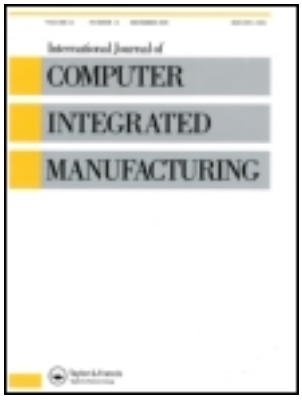

International Journal of Computer Integrated Manufacturing

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcim20

A game-theoretic approach to generating optimal process plans of multiple jobs in networked manufacturing

Guanghui Zhou [a b], Zhongdong Xiao [c d], Pingyu Jiang [a b] & George Q. Huang [e]

[a] State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an, 710049, China

[b] School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, 710049, China

[c] School of Management, Xi'an Jiaotong University, Xi'an, 710049, China

[d] The Key Lab of the Ministry of Education for Process Control & Efficiency Engineering, Xi'an Jiaotong University, Xi'an, 710049, China

[e] Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, China

Available online: 22 Nov 2010

To cite this article: Guanghui Zhou, Zhongdong Xiao, Pingyu Jiang & George Q. Huang (2010): A game-theoretic approach to generating optimal process plans of multiple jobs in networked manufacturing, International Journal of Computer Integrated Manufacturing, 23:12, 1118-1132

To link to this article: http://dx.doi.org/10.1080/0951192X.2010.524248

International Journal of Computer Integrated Manufacturing
Vol. 23, No. 12, December 2010, 1118–1132

Taylor & Francis
Taylor & Francis Group

A game-theoretic approach to generating optimal process plans of multiple jobs in networked manufacturing

Guanghui Zhou[a,b]*, Zhongdong Xiao[c,d], Pingyu Jiang[a,b] and George Q. Huang[e]

[a]*State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China;* [b]*School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an 710049, China;* [c]*School of Management, Xi'an Jiaotong University, Xi'an 710049, China;* [d]*The Key Lab of the Ministry of Education for Process Control & Efficiency Engineering, Xi'an Jiaotong University, Xi'an 710049, China;* [e]*Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, China*

(*Received 25 December 2009; final version received 12 September 2010*)

This study seeks to address an approach for generating optimal process plans for multiple jobs in networked manufacturing. Because of production flexibility, generating several feasible process plans for each job is possible. Concerning the networked manufacturing mode, the specific scenario of competitive relationships, like delivery time existing between different jobs, should be taken into account in generating the optimal process plan for each job. As such, in this study, an N-person non-cooperative game-theoretic mathematical solution with complete information is proposed to generate the optimal process plans for multiple jobs. The game is divided into two kinds of sub-games, i.e. process plan decision sub-game and job scheduling sub-game. The former sub-game provides the latter ones with players while the latter ones decide payoff values for the former one to collaboratively arrive at the Nash equilibrium (NE). Endeavouring to solve this game more efficiently and effectively, a two-level nested solution algorithm using a hybrid adaptive genetic algorithm (HAGA) is developed. Finally, numerical examples are carried out to investigate the feasibility of the approach proposed in the study.

Keywords: process plan; job scheduling; game theory; hybrid adaptive genetic algorithm; networked manufacturing.

1. Introduction

The recently emerged Networked Manufacturing is characterised by globalisation, collaboration, customisation, digitalisation and agility, providing a collaborative environment for customers, manufacturers and suppliers to work together (Jiang *et al*. 2007, Zhou *et al*. 2009). In networked manufacturing, the production mode has been changed from make-to-stock to make-to-order, in which the active participation of customers submitting manufacturing jobs in the manufacturing process is stressed. The competition of delivery times between different manufacturing jobs is becoming a focal point to be resolved in process planning and scheduling. In order to arrive at the objective of quick delivery to market, generating optimal process plans for different jobs to be manufactured is becoming a critical problem for enterprises to survive in the drastic global market competition.

Be in traditional manufacturing environments or in networked manufacturing ones, process planning and scheduling play important roles in production (Brandimarte and Calderini 1995, Huang *et al*. 1995, Kim and Egbelu 1999, Shafaei and Brunn 2000, Moon *et al*. 2002). Generally, a process plan specifies what raw materials or components are needed to produce a product and what operations are required to change those raw materials to the final product. The outcomes of process planning are information of manufacturing operations with all types of parameters and types of machines required to perform those operations. Scheduling attempts to allocate manufacturing resources to operations identified in process plans over time in such a way that some relevant criteria, e.g. minimal makespan, minimal cost and due date, could be met. Therefore, both process planning and scheduling involve the assignment of manufacturing resources to production tasks (Saygin and Kilic 1999, Kim *et al*. 2003).

Although there is a strong relationship between process planning and scheduling, these two functions have been studied independently at their initial stages. Subsequently, some researchers have realised the need to integrate process planning and scheduling to work together and have established the base for process planning and scheduling integration in networked

*Corresponding author. Email: ghzhou@mail.xjtu.edu.cn

ISSN 0951-192X print/ISSN 1362-3052 online

DOI: 10.1080/0951192X.2010.524248
http://www.informaworld.com

manufacturing environments. Huang *et al.* (1995) proposed a progressive approach for implementing the integration of process planning and scheduling. Kim and Egbelu (1999) developed a scheduling solution in which multiple process plans per job were taken into account. Shafaei and Brunn (2000) discussed the effects of process planning and scheduling functions and established a framework to integrate the two functions. Moon *et al.* (2002) presented an integrated approach for process planning and scheduling by taking the minimal total tardiness as the objective in the supply chain. Saygin and Kilic (1999) discussed the importance of integration between process planning and scheduling in flexible manufacturing systems (FMS) and developed a framework for integrating flexible process plans with scheduling in FMS. Kim *et al.* (2003) proposed a symbiotic evolutionary algorithm for dealing with the problems of integration between process planning and job shop scheduling. Li and McMahon (2007) developed a unified representation model and a simulated annealing-based approach to facilitate the integration and optimisation of process planning and scheduling. Shao *et al.* (2009) proposed a modified genetic algorithm-based approach for the integration of process planning and scheduling in which more efficient genetic representations and operator schemes were developed to enhance the optimised performance. Li *et al.* (2010) developed an agent-based approach for facilitating the integration of process planning and scheduling. In this method, an optimisation agent based on an evolutionary algorithm was used to manage the interactions and communications between agents to enable proper decisions to be made. However, considering new characteristics and requirements of networked manufacturing over traditional manufacturing, the models, solutions and algorithms addressed in the above literature should be extended and updated. First, the objective for traditional process planning and scheduling is to acquire overall optimal results for all jobs that is not fully concerned with the individual optimal requirement for each job. Nevertheless, because the competition factor between different jobs plays an important role in networked manufacturing, the objective for process planning and scheduling is somewhat different than that in traditional manufacturing, which must strive for individual optimal results for each job. In process planning and scheduling, the jobs submitted by different customers may become the opponents in which every one strives for their own optimal profit such as minimal makespan or manufacturing cost, etc. while decreasing the profits of their opponents to the best of their ability. Second, the machines related to jobs are always limited inside a single workshop or even an enterprise in traditional manufacturing. However, in networked manufacturing, the machines for jobs to be manufactured are geographically distributive, which are possibly derived from different workshops or even enterprises. In other words, the transportation time between two candidate machines would be one of the important factors of influencing the final decision for process planning and scheduling. Moreover, in networked manufacturing, like some types of flexible manufacturing systems, processing operations are becoming increasingly flexible, and most jobs may have several feasible process plans. Therefore, how to generate the optimal process plan for each job regarding correspondent dynamic constraints is becoming a new problem by considering the current status of resources at manufacturing spots.

Considering the above three characteristics and requirements of networked manufacturing, a new approach to generating optimal process plans for multiple jobs should be developed to satisfy the requirements derived from networked manufacturing. Therefore, the challenge is how to develop a suitable decision approach to effectively and efficiently support generating optimal process plans for jobs in networked manufacturing. Currently, game theory is becoming more popular and has been gradually introduced to deal with the scheduling related problems requiring high performance. Seredyński (1997, 1998) and Seredyński *et al.* (2001) discussed the feasibility of using game theory for scheduling problems and developed the game-based solution models and algorithms for typical distributed scheduling problems. Kim (2003) presented a new approach and developed a framework for maintenance scheduling of generating units (MSU) in competitive electricity markets. Zhou *et al.* (2009) introduced game theory for job-shop scheduling problems and proposed a game-theoretic approach for job scheduling in networked manufacturing. Thus, in order to give voice to the challenge, this article introduces game theory and establishes an N-person, non-cooperative game-theoretic mathematical model with complete information for generating optimal process plans for multiple jobs. This game is divided into two kinds of sub-games, i.e. process plan decision sub-game and job scheduling sub-game, which cooperate with each other to decide optimal process plan for each job. The former sub-game provides the latter ones with players, while the latter ones provide the former with payoff values. In the former sub-game, players correspond to jobs, strategies of each job to its feasible process plans and payoff of each job to its makespan provided by job scheduling sub-games when their Nash equilibrium (NE) points are found. In the latter sub-games, players correspond to process plans of jobs provided by the former one, strategies of each process plan to its related alternative geographical

distributive machines contained in networked manufacturing environment and payoff of each process plan to its makespan. Thus, deciding on the optimal process plans for jobs means finding the NE point in process plan decision sub-game. At the NE point, each job can win optimal process plan and no job wants to unilaterally deviate from NE status. Endeavouring to solve this game more efficiently, we propose a two-level nested solution algorithm by designing and utilising the hybrid adaptive genetic algorithm named HAGA to find the perfect NE points for two kinds of sub-games, and finally to obtain the optimal result for each job according to its own goal.

This remainder of this article is organised as follows. We first describe in Section 2 the research problem in detail to be dealt with in this article. In section 3, we develop and formulate a mathematical model to represent the research problem by using the N-person non-cooperation game. Section 4 explains the use of HAGA for solving the proposed game. We devote Section 5 to report and analyse the case simulation results. Section 6 describes some conclusions.

2. Problem description

For this study, we first define the problem to be resolved as described in detail as the following:

(1) Denote n as the number of jobs submitted by customers and $J_i(1 \le i \le n)$ to represent job i, J_i contains R_i feasible process plans, each process plan $pr_{i,j}(1 \le i \le n;\ 1 \le j \le R_i)$ contains $JN_{i,j}$ sequential operations and each operation corresponds to a set of alternative machines. Denote $O^k_{i,j}$ as operation k of $pr_{i,j}$ and $pr_{i,j} = \{O^k_{i,j} | 1 \le k \le JN_{i,j}\}$.
(2) Let m represent the number of machines related to jobs in networked manufacturing and f_l $(1 \le l \le m)$ an alternative machine.
(3) Denote $pt^k_{i,j}(f_l)$ as the processing time of $O^k_{i,j}$ of $pr_{i,j}$ on f_l.
(4) Denote $st^k_{i,j}(f_l)$ as the starting processing time of $O^k_{i,j}$ of $pr_{i,j}$ on f_l.
(5) Denote $t_{i,j}(f_l, f_{l'})$ as the transportation time of J_i between f_l and $f_{l'}$ according to $pr_{i,j}$.
(6) Denote T_i as the makespan of J_i and $T_i = st^{JN_{i,j}}_{i,j}(f_l) + pt^{JN_{i,j}}_{i,j}(f_l)$. The final objective we consider for this study is to minimise T_i to further obtain the corresponding optimal process plan for J_i from the set of feasible process plans. The objective function can be described as follows.

$$\tilde{F}_i = \min(T_i) = \min(st^{JN_{i,j}}_{i,j}(f_l) + pt^{JN_{i,j}}_{i,j}(f_l)) \qquad (1)$$

In order to satisfy Equation (1) effectively and efficiently, we should make some assumptions described as follows:

(1) When an operation of a job is being processed on a machine, it cannot be interrupted until finished.
(2) Job preemption is not allowed.
(3) No two jobs (operations) are scheduled to be processed on the same machine at the same time.
(4) The transportation time is considered in that after an operation of a job is processed on a machine; it will be immediately transported to next machine according to its own routing.
(5) All jobs can be simultaneously available at the time of zero.

Consulting the defined problem, the objective of generating optimal process plans of multiple jobs is to seek for the optimal process plan for each job in which all jobs consider they have obtained optimal profits.

Figure 1 describes the flexible characteristic among process plans contained in a job as a network diagram. Three kinds of nodes are designated: starting node, operation node and ending node. The starting node and ending node are respectively utilised to explicitly indicate the start and finish status of manufacturing process of a job. The operation node represents an operation that contains alternative machines on which the operation can be processed on its correspondent processing time. For example, in the case of operation

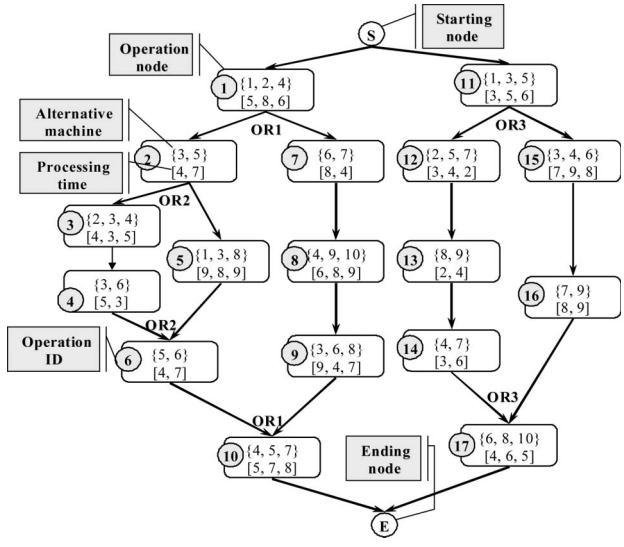

Figure 1. A network diagram for flexible process plans of a job.

node 2, {3,5} represents the set of alternative machines and [4,7] represents the set of processing times. This means that if the operation is processed on machine 3, the processing time is 4 time units and if on 5, the processing time is 7 time units. The arrows connecting nodes represent precedence between them. The 'OR' relationships are used to describe the processing flexibility that the same manufacturing feature can be finished with different operation procedures. If the links following a node are connected by an 'OR' symbol (called OR-connector), we only need to traverse one of the OR-links (the links connected by an OR-connector are called OR-links). An operation path that starts at an OR-link and ends as it merges with other paths is called an OR-link path. The end of OR-link path is also denoted by an OR-connector. For those links that are not connected by OR-connectors, all of them should be visited. In Figure 1, paths {12;13; 14} and {15;16} are two OR-link paths. An OR-link path may contain other OR-link paths, e.g. paths {3; 4} and {5}.

3. The game solution

Considering the above-described problem, a game solution is presented and developed to realise the generation of optimal process plans for multiple jobs. The presented game is an N-person non-cooperative game with complete information that is divided into two kinds of sub-games named process plan decision sub-game and job scheduling sub-game. Figure 2 illustrates logic framework of this game solution and relationship between the two kinds of sub-games. There exist one process plan decision sub-game and a series of job scheduling sub-games. In the process plan decision sub-game, players correspond to the jobs submitted by different customers, strategies of each job to its alternative process plans and payoff of each job to its makespan. In the job scheduling sub-game, players correspond to process plans provided by process plan decision sub-game, strategies of each process plan to its alternative machines dispersed in networked manufacturing environment and payoff of each process plan to its makespan. The shapes and number of job scheduling sub-games depend on the strategy combination of process plan decision sub-game. The final objective of this game solution is to obtain the NE point of process plan decision sub-game to further retro-deduce the optimal process plan for each job. In order to arrive at the objective, the two kinds of sub-games work together. The process plan decision sub-game provides the job scheduling sub-games with players (corresponding to process plans) while the job scheduling sub-games provide the process plan decision sub-game with payoff values. Through utilising payoff values provided by the job scheduling sub-games at their NE points, the optimal process plan for each job can be derived from the NE point of

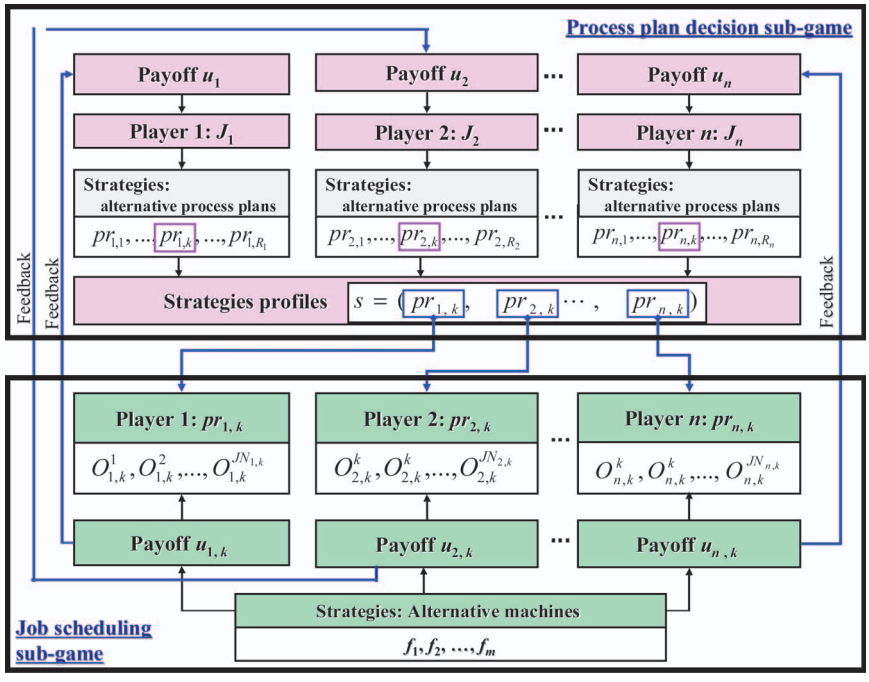

Figure 2. Logic structure of the game solution.

process plan decision sub-game. The definitions and functions of the above two sub-games are expressed in detail as follows.

3.1. Process plan decision sub-game

The process plan decision sub-game is described as a tupple, which is expressed as:

$$G = (N, S_i, u_i) \tag{2}$$

Where

(1) N represents the set of jobs, which is expressed as $N = \{J_i | 1 \le i \le n\}$.
(2) S_i represents the set of strategies of J_i corresponding to its alternative process plans expressed as $S_i = \{pr_{i,j} | 1 \le j \le R_i\}$.
(3) u_i represents a payoff function of J_i depending on strategies not of J_i but of other jobs that cannot be known in advance. Because the objective of this game solution is to minimise the makespan of every job, we define the payoff of J_i implicitly as a function of strategy combination s described in the following equation.

$$u_i(s) = F(pr_{i,j}, p\hat{r}_{u,v}) \tag{3}$$

where $p\hat{r}_{u,v} = (pr_{1,v_1}, \ldots, pr_{i-1,v_{i-1}}, pr_{i+1,v_{i+1}}, \ldots, pr_{n,v_n})$, $1 \le v_1 \le R_1, 1 \le v_{i-1} \le R_{i-1}, 1 \le v_{i+1} \le R_{i+1}, 1 \le v_n \le R_n$.

In equation (3), F is the function of $pr_{i,j}$ and $p\hat{r}_{u,v}$, which guarantees to generate the optimal process plan in which the minimal makespan of J_i can be obtained. The function value is derived from NE points of job scheduling sub-games. s represents the strategy profile (strategy combination) decided by all jobs at a time, which is described as $s = (s_1, s_2, \ldots, s_n)$, $s_i \in S_i$. Therefore, the length of s is n.

3.2. Job scheduling sub-game

The job scheduling sub-game can also be represented by a tupple shown in the following.

$$G' = (N', S_{i,j}, u_{i,j}) \tag{4}$$

Where

(1) N' represents the set of process plans provided by process plan decision sub-game in which one job corresponds to one and only one process plan. Therefore, $N' \equiv s$.
(2) $S_{i,j}$ represents the strategy set of $pr_{i,j}$ corresponding to the set of alternative machines for sequential operations in $pr_{i,j}$. Let M represent the set of machines related to all jobs in networked manufacturing environment; therefore, M can be described as $M = \{f_l | 1 \le l \le m\}$. Thus, we can deduce that $S_{i,j} \subseteq M$.
(3) $u_{i,j}$ is the payoff function of $pr_{i,j}$, which acts as a reference to choosing feasible machines for $pr_{i,j}$ referring to its sequential operations. $u_{i,j}$ is defined as follows.

$$u_{i,j}(s') = T_i = st_{i,j}^{JN_{i,j}}(f_l) + pt_{i,j}^{JN_{i,j}}(f_l) \tag{5}$$

where

$$s' = (s'_1, \ldots, s'_i, \ldots, s'_n), s'_i \in S_{i,j}$$

and subject to

$$st_{i,j}^{k+1}(f_{l'}) \ge st_{i,j}^{k}(f_l) + pt_{i,j}^{k}(f_l) + tt_{i,j}(f_l, f_{l'}) \tag{6}$$

and

$$st_{i,j}^{k}(f_l) + pt_{i,j}^{k}(f_l) \ge st_{x,y}^{k'}(f_l) \quad \text{or}$$
$$st_{i,j}^{k}(f_l) \le st_{x,y}^{k'}(f_l) + pt_{x,y}^{k'}(f_l) \tag{7}$$

where $x = 1, \ldots, n; y = 1, \ldots, R_x; k' = 1, \ldots, JN_{x,y}; i \ne x; f_l \in S_{i,j} \cap S_{x,y}$.

Equation (5) denotes that payoff of $pr_{i,j}$ is the sum of starting time and processing time of $O_{i,j}^{JN_{i,j}}$ of $pr_{i,j}$ processed on f_l by following strategy set of s'. Formula (5) describes the sequence constraint of operations, which means that $O_{i,j}^{k+1}$ cannot be started before finishing $O_{i,j}^{k}$ of J_i on f_l following $pr_{i,j}$. Formula (7) expresses the machine constraint, which means that each machine can only process one operation at a time.

According to equation (5), we transform equation (3) to the following equation:

$$u_i(s) = F(pr_{i,j}, p\hat{r}_{u,v}) = \min_{s=(s_1,\cdots,s_n), s_i \in S_i; s'=(s'_1,\cdots,s'_n), s'_i \in S_{i,j}} (u_{i,j}(s')) \tag{8}$$

Regarding equations (5) and (8), we draw a conclusion that the objective of generating optimal process plans of multiple jobs is to minimise the makespan of each job. Under this case, all jobs will choose their satisfied process plans to compete with one another to minimise their own makespans, while each one gets and tries to maximise makespans of opponents. Thus, how to balance profits to arrive at optimal results among all jobs according to correspondent constraints is becoming the bottleneck.

NE has been broadly considered and applied as an important pioneer solution for N-person non-cooperative games. An NE point is an N-tupple of

strategies, one for each player, such that anyone who deviates from it unilaterally cannot possibly improve its expected payoff (Kim 2003, Zhou *et al.* 2009). Considering the objective of presented game solution and definition of NE, we adopt NE to analyse the strategic behaviours of jobs in the presented game. As described in Equations (5) and (8), the payoff of J_i represents its makespan and the objective of each job is to minimise its own payoff. Therefore, the solution profiles for process plan decision sub-game and job scheduling sub-game are characterised by Equations (9) and (10), respectively.

$$u_i(s_i^{Nash}, s_{-i}^{Nash}) \leq u_i(s_i, s_{-i}^{Nash}) \tag{9}$$

for $i = 1, 2, \ldots, n$, $\forall s_i \in S_i$, and where $s_{-i}^{Nash} = (s_1^{Nash}, \ldots, s_{i-1}^{Nash}, s_{i+1}^{Nash}, \ldots, s_n^{Nash})$.

$$u_{i,j}(s_i'^{Nash}, s_{-i}'^{Nash}) \leq u_{i,j}(s_i', s_{-i}'^{Nash}) \tag{10}$$

for $i = 1, 2, \ldots, n, \forall s_i' \in S_{i,j}$ and where $s_{-i}'^{Nash} = (s_1'^{Nash}, \ldots, s_{i-1}'^{Nash}, s_{i+1}'^{Nash}, \ldots, s_n'^{Nash})$.

According to the presented solution, to generate the optimal process plans for multiple jobs is translated to search for the NEs of the two kinds of sub-games. In order to efficiently and effectively look for the NE points of the two kinds of sub-games, we develop a kind of HAGA to establish a two-level nested solution algorithm whose running mechanism will be set forth in detail in the next section.

4. HAGA-based solution algorithm for the game

As described in the above section, to deal with this problem is to obtain the NE points of the above two kinds of sub-games. When the NE point of process plan decision sub-game is found, the optimal process plan for each job can be acquired. Therefore, in this study, a two-level nested solution algorithm is designed for seeking the NE points based on genetic algorithm (GA). Considering the defect of easy local convergence of traditional GA, we develop the HAGA to improve global search capability by introducing hill-climbing search technique, and designing adaptive crossover and mutation operators. In order to find their own NE points of two kinds of sub-games, the presented solution algorithm is divided into two levels named process plan decision level and job scheduling level. One HAGA is designed and utilised in the process plan decision level to find the NE point for process plan decision sub-game, while a set of HAGAs are designed and run in parallel to find NE points for job scheduling sub-games whose number depends on the population size defined in HAGA in process plan decision level. The HAGAs contained in the two levels collaboratively work together to find the NE points for these two kinds of sub-games. Figure 3 shows the running procedure of proposed HAGA-based two-level nested solution algorithm in which the collaborative working mechanism among HAGAs in the two levels is specifically illustrated.

4.1. Chromosome design

In GA, each chromosome contained in a population represents one feasible solution for the complicated problem to be resolved. The chromosome is a carrier of genes and consists of the whole necessary information of genes. According to the characteristics of presented game and HAGA-based two-level nested solution algorithm, two kinds of encoding schemes for process plan decision level and job scheduling level are designed, respectively.

For individuals in process plan decision populations, Figure 4(a) shows the process-plan-based encoding method proposed. In this encoding method, the chromosome is characterised by a string and represents feasible process plans for all jobs as a sequential list. It should be noted that each character (gene) in the chromosome represents one feasible process plan and is only responsible for one and only one job. For example, if the ith element value in chromosome is j, it represents $pr_{i,j}$ of J_i. In this way, because the number of jobs is n, the length of chromosome is n. For individuals in job scheduling populations, the encoding method for chromosomes is illustrated in Figure 4(b). The chromosome is composed by a series of machines (genes) characterised by a string that is responsible for machine assignments. The string is divided into n substrings for n jobs as a sequential list following feasible process plans provided by the individual in process plan decision populations. Each substring represents a list of feasible machines related to sequential operations for completion of a job. For Table 3 in Experiment 1 (section 5), Figure 4 illustrates an example of the encoding schemes for the two levels in the HAGAs.

4.2. Fitness functions design

The fitness value provides the criteria to decide whether an individual will survive in next generation in GA. Therefore, a suitable fitness function must be presented that can satisfactorily reflect the fitness values of individuals in populations.

A great difference between traditional GA and HAGA is the process followed to evaluate the fitness of individuals. As described above, the objective of presented game solution is to seek the NE point at which each job thinks it has obtained the minimal makespan and if deviating from the NE point, it

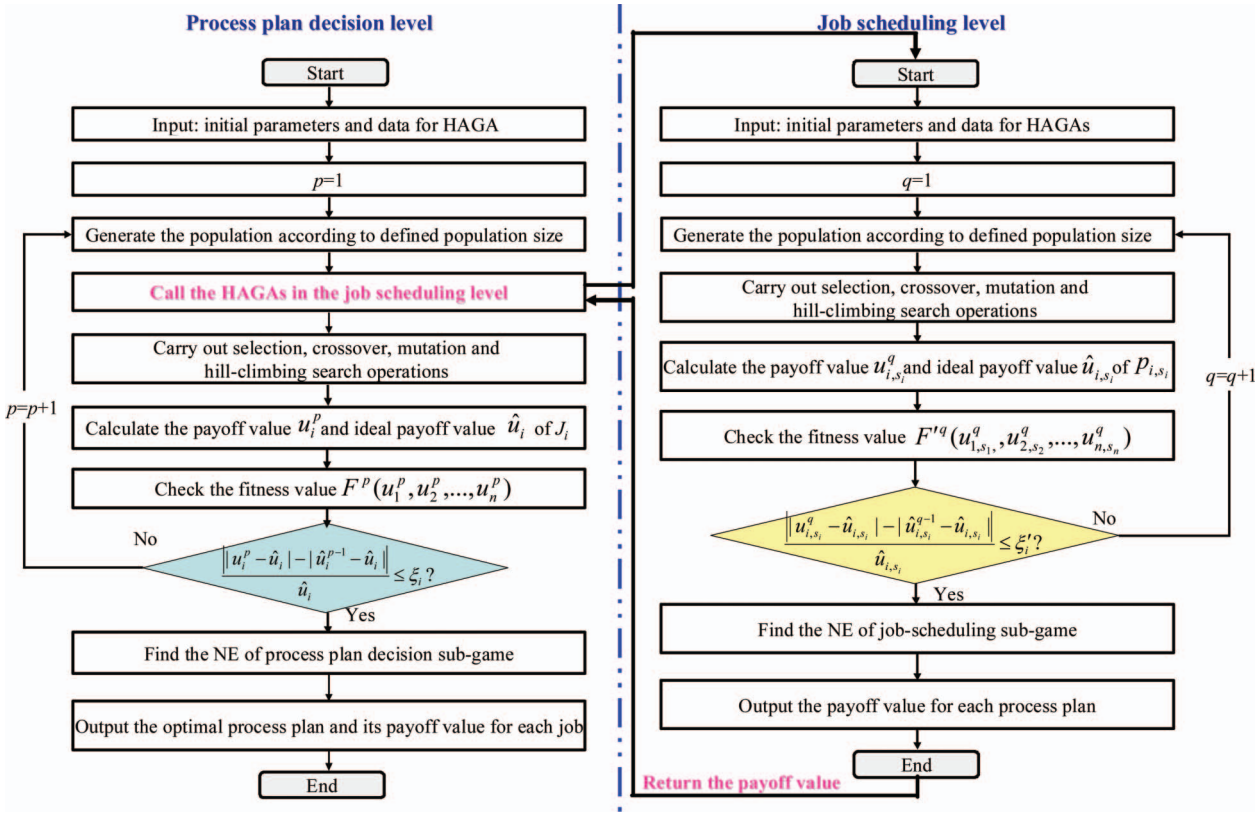

Figure 3. Running procedure of the solution algorithm for the game.

cannot possibly improve its expected makespan. In other words, jobs arrive at the status of profit equilibrium. However, because the traditional methods for generating process plans of multiple jobs do not take into account the competition factor existing in different jobs, the proposed fitness functions of their GA-based solutions do no longer satisfy the new requirements of presented HAGAs. That is why we need to design and develop new fitness functions for HAGAs.

For satisfying formula (9), we design the fitness function for HAGA in process plan decision level as Equation (11).

$$F^{\mathrm{p}}(u_1^{\mathrm{p}}, u_2^{\mathrm{p}}, \cdots, u_{\mathrm{n}}^{\mathrm{p}}) = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (u_{\mathrm{i}}^{\mathrm{p}} - \hat{u}_{\mathrm{i}})^2} \qquad (11)$$

Where $u_{\mathrm{i}}^{\mathrm{p}}$ represents the payoff value of J_i at p^{th} generation, and $\hat{u}_{\mathrm{i}}$ represents the perfect payoff value of J_i which is defined as

$$\hat{u}_{\mathrm{i}} = \min_{1 \le \mathrm{j} \le R_{\mathrm{i}}} \left(\min \left(\sum_{\mathrm{k}=1}^{\mathrm{JN_{i,j}}} pt_{\mathrm{i,j}}^{\mathrm{k}}(f_{\mathrm{l}}) + \sum tt_{\mathrm{i,j}}(f_{\mathrm{i}}, f_{\mathrm{i}'}) \right) \right) \qquad (12)$$

Equation (12) represents that perfect payoff value of J_{i} is its minimal makespan, which is derived from the assumption that only J_{i} exists in whole networked manufacturing environment to be manufactured.

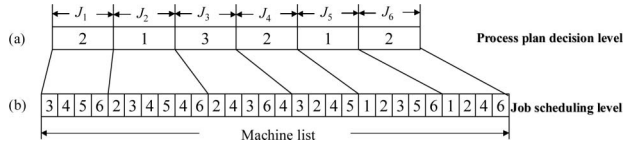

Figure 4. The encoding schemes for chromosomes. (a) Two-point crossover scheme for HAGA in the process plan decision level. (b) Two-point crossover scheme for HAGAs in the job scheduling level.

We also define that if the following constraint is satisfied, the presented fitness function can guarantee to find an NE point for process plan decision sub-game.

$$\frac{\left| |u_{\mathrm{i}}^{\mathrm{p}} - \hat{u}_{\mathrm{i}}| - |\hat{u}_{\mathrm{i}}^{\mathrm{p}-1} - \hat{u}_{\mathrm{i}}| \right|}{\hat{u}_{\mathrm{i}}} \le \xi_{\mathrm{i}} \qquad (13)$$

where $\hat{u}_{\mathrm{i}}^{\mathrm{p}-1}$ represents the best payoff value of J_{i} at the $(p-1)^{\mathrm{th}}$ generation, which is defined as follows:

$$\hat{u}_{\mathrm{i}}^{\mathrm{p}-1} = \arg\min \sum_{\mathrm{i}=1}^{\mathrm{n}} u_{\mathrm{i}}^{\mathrm{p}-1} \tag{14}$$

where $u_1^{\mathrm{p}-1}$, $u_2^{\mathrm{p}-1}$, ... , and $u_{\mathrm{n}}^{\mathrm{p}-1}$ are derived from the same individual in the $(p-1)^{\mathrm{th}}$ generation.

In formula (13), ξ_{i} is a threshold value used to decide the condition of finding the NE point. To simplify a computational complexity of the problem, we assure that $\xi_1 = \xi_2 = \cdots = \xi_{\mathrm{i}} = \cdots = \xi_{\mathrm{n}} = \xi$. The value of ξ has been given as an input parameter before the algorithm execution.

Considering formula (10), we throw out the fitness function for HAGAs in job scheduling level as the following equation:

$$F'^{\mathrm{q}}(u_{1,\mathrm{s}_1}^{\mathrm{q}}, u_{2,\mathrm{s}_2}^{\mathrm{q}}, \cdots, u_{\mathrm{n},\mathrm{s}_{\mathrm{n}}}^{\mathrm{q}}) = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (u_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}^{\mathrm{q}} - \hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}})^2} \tag{15}$$

Where $u_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}^{\mathrm{q}}$ represents payoff value of J_{i} at qth generation while $\hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}$ represents perfect payoff value of J_{i} under the selected process plan decided by s_{i}. Here, $\hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}$ is calculated by the following:

$$\hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}} = \min\left(\sum_{\mathrm{k}=1}^{\mathrm{JN}_{\mathrm{i},\mathrm{j}}} pt_{\mathrm{i},\mathrm{j}}^{\mathrm{k}}(f_{\mathrm{l}}) + \sum tt_{\mathrm{i},\mathrm{j}}(f_{\mathrm{l}}, f_{\mathrm{l}'})\right) \tag{16}$$

Equation (16) also stands for that the perfect payoff value of J_{i} is its minimal makespan, which is derived from the assumption that only J_{i} exists in whole networked manufacturing environment to be manufactured following one of its feasible process plans provided by s_{i}. We also denote that if the following constraint is satisfied, NE points for job scheduling sub-games are obtained.

$$\frac{\left|\left|u_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}^{\mathrm{q}} - \hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}\right| - \left|\hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}^{\mathrm{q}-1} - \hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}\right|\right|}{\hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}} \leq \xi_{\mathrm{i}}' \tag{17}$$

where $\hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}^{\mathrm{q}-1}$ represents the best payoff value of J_{i} at the $(q-1)^{\mathrm{th}}$ generation which is defined as follows:

$$\hat{u}_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}^{\mathrm{q}-1} = \mathrm{argmin} \sum_{i=1}^{n} u_{\mathrm{i},\mathrm{s}_{\mathrm{i}}}^{\mathrm{q}-1} \tag{18}$$

where $u_{1,\mathrm{s}_1}^{\mathrm{q}-1}$, $u_{2,\mathrm{s}_2}^{\mathrm{q}-1}$, ... , and $u_{\mathrm{n},\mathrm{s}_{\mathrm{n}}}^{\mathrm{q}-1}$ are derived from the same individual in the $(q-1)^{\mathrm{th}}$ generation.

In formula (17), the meaning of ξ_{i}' is the same to that of ξ_{i} utilised to guarantee to find NE points for job scheduling sub-games. Meanwhile, we also denote that $\xi_1' = \xi_2' = \cdots = \xi_i' = \cdots = \xi_{\mathrm{n}}' = \xi'$ for simplifying the computational complexity.

4.3. Genetic operations

The genetic operators used in the HAGA-based two-level nested solution algorithm mainly include selection, crossover, mutation and hill-climbing search, whose design schemes and detailed work logic are specified as follows.

4.3.1. Selection

The selection operator is used to decide the individuals that survive or not in the parent population and are passed to offspring. In this study, we adopt a roulette wheel method to realise selection of suitable individuals in the two-level HAGAs. The roulette wheel method is a direct proportion-based selecting method that is based on probability of fitness value of each individual to decide whether it survives or not. However, Equations (11) and (15) indicate that individuals having smaller fitness values will have greater chances to survive. Therefore, in order to satisfy the specified requirement, we must transform the fitness functions described in Equations (11) and (15) to new Equations (19) and (20), respectively for selection stage before adopting the roulette wheel method.

$$\bar{F}^{\mathrm{p}}(u_1^{\mathrm{p}}, u_2^{\mathrm{p}}, \cdots, u_{\mathrm{n}}^{\mathrm{p}}) = \frac{1}{F^{\mathrm{p}}(u_1^{\mathrm{p}}, u_2^{\mathrm{p}}, \cdots, u_{\mathrm{n}}^{\mathrm{p}})} \tag{19}$$

$$\bar{F}'^{\mathrm{q}}(u_{1,\mathrm{s}_1}^{\mathrm{q}}, u_{2,\mathrm{s}_2}^{\mathrm{q}}, \cdots, u_{\mathrm{n},\mathrm{s}_{\mathrm{n}}}^{\mathrm{q}}) = \frac{1}{F'^{\mathrm{q}}(u_{1,\mathrm{s}_1}^{\mathrm{q}}, u_{2,\mathrm{s}_2}^{\mathrm{q}}, \cdots, u_{\mathrm{n},\mathrm{s}_{\mathrm{n}}}^{\mathrm{q}})} \tag{20}$$

On the basis of these two equations, the individuals having smaller fitness values will survive in current population and become the eligible offspring.

4.3.2. Crossover

Crossover is a mechanism for diversification. This diversification mechanism enables GA to examine uninvolved regions and generate solutions that differ in various significant ways from those visited before. In the study, a two-point crossover scheme is adopted for recombining individuals in the two-level HAGAs whose work logics are illustrated in Figure 5. Two crossing points are randomly chosen with a probability in two selected chromosomes and each chromosome is divided into three substrings. The middle substrings of the two chromosomes are exchanged with each other to produce two offspring dynamically. The crossing points are marked with '▼' in Figure 5. For guaranteeing the high

performance of HAGAs, suitable value of crossover probability P_c plays an important role in crossover operation. The larger the P_c value, the sooner the new individuals are created, but excellent individuals may be destroyed; on the contrary, the smaller P_c value, the slower the searching process. Therefore, in order to enhance performance of HAGAs, an adaptive computing model for P_c is designed, which is described in Equation (19) to dynamically produce the adaptive P_c values for different individuals for surviving.

$$P_c = \begin{cases} \frac{(P_{c1}+P_{c0})}{2} + \frac{(P_{c1}-P_{c0})}{2}\cos\frac{F'-F_{avg}}{F_{max}-F_{avg}}\pi & F' \geq F_{avg} \\ \frac{(P_{c1}+P_{c2})}{2} + \frac{(P_{c1}-P_{c2})}{2}\cos\frac{F_{avg}-F'}{F_{avg}-F_{min}}\pi & F' < F_{avg} \end{cases} \quad (21)$$

Where

P_{c0}, P_{c1} and P_{c2}: constants defined as input parameters for HAGAs and $P_{c2} < P_{c1} < P_{c0}$;
F_{max}: the largest fitness value in a population;
F_{avg}: the average fitness value in a population;
F_{min}: the smallest fitness value in a population;
F': the larger fitness value derived from two individuals for crossover;

From Equation (19) we deduce that if $F' = F_{max}$, $P_c = P_{c0}$, which stands for the individual having the largest fitness value will be crossed to produce the new individual by P_{c0}. Conversely, if $F' = F_{min}$, $P_c = P_{c2}$, which stands for the individual having the smallest fitness value will be protected and will be inherited by the next generation. Based on the presented two-point crossover scheme, feasible offspring can be created without violating the feasibility of parents. With Table 3, an example of the two-point crossover schemes for two-level HAGAs are illustrated in Figure 5(a) and (b), respectively.

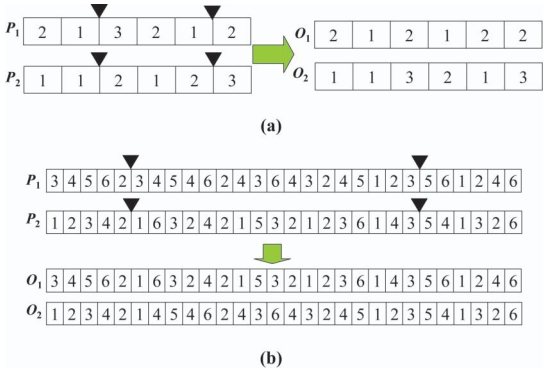

Figure 5. Two-point crossover scheme for the two-level HAGAs. (a) two-point crossover scheme for HAGA in the process plan decision level. (b) two-point crossover scheme for HAGAs in the job scheduling level.

4.3.3. *Mutation*

For mutation, some individuals are randomly selected with an individual mutation rate. In each selected individual, the gene for mutation is also randomly selected. In order to keep the offspring feasible in two-level HAGAs, we propose a mutating scheme for two-level HAGAs that is illustrated in Figure 6.

As the same as P_c, mutation probability P_m is also one of key factors for influencing the performance of GA. The larger the P_m value, the closer to the random search the algorithm; conversely, the smaller P_m value, the more difficult to produce new individuals. Therefore, we propose an adaptive computing model to dynamically produce P_m values for different individuals for surviving. Equation (20) describes the proposed adaptive computing model for P_m.

$$P_m = \begin{cases} \frac{(P_{m1}+P_{m0})}{2} + \frac{(P_{m1}-P_{m0})}{2}\cos\frac{F-F_{avg}}{F_{max}-F_{avg}}\pi & F \geq F_{avg} \\ \frac{(P_{m1}+P_{m2})}{2} + \frac{(P_{m1}-P_{m2})}{2}\cos\frac{F_{avg}-F}{F_{avg}-F_{min}}\pi & F < F_{avg} \end{cases} \quad (22)$$

Where P_{m0}, P_{m1}, P_{m2} are constants defined as input parameters for HAGAs and $P_{m2} < P_{m1} < P_{m0}$, F represents fitness value of an individual for mutation, and F_{max}, F_{min} and F_{avg} are the same as those defined in Equation (19).

From Equation (20), the following implications are deduced: if $F' = F_{max}$, $P_m = P_{m0}$ indicates that the individual possessing the largest fitness value has a high probability P_{m0} to carry out mutation operation to generate a new individual; if $F' = F_{min}$, $P_m = P_{m2}$ indicates that the individual possessing the smallest fitness value has a low probability P_{m2} to carry out a mutation operation and will have a high chance to be protected to the next generation.

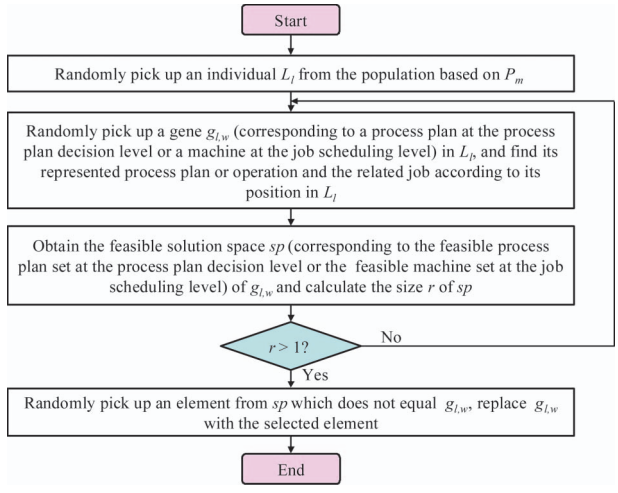

Figure 6. The mutation scheme for the two-level HAGAs.

4.3.4. Hill-climbing search

In order to further enhance the global convergence capability of two-level HAGAs, we develop a hill-climbing search method to produce better individuals in population. Because of adopting the neighbourhood search technique, the hill-climbing search method has strong capability of local search with high search efficiency. In the study, we utilise the developed hill-climbing search method to obtain the best individual after selection, crossover and mutation operations in the current generation and hope to produce a new better individual for the next generation by carrying out hill-climbing operation. Figure 7 illustrates the working mechanism and procedure of developed hill-climbing search method.

5. Experimental design and numerical results

In this section, we present two experimental case studies to demonstrate how the proposed game can be used as a solution for presented problem and how developed HAGA-based two-level nested solution algorithm can be adopted for resolving the game. Tables 1 and 2 provide the initial parameters for the two-level HAGAs and the transportation time between machines for the two experimental case studies.

5.1. Experiment 1

In the experiment, the problem is constructed with 6 jobs submitted by different customers and 6 machines geographically distributed in a networked manufacturing environment. Each job has one or more feasible process plans in which each operation has one or more alternative machines (named 6 × 6 problem). The objective of the problem is to gain the minimum makespan for each job (Equations (1), (9) and (10)). Table 3 shows the detailed initial data for the problem.

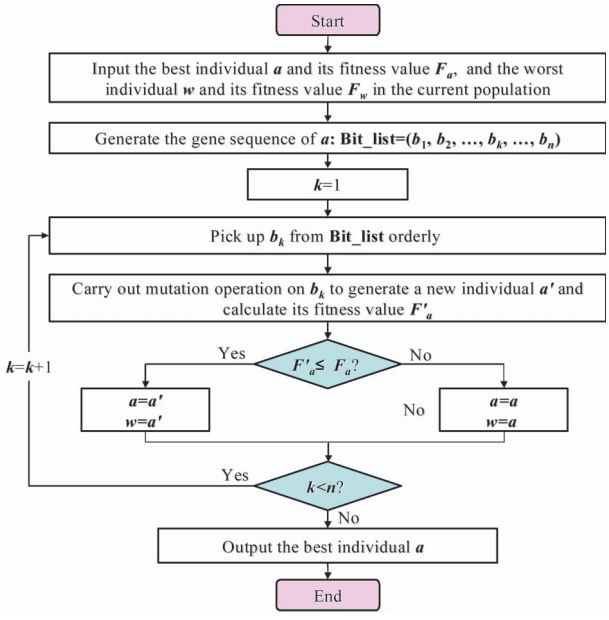

Figure 7. The working procedure of hill-climbing search method.

In Table 3, the cell contains two groups of data: one encircled by a pair of rounded parentheses represents candidate machines for operations and another encircled by a pair of quadrate parentheses that stands for process times on correspondent candidate machines. For example, cell[J_1][$pr_{1,2}$][O_2] contains (2, 4) and [4, 5], which indicates that operation $O^2_{1,2}$ can be processed on machine 2 or 4 and the correspondent process time is 4 or 5 time units.

Figure 8 illustrates the results in the form of a Gantt chart derived from process plan decision level for the experiment. From the Gantt charts, the process plan for each job and operations allocation on machines, as well as the times when operations start and finish their execution can be obtained visually. Figure 8(a) illustrates the initial results of process plans for the jobs at the 1st generation while Figure 8(b) illustrates that the solution algorithm finds a NE point at the 52nd generation for the problem.

Table 1. The initial parameters for the two-level HAGAs.

Parameter name	Parameter value			
	Process plan decision level		Job scheduling level	
Population	30		50	
Crossover probability P_c	P_{c0}	0.9	P_{c0}	0.85
	P_{c1}	0.8	P_{c1}	0.8
	P_{c2}	0.6	P_{c2}	0.65
Mutation probability P_m	P_{m0}	0.025	P_{m0}	0.015
	P_{m1}	0.010	P_{m1}	0.008
	P_{m2}	0.005	P_{m2}	0.001
Terminating value	ξ	0.01	ξ'	0.01
Max generation	300		500	

Table 2. Transportation time between machines.

Machine	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	0	1	3	2	3	2	3	4
f_2	1	0	2	2	3	3	5	4
f_3	3	2	0	2	3	3	5	4
f_4	2	2	2	0	2	3	2	4
f_5	3	3	3	2	0	2	2	3
f_6	2	3	3	3	2	0	1	2
f_7	3	5	5	2	2	1	0	1
f_8	4	4	4	4	3	2	1	0

Table 4 describes payoff values (makespans) and strategy profiles in process plan decision sub-game and job scheduling sub-games for all jobs at the 1st generation and the 52nd generation (NE point), respectively, for the problem that are derived from Figure 8. From Table 4, we can conclude that when the NE point is obtained at the 52nd generation in process plan decision sub-game, the optimal process plan for each job can be gained whose payoff value is notably better than that obtained at the 1st generation. For

Table 3. Initial data for the 6 × 6 problem.

Job	Process plan	Operation					
		O_1	O_2	O_3	O_4	O_5	O_6
J_1	$pr_{1,1}$	(1,2) [6,5]	(3,4,5) [7,6,6]	(6) [8]			
	$pr_{1,2}$	(1,3) [4,5]	(2,4) [4,5]	(3,5) [5,6]	(4,5,6) [5 ,5,4]		
J_2	$pr_{2,1}$	(2) [4]	(1,3) [2,3]	(2,4,6) [4,3,5]	(3,5) [2,4]	(2,4) [3,4]	(4,6) [3,5]
J_3	$pr_{3,1}$	(2,3) [5,6]	(1,4) [6,5]	(2,5) [5,6]	(3,6) [6,5]		
	$pr_{3,2}$	(1) [9]	(3,4) [8,8]	(5) [9]			
	$pr_{3,3}$	(2,3) [7,6]	(4) [7]	(3,5) [4,6]	(4,6) [5,5]	(2,4) [6,4]	
J_4	$pr_{4,1}$	(1,2) [7,8]	(3,4) [7,6]	(6) [9]			
	$pr_{4,2}$	(1,3) [4,3]	(2) [4]	(3,4) [4,5]	(5,6) [3,5]		
J_5	$pr_{5,1}$	(1) [3]	(2,4) [4,5]	(3) [4]	(5) [3]	(4,6) [5,4]	
	$pr_{5,2}$	(2,4) [5,6]	(5) [7]	(3,6) [9,8]			
J_6	$pr_{6,1}$	(1,2) [3,4]	(3,4) [4,3]	(2,5) [5,3]	(3) [4]	(4,5) [4,6]	(3,6) [5,4]
	$pr_{6,2}$	(1,3) [4,4]	(2,3) [5,6]	(2,4) [6,7]	(6) [7]		
	$pr_{6,3}$	(1,2,3) [3,5,8]	(4,5) [7,10]	(3,6) [9,9]			

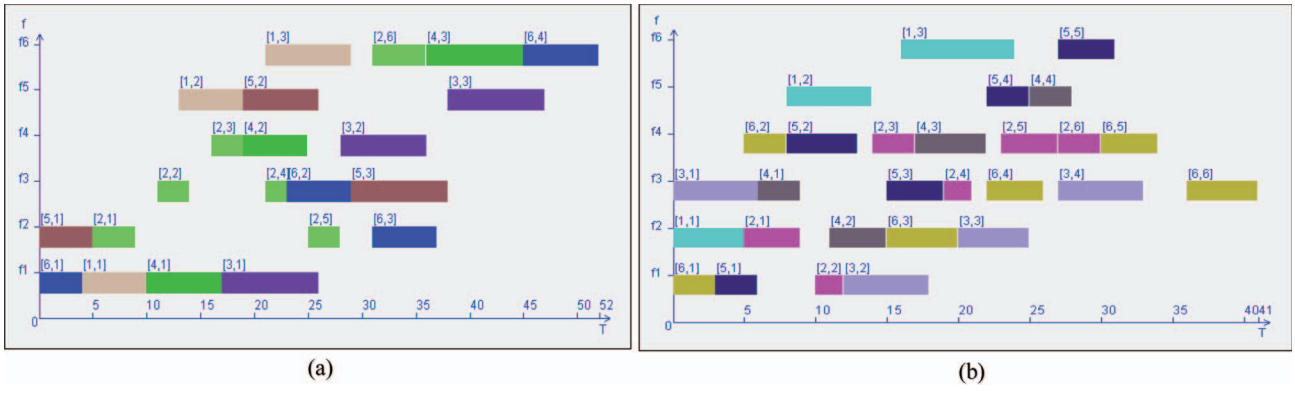

Figure 8. Gantt charts for optimal process plan decision of 6 × 6 problem. (a) Gantt chart for the 1st generation. (b) Gantt chart for the 80th generation. [*i*,*j*]: *i* represents job ID, *j* represents operation ID.

Table 4. Payoff values and feasible strategy profiles for 6 × 6 problem.

Job	1st generation			52nd generation (NE point)		
	Payoff value	Strategy profile		Payoff value	Strategy profile	
		Process plan decision sub-game	Job scheduling sub-games		Process plan decision sub-game	Job scheduling sub-games
J_1	29	$pr_{1,1}$	(1,5,6)	24	$pr_{1,1}$	(2,5,6)
J_2	36	$pr_{2,1}$	(2,3,4,3,2,6)	30	$pr_{2,1}$	(2,1,4,3,4,4)
J_3	47	$pr_{3,2}$	(1,4,5)	33	$pr_{3,1}$	(3,1,2,3)
J_4	45	$pr_{4,1}$	(1,4,6)	28	$pr_{4,2}$	(3,2,4,5)
J_5	38	$pr_{5,2}$	(2,5,3)	31	$pr_{5,1}$	(1,4,3,5,6)
J_6	52	$pr_{6,2}$	(1,3,2,6)	41	$pr_{6,1}$	(1,4,2,3,4,3)

Table 5. Initial data for the 8 × 8 problem.

Job	Process Plan	Operation							
		O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
J_1	$pr_{1,1}$	(1,2,7,8) [6,5,5,4]	(3,4,5) [7,6,6]	(6) [8]					
	$pr_{1,2}$	(1,3) [4,5]	(2,4) [4,5]	(3,5) [5,6]	(4,5,6,7,8) [5,5,4,5,9]				
J_2	$pr_{2,1}$	(2,7) [4,8]	(1,3,8) [2,3,8]	(2,4,6) [4,3,5]	(3,5) [2,4]	(2,4) [3,4]	(4,6) [3,5]	(1,3,5) [5,5,8]	(4) [2]
	$pr_{2,2}$	(1,3,5) [1,5,7]	(4,8) [5,4]	(4,6) [1,6]	(4,7,8) [4,4,7]	(4,6) [1,2]	(1,6,8) [5,6,4]	(4) [4]	
J_3	$pr_{3,1}$	(2,3) [5,6]	(1,4,7,8) [6,5,4,7]	(2,5,7) [5,6,5]	(3,6) [6,5]	(1,6) [6,6]	(5) [4]		
	$pr_{3,2}$	(1,8) [7,8]	(3,4,7) [8,8,11]	(5,8) [9,8]					
	$pr_{3,3}$	(2,3,8) [7,6,8]	(4,8) [7,7]	(3,5,7) [7,8,7]	(4,6) [7,8]	(1,2) [1,4]			
J_4	$pr_{4,1}$	(1,2,7,8) [7,8,7,8]	(3,4) [7,6]	(6) [9]	(1,7) [5,7]				
	$pr_{4,2}$	(1,3,5) [4,3,7]	(2,8) [4,4]	(3,4,6,7) [4,5,6,7]	(5,6,8) [3,5,5]				
J_5	$pr_{5,1}$	(1) [3]	(2,4) [4,5]	(3,8) [4,4]	(5,6,8) [3,3,3]	(4,6) [5,4]			
	$pr_{5,2}$	(2,4,7,8) [5,6,5,5]	(5) [7]	(3,6) [9,8]					
J_6	$pr_{6,1}$	(1,2) [3,4]	(3,4) [4,3]	(2,5,7,8) [5,3,5,4]	(3) [4]	(4,5) [4,6]	(3,6) [5,4]		
	$pr_{6,2}$	(1,3) [4,4]	(2,3) [5,6]	(2,4,7,8) [6,7,5,6]	(6) [7]				
	$pr_{6,3}$	(1,2,3) [3,5,6]	(4,5) [7,8]	(3,6) [9,8]					
J_7	$pr_{7,1}$	(1,2,3) [3,4,5]	(4,5,8) [4,5,5]	(6) [4]	(5) [5]	(3,6) [4,6]			
	$pr_{7,2}$	(3,5,6) [4,5,4]	(4,7,8) [4,6,4]	(1,3,4,5) [3,3,4,5]	(4,6,8) [4,6,5]	(1,3) [3,3]			
J_8	$pr_{8,1}$	(1,2,6) [3,4,4]	(4,5,8) [6,5,4]	(3,7) [4,5]	(4,6) [4,6]	(7,8) [1,2]			
	$pr_{8,2}$	(2,6) [5,6]	(4) [7]	(4,6,8) [7,6,8]					

example, for J_4, $u_4 = u_{4,1} = 45$, $s_4 = pr_{4,1}$, and $s'_4 = (1, 4, 6)$ at the 1st generation while $u_4^{\text{Nash}} = u_{4,2}^{\text{Nash}} = 28$, $s_4^{\text{Nash}} = pr_{4,2}$ and $s_4'^{\text{Nash}} = (3, 2, 4, 5)$ at the 52nd generation.

5.2. *Experiment 2*

In order to validate the feasibility of proposed game and solution algorithm, we further design another experimental case study. In this experiment, the problem is constructed with 8 jobs and 8 machines. Each job has one or more feasible process plans in which each operation contains one or more alternative machines (named 8 × 8 problem). Gaining the minimum makespan for each job is used as the objective. Table 5 shows detailed initial data for the problem.

Figure 9 shows the Gantt charts of this problem. Figure 9(a) shows the initial results of process plans for all jobs at 1st generation while Figure 9(b) indicates the optimal process plans for all jobs at 80th generation where a NE point is found for the problem.

Table 6 shows payoff values (makespans) and strategy profiles for the two kinds of sub-games for all jobs at the 1st generation and the 80th generation (NE point), respectively, for the problem. According to Table 6, the NE point is found at the 80th generation in process plan decision sub-game. The optimal process plan for each job can be gained whose payoff value is notably better than that obtained at the 1st generation.

6. Discussion

As described above, the problem presented in the study possesses several new characteristics that are somewhat different than the traditional ones listed in the

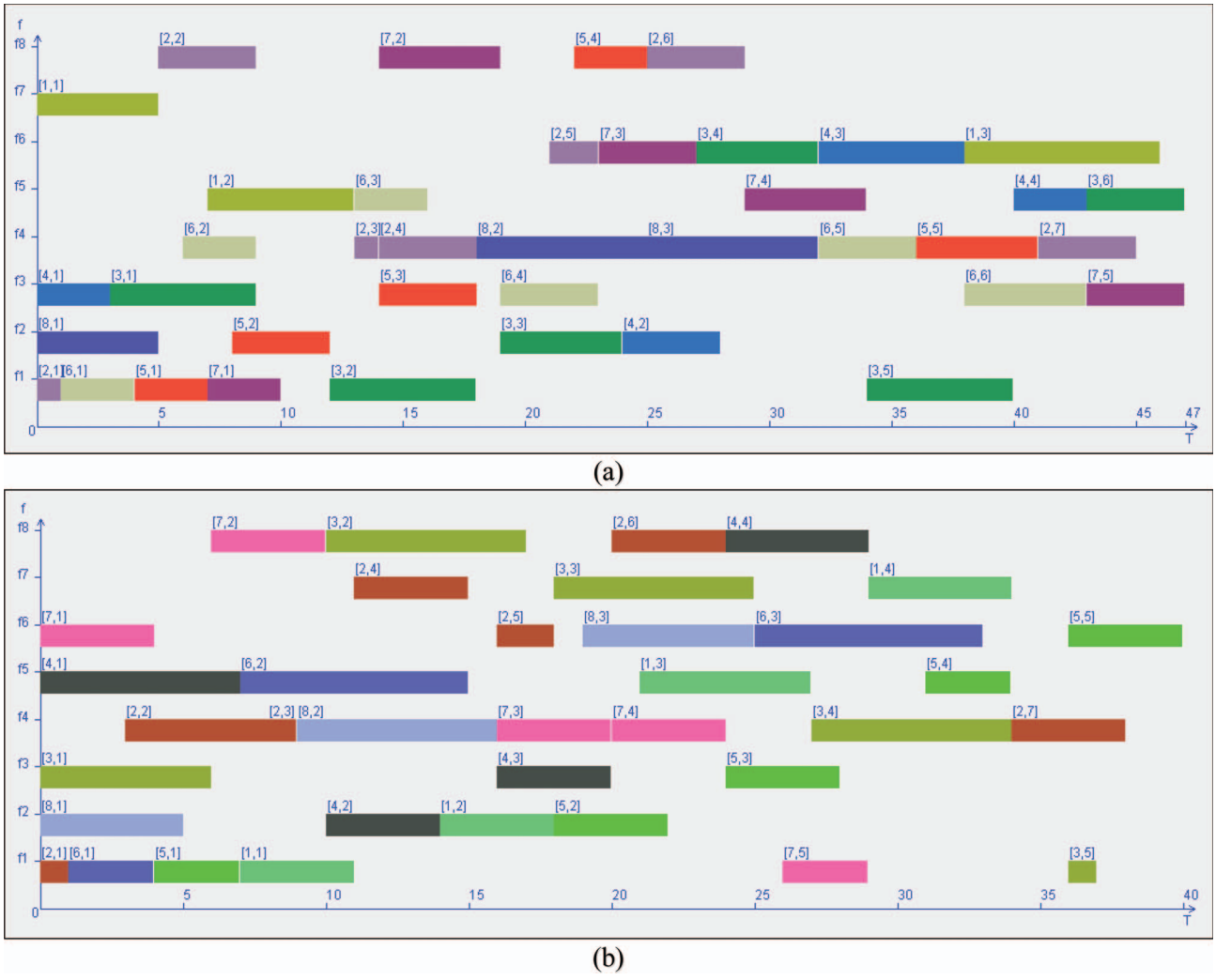

Figure 9. Gantt charts for optimal process plan decision of 8 × 8 problem. (a) Gantt chart for the 1st generation. (b) Gantt chart for the 80th generation. [*i*,*j*]: *i* represents job ID, *j* represents operation ID.

Table 6. Payoff values and feasible strategy profiles for 8 × 8 problem.

	1st generation			80th generation (NE point)		
		Strategy profile			Strategy profile	
	Payoff value	Process plan decision sub-game	Job scheduling sub-game	Payoff value	Process plan decision sub-game	Job scheduling sub-game
J_1	46	$pr_{1,1}$	(7,5,6)	34	$pr_{1,2}$	(1,2,5,7)
J_2	45	$pr_{2,2}$	(1,8,4,4,6,8,4)	38	$pr_{2,2}$	(1,4,4,7,6,8,4)
J_3,	47	$pr_{3,1}$	(3,1,2,6,1,5)	37	$pr_{3,3}$	(3,8,7,4)
J_4	43	$pr_{4,2}$	(3,2,6,5)	29	$pr_{4,2}$	(5,2,3,8)
J_5	41	$pr_{5,1}$	(1,2,3,8,4)	40	$pr_{5,1}$	(1,2,3,4,5)
J_6	43	$pr_{6,1}$	(1,4,5,3,4,3)	33	$pr_{6,3}$	(1,5,6)
J_7	47	$pr_{7,1}$	(1,8,6,5,3)	29	$pr_{7,2}$	(6,8,4,4,1)
J_8	32	$pr_{8,1}$	(2,4,4)	25	$pr_{8,1}$	(2,4,6)

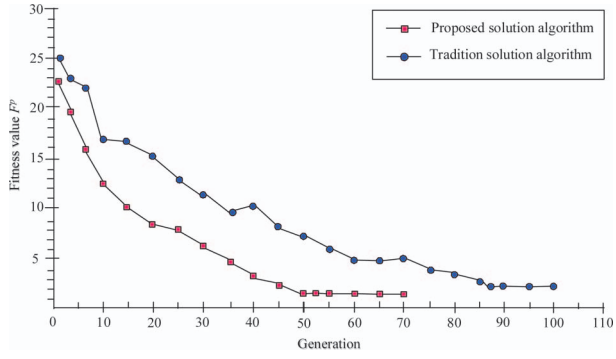

Figure 10. Convergence curves of proposed solution algorithm and traditional GA-based one.

references. Therefore, a new mathematical model and effective solution algorithm are inevitably required. Overall, the experimental results indicate that the proposed approach is an effective and acceptable approach for the problem for the following reasons: (1) the proposed approach considers all the new characteristics, especially the competition relationship between jobs of process planning and scheduling synthetically in networked manufacturing by adopting game theory, and (2) compared with the traditional GA-based solution algorithm, the proposed solution algorithm obtains better results and higher solving efficiency. Figure 10 shows the convergence curves of proposed solution algorithm and the traditional GA-based solution algorithm (Zhou *et al.* 2009) for the proposed 6 × 6 problem, which means that the proposed solution algorithm has more efficiency and better probability to obtain the best results.

7. Conclusions

In this study, we made several contributions to research literature with respect to generating optimal process plans of multiple jobs based on game theory in networked manufacturing environment. Foremost, we identified the characteristics of deciding process plans in networked manufacturing environment from those in traditional manufacturing environment. More importantly, we defined the conceptual model for generating optimal process plans in networked manufacturing based on its new characteristics that act as the reference for building game solution at next step.

As another contribution, we formulated a model for deciding optimal process plans for the jobs coming from different customers through adopting game theory. This game model was divided into two kinds of sub-games, i.e. process plan decision sub-game and job scheduling sub-games, which cooperate with each other to decide optimal process plan for every job. Owing to the problem's complexity and intractability, considering effectiveness and efficiency, we proposed a HAGA-based two-level nested solution algorithm to find the NE point of the game. The computational results show that the presented game solution is suitable for requirements of generating optimal process plans for multiple jobs in networked manufacturing environment, and the proposed solution algorithm is capable to obtain optimal results for the test problems.

The new approach presented in this study makes increasing the efficiency of networked manufacturing systems possible. An important further research issue is to extend the proposed approach in order that it may be applied to more practical and networked manufacturing systems. The increased use of this approach will most likely pave a new way for improving the developments in future networked manufacturing systems.

Acknowledgements

Gratitude is extended to the National Natural Science Foundation of China (Grant No.: 50605050) and Ministry of Education for New Century Excellent Talent Support Program of 2007 (NCET-07-0681) for the financial supports.

Special thanks also to Mr. Xuefeng Tian and Miss. Rui Wang for the programming efforts in implementing and demonstrating the presented game solution.

References

Brandimarte, P. and Calderini, M., 1995. A hierarchical bicriterion approach to integrated process plan selection and job shop scheduling. *International Journal of Production Research*, 33, 161–181.

Huang, S.H., Zhang, H.C., and Smith, M.L., 1995. A progressive approach for the integration of process planning and scheduling. *IIE Transactions*, 27, 465–464.

Jiang, P.Y., *et al*., 2007. e2-MES: an e-service-driven networked manufacturing platform. *International Journal of Computer Integrated Manufacturing*, 20, 127–142.

Kim, B.H., 2003. A new game-theoretic framework for maintenance strategy analysis. *IEEE Transactions on Power Systems*, 18, 698–706.

Kim, K.H. and Egbelu, P.J., 1999. Scheduling in a production environment with multiple process plans per job. *International Journal of Production Research*, 37, 2725–2753.

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

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, 80–95.

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

Moon, C., Kim, J., and Hur, S., 2002. Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. *Computers and Industrial Engineering*, 43, 331–349.

Saygin, C. and Kilic, S.E., 1999. Integrating flexible process plans with scheduling in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 15, 268–280.

Seredyński, F., 1997. Competitive coevolutionary multi-agent systems: the application to mapping and scheduling problems. *Journal of Parallel and Distributed Computing*, 47, 39–57.

Seredyński, F., 1998. Distributed scheduling using simple learning machines. *European Journal of Operational Research*, 107, 401–413.

Seredyński, F., Koronacki, J., and Janikow, C.Z., 2001. Distributed multiprocessor scheduling with decomposed optimization criterion. *Future Generation Computer Systems*, 17, 387–396.

Shafaei, R. and Brunn, P., 2000. Workshop scheduling using practical (inaccurate) data. Part 3: a framework to integrate job releasing, routing and scheduling functions to create a robust predictive schedule. *International Journal of Production Research*, 38, 85–99.

Shao, X.Y., *et al*., 2009. Integration of process planning and scheduling – A modified genetic algorithm-based approach. *Computers and Operations Research*, 36, 2082–2096.

Zhou, G.H., Jiang, P.Y., and Huang, G.Q., 2009. A game-theory approach for job scheduling in networked manufacturing. *International Journal of Advanced Manufacturing Technology*, 41, 972–985.