Int J Adv Manuf Technol (2009) 41:972–985
DOI 10.1007/s00170-008-1539-9

ORIGINAL ARTICLE

A game-theory approach for job scheduling in networked manufacturing

Guanghui Zhou · Pingyu Jiang · George Q. Huang

Received: 18 August 2006 / Accepted: 21 April 2008 / Published online: 3 June 2008

Abstract This paper presents a new kind of scheduling solution for jobs in networked manufacturing environments. The main contributions of this study can be focused on three points: The first is to distinguish the concepts and requirements of job scheduling in the networked manufacturing environment form those in the traditional manufacturing environment. The second is to construct a game-theory mathematical model to deal with this new job scheduling problem. In this presented mathematical model, this new job scheduling problem is formulated as an N-person non-cooperative game with complete information. The players correspond to the jobs submitted, respectively, by related customers and the payoff of each job is defined as its makespan. Each player has a set of strategies which correspond to the feasible geographical distributive machines. Therefore, obtaining the optimal scheduling results is determined by the Nash equilibrium (NE) point of this game. In order to find the NE point, the last point is to design and develop a genetic algorithm (GA)-based solution algorithm to effectively solve this mathematical model. Finally, a numerical example is presented to demonstrate the feasibility of the approach.

G. Zhou · P. Jiang
State Key Laboratory for Manufacturing Systems Engineering,
Xi'an Jiaotong University,
Xi'an 710049, China

G. Zhou (✉) · P. Jiang
School of Mechanical Engineering, Xi'an Jiaotong University,
Xi'an 710049, China
e-mail: ghzhou@mail.xjtu.edu.cn

G. Q. Huang
Department of Industrial and Manufacturing Systems
Engineering, The University of Hong Kong,
Hong Kong, China

Keywords Job scheduling · Networked manufacturing · Non-cooperative game · Nash equilibrium · Genetic algorithm

1 Introduction

It is now accepted that globalization is becoming a new trend for enterprises. The major factor that contributes to success in the global marketplace is the agile and rapid response speed to a customer's requirements. Therefore, the factor of quick delivery to market for products emerges as the crucial factor for enterprises wishing to survive in the extremely competitive global marketplace. To meet this requirement, several new manufacturing models have recently been proposed and networked manufacturing has become a pioneering method that is characterized by globalization, collaboration, customization, digitalization, and agility providing a collaborative environment for customers, manufacturers and suppliers to work together [1, 2]. In the networked manufacturing environment, job scheduling plays an important role that guarantees that suitable jobs be allocated and manufactured on suitable geographically distributive machines belonging to different shops or enterprises to achieving the respective objectives regarding the correspondent dynamic constraints.

Generally speaking, job scheduling involves scheduling, sequencing, and routing of jobs on various machines. The traditional job scheduling problem is always limited inside a shop which considers n jobs arrived at the shop at a certain points of time and schedule them on m machines regarding the technological constraints while achieving some relevant criteria such as minimal makespan, minimal mean flow time and minimal cost. However, in networked manufacturing, the job scheduling problem is somewhat

different from the traditional one. It extends the concept of traditional job scheduling and possesses some new network-oriented characteristics described as follows:

(1) It is a customer-centric job scheduling method. Because the jobs come from different customers having competitive relationships, this type of job scheduling mainly focuses on satisfying the individual objective of each job. However, traditional job scheduling always considers the whole scheduling objective of all jobs under some manufacturing constraints.
(2) The machines are distributive, ranging from an intra-shop to inter-shop and even inter-enterprise, which means the job scheduling bound is no longer limited to a single shop but extends to geographically distributive shops and even different enterprises.

Considering these new characteristics, job scheduling in networked manufacturing can be described as a series of jobs submitted by different customers compete with each other to occupy the corresponding machines according to their own respective objectives, e.g., minimal makespan and finally arrive at satisfying their own requirements. Therefore, this job scheduling problem can be stated as follows.

There are a given number of jobs on order submitted by customers denoted as n and each job contains a series of sequential operations. There are also a given number of machines dispersed in different shops or enterprises denoted as m. We determine the job schedule for each job while optimizing a certain objective without violating restrictions imposed on the networked manufacturing environment. In this study, we mainly consider the minimal makespan as the objective for every job which decides the time of delivery to market of products. Before solving this job scheduling problem, the following assumptions are made:

(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 are scheduled to be processed on the same machine at the same time;
(4) The transportation time exists. After an operation of a job is processed on a machine, it is immediately transported to the next machine according to its own routing;
(5) All jobs can be simultaneously available at the time of zero.

According to the above definition and assumptions, the job scheduling in networked manufacturing is actually a collection of individuals (jobs) acting selfishly and interacting to fulfill their own goals. The challenge we face is therefore in how to generate a scheduling solution to produce an effective and efficient scheduling result for each job. Currently, because traditional job scheduling solutions cannot effectively deal with the proposed job scheduling problem, exploring the new strategy and solution is becoming the key point.

In order to deal with the presented job scheduling problem satisfactorily, a mathematical model is constructed and a game-theory scheduling model is developed. In this scheduling model, the job scheduling problem is described as an N-person non-cooperative with complete information. The players correspond to the jobs submitted, respectively, by related customers and the payoff of each job is defined as its makespan. Each player has a set of strategies which corresponds to the feasible geographical distributive machines. The optimal result for each job is derived from the Nash equilibrium (NE) point of the game. After constructing the job scheduling game, we utilize the genetic algorithm (GA) as a solution procedure to find the perfect NE point and finally arrive at obtaining the optimal result for each job.

The rest of this paper is organized as follows. Section 2 presents a review of literature related to the job scheduling and the application of game-theory solution in different complex systems. In Sect. 3, we formulate a mathematical model to represent the job scheduling problem using the N-person non-cooperation game and define it as the job scheduling game. In Sect. 4, we explain the use of GA to solve the proposed job scheduling game. We devote Sect. 5 to report and analyze the case simulation of this job scheduling game and discuss the implications for these results. Finally, some concluding remarks are drawn in Sect. 6.

2 Literature review

2.1 Job scheduling

The job scheduling problem has been known as an NP-complete problem for several decades. Great efforts have been made and a variety of solutions including scheduling model and resolving algorithms have emerged to solve this problem in recent years.

Sousa et al. proposed a new architecture for scheduling in manufacturing systems. In this architecture, a series of holonics were developed to represent the jobs and resources and the Contract Net Protocol was adopted to handle the alterative constraints to deal with the scheduling conflicts [3]. Persi et al. presented a hierarchic approach for production planning and scheduling in a flexible manufacturing cell. In this hierarchic model, sets of jobs which could be concurrently processed were determined at the upper level, and at the low levels, the operations related to correspondent jobs were sequenced, linked, and scheduled [4]. Gou et al. demonstrated a holonics manufacturing scheduling architecture where machines, cells, factories,

parts, operations, etc., were modeled as holonics [5]. Yang et al. and Song et al. introduced the Petri-net to build the scheduling model for job shop scheduling [6, 7]. Concerning the solving algorithms for job scheduling, Moursli et al. introduced the branch-and-bound algorithm to deal with the scheduling problem in a flow shop [8]. Sakawa et al. considered the imprecise or fuzzy nature of the data in real-world problems, and proposed an efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy due-date [9]. Mattfled et al. and Senthilkumar et al. adopted GA and heuristic algorithm to deal with the job shop scheduling problems. In their presented solutions, the heuristic reduction of the search space helped the GA to find better solutions in a shorter computation time [10, 11]. Engin et al. formulated a scheduling method based on the immune algorithm to solve job and stage hybrid flow-shop scheduling problems [12]. Rajendran et al. and Ying et al. utilized an ant colony algorithm to model and deal with the permutation flow-shop scheduling problems and also demonstrated that this algorithm has higher efficiency than other meta-heuristics known for the benchmark problem set of Taillard [13, 14]. Chandrasekaran et al. proposed a computational method of artificial immune system algorithm (AIS) for finding optimal makespan values of different size problems in job scheduling. The presented artificial immune system algorithm provided solutions for larger-sized problems with near-optimal solutions [15].

Naturally, in the above-described job scheduling problems, the research objects are relatively limited into one job-shop and the job scheduling problem addressed in this paper almost were not involved. For this reason, a game-theory solution is proposed to deal with this kind of job scheduling problem, which will be described in detail in Sect. 3.

2.2 Game-theory solution for scheduling

At present, game theory is widely utilized in dealing with the economical-related issues. For scheduling, it mainly focuses on the scheduling issues in parallel computing systems or power systems. Seredyński et al. paid great attention on scheduling tasks in parallel computers. After analyzing the natures of parallel computing systems such as distribution and cooperation, he adopted multi-agent technologies to construct a multi-agent system with game-theory models of interaction between players to represent the parallel computing systems. For this purpose, a model of non-cooperative N-person games with limited interaction was considered and proposed and the genetic algorithms and evolutionary programming were successfully used to obtain the NE point with high performance [16–18]. Kim et al. presented a game-theory framework for the maintenance scheduling of generating units (MSU) problem to analyze strategic behaviors of generating companies (Gencos) in competitive electricity markets. In his study, the MSU problem is formulated as a dynamic non-cooperative game with complete information [19].

In conclusion, the above literature falls into two categories. One is that the job scheduling activities always take place in one shop. However, in the networked manufacturing environment, the job assignment may take place in several distributed shops or even enterprises, which makes the job scheduling problem more complex and therefore the job scheduling methods and models described in the literature obviously can not satisfy this new requirement well. Moreover, the described job scheduling methods and models in the literature are shop-profit-driven and the objectives are to arrive at the whole optimal results for all jobs, while the job scheduling problem addressed in this paper is customer-driven and the objectives are to arrive at the local optimization for each job. Therefore, though the job scheduling methods and models described in the literature maybe have more optimization, they cannot deal with customer-driven scheduling problems well, especially when the customers are opponents of one another. The other is that though the game theory is gradually used to deal with scheduling problems, it is not introduced to realize the job scheduling problems in networked manufacturing environments because of the complexity of this kind of job scheduling problem. Opportunities exist in these two areas for further rigorous research.

3 Problem description

3.1 Notations

In order to formulate the job scheduling problem and interpret the correspondent results efficiently and effectively, we should introduce the notations that are listed in Table 1.

3.2 The formulation of job scheduling game

The job scheduling problem addressed in this paper can be taken as an N-person non-cooperative game with complete information where each job acting as a player decides the processing strategies (i.e., selection of suitable machines) to arrive at its goal. We designate this game as a job scheduling game, which is expressed as a tupple: $G = (N, S_i, u_i)$. Here, u_i depends on strategies not of J_i but of other jobs that are not known in advance.

In order to make this game well suited for the job scheduling problem, we have to first customize this game to satisfy the specified requirements. The structure of this job

Table 1 The notations

Parameters or acronyms	Definition
$J_i(0 \le i < n)$	Job i where n is the number of jobs for schedule
$G=(N,S_i, u_i)$	A tupple for representing an N-person non-cooperative game
N	Set of players corresponding to jobs
S_i	Strategy set of J_i
s_i	Strategy of J_i
u_i	Payoff of J_i
U_i	The payoff function of J_i
$O_{i,j}(0 \le j < JN_i)$	Operation j of J_i where JN_i is the number of sequential operations of J_i
$M=\{f_k \mid 0 \le k < m\}$	Set of alternative machines in networked manufacturing environment were f_k represents a feasible machine and m the number of alternative machines
$pt_{i,j}$ (f_k)	The processing time of $O_{i,j}$ of J_i on machine f_k
$tt_{i,j}(f_k, f_l)$	The transportation time of J_i between machine f_k and f_l
$st_{i,j}$, (f_k)	The starting time of $O_{i,j}$ of J_i processed on machine f_k
P_c	Crossover probability of GA
P_m	Mutation probability of GA
ζ	The threshold constant
Pop	The population of GA
Max_Gen	The defined maximum generation of GA
Pop_Size	The size of population

scheduling game and its corresponding variables are defined in detail as follows.

3.2.1 The structure of the job scheduling game

The structure of the job scheduling game is illustrated in Fig. 1. In this figure, the jobs correspond to players, and each of them includes a series of sequential operations. The distributive machines to the strategies of this game are denoted as strategy profile. According to its included operations, each job tries to select the optimal machines to process its operations and arrives at the goal of maximizing its own payoff. Considering the factor that product delivery time plays an important role in global market competition, we select the makespan as the payoff function for each job. The detailed definition and description for payoff function will be discussed in the following section.

3.2.2 Players

In the job scheduling game, players correspond to the jobs submitted by different customers. According to the defined notations in Table 1, we assumed that there exist n jobs and J_i contains JN_i number of sequential operations, therefore, we can define that $N=\{J_i \mid 0 \le i < n\}$ and $J_i=\{O_{i,j} \mid 0 \le j < JN_i\}$.

3.2.3 Strategies

Due to the selection of machines playing an important role in job scheduling, in this job scheduling game, we adopt the distributive alternative machines belonging to different shops or enterprises to act as the worthwhile strategies for jobs to choose. Each job tries to maximize its own profit by selecting the suitable machines.

As described in Table 1, let M denote the set of all alternative machines in the networked manufacturing environment and S_i represents the strategy set of J_i. Therefore, we can deduce that S_i is a subset of M actually and $M \equiv \bigcup_{i=1}^{n} S_i$.

In this job scheduling game, each job tries to schedule their operations on correspondent machines to be processed, on the one hand, to maximize its own profit, and on the other hand, to minimize the profits of its opponents through adopting the suitable strategies. Thus, it is of course noted that the profit of a job is affected not only by its own scheduling strategies but also by the strategies of its opponents.

3.2.4 Payoffs

The payoff represents the welfare of the jobs at the end of the game. They act as the references for which each job chooses

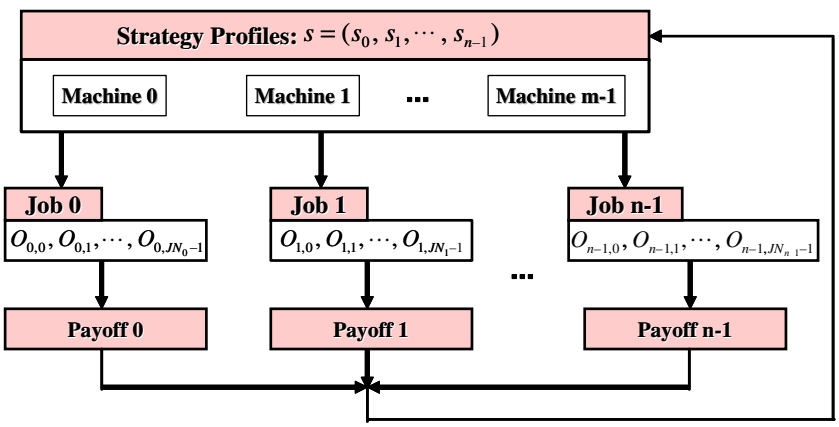


Fig. 1 Structure of the job scheduling game

its strategy. In this paper, the payoff of a job is defined as its makespan. As such, we define the payoff of J_i implicitly as a function of the strategy combination s described in (1).

$$U_i(s) = st_{i,JN_i}(f_k) + pt_{i,JN_i}(f_k) \tag{1}$$

where $s=(s_0, s_1, \ldots, s_{n-1})$, $s_i \in S_i$, $f_k \in S_i$ and subject to

$$st_{i,j+1}(f_k) - tt_{i,j+1}(f_k,f_l) \geq st_{i,j}(f_l) + pt_{i,j}(f_l) \tag{2}$$

where, $f_l \in S_i$, $k \neq l$ and

$$\begin{aligned} &st_{i,j}(f_k) + pt_{i,j}(f_k) \geq st_{x,y}(f_k) \\ &\text{or} \\ &st_{i,j}(f_k) \leq st_{x,y}(f_k) + pt_{x,y}(f_k) \end{aligned} \tag{3}$$

where $x=0,\ldots,n-1$; $y=0,\ldots, JN_x-1$; $i \neq x$; f; $f_k \in S_i \cap S_x$.

Equation (1) denotes that the payoff of J_i is the sum of the starting time and the processing time of its last operation O_{i,JN_i} processed on f_k. It implies that each job tries its best to minimize its own makespan by choosing the satisfied strategies. Inequality (2) describes the sequence constraint of operations which implies that two different operations of a job cannot be processed at the same time. Inequality (3) expresses the resource constraint, which means that each machine can only process one job (operation) at a time. In this job scheduling game, since all of the jobs want to choose the satisfied strategies to compete with one another to minimize their own makespans while each one gets and tries to maximize the makespans of the opponents, how to balance the profits to obtain the optimal scheduling results among all jobs is the bottleneck.

3.2.5 Solution

Nash equilibrium (NE) is broadly considered and applied as the solution for N-person non-cooperative games. An NE point is an N-tuple of strategies, one for each player, such that anyone who deviates from it unilaterally cannot possibly improve its expected payoff. In this study, we adopt NE to analyze the strategic behaviors of jobs in the job scheduling game.

As described in Eq. (1), the payoff of J_i represents its makespan and the objective of each job is to minimize its own payoff. Therefore, the solution profile $s^{Nash} = \left(s_0^{Nash}, s_1^{Nash}, \cdots, s_{n-1}^{Nash}\right)$ is characterized by (4)

$$U_i\left(s_i^{Nash}, s_{-i}^{Nash}\right) \leq U_i\left(s_i, s_{-1}^{Nash}\right),$$

for

$$i = 0, 1, \cdots, n-1, \forall s_i \in S_i, \tag{4}$$

where

$$s_{-1}^{Nash} = \left(s_0^{Nash}, s_0^{Nash}, \ldots, s_{i-1}^{Nash}, s_{i+1}^{Nash}, \ldots, s_{n-1}^{Nash}\right).$$

According to the presented solution, to deal with this job scheduling problem is translated to search for the NE of the job scheduling game. In this paper, we utilize GA as a solution algorithm for looking for the NE point of this game whose working mechanisms will be described in detail in the next section.

4 GA-based solution algorithm

As described in the above section, to deal with the job scheduling problem is translated to find the NE point of the job scheduling game, and therefore, in this paper, we proposed a solution algorithm based on genetic algorithm (GA) which can be used to find the NE point of this job scheduling game. Figure 2 summarizes the GA-based solution procedure of the job scheduling game.

4.1 Representation scheme

Creating the evolution objects plays an important role in solving practical problems by using GA. It becomes much more important how we transform the practical scheduling problems of job scheduling into "gene", "chromosome", and "population", which are the basic elements of GA.

In order to set up the GA-based solution model for the job scheduling game, the first thing for us to do is to design the suitable genes. In this study, the genes are classified into three different categories: genes for jobs, genes for machines, and genes for transportation time between two machines. Here, we adopt a series of characters including

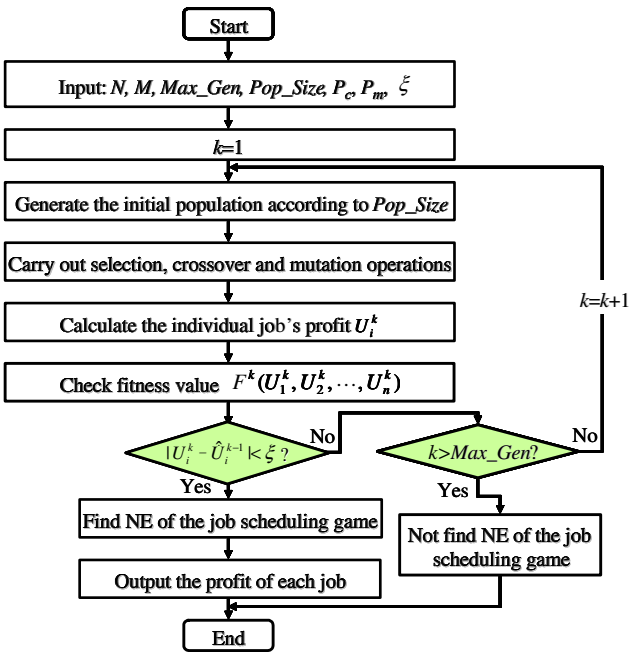


Fig. 2 Solution procedure of the job scheduling game

"0", …, "9", and "," to represent these genes. Through assorting these characters to form a character string, the related genes can be coded well. For example, the coding format for a job can be specified by the following:

$(''i'', ''j'', ''f'', ''st'', ''pt'')$,

where *"i"* represents the job ID, *"j"* represents the operation ID of the job *"i"*, *"f"* represents the machine ID related to operation *"j"*, *"st"* represents the starting processing time of operation *"j"* on machine *"f"*, and *"pt"* represents the processing time of operation *"j"*on machine *"f"*. For example, ("2","3","1","8","6") describes that operation "3" of job "2" is processed on machine "1", and the starting processing time is eight time units and processing time is six time units.

In the same way, we also can code the machines as the follows:

$(''f'', ''e'')$,

where *"f"* represents the machine ID and *"e"* represents the shop or enterprise ID where *"f"* belongs to.

In addition, the transportation time between two different machines can be described as follows:

$(''f'', ''f''', ''tt'')$,

where *"f"* and *"f'"* denote the IDs of different machines and *"tt"* represents the transportation time between these two machines.

On the basis of the presented gene-describing method, we can introduce the gene definition for this job scheduling game, and the overall information for jobs, machines, and transportation time can be expressed clearly.

4.2 Chromosome design

In the GA-based job scheduling game, each chromosome represents one feasible solution for all jobs which is a carrier of the genes and consists of the whole necessary information of the genes. In order to construct a suitable chromosome, the following rules should be considered seriously.

(1) The chromosome encoding scheme should be simple so as to improve the rate of evolution operations;
(2) The chromosome encoding scheme should be convenient for the design and operations of the genetic operators;
(3) The chromosome encoding scheme has to keep its solution validity even if it is transformed because of selection, crossover, and mutation, etc.

Concerning the above rules of encoding schemes, we propose a machine-based encoding method to construct the chromosomes. Figure 3 illustrates the designing scheme of the chromosome which is composed of a series of machines (genes) responsible for machine assignments. This chromosome is characterized by a string that is divided into n substrings whose sequence follows the jobs sequence ranging from J_0 to J_{n-1}. Each substring represents a list of machines related to the sequential operations of a job as a sequential list and the k^{th} gene of the chromosome represents the machine on which the correspondent operation is processed. Furthermore, we can obtain the length of the chromosome as $L_{chro} = \sum_{i=0}^{n-1} JN_i$.

4.3 Evaluation

As we know, the design of a fitness function is the key point in GA. Through calculating their fitness values, all the individuals in a population can be evaluated and the criteria can be provided for genetic operations at each generation. Therefore, now the important thing we have to do is to design a suitable fitness function for the job scheduling game.

As described in Eq. (1), the payoff of each job is defined as its makespan and the objective of each job is to try its best to acquire the minimal makespan and on the contrary to try to maximize the makespans of its opponents. Consequently, the presented fitness function should meet the described requirement. Equation (5) is the presented fitness function

$$F^k\left(U_0^k, U_1^k, \cdots, U_{n-1}^k\right) = \sum_{i=0}^{n-1} \left| U_i^k - \widehat{U}_i^{k-1} \right|,$$

for

$$i = 0, 1, \cdots, n-1; 1 \le k \le Max\_Gen \tag{5}$$

where U_i^k represents the payoff value, i.e., the makespan J_i of at the k^{th} generation, and $\widehat{U}_i^{k-1}$ represents the best payoff value of J_i at the $(k\text{-}1)^{th}$ generation, which is defined as follows:

$$\widehat{U}_i^{k-1} = \arg\ \min \sum_{i=0}^{n-1} U_i^{k-1} \tag{6}$$

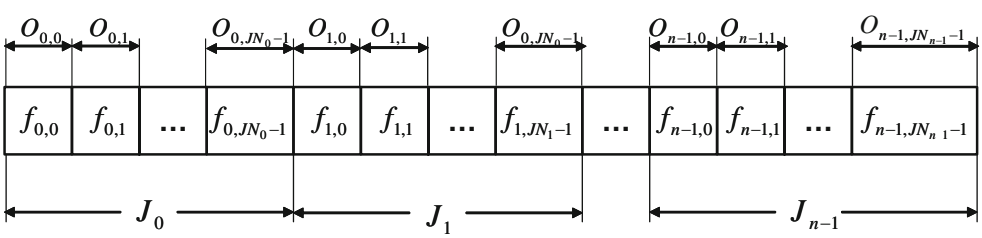


Fig. 3 The machine-based encoding for chromosome

where, U_0^{k-1}, U_1^{k-1},..., and U_{n-1}^{k-1} are derived from the same individual in the $(k-1)^{th}$ generation.

We also define that if the following constraints are satisfied, this fitness function can guarantee to find an NE point.

$$\left|U_i^k - \widehat{U}_i^{k-1}\right| < \xi_i \tag{7}$$

where ξ_i is a threshold value that is used to decide the condition of finding the NE point. To simplify a computational complexity of the problem, we will assure that

$$\xi_0 = \xi_1 = \cdots = \xi_k = \cdots = \xi_{n-1} = \xi \tag{8}$$

The value has been given as an input parameter before the algorithm execution.

4.4 Decoding

Besides the above influencing factors such as input parameters, representation scheme, chromosome design, and fitness function, the results of a job scheduling problem also rely on its priority rules. Each time a machine becomes idle, a job will be selected among those which are waiting to be processed. This decision is

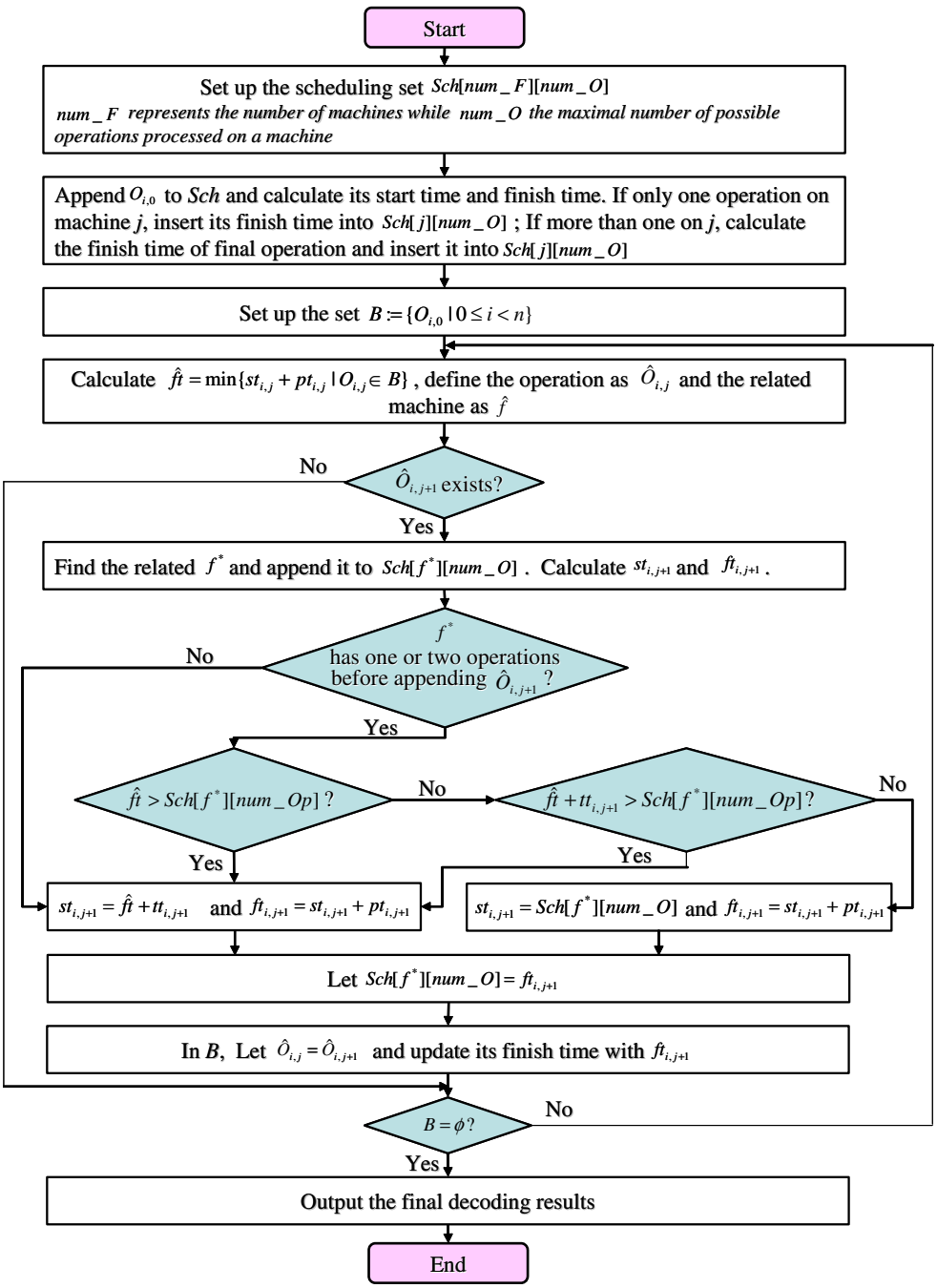


Fig. 4 The decoding procedure for the FCFS rule

made according to the priority rules and different priority rules leading to different scheduling results. Therefore, in this GA-based job scheduling game, the priority rules act as the constraints combined with the constraint (2) and (3) described in sub-section 3.2.4 to jointly decide the correspondent NE point. For simplicity, we only consider two typical rules which are known to be able to reduce the makespan of a job effectively in the following situations:

(1) **FCFS.** The first-come, first-served rule attempts to deal with unbiased scheduling problems, which means that when a job arrives at a machine, it will be appended at the tail of the queue and processed by the order of queue. This rule neglects properties of jobs and the state of machines.

(2) **SPT.** The shortest processing time rule attempts to give priority to those operations having the shortest processing time. The SPT rule can alleviate the idle time of machines through reducing the length of the queue in the fastest possible way.

Concerning these two priority rules, we design two different decoding methods able to satisfy the scheduling

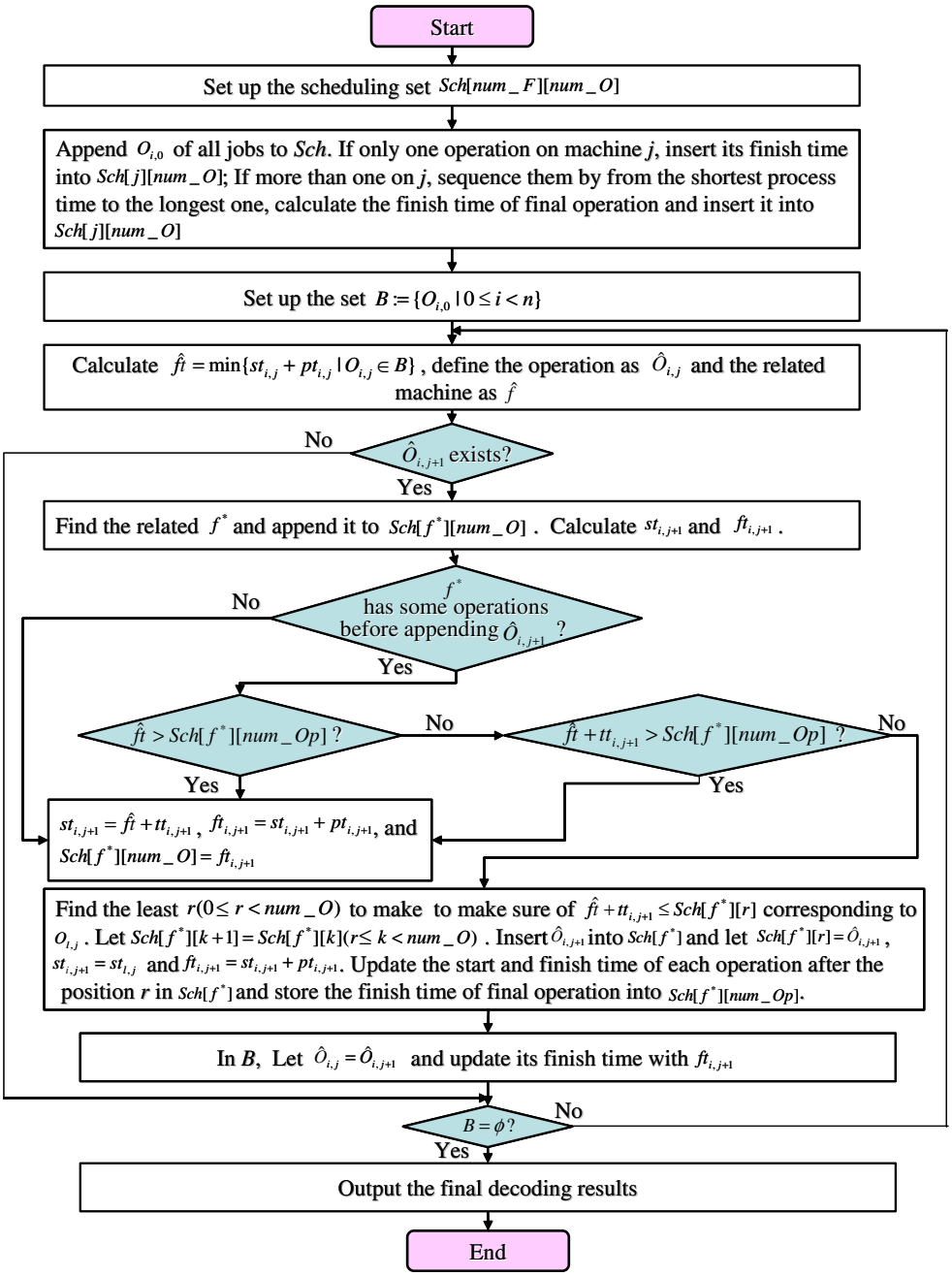


Fig. 5 The decoding procedure for the SPT rule

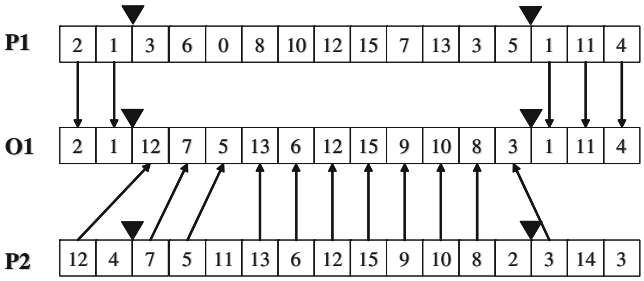


Fig. 6 Two-point crossover scheme

requirements. For the FCFS rule, its decoding procedure is illustrated in Fig. 4 and the SPT rule is illustrated in Fig. 5.

4.5 Genetic operations

The genetic operators used in this GA-based job scheduling game mainly include selection, crossover, and mutation, whose design schemes and detailed work logic are specified as follows:

Selection operator is used to decide the individuals' "survival or not" in the parent population and are passed to the next population. In this study, we adopt a roulette wheel method to realize the selection of suitable individuals. As we know, the roulette wheel method is a direct proportion-based selecting method that is based on probability of fitness value of each individual to determine its survival. In other words, the larger the value of probability of an individual, the greater chance of its survival. However, Eq. (5) indicates that individuals having smaller fitness values will obtain greater chances to survive. It is obviously opposite to the roulette wheel method. Therefore, before adopting the roulette wheel method to deal with selection decisions, we have to transform the fitness function described in Eq. (5) into the following new equation:

$$\widetilde{F}^k\left(U_0^k, U_1^k, \cdots, U_{n-1}^k\right) = \frac{1}{F^k\left(U_0^k, U_1^k, \cdots U_{n-1}^k\right)} \tag{9}$$

According to the above equation, the individuals who have the larger fitness values will survive in this population and become the eligible offspring.

For crossover, a variation of the two-point crossover scheme is adopted for recombination of individuals in this job scheduling game. The two-point crossover is modified to generate a feasible solution that satisfies the precedence constraints and avoids duplication or omission of operations. Two cut points are chosen randomly in the two selected individuals, and the elements before the first cut point and after the second cut point in one parent (P1) are passed on to the same positions as in the offspring (O1). These elements are removed from the other parent individual (P2) and the remnant elements are copied into the undetermined positions (i.e., positions between two cut points) in the offspring in the same order they appear in P2. Another offspring can be produced by switching the roles of P1 and P2. An example of the presented crossover scheme is illustrated in Fig. 6 in which cut points are marked with "▼". The crossover operator produces feasible sequences since both parents are feasible and offspring are produced without violating the feasibility of the parents.

For the mutation, some individuals are randomly selected with an individual mutation rate. In each selected individual, the gene for mutation is also selected randomly. In order to keep the offspring feasible, we propose a mutating scheme whose working procedure is illustrated in Fig. 7.

5 Experimental design and numerical results

To demonstrate how the proposed job scheduling game can be used as a solution for the presented kind of job scheduling problem and how the proposed GA-based mathematical solution can be adopted to solve the job scheduling game, we present an experimental study and report some initial results which are specified as follows.

5.1 Initial data

In this experimental study, we assume that there exist six jobs submitted by different customers and each job possesses six sequential operations. We also assume that there exist six machines related to these six jobs that are geographically distributive. Table 2 lists the initial data of the jobs and their correspondent strategy profile (candidate machines).

Each cell in the table contains two groups of data: one encircled by a pair of rounded parentheses represents the candidate machines for operations while the other encircled by a pair of quadrate parentheses represents the process

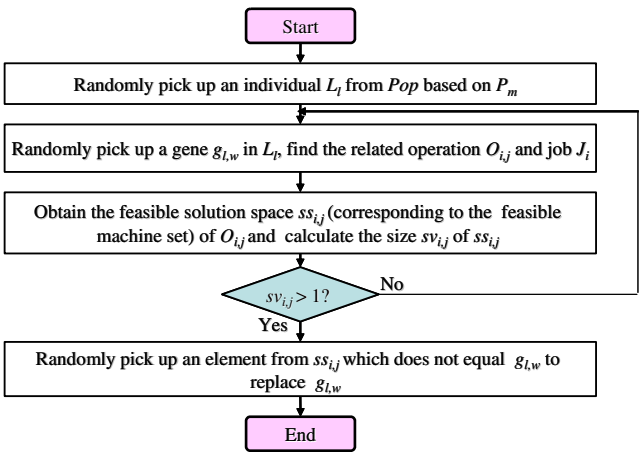


Fig. 7 The feasible mutation scheme

Table 2 The initial data of jobs

Job	Operation					
	O_0	O_1	O_2	O_3	O_4	O_5
J_0	(0, 2) [4, 6]	(1, 3, 5) [7, 5, 7]	(4, 0) [5, 4]	(1, 2) [7, 4]	(3, 4, 5) [3, 4, 5]	(1, 5) [5, 6]
J_1	(1) [4]	(0, 2) [2, 6]	(1, 3, 5) [4, 8, 5]	(2, 4) [7, 4]	(1,2, 3) [3, 4, 6]	(0, 5) [4, 5]
J_2	(1, 4) [8, 6]	(0) [5]	(2, 3, 5) [4, 6, 7]	(0, 5) [5, 5]	(4, 3) [6, 7]	(0, 1, 5) [8, 4, 6]
J_3	(0, 3, 4) [3, 5, 5]	(1) [4]	(2, 4) [5, 7]	(3, 1) [5, 6]	(1, 2, 5) [3, 4, 5]	(2, 4) [6, 7]
J_4	(2, 3) [4, 4]	(1, 3, 4) [3, 6, 3]	(0, 3, 5) [5, 6, 6]	(4) [7]	(2, 3) [6, 4]	(4, 5) [8, 7]
J_5	(1, 2, 5) [3, 6, 7]	(0, 2) [5, 6]	(1, 3, 4) [4, 6, 7]	(2, 4) [5, 3]	(0, 1) [5, 5]	(3, 5) [3, 4]

times on correspondent candidate machines. For example, Cell [3] [2] contains (2, 4) and [5, 7], which means that $O_{3,2}$ can be processed on machine 2 or 4 and the correspondent process time is 5 or 7 time units. In addition, the transportation time of jobs between any two machines is one of the key factors affecting the results of this job scheduling. Thus, in this experimental study, we also initialize the transportation time described in Table 3.

5.2 Numerical results

According to the above-presented priority rules, i.e., FCFS and SPT, we design two kinds of experiments for the job scheduling game, respectively, to demonstrate the influences on finding the correspondent NE points. Before performing these two kinds of experiments, we have to give some initial parameters for this GA-based solution algorithm beforehand, which are specified in Table 4.

Figures 8 and 9 illustrate the scheduling results in the form of a Gantt chart by FCFS rule and STP rule, respectively, which describe the allocation of operations of jobs on machines and the times when operations start and finish their execution. Figure 8a illustrates the initial scheduling results at the 1st generation and Fig. 4b illustrates that the solution algorithm finds a NE point for the job scheduling game at the 120th generation and Fig. 8c shows the scheduling results at the 200th generation, which are the same as those in Fig. 8b. This means that after finding the NE point, this solution algorithm converges at this NE point and every job keeps from falling away from its strategy profile.

Table 3 Transportation time between two different machines

Machine	f_0	f_1	f_2	f_3	f_4	f_5
F_0	0	1	2	1	2	2
F_1	1	0	1	2	2	1
F_2	2	1	0	2	1	2
F_3	1	2	2	0	1	1
F_4	2	2	1	1	0	2
F_5	2	1	2	1	2	0

Figure 9a also illustrates the initial scheduling results at the 1st generation and Fig. 9b illustrates an NE point is achieved at the 100th generation while Fig. 8c described when the NE point is found, the solution algorithm converges at this point.

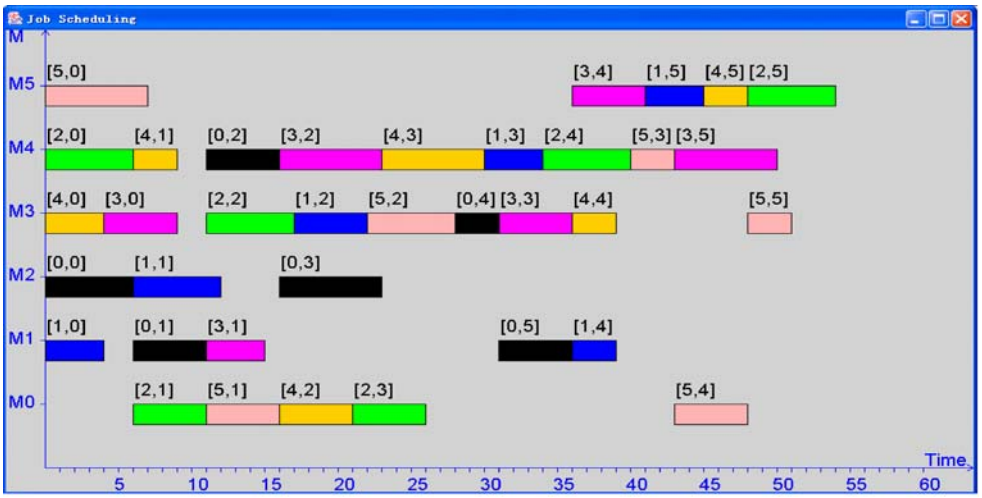
Regarding the FCFS rule, Table 5 lists the feasible strategy profiles at the 1st, 120th, and 200th generation, respectively, whilst Table 6 shows the correspondent payoff values for jobs related to these generations under the corresponding strategy profiles. Consulting Tables 5 and 6, we can draw a conclusion that when the NE point is obtained at the 120th generation, the strategy profiles and payoff values for jobs are no longer changed at the any succeeding generation.

Table 7 shows the strategy profiles of all jobs at the 1st, 100th, and 200th generation, respectively, following the SPT rule, and Table 8 shows the payoff values of jobs at these generations. According to the data derived from Tables 7 and 8, the conclusion that after the NE point is obtained at the 100th generation, the strategy profile and payoff values for all jobs are no longer changed is also drawn.

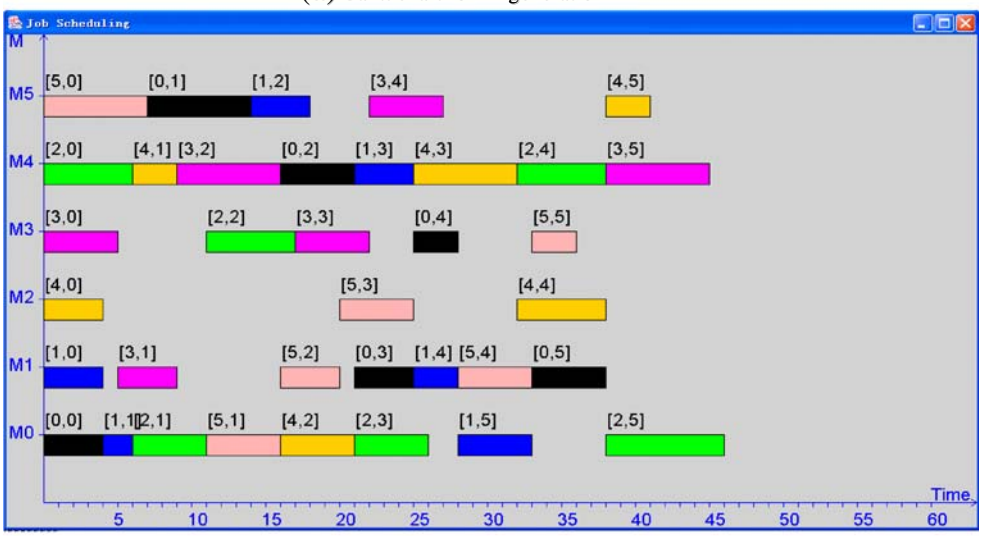
As discussed in Sect. 4.4, the priority rules are the important factors which act as the constraints to influence the job scheduling results. Consulting the above scheduling results by following FCFS and SPT priority rules, respectively, we notice that though the initial parameters for this job scheduling game are constant all the times, the NE point together with the strategy profile and payoff values of jobs is changed according to different priority rules. For example, for the FCFS rule, the makespan of J_2 is 33, while for the SPT rule, it is 40. It is demonstrated that the priority rules actually influence the NE of the presented job scheduling game.

Table 4 Initial parameters for GA-based solution algorithm

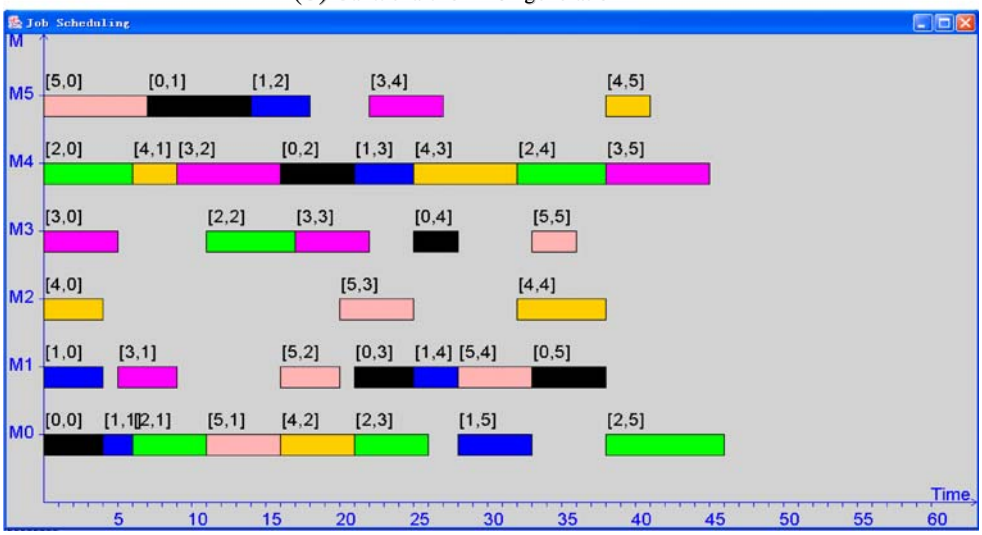
Parameter name	Parameter value
Population	50
Crossover probability P_c	0.8
Mutation probability P_m	0.02
Terminating value ζ	1.5
Max generations	2000



(a) Gantt chart for 1th generation



(b) Gantt chart for 120th generation

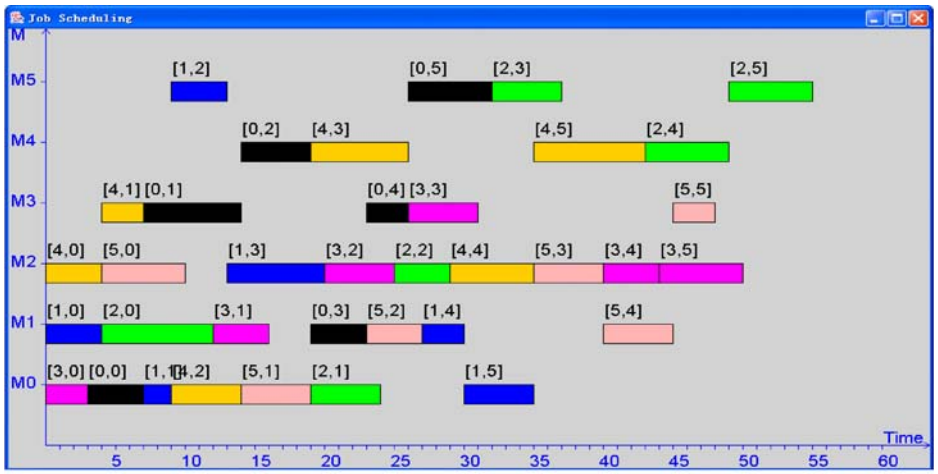


(c) Gantt chart for 200th generation

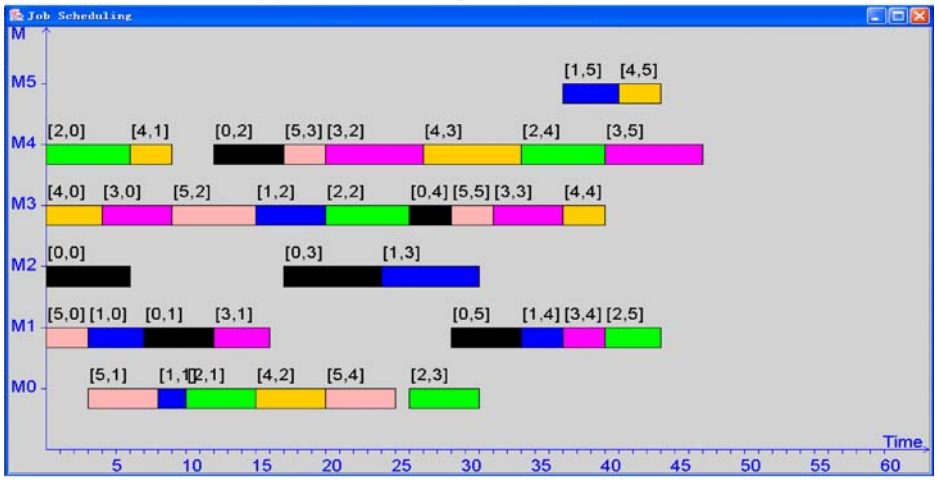
[*i*, *j*]: *i* represents job ID; *j* represent operation ID

Fig. 8 Gantt charts for FCFS rule in job scheduling game

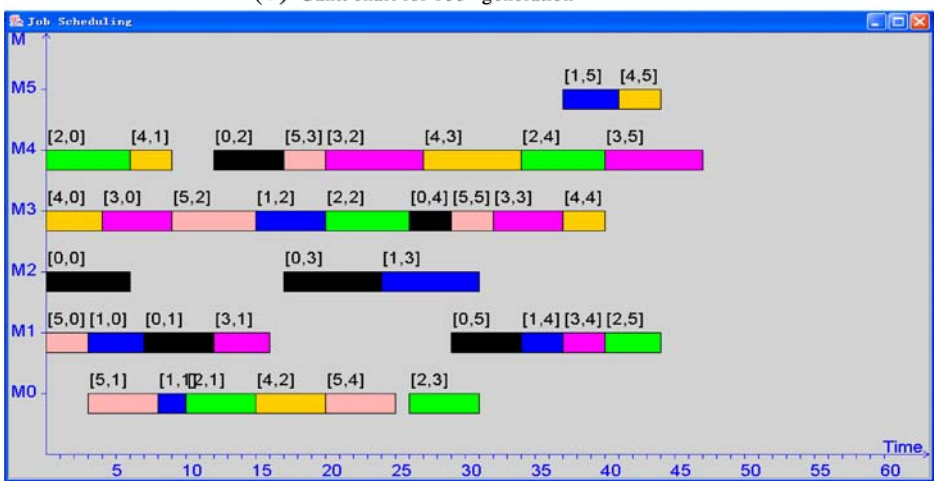
Fig. 9 Gantt chart for SPT rule in job scheduling game



(a) Gantt chart for 1th generation



(b) Gantt chart for 100th generation



(c) Gantt chart for 200th generation

[*i*, *j*]: *i* represents job ID; *j* represent operation ID

6 Concluding remarks

In this paper, we make several contributions to the research literature with respect to job scheduling based on game theory in the networked manufacturing environment. Foremost, we identify the characteristics of job scheduling in the networked manufacturing environment from those in the traditional manufacturing environment. More importantly, we also define the conceptual model for job scheduling in networked manufacturing based on its

Table 5 Feasible strategy profiles following FCFS rule

Generation	Strategy profile
1st	((2, 1, 4, 2, 3, 1), (1, 2, 3, 4, 1, 5), (4, 0, 3, 0, 4, 5), (3, 1, 4, 3, 5, 4), (3, 4, 0, 4, 3, 5), (5, 0, 3, 4, 0, 3))
120th	((0, 5, 4, 1, 3, 1), (1, 0, 5, 4, 1, 0), (4, 0, 3, 0, 4, 0), (3, 1, 4, 3, 5, 4), (2, 4, 0, 4, 2, 5), (5, 0, 1, 2, 1, 3))
200th	((0, 5, 4, 1, 3, 1), (1, 0, 5, 4, 1, 0), (4, 0, 3, 0, 4, 0), (3, 1, 4, 3, 5, 4), (2, 4, 0, 4, 2, 5), (5, 0, 1, 2, 1, 3))

Table 6 Payoff values (makespan) of jobs

Job	Payoff value (makespan)		
	1st generation	120th generation(NE point)	200th generation
J_0	36	38	38
J_1	45	33	33
J_2	54	46	46
J_3	50	45	45
J_4	48	41	41
J_5	51	36	36

Table 7 Feasible strategy profiles following SPT rule

Generation	Strategy profile
1st	((0, 3, 4, 1, 3, 5), (1, 0, 5, 2, 1, 0), (1, 0, 2, 5, 4, 5), (0, 1, 2, 3, 5, 2), (2, 4, 0, 4, 2, 4), (1, 0, 1, 2, 1, 3))
100th	((2, 1, 4, 2, 3, 1), (1, 0, 3, 2, 1, 5), (4, 0, 3, 0, 4, 1), (3, 1, 4, 3, 1, 4), (3, 4, 0, 4, 3, 5), (1, 0, 3, 4, 0, 3))
200th	((2, 1, 4, 2, 3, 1), (1, 0, 3, 2, 1, 5), (4, 0, 3, 0, 4, 1), (3, 1, 4, 3, 1, 4), (3, 4, 0, 4, 3, 5), (1, 0, 3, 4, 0, 3))

Table 8 Payoff values (makespan) of jobs

Job	Payoff value (Makespan)		
	1st generation	100th generation (NE point)	200th generation
J_0	36	35	35
J_1	35	40	40
J_2	65	45	45
J_3	51	46	46
J_4	53	43	43
J_5	43	32	32

characteristics which act as the reference for building the job scheduling game at the next step.

As another contribution, we formulate a job scheduling model for optimally scheduling the jobs coming from different customers through adopting game theory. In the job scheduling game, jobs are defined as the players, feasible machines are defined as strategies, the makespan of each job is denoted as the payoff, and the concept of NE is introduced and used to obtain the optimal job scheduling results. After proposing the mathematical model, we propose and develop the GA-based solution algorithm to solve our optimization problem. In the process of finding the optimal results for the job scheduling game, we consider two different priority rules, i.e., FCFS rule and SPT rule and design the correspondent decoding procedures to demonstrate the influences on the job scheduling game. From the computational results we concluded that the presented job scheduling game performed well as it is consistent with requirements and objectives of job scheduling in the networked manufacturing environment.

Acknowledgements Special thanks are due to Professor Huang of The University of Hong Kong for his gracious supervision of this work. Gratitude is also extended to the National Natural Science Foundation of China (Grant No.: 50605050), the Program for New Century Excellent Talents in University by China Ministry of Education (Grant No.: NCET-04–0928) for the financial supports.

References

1. Jiang PY, Zhou GH, Zhao G, Zhang YF, Sun HB (2007) e2-MES: an e-service-driven networked manufacturing platform for extended enterprises. Int J Comput Integr Manuf 20(2–3):127–142
2. Zhou GH, Jiang PY, Zhang DH (2005) Generating a task-driven extended enterprise for e-manufacturing. Int J Internet Enterprise Manage 3(3):264–279
3. Sousa P, Ramos C (1999) A distributed architecture and negotiation protocol for scheduling in manufacturing systems. Comput Ind 38:103–133
4. Persi P, Ukovich W, Pesenti R, Nicolich M (1999) A hierarchic approach to production planning and scheduling of a flexible manufacturing system. Robot Integr Manuf Syst 15:373–385
5. Gou L, Luh PB, Kyoya Y (1998) Holonic manufacturing scheduling: architecture, cooperation, mechanism and implementation. Comput Ind 37:213–231
6. Yang HS, Wang NS, Zhang JG, Cui XY (1998) Modelling, scheduling and simulation of flexible manufacturing systems using extended stochastic high-level evaluation Petri nets. Robot Auton Syst 14:121–140
7. Song JS, Lee TE (1998) Petri net modeling and scheduling for cyclic job shops with blocking. Comput Ind Eng 34(2):281–295
8. Moursli O, Pochet Y (2000) A branch-and-bound algorithm for hybrid flowshop. Int J Prod Econ 64:113–125
9. Sakawa M, Mori T (1999) An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy due date. Comput Ind Eng 36:325–341
10. Mattfeld DC, Bierwirth C (2004) An efficient genetic algorithm for job scheduling with tardiness objectives. Eur J Oper Res 155:616–630
11. Senthilkumar P, Shahabudeen P (2006) GA-based heuristic for the open job shop scheduling problem. Int J Adv Manuf Technol 30 (3–4):297–301
12. Engin O, Döyen A (2004) A new approach to solve hybrid flow shop scheduling problems by artificial immune system. Future Gener Comput Syst 20:1083–1095
13. Rajendran C, Ziegler H (2004) Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total fowtime of jobs. Eur J Oper Res 155:426–438
14. Ying KC, Liao CJ (2004) An ant colony system for permutation flow-shop sequencing. Comput Oper Res 31:791–801
15. Chandrasekaran M, Asokan P, Kumanan S, Balamurugan T, Nickolas S (2006) Solving job shop scheduling problems using artificial immune system. Int J Adv Manuf Technol 31(5–6): 580–593
16. Seredyński F (1997) Competitive co evolutionary multi-agent systems: the application to mapping and scheduling problems. J Parallel Distrib Comput 47:39–57
17. Seredyński F (1998) Distributed scheduling using simple learning machines. Eur J Oper Res 107:401–413
18. Seredyński F, Koronacki J, Janikow CZ (2001) Distributed multiprocessor scheduling with decomposed optimization criterion. Future Gener Comput Syst 17:387–396
19. Kim JH, Park JB, Park JK, Kim BH (2003) A new game-theoretic framework for maintenance strategy analysis. IEEE Trans Power Syst 18(2):698–706