaneously with the scheduling of jobs. This is an extension of the traditional JSP problem in terms of its planning feature; unfortunately, it lacks processing flexibility and sequencing flexibility. Moon et al. [3] proposed an evolutionary search approach based on a topological sort to solve the IPPS problem in a supply chain. Qiao and Lv [4] proposed an improved GA with a new initial selection method and new genetic representations to improve the ability of solving the IPPS problem. Li et al. [5] proposed an active learning GA whose learning operator learns from both the excellent individuals that the population has achieved so far and the excellent peers in the current generation to facilitate the integration and optimization of the IPPS problem. The simple GA method may fail to meet the desired expectations at few instances; thus, additional algorithms are needed to further enhance the GA. Despite the ability of GA to perform global searches, they sometimes fall into local optimums. Li et al. [6] and Amin-Naseri and Afshari [7] utilized local search procedure to overcome this weakness.

In addition to GA, other algorithm-based methods also have been proposed to deal with IPPS. Li and McMahon [8] utilized a unified representation model and a simulated annealing-based approach to facilitate the integration and optimization processes. Guo et al. [9, 10] employed a modern evolutionary algorithm, i.e., the particle swarm optimization (PSO) algorithm enhanced with new operators to optimize the IPPS problem.

Agent-based methods have also been considered in recent years. Li et al. [11] carried out an optimization agent based on an evolutionary algorithm to manage the interactions and communications between agents to enable proper decision making. Wong et al. [12] employed two types of agents, part agents and machine agents, to represent parts and machines, respectively. Leung et al. [13] integrated an ant colony optimization (ACO) algorithm into an established multi-agent system to solve the IPPS problem. Wong et al. [14] extended the ACO algorithm in two stages to improve the feasibility.

Given the research depth in model and the hybrid use of algorithms, the above methods can adequately handle the IPPS problem, and defects in algorithm search such as local optima are also overcome gradually. Furthermore, these methods increase search efficiency using distributed computations. However, to the best of our knowledge, few researches have applied IPPS in distributed environment. Besides, although the methods are fully implemented, they are incapable of dealing with the distributed manufacturing.

2.2. **The work in DJSP.** The DJSP broadens the traditional JSP to a more sophisticated level. The conditions and restrictions involved are not only the selection and scheduling of machines, but also the selection of distinct manufacturing cells. Accordingly, improved methods are constructed.

Li et al. [15] utilized the sequence representation of chromosomes to encode the identity of a factory. Jia et al. [16] proposed a modified GA that has two steps in the encoding procedure; the first step encodes factory information, and the second step encodes all jobs' operations and their processing sequences. These two steps enable a chromosome to present sufficient information and also offer convenient computations. Furthermore, Jia et al. [17] combined a GA with a Gantt chart to cope with the distributed environment. Chan et al. [18] not only introduced a new crossover mechanism, namely, dominated gene crossover, to enhance the optimization ability and eliminate the determination processes of the crossover rate, but also included a saturation operator to avoid excessive similarity of chromosomes. To heighten the power of the GA's local search, De Giovanni and Pezzella [19] employed a creative local search operator to improve available solutions by refining the most promising individuals of each generation.

Clearly, the GA used in the DJSP is quite mature, and with the appropriate encoding methods combined with other methods, it is competent of solving DJSP. However, although the DJSP is a more realistic scenario, it only highlights the scheduling part without emphasizing the planning one. The absence of planning skills prohibits the DJSP from adapting to more dynamic environments.

## 3. **An EGA for the DIPPS.**

3.1. **Problem definition and representation.** Generally, in the manufacturing system of DIPPS, there are several independent jobs and several independent and distributed MCs. Each job is permitted to be accomplished via one of the alternative plans. Furthermore, every plan consists of numerous, order-specific operations that can be processed by a couple optional machines. Because of the disparity of machines and technical skills, the plans and operations for a specific job are distinguished by different MCs. This urges the system to determine working MCs, schemes, and appropriate machines for given jobs simultaneously according to the designated criteria. In light of the distribution feature, the transportation time of finished parts from an MC to the control center is also taken into consideration.

DIPPS is generally based on the following assumptions and constraints.

(1) All MCs and their machines are exploitable at time zero and each MC has the capability to process every job.
(2) Because of the technique and manufacturing resources differences among MCs, each job has different plans and schedules in different MCs.
(3) Once a job is assigned to a specific MC, all operations of the job are processed in the assigned MC.
(4) Processing two or more operations of the same job simultaneously is not allowed.
(5) Every machine can only process one operation at a time, and operation interruption is not allowed.
(6) The setup times of the operations and machines are incorporated into the processing time, and the corresponding transfer time is ignored.
(7) The finished parts processed in the same MC are delivered together to the control center once all jobs in the MC are completed.

Here, we construct an example with two MCs and four independent jobs. Then, in order to represent and illustrate the alternative plans in DIPPS, the directed acyclic graph (DAG) is modified and employed. In general, DAG consists of a certain number of nodes and directed edges. For our purposes, one node here represents a specific operation and a directed edge connecting the nodes represents the processing priority order between a pair of operations. When a node has two or more edges to follow, only one of them can be chosen for a plan. Besides, we introduce two dummy operations with no processing time, i.e., the beginning node and ending node, to represent the start and end of a job. Figure 2 and Figure 3 show the alternative plans and schedules for jobs in two MCs with the representation of DAGs; for a node in the graph, the top number indicates the index, and the bottom array indicates the optional machines and corresponding processing time. Along with a group of directed edges from the beginning node to the ending node, one specific plan with a set of operations is determined. We define the serial numbers of plans in one DAG according to the following rule: when a node has more than one edge to choose from, the plan selecting the upper edge has a prior plan serial number. For

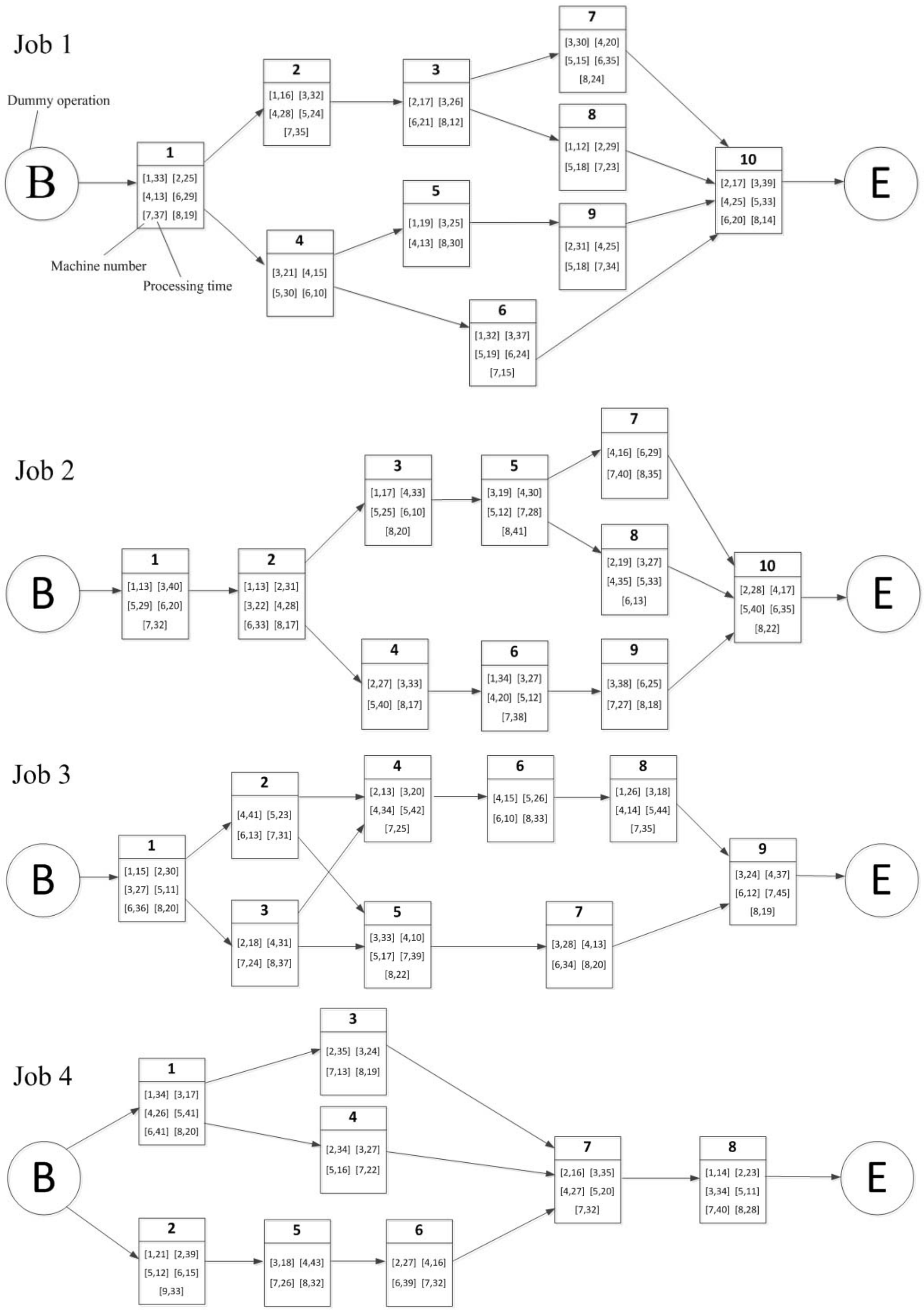

Figure 2. Representation of alternative plans and schedules in MC 1

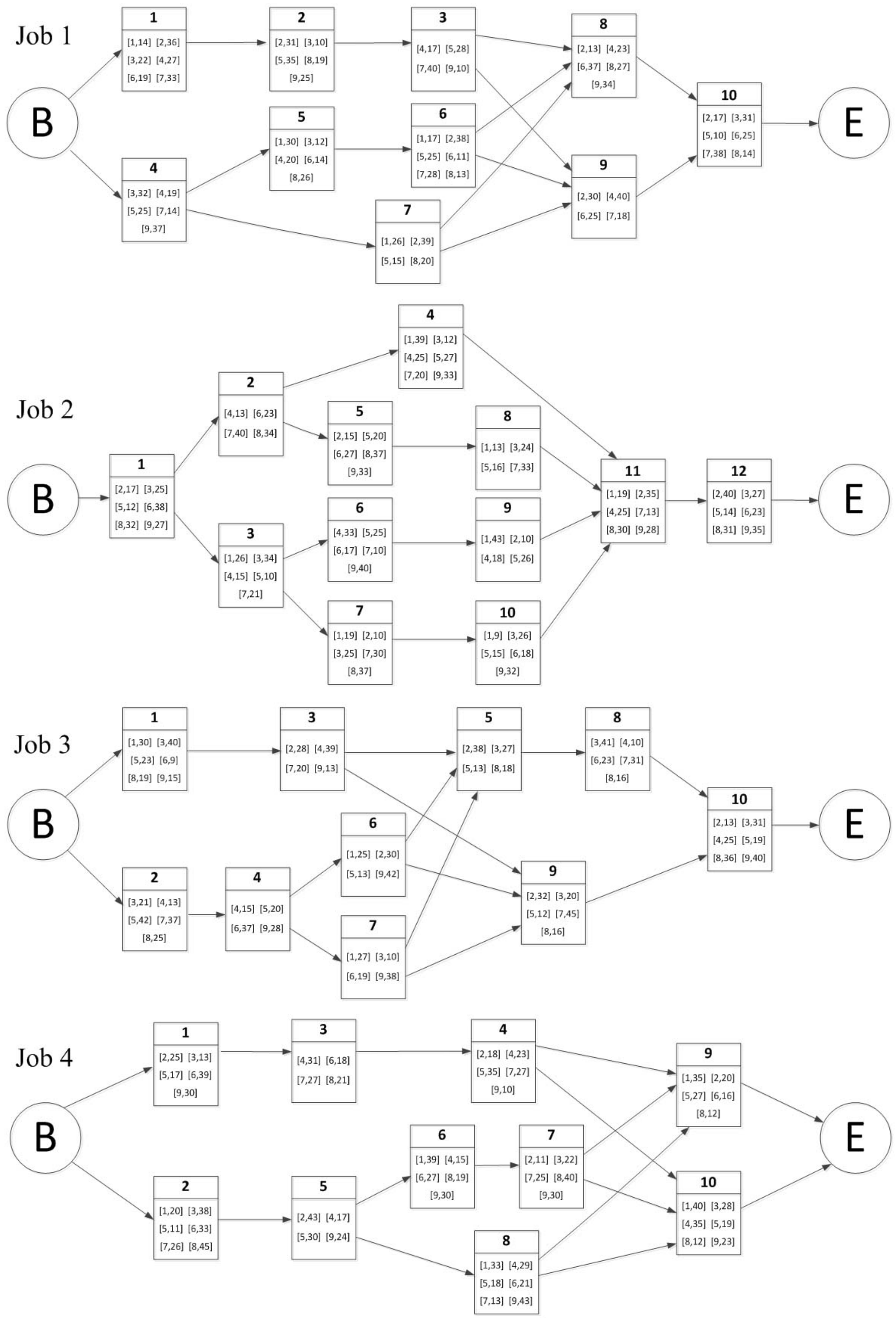

FIGURE 3. Representation of alternative plans and schedules in MC 2

example, for Job 1 processed in MC 1 as shown in Figure 2, the first plan, i.e., Plan 1, follows the operation sequence 1-2-3-7-10; and the second plan, i.e., Plan 2, follows 1-2-3-8-10. The remaining plans can be numbered in a similar manner. Hence, one DAG exhibits all possible plans for a job in a certain MC.

3.2. **The EGA.** As mentioned in the related work, both algorithm-based and agent-based methods show their capability in solving IPPS and DJSP problems [20,21]. In particular, GA is widely used and has achieved remarkable performance. In our previous work [22,23], GA was also adopted to tackle supply chain management problems. However, owing to the feature of distribution and integration, DIPPS is more complicated and requires additional flexibility and constraints. In this circumstance, a conventional GA falls short of our requirements. Therefore, we propose a creative EGA for DIPPS by equipping it with an upgraded representation and encoding technique to contain the information demanded. Furthermore, the genetic operations are improved to reinforce EGA's potential in exploiting the optimal plan and schedule.

3.2.1. *Fitness function.* The final makespan is adopted as the evaluation criteria. Meanwhile, it is converted to the fitness function to ensure that the fitness value is in the interval $[0, 1]$. Because the problem is distributed and MCs are independent, the makespan of each MC is determined first; then, the final makespan of all MCs is calculated with the addition of delivery time.

The following definitions and notation are necessary for the DIPPS problem:

| | |
|---|---|
| $o_{in}^{jkm}$ | The kth operation of alternative process plan $j$ of job $i$ in MC $n$ processed by machine $m$, where $k$, $j$, $i$, $n$, and $m$ are all indices. |
| $I$ | The number of jobs to be processed. |
| $N$ | The number of MCs. |
| $at_{in}$ | The actual accomplishment time of job $i$ from MC $n$ to the control center. |
| $x_{in}$ | $x_{in} \in \{0, 1\}$, where 1 indicates that job $i$ is processed in MC $n$, and 0 indicates that the job is processed in another MC. |
| $tt_{in}$ | The transportation time of the finished part of job $i$ from MC $n$ to the control center. |
| $EM_{fms}$ | The estimated maximal value of the final makespan. |
| $st_{in}^{jkm}$ | The starting time of $o_{in}^{jkm}$. |
| $st_{i^*n}^{j^*k^*m}$ | The starting time of the former operation of $o_{in}^{jkm}$ processed on the same machine $m$. |
| $pt_{in}^{jkm}$ | The processing time of $o_{in}^{jkm}$. |
| $pt_{i^*n}^{j^*k^*m}$ | The processing time of the former operation of $o_{in}^{jkm}$ processed on the same machine $m$. |
| $J_{in}$ | The number of alternative process plans of job $i$ in MC $n$. |
| $K_{in}^{j}$ | The number of operations of alternative process plan $j$ of job $i$ in MC $n$. |
| $M_n$ | The quantity of machines in MC $n$. |

The makespan for jobs in a certain MC ($ms_n$) is given by

$$ms_n = \max_{i\in[1,I]}(at_{in} \cdot x_{in}) \quad \forall n \in [1, N] \tag{1}$$

The final makespan ($fms$) is given by

$$fms = \max_{n\in[1,N]}\left(ms_n + \max_{i\in[1,I]}(tt_{in} \cdot x_{in})\right) \tag{2}$$

The fitness function ($F_{fms}$) is given by

$$F_{fms} = 1 - \frac{fms}{EM_{fms}} \tag{3}$$

The following formulas are formularizations of several constraints in Subsection 3.1:

$$\sum_{i=1}^{I} \sum_{n=1}^{N} x_{in} = I \tag{4}$$

$$st_{in}^{jkm} - st_{i^*n}^{j^*k^*m} - pt_{i^*n}^{j^*k^*m} \geq 0 \tag{5}$$

$$\forall i \in [1, I],\ \forall n \in [1, N],\ \forall j, j^* \in [1, J_{in}],\ \forall k, k^* \in [1, K_{in}^{j}],\ \forall m \in [1, M_n]$$

$$st_{in}^{jk} - st_{in}^{j(k-1)} - pt_{in}^{j(k-1)} \geq 0 \tag{6}$$

$$\forall i \in [1, I],\ \forall n \in [1, N],\ \forall j \in [1, J_{in}],\ \forall k \in [1, K_{in}^{j}]$$

$$\sum_{n=1}^{N} \min_{i \in [1,I]} \left( st_{in}^{j1} \cdot x_{in} \right) = 0 \tag{7}$$

Formula (4) imposes that all operations of one job should be processed in the same MC. Formula (5) imposes that one machine can only process one operation at the same time. Formula (6) implies that two operations of the same job cannot be processed simultaneously. Finally, Formula (7) implies that the starting time of operations is zero, i.e., machines are exploitable at time zero.

3.2.2. *Encoding and decoding.* Due to the increasing amount of information that must be considered, conventional GAs used in the previous work are incapable of representing and solving DIPPS; therefore, we propose an EGA that is competent to tackle this task. In this section, we discuss the encoding and decoding processes of a chromosome in the EGA.

Unlike the traditional IPPS problem, the magnitude of difficulty in DIPPS information encoding increases. The plans, schedules, and machines are distinct among different MCs, which results in a massive amount of information that must be encoded. To this end, EGA extends the traditional structure to a three-segmental chromosome with two-dimensional gene (Figure 4). These three segments represent the arrangement of the MC, process plans, and schedules, respectively. In particular, the third segment has two-dimensional genes to exhibit the schedule information.

The first and second segments of the chromosome contain the same number of genes, and the number depends on the number of jobs. In this example, there are four jobs and two MCs, so the numbers of genes in the first two segments are both four. These genes represent Job 1, Job 2, Job 3 and Job 4 in the sequences. Every gene in the first segment carries the MC information, while every gene in the second segment represents a selected plan of a job. For instance, the first genes in the first and second segments in Figure 4 are both 2, which means Job 1 is processed in MC 2, and Plan 2 is selected in this cell.

In the third one, i.e., the scheduling segment, every gene is adopted to represent an operation for a specific job. In each gene of this segment, the first digit is the job number, and the second one is the machine number the operation chooses. To avoid machine variations when plan changes occur, we propose a substituted machine number, i.e., a successive number beginning from 1 to represent the optional machines for each operation. For example, Figure 2 shows that the first operation of Job 1 can be processed on Machines 1, 2, 4, 6, 7, or 8. Therefore, we use substituted machine numbers 1, 2, 3, 4, 5 and 6 to represent these six machines in the chromosome. Generally, the gene number of this segment is the addition of the selected operations of all jobs. However, since operation quantities are different among different plans, the gene quantity that every job occupies varies once the plans and MCs change. To prevent this situation, the factitious operation is introduced to fix the gene quantity of each job as the operation number of the longest plan. When the chosen plan has fewer operations than the longest plan of the same job, the rest of the genes the job occupies are regarded as factitious operations. Consider the 21st gene of the third segment in Figure 4, i.e., $(3, 2)$, which stands for the 6th operation of Job 3. According to the first two segments and Figure 2, if Job 3 chooses process Plan 2 in MC 1, there are only five operations that need to be processed. Therefore, this gene is a factitious operation and will not be processed.

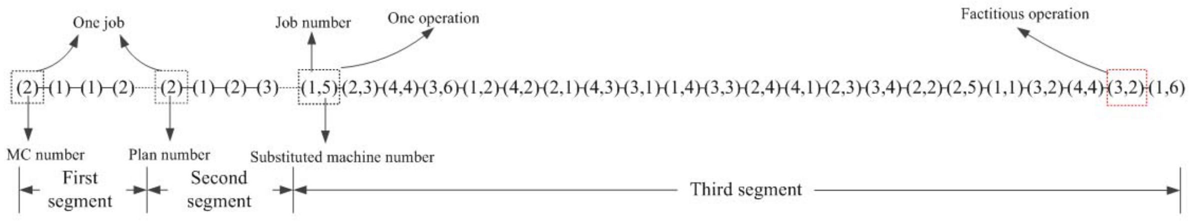

Figure 4. The encoding of EGA

In the decoding procedure, the operations in the third segment will be decoded from left to right. To decode them, Bierwirth and Mattfeld [24] discussed three schedules: active, semi-active, and non-delay. Among them, active schedule is the best method to generate an efficient schedule. Thus, in the decoding phrase, we will adopt active schedule. Meanwhile, since all MCs work independently without mutual interference, the decoding mechanism is definitely applicable in all of them. In this paper, we propose a clear and fast method to achieve active schedule.

For a specific MC, the processing steps of decoding are as follows. Note that JCT = 0 is the largest completion time of the former operations in the same job of the current operation, MCT = 0 is the largest completion time of the former operations on the same machine of the current operation, PT is the processing time of the current operation, and IP is the time of the first available idle point.

Step 1: Scan the operations in the same MC from left to right to detect the unprocessed gene.

Step 2: Operate the first operation by comparing its JCT and MCT. If JCT>MCT, occupy the time from JCT to (JCT+PT) of the assigned machine, and update JCT and MCT according to JCT = JCT+PT and MCT = JCT+PT, respectively. Proceed to Step 5; otherwise, check the idle areas from JCT to MCT on the machine.

Step 3: If no idle area exists, occupy the time from MCT to (MCT+PT) on the machine, and update JCT and MCT according to JCT = MCT+PT and MCT = JCT+PT, respectively. Proceed to Step 5; otherwise, if an idle area exists, seek the available idle area with a successive period of time the same length as PT.

Step 4: If an idle area is available, occupy the time from IP to (IP+PT) and update JCT = IP+PT, and then proceed to Step 5. If there is no available idle area, occupy the time from MCT to (MCT+PT) on the machine, and update JCT and MCT according to JCT = MCT+PT and MCT = MCT+PT, respectively. Proceed to Step 5.

Step 5: Check the set for unprocessed genes. If unprocessed genes are found, return to Step 2; otherwise, end the process and produce the next set.

After the process above is completed, all the makespans of MCs are calculated; the final makespan is then calculated by using (2) in Subsubsection 3.2.1. The flow chart of the process is shown in Figure 5.

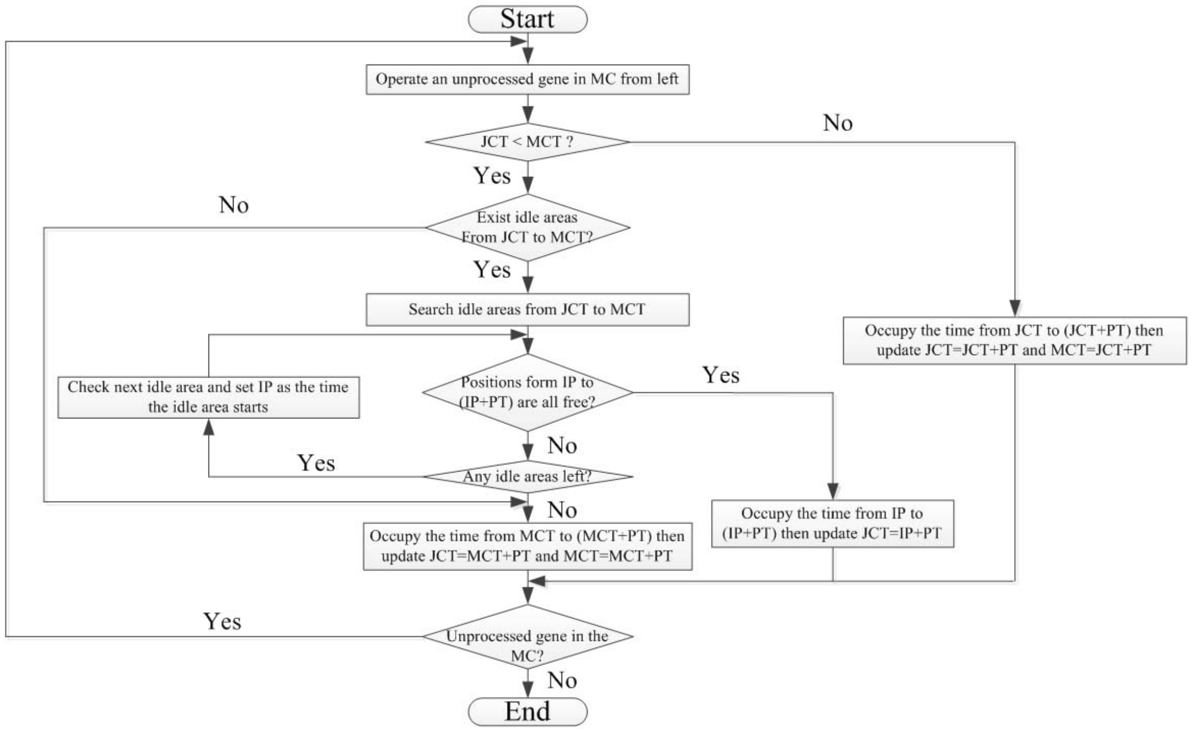
Start
Operate an unprocessed gene in MC from left
JCT < MCT ?
No
Yes
Exist idle areas From JCT to MCT?
No
Yes
Search idle areas from JCT to MCT
Occupy the time from JCT to (JCT+PT) then update JCT=JCT+PT and MCT=JCT+PT
Check next idle area and set IP as the time the idle area starts
Positions form IP to (IP+PT) are all free?
Yes
No
Yes
Any idle areas left?
No
Occupy the time from IP to (IP+PT) then update JCT=IP+PT
Occupy the time from MCT to (MCT+PT) then update JCT=MCT+PT and MCT=MCT+PT
Yes
Unprocessed gene in the MC?
No
End

Figure 5. The flow chart of the decoding algorithm

3.2.3. *Initial population.* According to the encoding principle discussed above, the gene numbers in the first segment, i.e., the MC codes, are generated first. Next, the plan numbers are settled according to the MC. After the MC and plan are determined, the operations of different jobs are randomly arranged in the third segment. The gene number every job occupies in the third segment is determined by their maximum-operation-number plan. Finally, the substituted machine numbers are randomly generated for every operation. Here, because the operations can choose no more than six machines in this example, the range of the substituted machine number for each factitious operation is $[1, 6]$.

3.2.4. *Reproduction.* We employ the roulette wheel selection strategy [25] as the reproduction scheme for the chromosome. In this method, the chromosome with higher fitness value has a greater probability to be selected for the next generation. In general, the roulette wheel is divided into $H$ sections based on $F_{fms}^{h} \Big/ \sum_{h=1}^{H} F_{fms}^{h}$, where $H$ is the number of chromosomes in one generation. However, because the magnitude of the fitness value discrepancy between different chromosomes is not significant, the proportions in the whole wheel are very close, which may cause a failure to differentiate excellent chromosomes. As a result, we adopt a scheme that adds the minimum fitness value in current generation $F_{fms}^{\min}$ to expand the area discrepancy, i.e., $\left(F_{fms}^{h} - F_{fms}^{\min}\right) \Big/ \sum_{h=1}^{H} \left(F_{fms}^{h} - F_{fms}^{\min}\right)$.

3.2.5. *Crossover.* One-point crossover is adopted in the first and second segments of the chromosome, and according to the cut point, the crossover scheme for the third segment is settled (Figure 6). First, two chromosomes are selected as parents Par 1 and Par 2 on the basis of the reproduction scheme explained above. At the same time, two blank chromosomes are initialized as offspring Off 1 and Off 2. Then, the schemes are employed for the three segments. The specific steps are as follows.

Step 1: Randomly set two identical cut points in the first two segments. The genes before the cut point of Par 1 and the genes after the cut point of Par 2 are transferred to the same positions as in Off 1. The remaining genes of Par 1 and Par 2 in this segment are passed down to Off 2 in their original order.

Step 2: For the third segment, the first digits in the genes of Off 1 and Off 2 are inherited from Par 1 and Par 2, respectively, by keeping their original positions. As the former cut point separates the job-specific genes into two parts, the second digits in the genes of the third segment belonging to the jobs ahead of the cut points of Par 1 and Par 2 are passed down to Off 1 and Off 2 in their original positions and orders. The remaining digits are replenished by copying the remaining digits in Par 2 and Par 1 to their corresponding positions in the other parent chromosome.

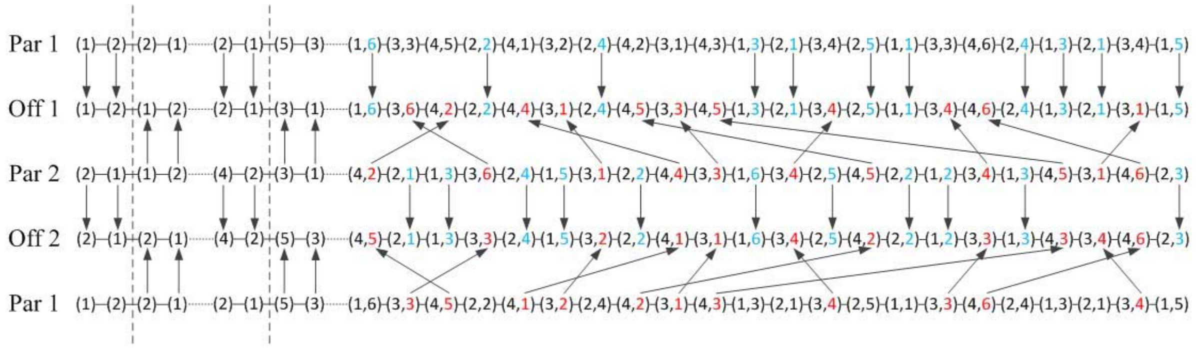
Par 1 (1)-(2)-(2)-(1)-(2)-(1)-(5)-(3)-(1,6)-(3,3)-(4,5)-(2,2)-(4,1)-(3,2)-(2,4)-(4,2)-(3,1)-(4,3)-(1,3)-(2,1)-(3,4)-(2,5)-(1,1)-(3,3)-(4,6)-(2,4)-(1,3)-(2,1)-(3,4)-(1,5)
Off 1 (1)-(2)-(1)-(2)-(2)-(1)-(3)-(1)-(1,6)-(3,6)-(4,2)-(2,2)-(4,4)-(3,1)-(2,4)-(4,5)-(3,3)-(4,5)-(1,3)-(2,1)-(3,4)-(2,5)-(1,1)-(3,4)-(4,6)-(2,4)-(1,3)-(2,1)-(3,1)-(1,5)
Par 2 (2)-(1)-(1)-(2)-(4)-(2)-(3)-(1)-(4,2)-(2,1)-(1,3)-(3,6)-(2,4)-(1,5)-(3,1)-(2,2)-(4,4)-(3,3)-(1,6)-(3,4)-(2,5)-(4,5)-(2,2)-(1,2)-(3,4)-(1,3)-(4,5)-(3,1)-(4,6)-(2,3)
Off 2 (2)-(1)-(2)-(1)-(4)-(2)-(5)-(3)-(4,5)-(2,1)-(1,3)-(3,3)-(2,4)-(1,5)-(3,2)-(2,2)-(4,1)-(3,1)-(1,6)-(3,4)-(2,5)-(4,2)-(2,2)-(1,2)-(3,3)-(1,3)-(4,3)-(3,4)-(4,6)-(2,3)
Par 1 (1)-(2)-(2)-(1)-(2)-(1)-(5)-(3)-(1,6)-(3,3)-(4,5)-(2,2)-(4,1)-(3,2)-(2,4)-(4,2)-(3,1)-(4,3)-(1,3)-(2,1)-(3,4)-(2,5)-(1,1)-(3,3)-(4,6)-(2,4)-(1,3)-(2,1)-(3,4)-(1,5)

FIGURE 6. The crossover operation of EGA

3.2.6. *Mutation.* Mutation is a crucial operation in EGA because of its unpredictable nature. Generally, large-scale mutations change the chromosome dramatically and are favorable for preventing local convergence, while small-scale mutations can retain the feature of excellent chromosome. To exploit the merits of both large-scale and small-scale mutations, we employ a double-layer mask to implement three schemes with different mutation magnitudes on the chromosome (Figure 7). First, a binary set with three numbers in Layer 1 and four numbers in Layer 2 is initialized. In Layer 1, the binary numbers from left to right in the mask are mapped to Schemes 1, 2, 3 and 4. In Layer 2, they are mapped to Jobs 1, 2, 3 and 4. Basically, if some binary numbers are set "1", it means the corresponding scheme and job will participate in the mutation. Notice that only one scheme and one job can be chosen for mutation each time. For a certain chromosome, the three schemes are carried out as follows after the numbers in the mask are randomly set.

Scheme 1: Once the first number in Layer 1 is settled as "1", Scheme 1 is put into effect. Then, according to Layer 2, the corresponding genes in the third segment are mutated by randomly changing the substituted machine numbers in the reasonable range. For example, in Figure 7(a), the second number in Layer 2 is set as "1"; thus, Job 2's related genes in the third segment are mutated by changing the machine numbers.

Scheme 2: This scheme works when the second number in Layer 1 is settled as "1". In addition to changing the third segment, this scheme also requires the EGA to mutate the plan of the corresponding job. According to Figure 7(b), both plan and selected machines of Job 4 are altered.

Scheme 3: The largest scale of mutation happens when the last number in Layer 1 is "1". In this scheme, all corresponding MCs, plans, and machine numbers are transformed

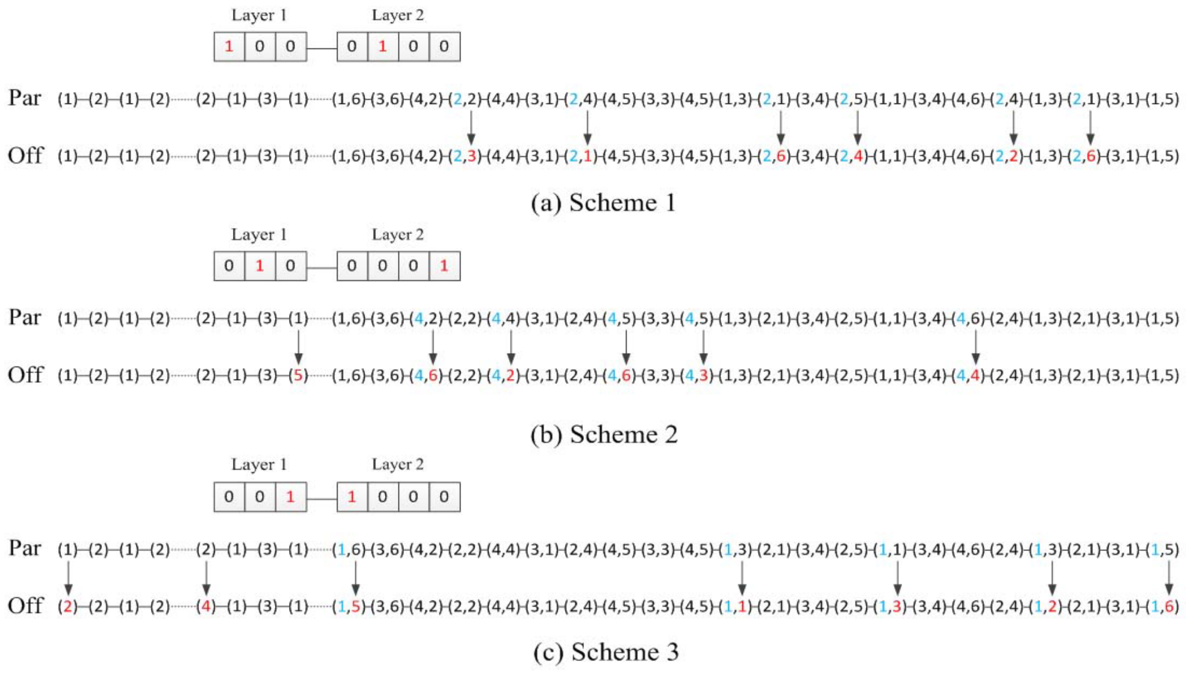

Figure 7. The mutation operation of the EGA

when the specific job-mapping number in Layer 2 is put into "1". For instance, the MCs, plans, and machine numbers of Job 1 are completely changed on in Figure 7(c).

By adopting this creative mutation method, different mutation schemes in various magnitudes have the chance to be implemented. This diversity helps overcome the drawbacks of only using a single scheme.

## 4. Experiment.

### 4.1. Case study.

To verify the capability of the EGA in solving the DIPPS problem, we conduct a case study with two MCs and four jobs; the relevant information is illustrated in Figure 1, Figure 2, Figure 3 and Table 1. Figure 1 exhibits the distributed environment structure of DIPPS with two MCs. Figure 2 and Figure 3 are representations of alternative plans and schedules of four jobs in their respective MCs. Table 1 displays the transportation time of jobs from different MCs to the control center.

Table 1. Transportation time

| $tt_{in}$ Job (i) / MC (n) | Job 1 | Job 2 | Job 3 | Job 4 |
|---|---|---|---|---|
| MC 1 | 34 | 27 | 36 | 40 |
| MC 2 | 43 | 29 | 25 | 43 |

The experiment was conducted on a computer running Windows 7, with 2.40 GHz Intel Core2 Duo CPU and 4.00 GB RAM. The population of every generation was fixed to 30, and the rates of crossover and mutation were 0.85 and 0.05, respectively. The program terminated when one of the two conditions was satisfied: (1) five successive generations' adjacent final makespan difference are all no more than 0.015; (2) the 800th generation is reached. The evolutionary process is displayed in Figure 8 where the final makespan and

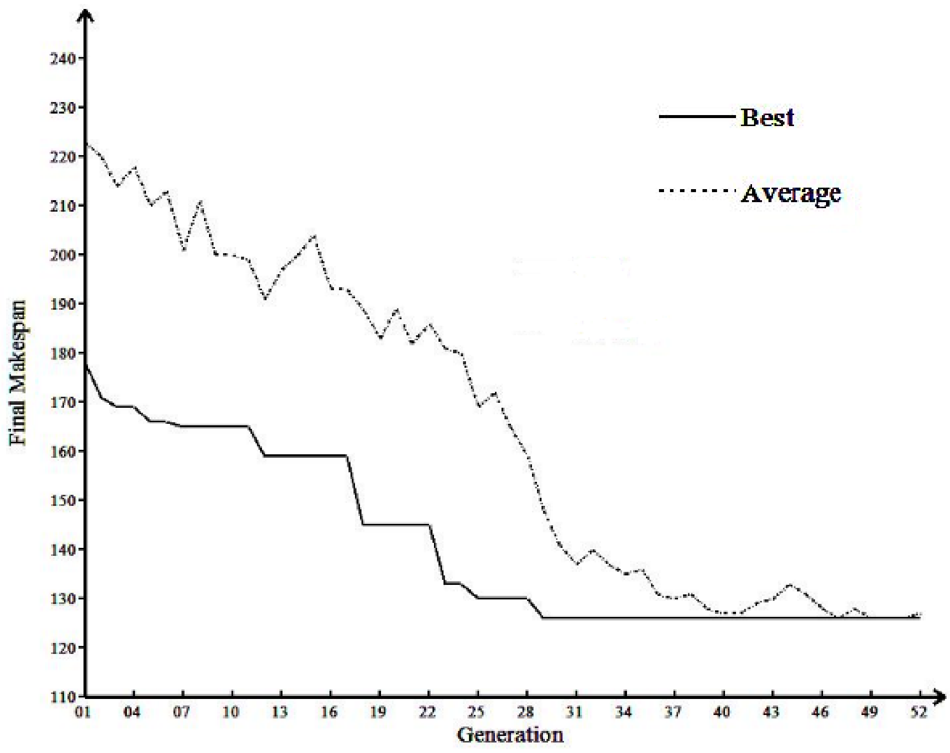

FIGURE 8. The evolutionary process of the EGA

generation correspond to the vertical and horizontal axes, respectively. Also, the dotted line and full line represent the mean final makespan and best final makespan, respectively.

The optimal chromosome was determined to have a processing time of 78s and a final makespan of 126: (1)-(2)-(2)-(1)- - -(4)-(1)-(2)-(2)- - -(4,3)-(1,3)-(2,1)-(2,1)-(3,3)-(2,2)-(3,3)-(2,3)-(3,2)-(4,4)-(2,4)-(3,4)-(4,3)-(3,2)-(4,4)-(1,4)-(1,5)-(1,1)-(1,4)-(2,4)-(3,3)-(4,3). This optimal chromosome indicates that Job 1 and Job 4 are processed in MC 1 by choosing Plan 4 and Plan 2, respectively, with relevant machines, while Job 2 and Job 3 are operated in MC 2 with Plan 1 and Plan 2, respectively, using corresponding machines. Table 2 shows the best schedule for each job, where O and M stand for the index number of operation and the operating machine in Figure 2 and Figure 3, respectively.

TABLE 2. The manufacturing schedule obtained from EGA

| | | |
|---|---|---|
| MC1 | Job 1 | O1(M4) - O4(M6) - O6(M7) - O10(M2) |
| | Job 4 | O1(M4) - O4(M7) – O7(M4) – O8(M5) |
| MC2 | Job 2 | O1(M2) – O2(M4) – O4(M3) – O11(M4) – O12(M6) |
| | Job 3 | O1(M5) – O3(M7) – O9(M3) – O10(M5) |

4.2. **Comparison with the conventional genetic algorithm approach in IPPS.** In this part, a conventional genetic algorithm (CGA) approach used to deal with IPPS is adopted to solve the same case in the first part of the experiment. Then, we compare EGA with it to demonstrate the efficiency and effectiveness of our method.

The chromosome of the CGA contains only the planning part and the scheduling part. For convenience, the last two segments in our EGA are adopted here to represent its chromosome. To apply the CGA in our distributed case, we need to assign a specific MC to each job before the application of CGA, and then compare the best results generated from all conditions to find the best solution. For example, in this case which contains two MCs and four jobs, there are 16 ($2^4$) ways to settle jobs to MCs. That means we need to apply CGA 16 times and then get the best solution from comparing 16 results.

For CGA, the crossover procedure adopts the proposed strategy we illustrated in Sub-subsection 3.2.5, and the mutation procedure is completed by randomly changing the machines of a same job's operations. The population of every generation, the rates of crossover and mutation are 30, 0.85 and 0.05, respectively. Once each CGA reaches its 40th generation (a total of 640 generations for 16 CGAs) or satisfies the first stop criterion of EGA, it will terminate. We run the comparison of 16 CGAs 10 times, and then get the best solution and its evolutionary process (Figure 9). This best solution is from the CGA where Job 2 is assigned to MC 1 and Jobs 1, 3 and 4 are in MC 2. Table 3 shows the comparison between EGA and CGA. EGA exceeds the CGA in both finding the best solution and saving the computing time. In addition, with the increase of MCs and jobs, the computing times of CGA increase exponentially, and will inevitably make the method more unpractical.

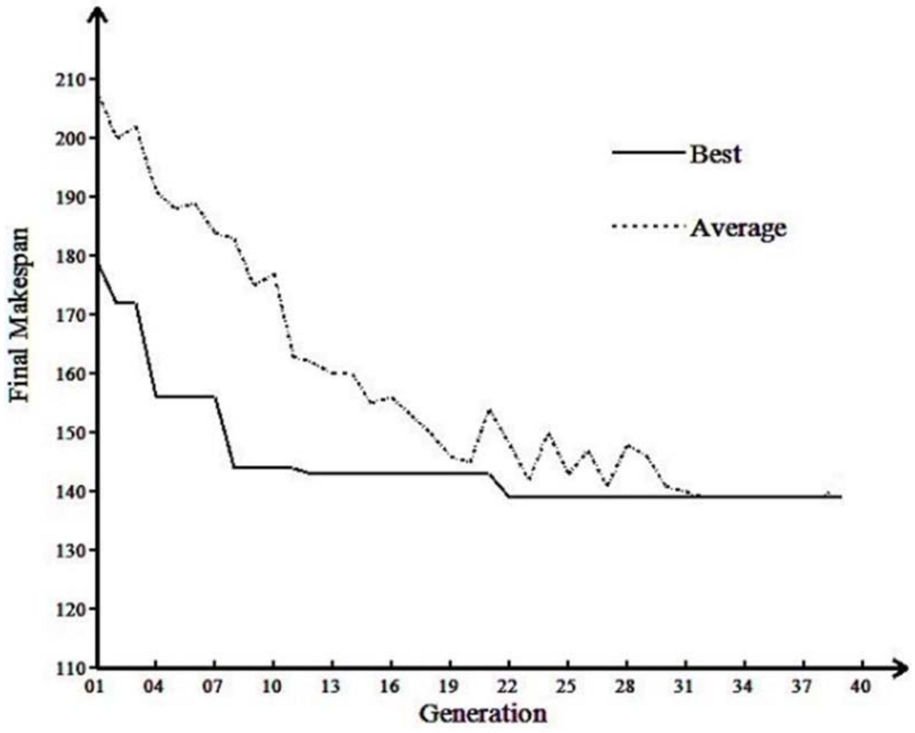

Figure 9. The evolutionary process of the CGA

Table 3. The comparison between EGA and CGA

| | Best final makespan | Computing time |
|---|---|---|
| EGA | **126** | **78s** |
| CGA | 139 | 338s |

## 5. **Conclusions.**

In this paper, a DIPPS model that considers the integration of process planning and scheduling in distributed manufacturing environment is constructed. Meanwhile, a creative EGA is proposed to find the best plan and schedule in DIPPS. Overall, the main contribution of this paper can be concluded as follows.

- Propose a DIPPS model that not only integrates the planning part and the scheduling part of the manufacturing process but also explores the integrated process in a distributed environment.
- Propose an EGA that adopts a new chromosome representation with three-segmental string and two-dimensional genes to encode the information of the DIPPS, and improves the genetic operations with advanced crossover and creative mutation schemes to enhance the global search capability. Also, a convenient decoding method was combined to decode the chromosome into active schedules.

Through a case study and a comparison with CGA used in IPPS, the EGA demonstrates its capability in searching for the optimal plan and schedule for DIPPS, and is competent to deal with the manufacturing activities with both integration and distribution features.

**Acknowledgement.** The work has been supported by China National Natural Science Foundation (No. 51475410, No. 51375429), Zhejiang Science and Technology Plan of China (No. 2013C31099, No. 2013C33046), and Zhejiang Natural Science Foundation of China (No. LY13E050010).

## REFERENCES

[1] Y. K. Kim, K. Park and J. Ko, A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling, *Computers & Operations Research*, vol.30, no.8, pp.1151-1171, 2003.
[2] N. Morad and A. Zalzala, Genetic algorithms in integrated process planning and scheduling, *Journal of Intelligent Manufacturing*, vol.10, no.2, pp.169-179, 1999.
[3] C. Moon, J. Kim and S. Hur, Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain, *Computers & Industrial Engineering*, vol.43, no.1-2, pp.331-349, 2002.
[4] L. H. Qiao and S. P. Lv, An improved genetic algorithm for integrated process planning and scheduling, *The International Journal of Advanced Manufacturing Technology*, vol.58, no.5-8, pp.727-740, 2012.
[5] X. Y. Li, L. Gao and X. Y. Shao, An active learning genetic algorithm for integrated process planning and scheduling, *Expert Systems with Application*, vol.39, no.8, pp.6683-6691, 2012.
[6] 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.
[7] M. R. Amin-Naseri and A. J. Afshari, A hybrid genetic algorithm for integrated process planning and scheduling problem with precedence constraints, *The International Journal of Advanced Manufacturing Technology*, vol.59, no.1-4, pp.273-287, 2012.
[8] W. D. Li and C. A. McMahon, A simulated annealing-based optimization approach for integrated process planning and scheduling, *International Journal of Computer Integrated Manufacturing*, vol.20, no.1, pp.80-95, 2007.
[9] Y. W. Guo, W. D. Li, A. R. Mileham and G. W. Owen, Optimisation of integrated process planning and scheduling using a particle swarm optimisation approach, *International Journal of Production Research*, vol.47, no.14, pp.3775-3796, 2009.
[10] Y. W. Guo, W. D. Li, A. R. Mileham and G. W. Owen, Applications of particle swarm optimisation in integrated process planning and scheduling, *Robotics and Computer-Integrated Manufacturing*, vol.25, no.2, pp.280-288, 2009.
[11] X. Y. Li, C. Y. Zhang, L. Gao, W. D. Li and X. Y. Shao, An agent-based approach for integrated process planning and scheduling, *Expert Systems with Applications*, vol.37, no.2, pp.1256-1264, 2010.

[12] T. N. Wong, C. W. Leung, K. L. Mak and R. Y. K. Fung, An agent-based negotiation approach to integrate process planning and scheduling, *International Journal of Production Research*, vol.44, no.7, pp.1331-1351, 2006.
[13] C. W. Leung, T. N. Wong, K. L. Mak and R. Y. K. Fung, Integrated process planning and scheduling by an agent-based ant colony optimization, *Computers & Industrial Engineering*, vol.59, no.1, pp.166-180, 2010.
[14] T. N. Wong, S. C. Zhang, G. Wang and L. P. Zhang, Integrated process planning and scheduling – Multi-agent system with two-stage ant colony optimisation algorithm, *International Journal of Production Research*, vol.50, no.21, pp.6188-6201, 2012.
[15] L. Li, J. Y. H. Fuh, Y. F. Zhang and A. Y. C. Nee, Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments, *Robotics and Computer-Integrated Manufacturing*, vol.21, no.6, pp.568-578, 2005.
[16] H. Z. Jia, A. Y. C. Nee, J. Y. H. Fuh and Y. F. Zhang, A modified genetic algorithm for distributed scheduling problems, *Journal of Intelligent Manufacturing*, vol.14, no.3-4, pp.351-362, 2003.
[17] H. Z. Jia, J. Y. H. Fuh, A. Y. C. Nee and Y. F. Zhang, Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems, *Computers & Industrial Engineering*, vol.53, no.2, pp.313-320, 2007.
[18] F. T. S. Chan, S. H. Chung and P. L. Y. Chan, An adaptive genetic algorithm with dominated genes for distributed scheduling problems, *Expert Systems with Applications*, vol.29, no.2, pp.364-371, 2005.
[19] L. De Giovanni and F. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, *European Journal of Operational Research*, vol.200, no.2, pp.395-408, 2010.
[20] H. K. Niazi, H. F. Sun, F. P. Zhang and R. Ahmed, Extending MATLAB and GA to solve job shop manufacturing scheduling problems, *WSEAS Trans. Systems*, vol.5, no.4, pp.805-810, 2006.
[21] T. P. Hong, P. C. Sun and S. D. Li, A heuristic algorithm for the scheduling problem of parallel machines with mold constraints, *WSEAS Trans. Systems*, vol.7, no.6, pp.642-651, 2008.
[22] 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, *The International Journal of Advanced Manufacturing Technology*, vol.53, no.9-12, pp.1247-1260, 2011.
[23] 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.
[24] C. Bierwirth and D. C. Mattfeld, Production scheduling and rescheduling with genetic algorithms, *Evolutionary Computation*, vol.7, no.1, pp.1-17, 1999.
[25] D. B. Fogel, An introduction to simulated evolutionary optimization, *IEEE Trans. Neural Networks*, vol.5, no.1, pp.3-14, 1994.