A Kind of Reinforcement Learning to Improve Genetic Algorithm for Multiagent Task Scheduling

Zhipeng Li,1 Xiumei Wei,2 Xuesong Jiang,2 and Yewen Pang2

1Institute of Machine Learning and Systems Biology, School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China
2School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China

Correspondence should be addressed to Xiumei Wei; wxm@qlu.edu.cn

Received 10 August 2019; Revised 22 November 2020; Accepted 24 December 2020; Published 13 January 2021

1. Introduction

The process industry provides important support for Chinese economic and social development. The integrated manufacturing system of modern process industry is one of the important competitive technologies to improve the competitiveness of processing enterprises [1]. In process manufacturing companies, the productivity of a company often depends on the level of automation of the company. The level of automation is largely dependent on the level of the intelligent control system. With the innovation and development of information technology, its control structure also updates and evolves with the development of computer technology. The current control system structure should take two aspects into account. On the one hand is production safety. The process industry is highly automated, and complicated chemical reactions occur during the production process, which is dangerous and therefore requires high system requirements. The system should have the ability to proactively defend and predict the risks in the production process, reducing the risk of downtime. On the other hand, it has system autonomy. Although the subsystems are actually divided according to the working conditions, information exchanges will be carried out at any time between different information systems. The information exchange is done automatically without the other human participation and control. Therefore, when building a model for a subsystem, it is required that the subsystem has the autonomous ability to control its own state and behavior autonomously. If you use object-oriented aspects to model from these two aspects, it is obviously not feasible and impossible to complete.

According to the characteristics and requirements of the control structure, we can use agent for modeling. Because agent has the characteristics of autonomy, responsiveness, adaptability, sociality, etc., this coincides with the requirements of the control system. As for the characteristics of the process industry system, agent technology can be regarded as a typical distributed multiagent system. The control system of the manufacturing process should have modular, distributed, and open features, as well as an integrated framework that is well connected to the application. The
distributed multiagent modeling is beneficial to the realization of this integration framework. At the same time, agent can be a good addition to the new integration platform.

Many scholars at home and abroad have introduced multiagent into manufacturing enterprises and done a lot of research.

Wang modeled the manufacturing process and proposed a multiagent system model to promote the transformation of enterprises into intelligent manufacturing. The physical entities of the model are abstracted into agents in the multiagent system model, and Petri nets are used in the system to express the characteristics of the agent [2].

Fu analyzed the China Mobile Multimedia Broadcasting System and built a multiagent system integrating management, control, and maintenance. It explored the basic structure of the model and the role of various parts of the system and explored the agent [3].

Han proposed a multiagent model that can realize distributed combination and management, which can quickly and scientifically complete decision making in a distributed environment [4].

Cao analyzed the structure of the control system of the current manufacturing enterprise and designed a multiagent system suitable for the hybrid enterprise of manufacturing enterprises. It introduced the workflow of each agent in this model in detail. It is a versatile system framework for most manufacturing companies [5].

Based on the traditional hierarchical control method and the distributed control method, Gao proposed a multiagent system for order production based on the analysis of the existing control structure and the control structure between the manufacturing system. The system has multiple dynamic logic units. The system divides the shop control system into several layers, which are the shop control layer, the field device layer, and the dynamic logic unit layer [6].

Xu analyzed the manufacturing process of the dyeing workshop and analyzed the manufacturing process of the dyeing workshop. Combined with the process requirements of the production process of the dyeing workshop, a dynamic scheduling model suitable for the workshop was constructed. In order to improve the global optimization ability of the scheduling system, a dynamic dyeing shop scheduling method suitable for the model is constructed by combining reinforcement learning and ant colony algorithm. The model and algorithm are applied to the printing and dyeing workshop for simulation research. The simulation experiment results show that the method is feasible and worthy of popularization and application [7].

In the production process of 2M1B production line for two devices and one inventory buffer, Wang and Wang proposed a multiagent reinforcement learning method for pipeline maintenance [8].

From a global perspective, many models cannot satisfy global optimization. At present, the process enterprise manufacturing process control model is divided into three types: centralized, hierarchical, and distributed. The centralized system has low fault tolerance and is prone to system failure. If the central control computer fails, it will cause the entire system to collapse, so centralized use is rarely used now. In the hierarchical hierarchy, the upper and lower layers are subordinate, and the upper and lower layers are highly resistant. From a partial point of view, it is still a centralized control structure, so there are still some defects. Compared with centralized and distributed, each subsystem is relatively independent, and each subsystem can achieve local optimization of each subsystem, but it is difficult to achieve global optimization and overall coordinated control of the whole system. In order to achieve the cooperation of each subsystem, it has the goal of good network bandwidth and efficient computing power. Therefore, it is the focus of this paper to construct a suitable intelligent coordination control model for the production process and use the reinforcement learning algorithm to realize the task scheduling of the production process.

To address the shortcomings of the above model, by analyzing the characteristics of the process industry manufacturing process and the requirements of collaborative control optimization in the manufacturing process, we designed a multiagent distributed hierarchical intelligent control model for the process industry manufacturing process. Aiming at the task scheduling problem of multi-agent systems, we use reinforcement learning ideas and improve genetic algorithms for multiagent production task scheduling. This article takes the production line processing production order as an example, with the goal of minimizing the completion time, and the open source dataset provided in OR-Library is used for experiments. The experimental results prove the effectiveness of the algorithm. The organization structure of the article is as follows. Section 2 introduces our multiagent distributed hierarchical intelligent control model. Section 3 introduces our improved QGA in detail. In Section 4, we carried out the task scheduling experiment and experiment result analysis of the multiagent system based on the QGA. Section 5 is the summary and outlook of this article.

2. Multiagent Model Construction in Manufacturing Process

According to the characteristics of Industry 4.0, the current multiagent control model is difficult to achieve global optimization control. In this paper, a multiagent distributed hierarchical control model for production process is proposed by combining the three architectures of multiagent system alliance, hierarchy, and distribution with artificial intelligence algorithm. This paper proposes a multiagent distributed hierarchical control model for production processes. The model is shown in Figure 1. The model is hierarchically layered and divided into upper and lower layers. The upper layer is the system management agent and the interface agent. The lower layer is the workshop control layer, and the workshop control layer consists of a shop control agent and several equipment agents. The model is a hierarchical structure from the perspective of organizational
structure, but it is actually a distributed intelligence control structure in production operation. The upper interface agent reflects the openness and expansibility of the multiagent system. Interface agent can connect ERP, process, man-machine interaction, and so on. The lower workshop control layer consists of multiple workshops, each equivalent to a small control system. They are independent and can communicate with one another via the bus. Each workshop has a workshop control agent and multiple equipment agents, each of which can communicate with each other. For the tasks issued by the system agent, the workshop agents in each subworkshop can cooperate with each other to realize the decomposition of the task agent. At the same time, each subworkshop can assign the assigned tasks to the related equipment agents as subtasks. After the equipment agent gets the tasks, the task cooperation can be completed through communication negotiation. For such a global optimization control model, searching for the best intelligent algorithm is the direction of our continuous efforts.

The multiagent distributed hierarchical control model divides the system into several subsystems, each of which is distributed. Considering that local resources and data are distributed independently, the workshop control agent of each subsystem contains all the information of the subsystem, which can conveniently control the local production. The local production control subsystem is also a multiagent system, which consists of multiple production equipment agents and workshop control agents. The device agent has independence and autonomy. Considering the different process requirements of the production process, each local subsystem and system management agent form a centralized control so that the subsystems are well managed.

The main structure and function of each part of the system management agent, interface agent, workshop control agent, and equipment agent in this model are introduced in detail.

2.1. System Management Agent. The system management agent mainly manages the entire system, and it has the highest management authority of the entire system. It can manage the access information through the interface agent. If the interface agent accesses the process information, it interacts with the interface agent to implement the management of the process data. If the interface agent accesses the ERP management system and man-computer interaction, it can interact with the interface agent to realize information management. The process agent can also interact and communicate with other agents to realize the data management and production process monitoring of the production process. The system management agent also contains intelligent modules to realize intelligent management of the entire system. The system management agent is shown in Figure 2.
2.2. **Interface Agent.** Interface agent can realize the function expansion of the whole multiagent system. The interface agent can be connected to process information, ERP system, man-computer interaction, and so on. The interface agent can well reflect the extensibility and development of the entire system. If there is no interface agent, when the system needs to add modules or needs to add functional requirements, then it may be necessary to redesign the multiagent distributed hierarchical intelligent control model, which is not very friendly to production. The structure of the interface agent is shown in Figure 3.

2.3. **Workshop Control Agent.** The workshop control agent is located in the workshop control layer of the model. The workshop control layer consists of multiple workshop control subsystems. Each workshop control subsystem has a workshop control agent and multiple equipment agents. Workshop control agent is the administrator of local subsystem, which has the highest level of authority of local subsystem. Workshop control agent plays the role of system management agent management and control equipment agent bridge. On the one hand, the workshop control agent accepts macrocontrol or static planning from the system management agent. It manages the equipment agent of the workshop and accepts the tasks assigned by the system management agent. It completes task assignment and scheduling for each device agent by using intelligent algorithms. On the other hand, the system manages the information required by the agent. Through the workshop control agent transfer, it can monitor the task execution and resource utilization and feed back to the system management agent to realize the optimal scheduling, status evaluation, and resource monitoring of the control system. The workshop control agent structure is shown in Figure 4.

2.4. **Equipment Agent.** Equipment agent is at the bottom of the workshop control layer. It interacts with the workshop control agent to perform the assignment of the workshop control agent to its own tasks. It monitors the equipment in the production process and collects and analyzes the data from the production process. It predicts the resources required to complete a task and reports the relevant results to the workshop control agent. The equipment agent also contains an intelligent algorithm module, which facilitates the device agent to predict the resources required for the processing task. The equipment agent also has a distributed database, which is convenient for storing the data collected by itself. The structure of the convenient agent is shown in Figure 5.

### 3. Multiagent Task Scheduling Based on QGA

Task scheduling is also one of the important contents of the multiagent system. Task scheduling is also an important component of process enterprise production process management. Rational scheduling of production tasks plays an important role in improving the productive efficiency of enterprises. Job shop is a strong NP-hard problem as a production task scheduler. Since the issue was raised, people have been researching. The process industry’s production process is usually continuous, with uncertainties, nonlinearity, multiple objectives, multiple constraints, and other characteristics. Process industrial production process is a NP-hard problem [9]. Many researchers have applied heuristic algorithms to solve such NP-hard problems. The
most used algorithm is the genetic algorithm (GA) [10]. Genetic algorithm is widely used in solving complex problems such as nonlinearity and optimization [11, 12]. However, genetic algorithms are also flawed, such as the disadvantages of falling into local optimum and low computational efficiency when solving large-scale task scheduling [13]. So, looking for a more efficient algorithm is the direction we are always looking for.

Reinforcement learning is a semisupervised algorithm. It emphasizes the process of interaction between the agent and the environment without the interference of the external environment. Reinforcement learning provides new solutions and methods for multiagent task scheduling. Many scholars have applied the Q-learning algorithm in reinforcement learning to solve large-scale complex problems and have achieved good results [14, 15]. However, Q-learning also has some shortcomings, such as its convergence speed needs to be improved and its Q table storage information is limited. So, this paper proposes a new solution. Combining the characteristics of GA algorithm and Q-learning algorithm, a genetic algorithm based on Q-learning (QGA) is proposed. The simulation experiment analysis is carried out using the standard task scheduling dataset in OR-Library [16]. The experimental results demonstrate the superiority of the algorithm.
3.1. Description of Manufacturing Process Task Scheduling Problem. Manufacturing process task scheduling refers to spatial, temporal planning, scheduling, and scheduling of multiple production tasks under the conditions of meeting process requirements and existing production equipment requirements. Since the process industry produces products or multiple processes of the same product need to share resources and equipment, it is necessary to rationally plan production through algorithms. The purpose of production task scheduling is to rationally plan and allocate resources, determine the processing time and the sequence of products in different equipment, and improve production efficiency. Process industry manufacturing process task scheduling can be described as follows: n jobs should be processed on m machines, while minimizing job completion time with the following constraints and assumptions:

1. Each machine can only perform one operation at a time.
2. The operation of the job can only be performed by one machine at a time.
3. Once you start working on the machine, you cannot interrupt it.
4. No other job operations can be performed until the previous operation is completed.
5. There is no alternate route, that is, the job operation can only be performed in one type of machine, and the operation processing time and the number of operable machines is known in advance.

3.2. Mathematical Model of Manufacturing Process Task Scheduling Problem. To facilitate the above description of the problem, we define the following mathematical symbols:

- \( n \): indicates the number of tasks.
- \( m \): indicates the number of machines.
- \( \sigma \): indicates the operation of the task.
- \( T \): indicates the set of tasks \( T = \{ T_i \} | i = 1, 2, \ldots, n \}; \)
- \( T_i \) represents the i-th task.
- \( M \): indicates the collection of machines \( M = \{ M_k | k = 1, 2, \ldots, m \} \); \( M_k \) represents the k-th machine.
- \( O_i = \{ O_{i1}, O_{i2}, \ldots, O_{im} \} \): the operation sequence set of the task \( i \); \( O_{ij} \) represents the machine number selected by the j-th process of the task \( i \), \( j = 1, 2, \ldots, m \).
- \( H_i = \{ H_{ij1}, H_{ij2}, \ldots, H_{ijm} \} \); \( H_{ij} \) represents operating time required for the j-th process of task \( i \), \( j = 1, 2, \ldots, m \).

Next we build the task scheduling function model [17, 18]:

\[
\min \left\{ \max_{1 \leq k \leq m} \left\{ \max_{1 \leq i \leq n} c_{ik} \right\} \right\}, \quad (1)
\]

\[
s.t., \quad (2)
\]

\[
c_{ik} - f_{ik} + M(1 - a_{iek}) \geq c_{i\sigma}, \quad i = 1, 2, \ldots, n; e, j = 1, 2, \ldots, m, \quad (3)
\]

\[
c_{dk} - c_{ik} + M(1 - b_{idk}) \geq f_{jk}, \quad d = 1, 2, \ldots, n; c_{ijk} \geq 0, \quad (4)
\]

\[
a_{iek} = \begin{cases} 1, & \text{if the machine } e \text{ processes the workpiece } i, \\ 0, & \text{not the above situation}, \end{cases} \quad (5)
\]

\[
b_{idk} = \begin{cases} 1, & \text{if task } i \text{ is processed on machine } k \text{ before task } d, \\ 0, & \text{not the above situation}. \end{cases} \quad (6)
\]

The first of the above is the total objective function, which minimizes the completion time of all tasks. The
3.3. Genetic Algorithm. The genetic algorithm [19] was proposed in 1975. Its proposal mainly draws on the ideas of natural selection and genetic evolution in the biological world. The solution process of the genetic algorithm is to use the ideas of reproduction, selection, crossover, and mutation in the natural world to carry out continuous iteration and select the best individual from the population. Compared with other heuristic algorithms, genetic algorithm can break through the limitation of search area and realize complete exploration of solution space. The genetic algorithm uses the fitness function as an evaluation index. So, its search process reduces the dependence on man-machine interaction. Therefore, genetic algorithms are favored in engineering optimization.

3.3.1. Principle of Genetic Algorithm. Genetic algorithm is based on the genetic characteristics of nature combined with natural selection. It implements the mapping of problem solving to natural populations by coding the problem to be solved. The genetic algorithm first initializes the population, follows the idea of evolution in nature, and then performs operations such as crossover and mutation to generate new populations. Set algebra of population reproduction to control the evolution process of the genetic algorithm. If the number of iterations is reached, the individuals with high fitness will be left. Decoding this highly adaptive individual will result in an optimal solution to solve the problem [20].

3.3.2. Encoding and Decoding Operations. Coding is the spatial mapping that the solution space to solve the problem can be processed by the genetic algorithm. Encoding is generally in binary form. If the search space is turned into a feasible solution space, this process is decoded. Encoding and decoding are indispensable parts of solving problems using genetic algorithms. At present, it is more common to encode with a binary number or the like.

Binary is encoded using a binary number of 01. The genetic algorithm encodes the feasible solution of the problem in binary. The binary encoding method is simple and flexible. If the accuracy of the solution is high, the length of the chromosome will be very long. If the space for solving the problem is increased, it will not be good for obtaining the best solution. At this time, we can use real numbers to encode.

3.3.3. Genetic Operator. Genetic operators are an important way to complete population evolution. It is also an important part of genetic algorithms. Selection, crossover, and mutation are common operators of genetic algorithms.

(1) Selection Operator. The selection operator is usually selected from the population as a next generation population with high fitness. In the selection operation, the fitness function is generally defined in advance, and then the chromosome with high fitness is selected. These chromosomes will undergo subsequent genetic and evolutionary operations, and those with lower fitness will be discarded. Choosing an operator is an operation of “survival of the fittest.” Generally, elite reservation and roulette are selected as selection operators.

(2) Crossover Operator. The crossover operator is two chromosomes selected by the parent in some way. Exchange some of the genes on a chromosome according to certain rules to produce new chromosomes. This rule is usually considered as the crossover probability. The magnitude of this probability value determines the likelihood of an exchange-gene operation occurring in the population. Crossover is actually a way of genetic recombination, in which genes come from genes on the previous generation of chromosomes. Crossing is a way to create new individuals. At present, common crossover operations mainly include arithmetic crossover, multipoint crossover, etc..

(3) Mutation Operator. Mutations are similar to genetic mutations in biology. Variation in genetic algorithms refers to the process by which the value of a gene at a certain position on a chromosome becomes another value. The magnitude of this probability value indicates the change in the gene in the chromosome. Mutations can generate new genes. Mutation operation can well prevent the premature convergence of genetic algorithms. Mutation operator is a relatively important genetic operator.

3.3.4. Fitness Function. In order to choose a better chromosome from the population, this requires designing the fitness function. The fitness function is usually the objective function, which is used to judge the quality of the population. It is an important reference source for natural selection.

The quality of the fitness function design directly affects the quality of the solution and the speed at which the algorithm converges. The fitness function should be able to better reflect the quality of the chromosome and need to meet continuous, single-valued, and non-negative conditions and minimize the calculation.

Generally, genetic algorithm is to convert the target function of the problem to the fitness function. The fitness function is a good evaluation of the quality of individuals in the population. Fitness can guarantee the reproduction opportunities of good individuals and preserve their good characteristics. Therefore, the design and selection of fitness function are related to the quality of the whole solution.
Good fitness function can accelerate the convergence of the algorithm, so the design and selection of fitness function in genetic algorithms are also very important.

3.4. Q-Learning Algorithm. The Q-learning algorithm is one of the classic algorithms in reinforcement learning. It is a model-free learning method. It can be a process for the agent to gain experience through continuous learning in the environment. The Q-learning algorithm considers the interaction between agent and environment as a Markov decision process. This process is the process of the agent in the current state and the selected action, determining a fixed state transition probability, reaching the next state, and getting an instant reward. The goal of the Q-learning algorithm is to find a strategy that maximizes the cumulative rewards obtained.

When building the Q-learning algorithm, we first need to build an instant reward matrix R. The instant reward matrix R guides the agent to select actions, thereby obtaining a Q matrix, and the Q value is updated as follows:

\[
Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right].
\]

The Q-learning algorithm is as follows (Algorithm 1) [21].

3.5. QGA. The combination of reinforcement learning algorithm and genetic algorithm has been widely concerned by researchers at home and abroad since the 1980s. There are three main ideas in which reinforcement learning and genetic algorithms are combined. One is reinforcement learning and genetic algorithm for the same goal division of labor. One is to introduce genetic algorithm and reinforcement learning algorithm into the multiagent system. It uses genetic algorithm to learn the interaction strategy between agents in the multiagent system and uses genetic algorithm to complete the evolution of agent. The third is the genetic operator of the adaptive control genetic algorithm. This is a deep, intrinsic inner fusion.

The third fusion idea is based on genetic algorithm and reinforcement learning algorithm. This paper builds an algorithm which combines Q-learning with genetic algorithm. The main idea of the algorithm is to regard the gene space of genetic algorithm problem as the action strategy space of Q-learning algorithm. The fitness function that performs an action within the gene space is considered to be the reward that is obtained by performing the action. This makes it easy to translate the problems of genetic algorithms into reinforcement learning problems.

The basic idea of the QGA is to first encode the feasible solution of the problem. In binary form, the encoded gene space is \( x_g = \{0, 1\}^L \), the chromosome is used to represent the feasible solution, and \( L \) is the coding length of the gene on the chromosome.

The pseudocode is given in Algorithm 2.

It can be seen from the pseudocode of the algorithm that the Q-learning algorithm selects a good action genetic algorithm to find a good structure. The selection action in Q-learning corresponds to the genetic selection operator in genetic algorithm. The strategy selection in reinforcement learning corresponds to the mutation operation in the genetic algorithm. This achieves a deep integration of GA and Q-learning. In the next section, we conduct the test with QGA.

4. Task Scheduling Based on QGA

4.1. Description of Task Scheduling Strategy for Multiagent Distributed Hierarchical Intelligent Control Model. Enter the task by the interface agent or system management agent. After the system management agent accepts the task, it searches the knowledge base according to the task requirements and characteristics and then the process flow and requirements of the interface agent to access the production task. After receiving the process data information, the system management agent decomposes the task into subtasks according to the different task processing capabilities of each subsystem. Then, it assigns subtasks to subsystems with different production capacities and production requirements. After each subsystem receives the subtask assigned by the system management agent, the workshop space agent further decomposes the task assigned by itself. The shop control agent realizes the assignment of tasks according to the QGA, and the equipment agent then completes the production according to the assigned tasks. The multiagent distributed hierarchical intelligent control model task scheduling strategy is shown in Figure 6.  

4.2. Simulation. In this section, we conduct experiments to verify the effectiveness of the improved algorithm. We conducted simulation experiments on an experimental machine with Intel Xeon (R) CPU E7-8867 v4 @ 2.00 GHz, 80, GPU Nvidia GTX 1080Ti, memory 62.8GiB, and disk 698.4 GB. In order to verify the validity of the QGA, we selected the international standard job shop dataset provided by OR-Library [8] for simulation experiments. There are 10 tasks (\( j_{12} \ldots j_{10} \)) and 10 operating equipment (\( m_1, m_2, \ldots, m_{10} \)) production task scheduling problems; the experimental data are shown in Tables 1 and 2.

The relevant parameter settings of GA in the experiment: the initial population size is \( N = 200 \), the crossover probability is set to 0.8, the mutation probability is set to 0.2, and the number of iterations is 200. The relevant parameters of the QGA algorithm are set as follows: greedy strategy selection probability \( \varepsilon = 0.2 \), learning rate \( a = 0.1 \), \( \gamma = 0.9 \), QGA maximum iteration number is 40,000. We have drawn several common algorithms for makespan (as shown in Figure 7) and QGA on task scheduling for Gantt charts (as shown in Figure 8). It can be seen from the comparison of several common algorithms, which proves the effectiveness of the proposed algorithm. The main idea of the QGA is to regard the gene space of the genetic algorithm problem as the action strategy space of the Q-learning algorithm, and the fitness function of the action executed in the gene space is regarded as the reward for
(1) Initialize $Q(S_t, A_t) \leftarrow$ arbitrary value
(2) Repeat (for each episode):
   Initialization state $S_t$
   Repeat (for each step of episode):
      Select action $A_t$ according to $\varepsilon$-greed strategy, get instant reward $R_t$ and next state $S_{t+1}$,
      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha (R_{t+1} + \gamma \max Q(S_{t+1}, a) - Q(S_t, A_t))$
   $S_t \leftarrow S_{t+1}$
   Until $S_t$ is terminated,
   Until all $Q(S_t, A_t)$ converges.
(3) Output final policy $\pi$.

Algorithm 1: Pseudo code of the Q-learning algorithm.

Set the gene encoding length $L$
For all $j = 1, \ldots, 2^L$
         $Q(a_j) = 0$
         $k = 1$
Repeat
   Use greedy strategy $\varepsilon$ to select action $a$
   $r = \text{fitness}(x(a))$ #Calculate the fitness of $x$
   $Q(a_{jk}) \leftarrow Q(a_{jk}) + \alpha (r + \gamma \max Q(a_{jk}))$
   If $\text{best}_r > r$ then
      $\text{best}_r = r, \text{best}_x = x$ #Elite retention
      $k = k + 1$
   Until the termination condition is met,
Output $\text{best}_x$.

Algorithm 2: Pseudo code of the proposed QGA algorithm.

Figure 6: Task scheduling process when multiagent works normally.

Table 1: $10 \times 10$ task scheduling problem operation schedule.

| Time | O1 | O2 | O3 | O4 | O5 | O6 | O7 | O8 | O9 | O10 |
|------|----|----|----|----|----|----|----|----|----|-----|
| J1   | 29 | 78 | 9  | 36 | 49 | 11 | 62 | 56 | 44 | 21  |
| J2   | 43 | 90 | 75 | 11 | 69 | 28 | 46 | 46 | 72 | 30  |
| J3   | 91 | 85 | 39 | 74 | 90 | 10 | 12 | 89 | 45 | 33  |
| J4   | 81 | 95 | 71 | 99 | 9  | 52 | 85 | 98 | 22 | 43  |
| J5   | 14 | 6  | 22 | 61 | 26 | 69 | 21 | 49 | 72 | 53  |
| J6   | 84 | 2  | 52 | 95 | 48 | 72 | 47 | 65 | 6  | 25  |
| J7   | 46 | 37 | 61 | 13 | 32 | 21 | 32 | 89 | 30 | 55  |
| J8   | 31 | 86 | 46 | 74 | 32 | 88 | 19 | 48 | 36 | 79  |
| J9   | 76 | 69 | 76 | 51 | 85 | 11 | 40 | 89 | 26 | 74  |
| J10  | 85 | 13 | 61 | 7  | 64 | 76 | 47 | 52 | 90 | 45  |
### Table 2: 10 × 10 task scheduling problem operation sequence.

| Time | O1 | O2 | O3 | O4 | O5 | O6 | O7 | O8 | O9 | O10 |
|------|----|----|----|----|----|----|----|----|----|-----|
| /1   | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10  |
| /2   | 1  | 3  | 5  | 10 | 4  | 2  | 7  | 6  | 8  | 9   |
| /3   | 2  | 1  | 4  | 3  | 9  | 6  | 8  | 7  | 10 | 5   |
| /4   | 2  | 3  | 1  | 5  | 7  | 9  | 8  | 4  | 10 | 6   |
| /5   | 3  | 1  | 2  | 6  | 4  | 5  | 9  | 8  | 10 | 7   |
| /6   | 3  | 2  | 6  | 4  | 9  | 10 | 1  | 7  | 5  | 8   |
| /7   | 2  | 1  | 4  | 3  | 7  | 6  | 10 | 9  | 8  | 5   |
| /8   | 3  | 1  | 2  | 6  | 5  | 7  | 9  | 10 | 8  | 4   |
| /9   | 1  | 2  | 4  | 6  | 3  | 10 | 7  | 8  | 5  | 9   |
| /10  | 2  | 1  | 3  | 7  | 9  | 10 | 6  | 4  | 5  | 8   |

### Figure 7: Four algorithms of makespan.

#### Figure 8: 10 × 10 scheduling problem QGA task scheduling Gantt chart.

The Gantt chart illustrates the scheduling of tasks on 10 machines over 10 operations. Each bar represents the duration of a task, and the sequence highlights the order in which tasks are executed. The makespan for this configuration is 930 units, with individual task durations as follows: $j_1 = 915, j_2 = 915, j_3 = 927, j_4 = 856, j_5 = 902, j_6 = 530, j_7 = 839, j_8 = 892, j_9 = 792, j_{10} = 930$. This chart visually demonstrates the efficiency of the QGA algorithm in task management, where each bar color corresponds to a specific operation.
performing the action. Q-learning algorithm is responsible for selecting good actions. Genetic algorithm is responsible for finding good structures, so QGA is easier to reach the optimal value in a short time, so its makespan is the smallest.

5. Conclusions

This paper analyzes the defects of traditional manufacturing process control system structure. Through in-depth research on the manufacturing process of the process industry, combined with multiagent systems and reinforcement learning technology, a multiagent distributed hierarchical intelligent control model for the process industry is constructed. In order to solve the multiagent task assignment problem in the multiagent distributed hierarchical control model, the idea of reinforcement learning and genetic algorithm is combined, and the improved QGA is used for task scheduling. Taking the production order processing in the production line as an example, the improved QGA and several common task scheduling algorithms are used to experiment on the task assignment. The experimental results prove the effectiveness of the algorithm.

For task assignment among multiagent, the QGA we built also has some defects. For example, when the length of gene space coding on the chromosome is too long, our algorithm needs a large Q table, which is not desirable. If the action space is large, each action cannot be accessed multiple times. We need to do further research to improve the performance of the algorithm. In addition, we can also combine graph neural networks [22–28] with traditional evolutionary algorithms and reinforcement learning algorithms for task scheduling.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the National Key R&D Program of China (no. 2019YFB1707004).

References

[1] T. Chai, “Modern integrated manufacturing system for process industry based on three-layer structure,” Control Engineering, vol. 9, pp. 1–6, 2002.
[2] Z. Wang, “Research on multi-agent model of intelligent manufacturing system,” China Mechanical Engineering, vol. 14, pp. 1390–1393, 2003.
[3] W. Fu, “Intelligent control-maintenance-management system multi-agent model,” Computer Integrated Manufacturing System, vol. 8, pp. 786–791, 2002.
[4] X. Han, “Integrated decision support system for complex problem solving,” Computer Integrated Manufacturing System, vol. 11, pp. 109–115, 2005.
[5] C. Cao, Research on Key technologies of integrated manufacturing execution system (I-MES) based on multi-agent, Nanjing University of Aeronautics and Astronautics, Nanjing, China, PhD Type, 2003.
[6] W. Gao, Research on control structure and control technology of workshop manufacturing system based on multi-agent, Hefei University of Technology, Hefei, China, PhD Type, 2003.
[7] X. Xu, “Application of multi-agent dynamic scheduling method in dyeing shop scheduling,” Computer Integrated Manufacturing System, vol. 16, pp. 611–620, 2010.
[8] C. Qi, “Pipeline maintenance strategy based on multi-agent reinforcement learning,” Journal of Systems Engineering, vol. 28, pp. 702–708, 2013.
[9] X. Zhao, “Summary of process industry production scheduling problems,” Chemical Automation and Instrumentation, vol. 31, pp. 8–13, 2004.
[10] N. Geng, “Task scheduling and assignment based on adaptive selection genetic algorithm,” Computer Engineering, vol. 34, no. 43–45, p. 65, 2008.
[11] R. Deepa, "An efficient task scheduling technique in heterogeneous systems using self-adaptive selection-based genetic algorithm in parallel computing in electrical engineering," in Proceedings of the International Symposium 2006. PAR ELEC 2006, pp. 343–348, Białystok, Poland, September 2006.
[12] L. Du, “A research on the improvement of task scheduling algorithm based on Q learning,” in Proceedings of the 20th National Conference on Computer Technology and Applications (CACIS), pp. 236–240, Nanning, China, February 2020.
[13] S. Chen, "Multi-step Q learning algorithm and performance simulation based on Metropolis criterion," Journal of System Simulation, vol. 19, pp. 1284–1287, 2007.
[14] Y. He, “A scheduling method for reducing energy consumption of machining job shops considering the flexible process plan,” Journal of Mechanical Engineering, vol. 52, no. 19, pp. 168–179, 2016.
[15] X. Shao, “Shuai rate; majike. modeling of multi-target composite AGV dispatching system and its application in power metering verification,” Jiangsu Electrical Engineering, vol. 35, pp. 24–27, 2016.
[16] J. E. Beasley, "OR-library: distributing test problems by electronic mail," Journal of the Operational Research Society, vol. 41, no. 11, pp. 1069–1072, 1990.
[17] L. Wang, "Workshop scheduling and its genetic algorithm," Industrial Engineering and Management, vol. 9, p. 78, 2004.
[18] H. Zhang, "Research on job-shop scheduling problem based on genetic algorithm," Journal of Shenyang Ligong University, vol. 35, pp. 60–64, 2016.
[19] C. E. Taylor, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, Vol. 69, University of Chicago Press, Chicago, IL, USA, 1994.
[20] D. Gong, J. Sun, and Z. Miao, "A set-based genetic algorithm for interval many-objective optimization problems," IEEE Transactions on Evolutionary Computation, vol. 22, no. 1, pp. 47–60, 2018.
[21] C. J Watkins and P. Dayan, "Q-learning," Machine learning, vol. 8, no. 3–4, pp. 279–292, 1992.
[22] J. Zhou, “Graph neural networks: a review of methods and applications,” 2018, https://arxiv.org/abs/1812.08434.
[23] S. Garg, A. Bajpai, and A. Mausam, “Symbolic network: generalized neural policies for relational MDPs,” 2020, https://arxiv.org/abs/2002.07375.

[24] M. Qu, “Few-shot relation extraction via bayesian meta-learning on relation graphs,” 2020, https://arxiv.org/abs/2002.07375.

[25] J. Jiang, “Graph convolutional reinforcement learning,” 2018, https://arxiv.org/abs/1810.09202.

[26] Y. Chen, L. Wu, and M. J. Zaki, “Reinforcement learning based graph-to-sequence model for natural question generation,” 2019, https://arxiv.org/abs/1908.04942.

[27] Z. Zhang, P. Cui, and W. Zhu, “Deep learning on graphs: a survey,” IEEE Transactions on Knowledge and Data Engineering, vol. 99, p. 1, 2020.

[28] J. Skarding, B. Gabrys, and K. Musial, “Foundations and modelling of dynamic networks using dynamic graph neural networks: a survey,” 2020, https://arxiv.org/abs/2005.07496.