

Coordinated Scheduling of Two-Agent Production and Transportation Based on Non-Cooperative Game

Ke Xu^{1,2}, Peng Liu^{1,*} and Hua Gong^{1,2}

¹School of Management, Shenyang University of Technology, Shenyang, 110870, China

²School of Science, Shenyang Ligong University, Shenyang, 110159, China

*Corresponding Author: Peng Liu. Email: liupeng@sut.edu.cn

Received: 13 September 2022; Accepted: 04 November 2022

Abstract: A two-agent production and transportation coordinated scheduling problem in a single-machine environment is suggested to compete for one machine from different downstream production links or various consumers. The jobs of two agents compete for the processing position on a machine, and after the processed, they compete for the transport position on a transport vehicle to be transported to two agents. The two agents have different objective functions. The objective function of the first agent is the sum of the makespan and the total transportation time, whereas the objective function of the second agent is the sum of the total completion time and the total transportation time. Given the competition between two agents for machine resources and transportation resources, a non-cooperative game model with agents as game players is established. The job processing position and transportation position corresponding to the two agents are mapped as strategies, and the corresponding objective function is the utility function. To solve the game model, an approximate Nash equilibrium solution algorithm based on an improved genetic algorithm (NE-IGA) is proposed. The genetic operation based on processing sequence and transportation sequence, as well as the fitness function based on Nash equilibrium definition, are designed based on the features of the two-agent production and transportation coordination scheduling problem. The effectiveness of the proposed algorithm is demonstrated through numerical experiments of various sizes. When compared to heuristic rules such as the Longest Processing Time first (LPT) and the Shortest Processing Time first (SPT), the objective function values of the two agents are reduced by 4.3% and 2.6% on average.

Keywords: Coordinated scheduling; two-agent production and transportation; non-cooperative game; genetic algorithm

1 Introduction

In modern manufacturing models, various production connections inside a business or distinct clients outside a firm will have different personalized wants and compete for production resources. At this time, cooperative planning and decision-making among different decision makers must be considered [1].



This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Agnetis et al. [2] studied this sort of problem and proposed multi-agent production scheduling by viewing production links or customers as agents. Multi-agent production scheduling refers to multiple agents competing for the use of the common machines to process their respective jobs so that the objective of depending on the completion time of their own jobs is optimal [3,4]. In a distributed manufacturing environment, manufacturing equipments and individual agents may be in various geographical locations, and the manufacturing capacity of resources should be taken into account together with the synergy between transportation and production. The coordination of production and transportation is common in steel manufacturing, automotive sector suppliers and other industries [5–8].

Examples involving two-agent production and transportation may be found in the steel plant manufacturing process. As one practical example, the steelmaking-continuous casting processes in the steel plant demonstrate the real production settings. The manufacturing process is depicted in Fig. 1. The crude steel produced in the converter is refined and converted into liquid steel, which is transported to the continuous casting machine through transport vehicles. The molten steel can take two alternative modes according to the desired ingot shape in the ingot formation process. One is the mould casting mode that the molten steel in a ladle is poured into molds to solidify cylindrically-shaped ingots. The other is the continuous-casting mode that the molten steel is cast and cooled to form slabs, blooms or billets in the continuous caster. Due to the differences between the mould casting mode and the continuous-casting mode, they are regarded as two agents competing for the refined molten steel. The purpose is to schedule all of the jobs of both agents such that each agent's objects are optimized.

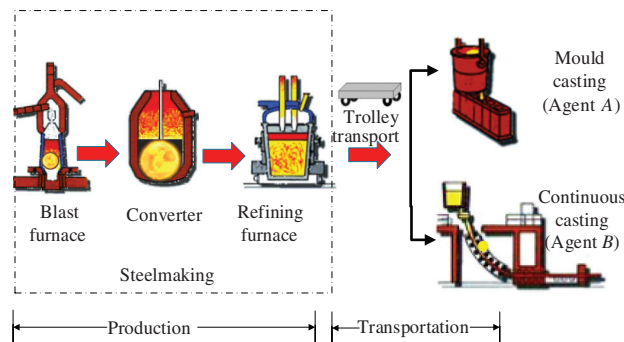


Figure 1: Steelmaking-continuous casting production process

The majority of research on multi-agent scheduling has been on a single machine. Baker et al. [9] and Agnetis et al. [10] studied the single-machine two-agent scheduling problem with the linear weighted sum model, the constrained optimization model, and the Pareto optimization model as objective functions. Liang et al. [11] studied the single-machine two-agent scheduling problem with release time and the makespan objective function, and established a mixed integer programming model. Wang et al. [12] described a multi-agent single-machine scheduling problem with release dates, where several packets (jobs) share a common network (processor) but are maintained by several competitive applications (agents) that optimize their criteria. The objective is to minimize the sum of the makespan belonging to several agents individually. Zhang et al. [13] extended the resource allocation fair pricing problem to single-machine two-agent scheduling, where the objectives of the two agents are the total number of tardy jobs and maximum cost, respectively. Furthermore, several researchers have studied multi-agent scheduling on parallel machines [14], batch machines [15,16] and flowshop [17].

There is less research on multi-agent scheduling issues that address transportation and processing cooperatively. Fan et al. [18] defined the objective function as the sum of the makespan and transportation time, and proposed a dynamic programming algorithm. Yin et al. [19] investigated a two-agent process and

finished product batch transportation scheduling problem. A polynomial-time algorithm for minimizing the objective of agent A is given under the constraint that the objective of agent B does not exceed a given threshold. Qi et al. [20] showed that their algorithm is incorrect by a counterexample and presented polynomial-time algorithms to solve the problem in Yin et al. [19] in polynomial time.

In summary, the multi-agent scheduling problem is mostly studied as a two-agent scheduling problem under a single machine, with less consideration of the collaboration between processing and transportation. Most of the solution methods transform the multi-objective problem into a single-objective problem and solve the small-scale problem by an exact algorithm, ignoring the competitive relationship between two agents.

Mathematical programming approaches, meta-heuristic algorithms, intelligent optimization algorithms, and other methods are commonly adopted to solve the production scheduling problem [21,22]. Game theory takes into account the collaboration and competition between different subjects. Many researchers have adopted game theory to investigate the production scheduling problem. Zhou et al. [23] mapped each customer's manufacturing task as a player, establishing a scheduling model and solving the Nash equilibrium point of the non-cooperative game model based on a hybrid adaptive genetic algorithm. Nong et al. [24] studied the scheduling problem of parallel batch machines with the objective of minimizing the makespan of each agent and mapping them to the player so as to build a game model. Zhang et al. [25] studied the real-time multi-objective flexible job shop scheduling problem and proposed a two-level scheduling method based on dynamic game theory to find the optimal solution through sub-game perfect Nash equilibrium. Wang et al. [26] constructed a game theory-based dynamic scheduling model for flexible job shop under emergency order addition, aiming at the rapidity and stability of the scheduling plan. Nie et al. [27] established a non-cooperative game model and proposed a game scheduling algorithm based on the genetic algorithm to solve the D-min Nash equilibrium, aiming at the shop scheduling problem in virtual manufacturing network and minimizing the makespan of all jobs.

It can be seen that more scholars have studied production scheduling problems based on non-cooperative games, although game theory is rarely applied to analyze multi-agent scheduling problems. As a result, non-cooperative game theory is used to investigate the two-agent production and transportation coordinated scheduling problem by taking into account the competition between the jobs of two agents for the processing sequence and transportation sequence from the standpoint of maximizing the payoff of agents.

This paper carries out the following three works listed below.

- (1) Two different production links or customers are regarded as agents in view of the competition for the same machine resources between the jobs from different production links or customers. After the job has been processed, it needs to be delivery to the respective agent, and the two agents have distinct objectives. Considering production and post-production transportation cooperatively, a scheduling problem of two-agent production and transportation coordination in a single machine is suggested.
- (2) A non-cooperative game model for the competition relationship between the jobs of two agents for the machine and the transport vehicle is constructed. The two agents are mapped as players, and the two agents compete for the processing position on the machine and the transportation position on the transport vehicle. The processing position and transportation position of the jobs are mapped as strategies, and the objective function of the two agents is mapped as payoff functions.
- (3) An approximate Nash equilibrium solution algorithm based on an improved genetic algorithm (NE-IGA) is proposed to solve the approximate Nash equilibrium solution of the problem. The fitness function is designed from the Nash equilibrium definition, and the genetic operator based on production and transportation is designed according to the characteristics of the two-agent production and transportation coordination scheduling problem in single-machine.

This paper is outlined as follows: Section 2 develops a non-cooperative game model to handle the problem of two-agent production transportation coordination scheduling in single machine. Section 3 designs an approximate Nash equilibrium solution method based on an improved genetic algorithm. Section 4 describes the experiment setting and the results of the experiment. Finally, Section 5 summarizes the conclusions.

2 A Non-Cooperative Game Model for Two-Agent Production and Transportation Scheduling in Single Machine

2.1 Problem Description

The jobs of two agents compete to be processed on a single machine and transported to their respective agents after completion, i.e., the jobs of two agents compete to get the order of processing on the machine and transport on the transport vehicle.

Let there be two agents: agent A and agent B , whose job sets are $J^A = \{J_1^A, \dots, J_m^A\}$, $J^B = \{J_{m+1}^B, \dots, J_n^B\}$, and the total number of jobs is n . The number of jobs of agent A is m and the number of jobs of agent B is $(n-m)$. The jobs of two agents compete for processing on a single machine, and the finished jobs on this machine need to be transported to agent A and agent B by a transport vehicle. The time for the transport vehicle to travel between machine and agent X is t_X ($X = A, B$). The processing time of the J_i ($i = 1, \dots, n$) is P_i , the completion time is C_i . The finish time of the transport vehicle transporting the J_i is T_i , and the finish time T_i is defined as the time for the vehicle to transport J_i to the corresponding agent and back to the machine. Related parameters and variables are shown in Table 1.

Table 1: Table of notation descriptions

Set of indices	
J	Set of jobs, $J = \{J_i, i = 1, \dots, n\}$
X	Agent, $X = A, B$
J^A	Set of jobs for agent A , $J^A = \{J_1^A, \dots, J_m^A\}$
J^B	Set of jobs for agent B , $J^B = \{J_{m+1}^B, \dots, J_n^B\}$
Parameters	
P_i	The processing time of the J_i , $i = 1, \dots, n$
t_X	The time for the transport vehicle to travel between machine and agent X , $X = A, B$
Variables	
C_i	The completion time of J_i
T_i	The finish time of the transport vehicle transporting the J_i ; that is, the time for the vehicle to transport J_i to the corresponding agent and back to the machine.
W_i	The waiting time for the transport vehicle to transport J_i .
f_1	The optimization objective of agent A
f_2	The optimization objective of agent B

Assume that the following assumptions are satisfied to simplify the problem.

- (1) The single machine can process continuously without the requirement for setup or switching time.

- (2) All jobs have arrived at the initial time and are ready to be processed.
- (3) The transport vehicle has a capacity restriction, and only one job can be loaded onto the transport vehicle at a time.

Considering that there is a competitive relationship between the two agents, each agent needs to maximize its own utility. The objective of agent A is the sum of the makespan and the total transportation time, while the objective of agent B is the sum of the total completion time and the total transportation time, as shown in Eqs. (1)–(2).

$$\text{Min} f_1 = \max_{i=1,2,\dots,m} \{C_i\} + mt_A + \sum_{i=1}^{m-1} W_i \quad (1)$$

$$\text{Min} f_2 = \sum_{i=m+1}^n C_i + (n - m)t_B + \sum_{i=m+1}^{n-1} W_i \quad (2)$$

where W_i denotes the waiting time for the transport vehicle to transport J_i . If J_i has finished processing when the transport vehicle has finished transporting the last job and returned to the machine, the transport vehicle does not need to wait, i.e., $W_i = 0$; if J_i has not finished processing when it returns to the machine, the transport vehicle needs to wait for J_i to finish processing, and the waiting time is the completion time of J_i minus the finish time of the previous transport job l , i.e., $W_i = C_i - T_l$.

According to the extended three-domain representation $\alpha | \beta | \gamma_1, \gamma_2$ to describe the two-agent problem, α and β denote the characteristics of the machine environment and job, r_i denotes the arrival time of J_i , and if all jobs arrive at the same time, then $r_i = 0$. γ_1, γ_2 denotes denote the objective functions of the two agents, respectively. So the two-agent production and transportation coordination scheduling problem in a single-machine environment can be expressed as $1 | r_i = 0 | f_1, f_2$.

2.2 Non-Cooperative Game Model

In a non-cooperative game model, multiple players choose their strategies based on their own interests, and there is both conflict and cooperation between players, which conforms to the competition and cooperation between two agents for resources in the two-agent scheduling problem. As a result, we adopt non-cooperative game theory to address the problem of two-agent production and transportation scheduling in a single machine.

In a single machine setting, a non-cooperative game model for the two-agent production and transportation coordinated scheduling problem is established. The model can be formulated as Eq. (3).

$$G = \{I, S, U\} \quad (3)$$

where I denotes the number of game players. $S = \{S_1, \dots, S_k\}$ denotes the strategy of players, where S_o denotes the strategy of the players $o(o = 1, 2, \dots, k)$. $U = \{U_1, \dots, U_k\}$ denotes the payoff function of player with respect to the strategy, where U_o denotes the payoff of the players o under the strategy.

Player: The jobs of the two agents fight for the sequence of processing on the machine and transit on the transport vehicle. The two agents are mapped as players.

Strategy: The strategy is defined as $S = \{S_1, S_2\}$, which is the set of actions or decision choices of the players. Each agent is responsible for two decisions: the processing position and transportation position of each job. Then the strategies of the two agents are as Eq. (4).

$$\begin{aligned}
S_1 &= (M_{i,l}^A, T_{i,j}^A) i = 1, 2, \dots, m, l, j \in n \\
S_2 &= (M_{i,l}^B, T_{i,j}^B) i = m+1, \dots, n, l, j \in n
\end{aligned} \tag{4}$$

where $M_{i,l}^A$ denotes that J_i^A is processed at location l , $T_{i,j}^A$ denotes that J_i^A is transported at trip j . $M_{i,l}^B$ denotes that J_i^B is processed at location l , and $T_{i,j}^B$ denotes that J_i^B is transported at trip j .

Not all strategies are valid. Strategy S is invalid when the following conditions occur.

- (1) Sequences of processing conflict: Because the two players only consider their own strategies, there will be two jobs processed at the same position, rendering the strategy invalid. In other words, for any two different jobs i_1 and i_2 in strategy S , the strategy is invalid when there is $M_{i_1,l}^A = M_{i_2,l}^A$ or $M_{i_1,l}^A = M_{i_2,l}^B$.
- (2) Sequences of transport conflict: When two jobs are transported at the same position, the strategy is invalid. In other words, for any two different jobs i_1 and i_2 in strategy S , the strategy is invalid when there is $T_{i_1,j}^A = T_{i_2,j}^A$ or $T_{i_1,j}^A = T_{i_2,j}^B$.

For example, the job sets of agent A and agent B are $\{J_1, J_2\}$ and $\{J_3\}$, respectively. Then, without considering conflicts, there are 3 processing positions and 3 transport positions from which all jobs can choose on the machine or vehicle, resulting in $3^3 \times 3^3 \times 3^3 = 729$ different strategies, but some of them are invalid. For example, strategy $S_1 = \{(2, 2), (1, 1)\}$, $S_2 = \{(1, 3)\}$, where the J_2^A and J_1^B are processed at the same position, so the strategy is invalid. Strategy $S_1 = \{(2, 2), (1, 1)\}$, $S_2 = \{(3, 2)\}$, the J_1^A and J_1^B are transported at the same position, making this strategy invalid as well. After deleting all invalid strategies, there remained 36 feasible strategies.

Payoff function: The payoff function is the gain obtained by the game parties under the strategy. Therefore, for agent A and agent B under the strategy, the payoff function $U = \{U_1, U_2\}$. Taking the objective functions of the two agents as their payoff functions: $U_1 = f_1$, $U_2 = f_2$, the game model of the two-agent scheduling problem can be described as Eq. (5).

$$G = \{S_1, S_2; U_1, U_2\} \tag{5}$$

Nash equilibrium: In the non-cooperative game model, there exists a certain strategy $S^* = \{S_1^*, S_2^*\}$, satisfying the two game parties unilaterally changing their own strategies without increasing their own payoff, that is, the strategy S^* is a Nash equilibrium solution when Eq. (6) is satisfied.

$$u_o(S_o^*, S_{-o}^*) \geq u_o(S_o, S_{-o}^*) \tag{6}$$

where u_o denotes the payoff function of the game player o and S_{-o}^* denotes the combination of strategies of the game players other than the game player o .

For example, the job sets of agent A and agent B are $\{J_1, J_2\}$ and $\{J_3\}$, respectively, with the relevant parameters as shown in Table 2.

Table 2: Parameters related to production and transport tasks for $m = 2, n = 3$

	Agent A		Agent B
J_i	1	2	3
P_i	0.5	1.5	1.8
t_X	1.2	1.2	1.9

According to the non-cooperative game model built for the two-agent production and transportation coordination scheduling problem, there are 36 alternative strategies for agent A and 9 alternative strategies for agent B . According to the constructed payoff matrix, nine pure strategy Nash equilibriums can be obtained using the delineation method: $S^1 = \{(1, 1), (2, 2); (3, 3)\}$, $S^2 = \{(1, 3), (2, 1); (3, 2)\}$, $S^3 = \{(1, 3), (2, 2); (3, 1)\}$, $S^4 = \{(1, 1), (3, 2); (2, 3)\}$, $S^5 = \{(1, 3), (3, 1); (2, 2)\}$, $S^6 = \{(1, 3), (3, 2); (2, 1)\}$, $S^7 = \{(2, 2), (3, 1); (1, 3)\}$, $S^8 = \{(2, 3), (3, 1); (1, 2)\}$, $S^9 = \{(3, 3), (2, 2); (1, 1)\}$, corresponding to the payoffs of (5.2, 6.3), (4.4, 9.5), (4.4, 9.5), (8.8, 4.3), (6.2, 6.5), (6.2, 6.5), (8.8, 5.2), (6.2, 8.5), (6.2, 5.5).

3 An Approximate Nash Equilibrium Solution Algorithm Based on an Improved Genetic Algorithm

The strategy space of the game model grows exponentially as the number of jobs of two agents increases, and the genetic algorithm can undertake a speedy search for the best solution without the limitation of scale change. Therefore, an approximate Nash equilibrium solution algorithm based on an improved genetic algorithm (NE-IGA) has been developed to seek the approximate Nash equilibrium of the provided game model.

3.1 Coding

According to the action strategy in the game model, integer coding is applied, and each job needs to choose a processing position and a transportation position, so each chromosome is divided into two sections, as shown in Fig. 2. The length of the chromosome is $2n$, with the initial n bits representing the processing sequence and the latter n bits representing the transportation sequence, and each component representing the whole arrangement of n jobs.

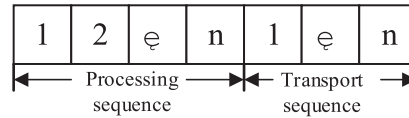


Figure 2: Coding scheme

For example, there are 5 jobs with chromosome codes (3, 4, 1, 2, 5, 2, 5, 1, 3, 4), which implies that the processing sequence of the jobs is 3, 4, 1, 2, 5, namely $J_3 \rightarrow J_4 \rightarrow J_1 \rightarrow J_2 \rightarrow J_5$, and the transportation sequence of the jobs is 2, 5, 1, 3, 4, namely $J_2 \rightarrow J_5 \rightarrow J_1 \rightarrow J_3 \rightarrow J_4$.

3.2 Fitness Function

The design of the fitness function is critical to resolving the Nash equilibrium. The objective of the two-agent production and transportation coordination scheduling problem is to minimize the objective functions of agents A and B. According to the definition of Nash equilibrium in the non-cooperative game model, when the Nash equilibrium is reached, it is difficult for the players to improve their own payoffs by modifying their own strategies, and the payoffs of the two agents are close enough to the best payoffs of the previous generation. As a result, the Nash equilibrium is determined by the difference between individual payoffs in the current generation and the best individual payoff in the previous generation. Second, the genetic algorithm calculates the likelihood of individual inheritance to the next generation based on the proportion of individual fitness, which must be positive. The fitness function is defined as Eq. (7).

$$F^g(U_1^g, U_2^g) = \sum_{o=1}^2 |U_o^g - \tilde{U}_o^{g-1}| \quad (7)$$

$$\tilde{U}_o^{g-1} = \min_{q \in Q} \{U_{oq}^{g-1}\} \quad (8)$$

where U_1^g denotes the payoff of agent A at generation g ; U_2^g denotes the payoff of agent B at generation g ; \tilde{U}_1^{g-1} denotes the best payoff of agent A at generation $g-1$; \tilde{U}_2^{g-1} denotes the best payoff of agent B at generation $g-1$, and Q is the set of populations.

According to the definition of the Nash equilibrium of the non-cooperative game model, the smaller the fitness value, the better. Defining Eq. (7) satisfies Eq. (9) when the algorithm finds the Nash equilibrium.

$$|U_o^g - \tilde{U}_o^{g-1}| < \eta_o \quad o = 1, 2 \quad (9)$$

where η_o is the threshold applied to determine the solution to the scheduling of the game model, i.e., the criterion for determining the Nash equilibrium, taken as a smaller positive number. To simplify the problem, we take $\eta_1 = \eta_2 = \eta$. Obviously, the Nash equilibrium threshold η determines the approximation degree between the solution obtained by the algorithm and the Nash equilibrium. If the value of η is too large, the approximation degree of the solution will be poor, while if the value of η is too small, the convergence time of the algorithm will be prolonged. Therefore, the results obtained by fast global search of the genetic algorithm approximate the Nash equilibrium.

3.3 Genetic Operator

3.3.1 Selection Operator

The roulette selection operator is applied, and the likelihood of each chromosome joining the next generation is equal to the ratio of its fitness value to the total fitness values of the whole population. Then, according to the single lottery logic, chromosomes are chosen based on the random number's area.

The fitness is divided into $F^g(U_1^g, U_2^g)$ parts, with each individual accounting for $1/F^g(U_1^g, U_2^g)$, and then the chromosomes are selected according to the region where the random numbers are located, according to the single draw logic.

3.3.2 Production and Transportation-Based Two-Point Cross Operator

Paternal chromosomes y_1 and y_2 are chosen. Two gene loci are chosen at random from the first n production sequence of chromosome y_1 , and the matching two gene loci are chosen at random from the final n transport sequence and given to the offspring chromosome y . To avoid the formation of invalid chromosomes, offspring y is assigned from parent chromosome y_2 in left-to-right order, with the exception of loci previously selected for chromosome y_1 . The crossover process is shown in Fig. 3.

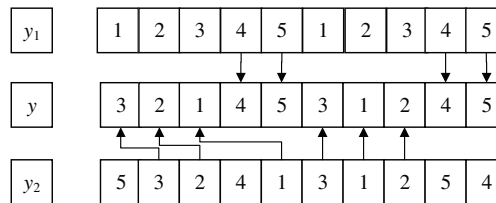


Figure 3: Production and transportation-based two-point crossover operation

3.3.3 Production and Transportation-Based Two-Point Mutation Operator

Chromosome y_3 is chosen. To obtain the mutant chromosome y , two gene loci are randomly picked from the first n production loci of y_3 for exchange, and two gene loci are randomly selected from the final n transport loci for exchange. The mutation process is shown in Fig. 4.

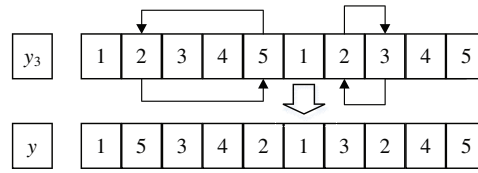


Figure 4: Production and transportation-based two-point mutation operation

Crossover or mutation operations are decided based on the similarity between parental chromosomes in order to improve population diversity and avoid slipping into a local optimum. The similarity between chromosomes y_1 and y_2 is expressed by the Hamming distance $D(y_1, y_2)$. The smaller the $D(y_1, y_2)$, the greater the difference between y_1 and y_2 , and the greater the likelihood of obtaining new offspring chromosomes after crossover operation. The larger the $D(y_1, y_2)$, the smaller the difference between y_1 and y_2 , and the crossover operation tends to leave the offspring chromosomes unchanged, and the mutation operation is employed at this time. To decide whether to do the mutation operation, according to the similarity relative percentage threshold φ . When the formula (10) is satisfied, the algorithm considers the similarity of the selected parent chromosome to be insufficient and executes the crossover operation. Otherwise, the selected parent chromosome's resemblance is regarded as strong, and the mutation operation is carried out.

$$\frac{D(y_1, y_2)}{n} \leq \varphi \quad (10)$$

where n is the number of jobs.

3.4 Algorithm Flow

The optimal individual retention strategy is adopted by NE-IGA to safeguard the best individuals in each generation of the population from being eliminated during subsequent selection, crossover, or mutation. This is accomplished by replacing the one with the largest value of the fitness function calculated in the offspring generation with the one with the smallest value of the fitness function in the parent generation.

The following are the steps of NE-IGA for solving the two-agent production and transportation coordination scheduling problem. And Fig. 5 depicts the algorithm flow.

Step1: Parameter settings: population size N , crossover probability P_c , variation operation threshold φ and maximum number of iterations T_{max} .

Step2: Initialize the population and set $g = 1$.

Step3: The fitness value $F^g(U_1^g, U_2^g)$ of each chromosome is calculated according to Eq. (7), and the parental chromosome is selected by roulette.

Step4: Update the population. Two chromosomes y_1 and y_2 are chosen at random from the parent generation, and the similarity is calculated to see if Eq. (10) is satisfied. If not, the mutation operation is performed. Otherwise, the crossover operation is performed according to the crossover probability P_c .

Step5: Retain the best individuals. Calculate the fitness value of the current population using Eq. (7). And replace the individuals with the worst fitness value in the offspring population with the best individuals in the parent population.

Step6: If Eq. (9) is met, it outputs the optimal solution, which is the approximate Nash equilibrium solution. Otherwise, proceed to Step 7.

Step7: Determine whether the maximum number of iterations has been reached, i.e., whether $g = T_{max}$ is true; if so, the optimal solution of this generation is output as the Nash equilibrium approximate solution to end the algorithm; otherwise, $g = g+1$, return to step 3.

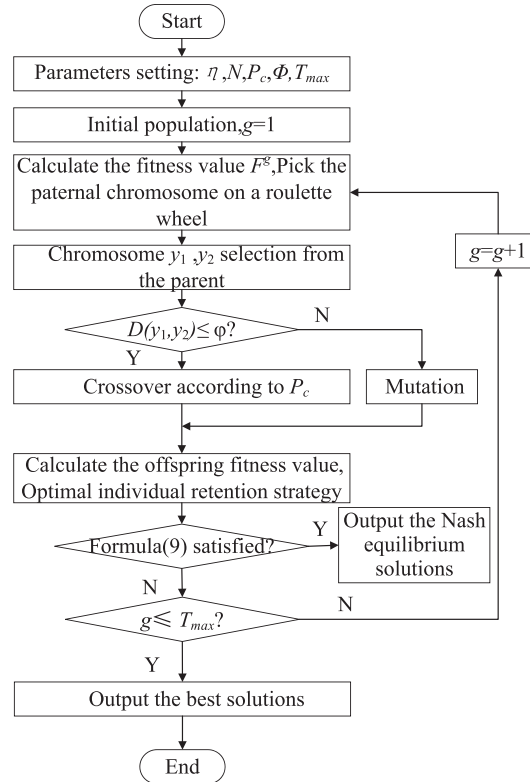


Figure 5: Algorithm flow chart

4 Numerical Experimental Analysis

The following numerical example is created for simulation and analysis in order to validate the effectiveness of the proposed algorithm. The experiments are carried out on an Intel(R) Core(TM) i5-7200U CPU 2.50 GHz 2.71 GHz processor, 8 GB RAM, and Unity 2017 software.

4.1 Selection of Initial Circumstances and Parameters

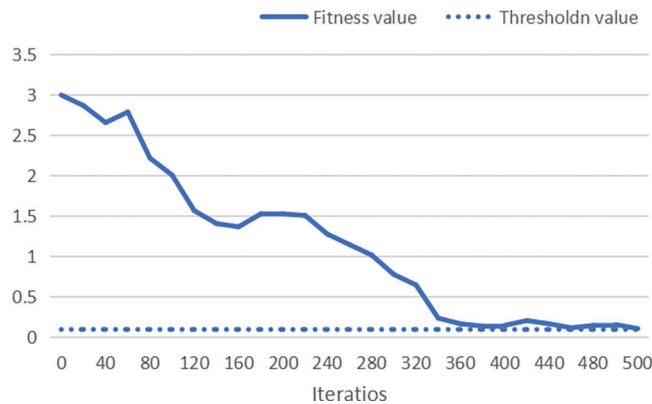
Take $m = 6$ and $n = 12$, which means that agent A and agent B each have 6 jobs to process, while P_i and t_x are chosen at random on the interval $[1, 10]$. The following orthogonal experiments are aimed at determining the algorithm's parameter selection. The population size N is set to 50, 100, and 200; the crossover probability P_c is set to 0.7, 0.8, and 0.9; the similarity threshold ϕ is set to 0.5, 0.6, and 0.7; and the Nash equilibrium threshold η is set to 0.01, 0.05, and 0.1. Nine groups of experimental data are produced at random, and the mean values of the objective functions of the two agents are determined for each group of data. The parameters are then determined by calculating the mean values of the objective functions over 9 groups of varied experimental data. Table 3 displays the results.

It can be seen from Table 3 that the fifth group has the smallest mean, so the population size N is determined to be 100, the crossover probability P_c is 0.8, the similarity threshold ϕ is 0.6, and the Nash equilibrium threshold η is 0.05.

Table 3: Orthogonal experimental results of NE-IGA

Serial number	N	P_c	φ	η	Average value
1	50	0.7	0.5	0.01	139.75
2	50	0.8	0.6	0.05	132.09
3	50	0.9	0.7	0.1	133.01
4	100	0.7	0.5	0.01	129.03
5	100	0.8	0.6	0.05	128.73
6	100	0.9	0.7	0.1	130.01
7	200	0.7	0.5	0.01	129.21
8	200	0.8	0.6	0.05	133.70
9	200	0.9	0.7	0.1	140.47

Fig. 6 shows the convergence of the fitness function when the algorithm iterates to 500 generations with the above experimental parameters. It can be seen that the fitness function converges to the given threshold when the number of iterations is close to 400, and it has stabilized at this time. Therefore, the maximum number of iterations for this experiment is chosen to be 400.

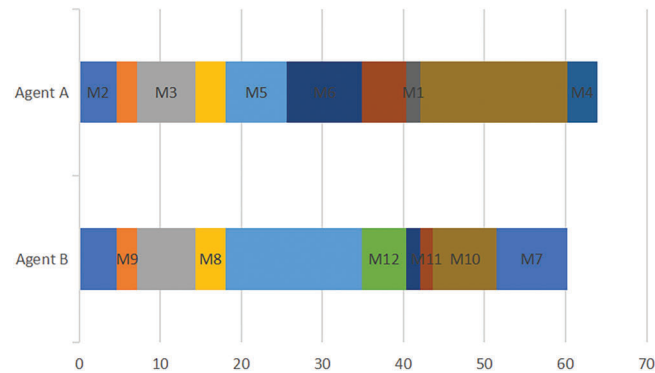
**Figure 6:** The iterative convergence curve for the best fitness value

4.2 Analysis of Numerical Experimental Results

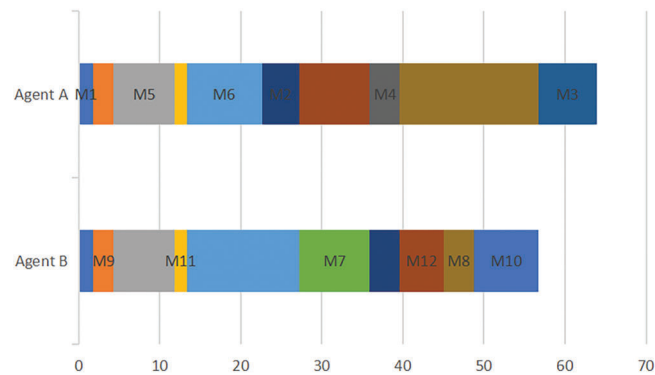
Fig. 7 shows the scheduling Gantt charts of the algorithm when it runs for the first, 100th, and 400th generations. Table 4 shows the scheduling schemes and the payoff of the two agents for the corresponding number of iterations. It can be seen that the objective function values of both agents A and B in the 400th iteration scheduling scheme are reduced when compared to the first and 100th iterations, and it is difficult to change the objective functions of both agents by unilaterally changing the positions of the artifacts at the 400th iteration.

4.3 Experimental Comparison

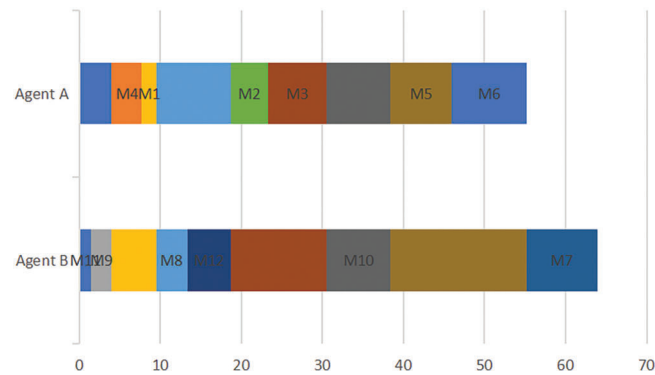
To further validate the efficacy of the proposed algorithm, a comparison analysis is done at various experimental sizes, utilizing the Longest Processing Time first (LPT) and the Shortest Processing Time first (SPT).



(a) Gantt chart for the first generation of scheduling



(b) Gantt chart for the 100th generation of scheduling



(c) Gantt chart for the 400th generation of scheduling

Figure 7: NE-IGA scheduling Gantt chart

Fig. 8 and Table 5 illustrate the scheduling Gantt charts and payoff of the two agents for $m = 6$ and $n = 12$ under the LPT, SPT, and NE-IGA. It can be seen that the LPT rule has the worst payoff for agent A and agent B , and the scheduling plan is not reasonable. When compared to the other two rules, NE-IGA has a relatively superior payoff for both agent A and agent B , and the payoff for agent A and agent B is improved by 8.3% and 2.8% compared to SPT, and by 8.7% and 42.5% compared to the LPT rule. The average value of the two agents' objective functions is improved by 34.3% and 4.6%, respectively, when compared to the LPT and SPT. As a result, the NE-IGA obtains a superior scheduling plan for both agent A and agent B .

Table 4: NE-IGA scheduling plan and payoff values for the first, 100th, and 400th generations

Iterations	Scheduling plan	Payoff	
		Agent A	Agent B
1	$(J_2, J_9, J_3, J_8, J_5, J_6, J_{12}, J_1, J_{11}, J_{10}, J_7, J_4, J_2, J_9, J_3, J_8, J_5, J_6, J_{12}, J_{11}, J_1, J_{10}, J_7, J_4)$	98.50	280.80
100	$(J_1, J_9, J_5, J_{11}, J_6, J_2, J_7, J_4, J_{12}, J_8, J_{10}, J_3, J_1, J_9, J_{11}, J_6, J_5, J_2, J_7, J_4, J_{12}, J_{10}, J_8, J_3)$	95.70	264.00
400	$(J_{11}, J_9, J_4, J_1, J_8, J_{12}, J_2, J_3, J_{10}, J_5, J_6, J_7, J_{11}, J_9, J_4, J_1, J_8, J_{12}, J_2, J_3, J_{10}, J_5, J_6, J_7)$	85.20	201.30

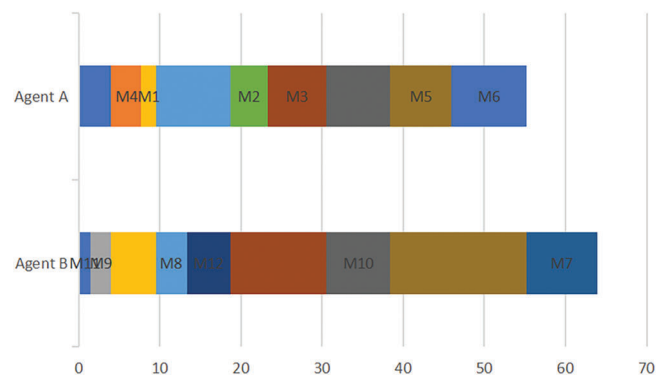
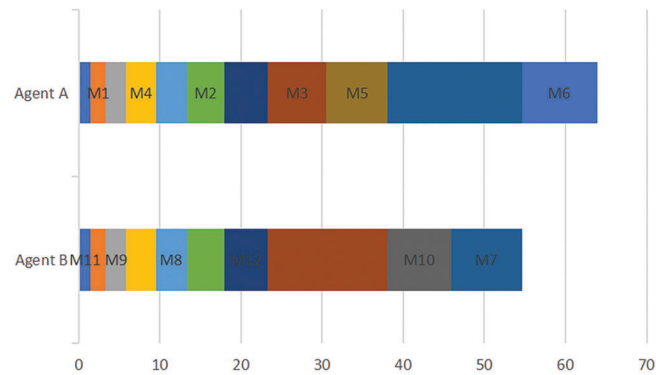
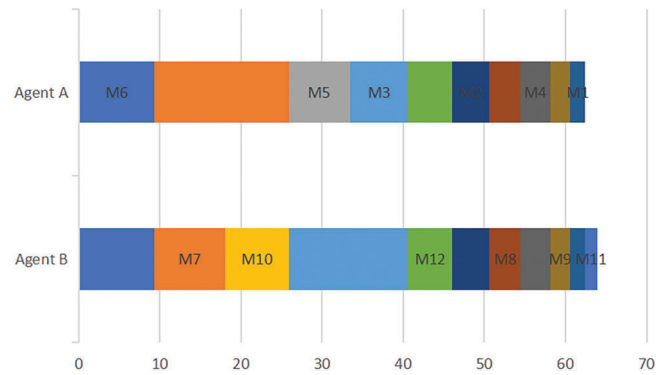
**Figure 8:** Gantt chart comparison of different algorithms when $n = 12$ and $m = 6$

Table 5: Payoff of different algorithms when $n = 12$ and $m = 6$

Algorithm	Payoff		Average value
	Agent A	Agent B	
LPT	96.20	301.10	198.65
SPT	95.70	178.00	136.85
NE-IGA	87.80	173.10	130.45

Table 6 compares the payoff of the NE-IGA to the LPT and SPT for two agents with $m = 12$ and $n = 24$. The payoff for agent B under the LPT rule is the worst, which is adverse for agent B , therefore the scheduling plan is not rational. Although the payoff of agent B is the best under the SPT, the payoff of agent A is the poorest at this moment, and the scheduling plan is likewise inappropriate. The payoff for both agent A and agent B under the NE-IGA is quite good when compared to the other two rules. The payoff of agent A is enhanced by 12.4% when compared to the SPT, while the payoffs of agents A and B are increased by 6.0% and 62.8%, respectively, when compared to the LPT. Furthermore, the average value of the objective function of both agents is the least under the NE-IGA, which is improved by 54.9% and 3.8%, respectively, over the LPT and SPT rules.

Table 6: Payoff of different algorithms when $n = 24$ and $m = 12$

Algorithm	Payoff		Average value
	Agent A	Agent B	
LPT	244.70	1500.80	872.75
SPT	259.80	557.80	408.80
NE-IGA	227.60	559.0	393.30

It can be seen that the objective functions of the two agents under the NE-IGA algorithm can converge to the optimum at the same time under different experimental scales, and a solution closer to the Nash equilibrium is obtained.

5 Conclusions

In this paper, we investigate the problem of coordinated production and transportation scheduling for two agents in a single machine. The jobs of two agents compete to be processed on a single machine and delivered to their respective agents after completion, and there is also competition for the transport vehicle. The objective of agent A is the sum of the makespan and the total transportation time, whereas the objective of agent B is the sum of the total completion time and the total transportation time. A non-cooperative game model is developed with two agents as players whose jobs fight for processing and transportation on the machine and vehicle; the processing and transportation position of the jobs as the game strategy; and the objective functions of the two agents as their payoff. The NE-IGA is meant to solve the approximate Nash equilibrium by designing the coding and genetic operators based on the processing and transportation sequence, as well as the fitness function based on the Nash equilibrium definition. The classic heuristic criteria are contrasted with simulation results at various experimental sizes. The experimental results show that the NE-IGA scheduling plan produces better results, with average performance improving by 44.6% and 4.2%, respectively, when compared to the LPT and SPT.

This paper assumes that the transport vehicle can only transport one job at a time and does not consider batch transport. Furthermore, the job may need to be delivered to the matching agent for the next processing within a limited time after the current machine processing is completed, which is known as the transportation waiting constraint. Therefore, the problem of two-agent production and transportation coordination scheduling considering transportation batch and transportation waiting time constraints can be further investigated in a single machine environment in the future. Furthermore, the development of new efficient algorithms is a potential research topic for the problem of two-agent production and transportation coordination.

Acknowledgement: The authors wish to acknowledge the contribution of Liaoning Key Lab of Equipment Manufacturing Engineering Management, Liaoning Research Base of Equipment Manufacturing Development, Liaoning Key Research Base of Humanities and Social Sciences: Research Center of Micro-management Theory of SUT.

Funding Statement: This work was supported in part by the Project of Liaoning BaiQianWan Talents Program under Grand No. 2021921089, the Science Research Foundation of Educational Department of Liaoning Province under Grand No. LJKQZ2021057 and WJGD2020001, the Key Program of Social Science Planning Foundation of Liaoning Province under Grant L21AGL017, the special project of SUT on serving local economic and social development decision-making under Grant FWDFGD2021019, the “Double First-Class” Construction Project in Liaoning Province under Grant ZDZRGD2020037.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

- [1] M. M. E. Alemany, F. Alarcón, F. C. Lario and J. J. Boj, “An application to support the temporal and spatial distributed decision-making process in supply chain collaborative planning,” *Computers in Industry*, vol. 62, no. 5, pp. 519–540, 2011.
- [2] A. Agnetis, P. B. Mirchandani and D. Pacciarelli, “Nondominated schedules for a job-shop with two competing users,” *Computational & Mathematical Organization Theory*, vol. 6, no. 2, pp. 191–217, 2000.
- [3] O. J. Shukla, G. Soni and R. Kumar, “A review of multi agent-based production scheduling in manufacturing system,” *Recent Patents on Engineering*, vol. 15, no. 5, pp. 15–32, 2021.
- [4] Y. Q. Liu, S. D. Sun, X. V. Wang and L. H. Wang, “An iterative combinatorial auction mechanism for multi-agent parallel machine scheduling,” *International Journal of Production Research*, vol. 60, no. 1, pp. 361–380, 2022.
- [5] E. Guzman, B. Andres and R. Poler, “A MILP model for reusable containers management in automotive plastic components supply chain,” in *The Proc. of Working Conf. on Virtual Enterprises*, Saint Etienne, France, pp. 170–178, 2021.
- [6] Z. Y. Zhao, S. Y. Li, S. X. Liu, S. Liu and Y. F. Zhao, “A review of dynamic scheduling in iron and steel production process,” *Metallurgical Industry Automation*, vol. 46, no. 2, pp. 65–79, 2022.
- [7] B. Andres, R. Sanchis, J. Lamothe, L. Saari and F. Hauser, “Integrated production-distribution planning optimization models: A review in collaborative networks context,” *International Journal of Production Management and Engineering*, vol. 5, no. 1, pp. 31–38, 2017.
- [8] B. Andres and R. Poler, “Models, guidelines and tools for the integration of collaborative processes in non-hierarchical manufacturing networks: A review,” *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 2, pp. 166–201, 2016.
- [9] K. R. Baker and J. C. Smith, “A multiple-criterion model for machine scheduling,” *Journal of Scheduling*, vol. 6, no. 1, pp. 7–16, 2003.
- [10] A. Agnetis, P. B. Mirchandani and D. Pacciarelli, “Scheduling problems with two competing agents,” *Operations Research*, vol. 52, no. 2, pp. 229–242, 2004.

- [11] J. H. Liang, H. Y. Xue, D. Y. Bai and Y. H. Miao, "A branch and bound algorithm for the bi-agent single-machine scheduling problem with release dates," *Operations Research and Management Science*, vol. 28, no. 10, pp. 83–88, 2019.
- [12] X. Y. Wang, T. Ren, D. Y. Bai, C. Ezech, H. D. Zhang *et al.*, "Minimizing the sum of makespan on multi-agent single-machine scheduling with release dates," *Swarm and Evolutionary Computation*, vol. 69, pp. 100996–1–100996–19, 2022.
- [13] X. G. Zhang, J. Y. Liu and T. X. Cui, "Single machine KS fair pricing problem with two-agent based on the number of delayed jobs and the maximum cost function," *Journal of Chongqing Normal University (Natural Science)*, vol. 37, no. 1, pp. 16–21, 2020.
- [14] K. J. Zhao and X. W. Lu, "Two approximation algorithms for two-agent scheduling on parallel machines to minimize makespan," *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 260–278, 2016.
- [15] Q. Feng, W. P. Shang and C. W. Jiao, "Two-agent scheduling on a bounded parallel-batching machine with makespan and maximum lateness objectives," *Journal of the Operations Research Society of China*, vol. 8, no. 1, pp. 189–196, 2020.
- [16] C. He and X. X. Han, "Two-agent scheduling on an unbounded parallel-batching machine to minimize maximum cost and makespan," *Operations Research Transactions*, vol. 22, no. 3, pp. 109–116, 2018.
- [17] D. Y. Bai, A. Diabat, X. Y. Wang, D. D. Yang, Y. Fu *et al.*, "Competitive bi-agent flowshop scheduling to minimise the weighted combination of makespans," *International Journal of Production Research*, 2021. <https://doi.org/10.1080/00207543.2021.1923854>.
- [18] J. Fan, "Integrated production and delivery scheduling problem with two competing agents," in *The Proc. of IEEE Int. Conf. on Advanced Management Science*, Chengdu, China, pp. 157–160, 2010.
- [19] Y. Yin, Y. Wang, T. C. E. Cheng, D. Wang and C. C. Wu, "Two-agent single-machine scheduling to minimize the batch delivery cost," *Computers & Industrial Engineering*, vol. 92, pp. 16–30, 2016.
- [20] X. L. Qi and J. J. Yuan, "A further study on two-agent scheduling on an unbounded serial-batch machine with batch delivery cost," *Computers & Industrial Engineering*, vol. 111, pp. 458–462, 2017.
- [21] E. Guzman, B. Andres and R. Poler, "Models and algorithms for production planning, scheduling and sequencing problems: A holistic framework and a systematic review," *Journal of Industrial Information Integration*, vol. 27, no. 5, pp. 100287, 2022.
- [22] J. Mula, M. Díaz-Madroño, B. Andres, R. Poler and R. Sanchis, "A capacitated lot-sizing model with sequence-dependent setups, parallel machines and bi-part injection moulding," *Applied Mathematical Modelling*, vol. 100, pp. 805–820, 2021.
- [23] G. H. Zhou, R. Wang, P. Y. Jiang and G. H. Zhang, "Non-cooperation game model and hybrid adaptive genetic algorithm for job-shop scheduling," *Journal of Xi'an Jiaotong University*, vol. 44, no. 5, pp. 35–39, 2010.
- [24] Q. Q. Nong, S. J. Guo and L. H. Miao, "The shortest first coordination mechanism for a scheduling game with parallel-batching machine," *Journal of the Operations Research Society of China*, vol. 4, no. 4, pp. 517–527, 2016.
- [25] Y. F. Zhang, J. Wang and Y. Liu, "Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact," *Journal of Cleaner Production*, vol. 167, pp. 665–679, 2017.
- [26] J. Wang, Y. J. Peng and G. H. Luo, "Non-cooperative game theory and RFID based integrating rush orders flexible shop floor dynamic scheduling method," *Manufacturing Technology and Machine Tools*, no. 6, pp. 164–170, 2018.
- [27] L. Nie, G. H. Zhang, X. G. Wang and Y. W. Bai, "A Game-theory based optimization approach for job scheduling in virtual manufacturing network," *China Mechanical Engineering*, vol. 30, no. 12, pp. 1492–1497, 2019.