Agent-based Simulation of Job Shop Production

Tao Zhang*, Shufang Xie, Oliver Rose

Fakultät für Informatik, Universität der Bundeswehr München,
Werner-Heisenberg Weg 39, D-85577 Neubiberg, Germany; *tao.zhang@unibw.de

SNE 29(3), 2019, 141 - 148, DOI: 10.11128/sne.29.tn.10487
Received: June 15, 2019 (Selected ASIM SST Hamburg 2018 Postconf. Publ.), Accepted: July 22, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna,
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The most important design principles of Industry 4.0 are decentralization and intelligence. The entities, like resources and materials, can make decisions on their own by means of cyber-physical systems. For research purposes, we unify cyber entities and physical entities and build an agent-based simulation model. The agents learn knowledges offline during simulation runs and become smarter and smarter. The model will finally connect to the physical systems and carry out online decision-making. The study is the first stage of the whole project. A framework for the agent-based simulation is developed and an agent-based model of the job shop production including release agents, machine group agents, and job agents are built.

Introduction

Industry 4.0 [1] is currently one of the most frequently discussed topics among practitioners and academics. Taking advantages of cyber-physical systems [2], the internet of things [3] and the internet of services, it enables faster, more flexible, and more efficient processes to produce higher-quality goods at reduced costs in an environmental-friendly and resource-conserving way. This in turn will increase manufacturing productivity, shift economics, foster industrial growth, and modify the profile of the workforce – as ultimately change the competitiveness of companies and regions. Most of studies and industry cases about Industry 4.0 still stay at a very basic level, such as studies and cases about sensors, communications, and automations. These are certainly foundations of Industry 4.0. However, once these foundation works are done, there will be a very urgent requirement for more effective decision-making methods on the platform of Industry 4.0.

As we all know, the most important design principles of Industry 4.0 are decentralization and intelligence [4]. The cyber-physical systems can make decisions on their own. To make the cyber-physical systems intelligent, the systems must have abilities to learn knowledge. Online learning and offline learning are two common learning techniques. For the research purposes, the online learning is unpractical. In the study, we will create an agent-based simulation model (cyber system) and the agents (cyber entities) will learn to make decisions from the simulation. At last, the model will connect to the physical systems and form the cyber-physical systems.

The study is an extension of our previous works [5, 6] in which we realized that the agent-based simulation (ABS) is less efficient when being used into a non-real-time system even though it can be speeded up by giving a timescale. So, to speed up the ABS, we introduced a process-interaction worldview (PIW) originated in the discrete event simulation to the ABS.

In this study, we are going to design a framework for the ABS with the PIW and build a model of the job shop production based on the framework. In the model, each agent is related to one physical entity. The agents can make decisions by either some decision rules or their knowledge learnt from some other independent simulation-based approaches that we have proposed [7-9].

1 Agent-based Simulation with PIW

The ABS&PIW approach [5, 6] was proposed on the basis of the agent-based model(ABM). We provided a four-tuple $SIM = (I, TM, ABM, O)$ with elements inputs (I), time manager (TM), ABM, and outputs (O) to describe the approach strictly. The procedure of the approach is presented mathematically in which the simulation clock advances in a sequence of activation points and all concurrent activations are activated at a time, and associated agents respond in parallel.

To reuse the code and make the development of the ABS with the PIW easy, in this section we will give a general idea that how to develop a framework for the

ABS&PIW. We will design individual agent and time manager, and draw a static structure of the framework.

1.1 Individual Agent

The structure of the agent is shown in Figure 1. The agent is composed of attributes, an initialization method, a behavior controller, and a message handler.

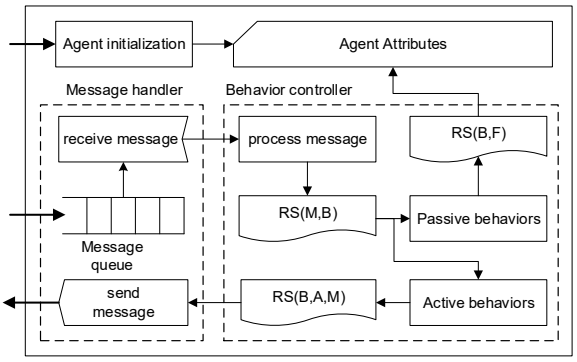

Figure 1: Structure of individual agent.

Agent Initialization

Agent initialization, which is applied to set the values of attributes when an agent is created, is the only possibility to change the state of an agent directly by the external environment and enables the simulation to start in any state of the agent. There are also many other common methods (e.g., reset, start, and stop) for controlling agents, but they are not visible to the environment and called only by agents themselves.

Behavior Controller

The behavior controller decides which behavior to be executed when receiving a message. The decision is made depending on the relationships RS. Here we just summarize them in three types of relationships: RS (M, B) is the relationship between messages (in) and behaviors; RS (B, F) maps attributes to behaviors; RS (B, A, M) denotes the relationship between behaviors and messages (out). The behavior controller can also control behaviors, such as adding behavior, removing behavior, and changing behavior's state according to the environment.

Message Handler

Through the message handler, the agent communicates with other agents. A message handler includes one message queue and two methods: "send" and "receive." Messages from other agents will be stored in the message queue and received by "receive" method. Similarly, the agent can send the messages to other agents by using the method "send." These two methods are behaviors of the agent.

1.2 Time Manager

In order to keep consistent with the agent-based model, the time manager is also developed as an agent to be in charge of advancing time, activating agents, and managing activation points. The time manager extends the class of agents, and Figure 2 shows its structure.

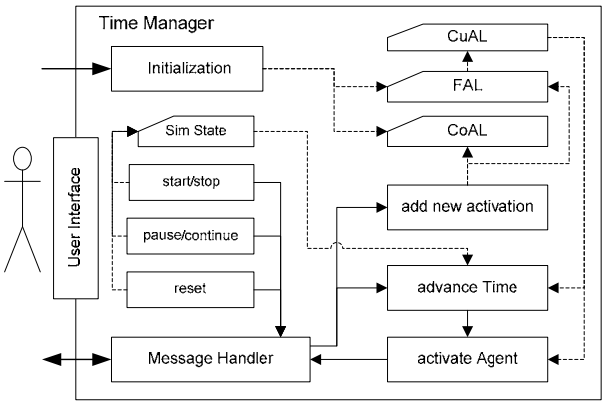

Figure 2: Structure of the time manager.

Behavior of the Time Manager

There are three main behavior types: advance simulation clock, receive activation point and activate agent. A user interface is provided to control the simulation. Before the simulation runs, at least one activation point needs to be given in advance. Activation points are conveyed in the form of messages between the time manager and other agents. After the activation, the time manager is blocked until a new activation point is received or the ABM notifies it to advance the simulation clock when the model state is ready.

Activation Point Lists in the Time Manager

Three activation point lists are created in the time manager: conditional activation list (CoAL), future activation list (FAL) and current activation list (CuAL).

The time manager puts new received activation points into the appropriate list. The earliest activation points are moved from the future list to the current list every time the simulation clock advances. After activation, the current list is cleared. Conditional activation points are tried again and again and removed from the list when the conditions are met.

1.3 Agent-based Model

The agent-based model includes the agent environment, the agent manager, and a set of agents (shown in Figure 3).

The agent environment is a medium for communication among the agents. The time manager and the agents send or receive activation points in the environment too. The agent manager is in charge of all agents and provides the model state to the time manager when the simulation is running. A new agent needs to register with the manager. The agents report their physical states and updated flags when they become blocked. The time manager also needs to register. In the model, each agent has a unique name which is used to specify the target agent in the communication.

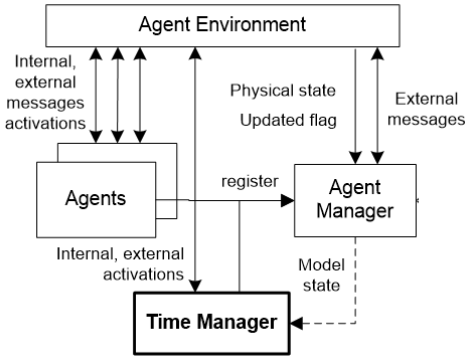

Figure 3: Structure of the Agent-based Model.

2 Agents in Job Shop Production

Agent-based modeling enables us to model the job shops more realistically and systematically comparing to other modeling methods, like discrete event simulation modeling, Petri nets, and so on. We know that an agent-based model is composed of agents and their relationships. Thus, we should determine who the agents are in the job shop production first. Obviously, material plays a leading role in the job shops. However, materials are usually transformed to other types of materials in shops, and their lifecycle is very short. Paying attention to them makes no sense.

We usually focus on jobs which organize all materials that one product needs in a serial of sequential operations. Besides, machines and transporters also make up a large proportion of the job shops. Because we assume that the transportation capacity is finite, the transporters are out of scope. The transportation in our study is treated as a delay. We only consider the machines here. As a type of flow, the material flow must have one or more sources and let jobs derive from it. The sources are usually job pools. Each type of job, i.e., product, is connected to a job pool.

There is also a valve controlling the flow from each pool. In the job shops, this valve is a job release procedure. The release procedure is what we will concentrate rather than the job pools. To date on the basis of analysis above, we list three entities concerned in the job shops: job, machine group, and release procedure. We will model these three entities as agents in the agentbased model: release agent, job agent, and machine agent. In this section, we will describe how to create them on the basis of the framework, including their attribute and behavior definitions, communication and cooperation design, and simulation-related delay and activation design.

2.1 Release Agent

In a release agent, there are one piece of product data, multiple release policies, and one buffer. The release agent creates the job agents who are given in the product data based on a release policy. The buffer is located behind the release agent (see Figure 4). If the corresponding buffer is full in the first operations, the released job cannot start the operations. It will be blocked and stays in the release buffer until the buffer of the first operations has free space. If the buffer of the release agent is full, the release agent stops releasing until the buffer is not fully occupied.

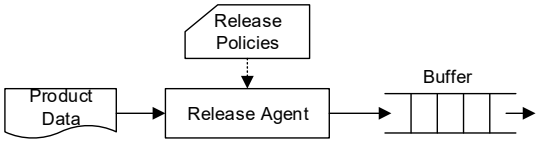

Figure 4: The release agent with one buffer.

Product Data in the Release Agent

At the beginning of the simulation, the simulator reads product and process data from files and generates each product a release agent.

Each product has a unique name and five data elements: a production probability, process, priority, the release interarrival time, and the target work-in-process (WIP) level. The process element is the name of the process flow which is defined in the process file. Once the job agents are created, they will obtain their process flows from the product. The priority specifies the urgency of products. The value of the priority is from 0 to 1 and is used by some dispatch rules. The elements of interarrival time and WIP level are used by the release policies.

Release Policies of the Release Agent

The release policy is the criterion which decides when to release jobs. The release policies are a very common way to solve the job release problem. For now, the release agent has three very common types of release policies: constant interarrival time (CONINT), constant WIP level (CONWIP), and avoiding starvation (AS). CONINT releases the job in the same interval. To the policy of CONWIP, one interval time is still needed to be designated before reaching the target WIP. After reaching the target WIP, the policy of CONWIP takes effect. If the WIP level is less than the target WIP level, a new job is released. In the AS policy, a target buffer size is set for a bottleneck machine group. When the buffer size of the bottleneck is more than the target value, the releasing stops. When the buffer size is less than the target, new jobs are released. The quantity of the released jobs is the difference between the buffer size and the target value.

Communication with the Time Manager and Other Agents

If the release policy is CONINT, "releasing job" is an activation point which will be sent to the time manager and put in the activation list. When reaching the activation time, the time manager will send an activation message to the release agent. Receiving the activation message from the time manager, the release agent will be activated and start to release jobs. In the case of CONWIP, the finished job will send a message to the related release agent which will release a job immediately after receiving the message. For the AS policy, the machine agent of the bottleneck will request the release agent to stop releasing or ask the release agent to release jobs. If a released job is refused by the machine agent in the first operation due to the fullness of its buffer, the job sends blocked message to the release agent, and the release agent puts it into the buffer. When accepted, the job sends unblocked message to the release agent. The job will be removed from the buffer and moved to the first operation.

2.2 Job Agent

The job agent is a temporary entity. After released, it will be processed on many machines in the order of its process flow which depends on the target product, and after finished it will be destroyed. The job agent has a unique name in the model and four logical states: transporting, waiting, processing, and blocking.

Behaviors and Life cycle of the Job Agent

A job agent has five behaviors: to request resources, to be transported, to enter the buffer, to wait, to be processed, and to be blocked. The lifecycle of a job agent is shown in Figure 5. After release, the job requests the next operations. If accepted, it begins transporting and then enters the buffer to wait. If refused, it is blocked on the current machine. After receiving the start message from the machine group, the job starts. The job finishes when receiving the end message from the machine group and then requests the next operation. If it is the last operation, the job completes.

2.3 Delays and Activations in the Job Agent

There are four types of delays related to the logical state: transporting, blocking, waiting and processing. Blocking delay means that a finished job cannot move to the next operation due to the fullness of the related buffer and will continue to stay on the current machine. This is a conditional delay, and it will be activated by the time manager every time the simulation time advances. When a job begins to be transported or is blocked, associated activation points (transporting end, blocking end) will be sent to the time manager. After that, the job agent will wait for activations from the time manager and the delay will end when it receives the activation messages. The waiting delay and the processing delay end when the job receives the related message from a machine.

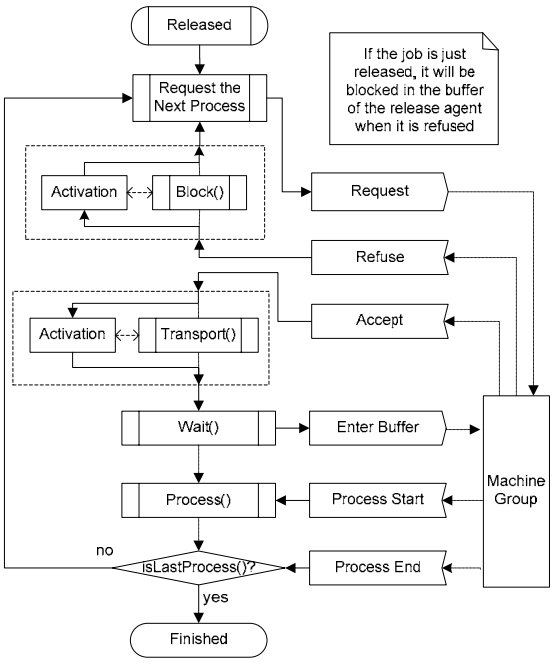

Figure 5: Lifecycle of a job agent.

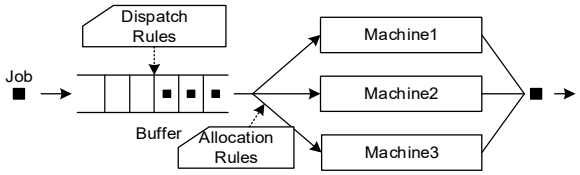

Figure 6: Structure of the machine group.

2.4 Machine Group Agent

The machine group agent is a permanent and active entity. It includes one buffer and several machines (see Figure 6). The machines have the same function and share the buffer.

There are five logical states of the machines: idle, busy, setup, breakdown, and maintenance. The machines may be single processing machines or batch processing machines. When a job arrives, and cannot be processed at once, the job joins the buffer. The buffer has a finite capacity, and it dispatches the waiting job to the idle machine according to a dispatch rule. There is no buffer behind the machine group. When a job finishes, the ther be transported to the next machine group or be blocked on the current machine.

Behaviors of Machine Group Agent

The machine group agent has two behaviors: to respond to the request from the job agent and to select the best machine to process the job. The buffer has two behaviors: to store the job and to dispatch the stored jobs to the machines in a certain batch and the order of the given priority. The machines have five behaviors: to request the jobs from the buffer, to setup, to process, to interrupt, and to recover.

When a job enters a buffer or a machine just finishes one job, the machine group will decide on the start of a new process. If the buffer is not empty (in case of batch processing, a batch must be ready) while the machine group has an idle machine, the idle machine will start processing and inform related job agents. If there is more than one idle machine, one machine will be selected according to an allocation rule. If the setup is needed, the machines will start the process after the setup. Figure 7 shows the behavior flow of the machine group agents.

Dispatch and Allocation Rules in Machine Group Agent

A dispatch rule is a criterion for determining the jobs' process priority in the buffer. An allocation rule is for selecting one machine from the idle machines. The dispatch and allocation rules are very common ways to solve the sequencing and routing problems.

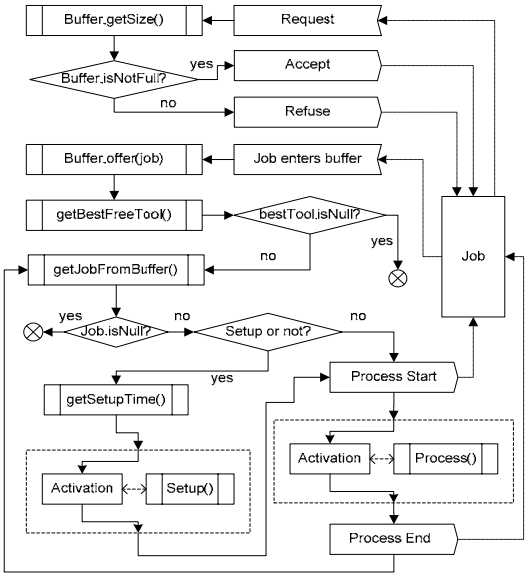

Figure 7: Behavior flow of a machine group agent.

A buffer may include a set of dispatch rules. When a job joins in the buffer, all messages associated with dispatch rules are sent to the buffer. Based on these messages and under a given dispatch rule, the buffer determines the priority of each job and queues them in that sequence. Once a job joins the buffer, the priority is updated. Currently, 17 types of common dispatch rules, such as FIFO (First In First Out), EDD (Earliest Due Date), CR (Critical Ratio), etc., have been preset in the buffer. The allocation rules are used by the group agent.

Once a job comes out from the buffer, the group agent will collect information of all machines and select one to process the job.

Delays and Activations in Machine Group Agent

There are four types of delays related to the logical state in the machine group agents: setup, processing, breakdown and preventive maintenance. When the setup delay occurs, the machine group agent sends an activation message (setup end) to the time manager and informs related jobs. The jobs will wait and cannot be processed by other machines. When the setup delay ends, the machine group agent sends an activation (process end) message to the time manager. Meanwhile, it informs the jobs and starts processing. Activated by the time manager, the machine finishes processing and informs the jobs.

2.5 Communication among the Agents

We summarize the communication in Figure 8. It includes the delays and activations conveying between the time manager and the agents as well as the communication among the agents.

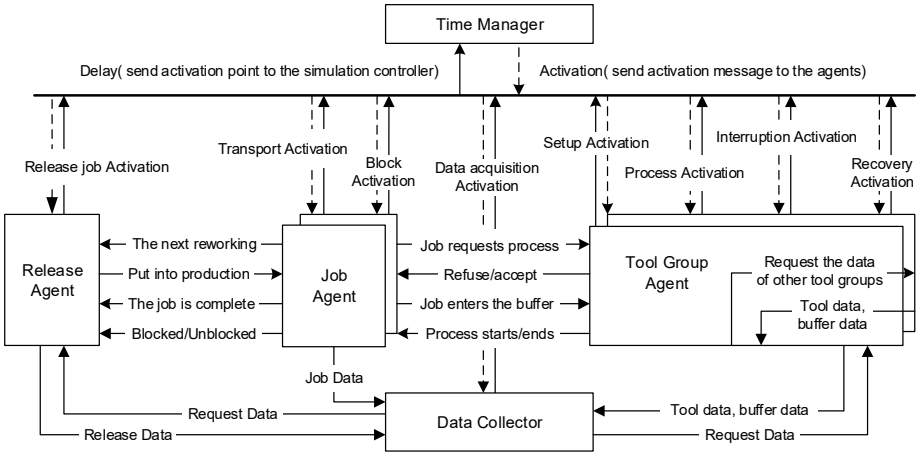

Figure 8: Communications among the agents and the time manager.

3 Agent-based Simulation of the Job Shop Production

3.1 Overview of the Simulation

The agent-based simulation of the job shop production consists of a simulator, production data, and performance measures (see Figure 9). The production data are inputs, and the performance measures are outputs.

Production Data

We classify the production data into three parts: machine group data, product data, and process data, and then adopt XML to store the data. Thus, three XML files are created to store the data. The machine group file includes data about all machine groups, e.g., the number of machines, buffer size, dispatching rule, etc. The process file consists of the process flows of all products. Each processing flows has many operations. The product file consists of buffer size, waiting time (by machine group), blocking data about all products which will be produced. Each product has a name of the process flow which is defined in the process file. The process file links the product file and the machine group file together.

Simulator for the Job Shop Production

The agent-based model (including release agents, machine group agents, and job agents), data collector and time manager make up the simulator for the job shop production. The time manager creates the release agents and the machine group agents according to the machine group data at the beginning of the simulation, and it is responsible for advancing the simulation time and handling the simulation control (e.g., start, stop, pause, etc.).

The data collector is responsible for collecting simulation data and computing the performance measures. It can collect all simulation data in detail, as well as part of sample data.

Performance Measures

The performance measures include WIP level, cycle time, time (by machine group), and machine utilization information (e.g., idle time, processing time, breakdown time and setup time). All data can be shown in graphs and tables. These measures can be used to improve the job shop production and can be provided for the optimization or control algorithms to achieve the goal of the optimization and control.

3.2 Static Structure of the Simulator

The static structure of the simulator (see Figure 10) has three layers: framework layer, agent layer, and production characteristics layer.

The framework layer provides the agent base class and the time manager. The agent layer contains the agents which are abstracted from the job shops, and these agents extend the base class of the agent in the framework. A machine group agent has multiple machines and one buffer. Each release agent can release one kind of product.

The data collector, which is also an agent and in charge of data collection from machines, buffers, jobs, and release agents, is contained in the agent layer. In the production characteristics layer, many characteristics are considered, such as reentrant flows, rework, setups, batch processing, breakdowns, and preventive maintenance. Dispatch rules and release policies are part of this layer.

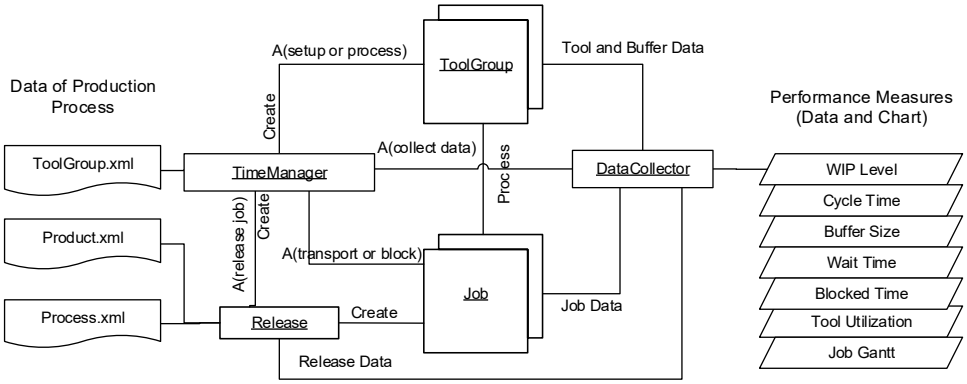

Figure 9: Agent-based simulation of the job shop production.

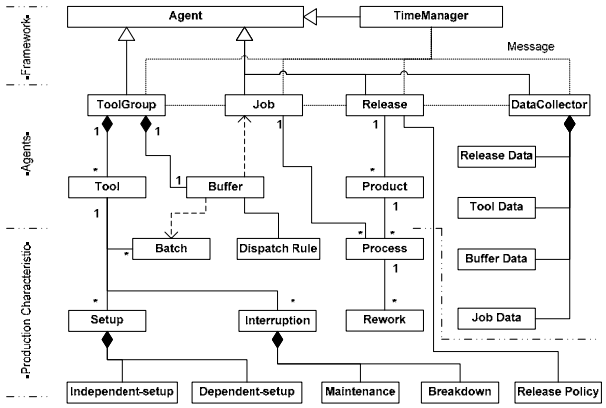

Figure 10: Static structure of simulator.

4 Applications

We create a simulation model of an example job shop. The job shop contains five machines and produces two products (Pa and Pb) with two process flows and two different throughputs. The throughputs of product Pa and Pb are 110 and 197 units per week. There are no batch processing machines. The machines need sequence-dependent setups. The interval between two breakdowns on the machines is subject to the exponential distribution and the repairing time follows an exponential distribution too.

The simulator is used to solve real-time sequencing problems in the job shop in which we should decide which job should be processed first once a machine becomes available. The objective is to minimize the cycle time. The sequencing problems are sequential decision-making problems. The decision-making is the selection of the best job from possible jobs. In other words, it is a priority calculation problem. For each job, a priority value is computed. The job with the highest priority is selected.

We proposed three simulation-based approaches to calculate the priority values of jobs, including the simulation try-then-decide method (STTD) [8], the intelligent method based on the simulation try-then-decide method (INT1) [7], and the intelligent method based on Markov decision process (INT2) [9].

The STTD method is a pure simulation approach. It uses the alternative simulations to predict the future after a job is taken and select jobs according to the future information from the simulation. The most important innovation is the usage of the base-rule in the alternative simulations. The base-rule avoids the exponential explosion of the number of the alternative simulations. To evaluate the STTD method, we replace the system with an environment simulation. The decisions we made are executed in the environment simulation. The results from the environment simulation can be used to evaluate the STTD method.

The INT1 method combines the experiences & data approach with the simulation approach. A data-driven model is introduced to calculate the priority values for jobs. It is built on the data from the simulation with the STTD method. So, it manages to learn the knowledge of the STTD method. Two types of factor influence the priority value of the job: global factors and local factors. The state of the job shops is divided into several patterns by clustering the global data. In each pattern, the priority value is only up to the local factors. The relationship between the priority and the local factors is mapped in the neural networks. For each pattern, one neural network is created. In the decision maker, the centroids of the patterns make up a pattern pool, and the neural networks make up a network pool. While making the decision, the decision maker determines the pattern of the current state according to the pattern pool and selects one corresponding neural network from the network pool.

The neural network will calculate the priority for each job according to the local factors.

The INT2 method combines the experiences & data approach, mathematical approach, and simulation approach. It models the sequencing as Markov decision process with multiple decision makers. We defined the five-tuple for the problems, including decision points, state space, action sets, transition procedure, and a reward function.

The data-driven model is still used to map the value of the action to the state-action pair. The simulation-based batch-mode Q-learning algorithm explores the state space by the simulation. Each simulation run is one iteration. The data-driven model is updated and improved gradually after each iteration.

The three methods compare with each other as well as other two decision rules, First In First Out (FIFO) and Shortest Processing Time (SPT). The experiment results are shown in Table 1. The results show that the STTD and INT1 methods always outperform the rules. The STTD method performs best but consumes much time. Contrarily, the INT1 method takes less time while the performance is just a little bit worse than the STTD. Thus, the STTD is more suitable for offline applications, and the INT1 can be used in the real-time control. Unfortunately, the INT2 method performs unsteadily. It will be further studied in the future.

5 Conclusions

On the basis of the previous works, a framework for the ABS is designed and an agent-based model of the job shop production including release agents, machine group agents, and job agents are built. The behaviors and interactions among the agents are explicit defined. A large variety of decision rules are preset in the model as well. The agents can make decisions simply according to the rules. The agent-based model, the data collector, and the time manager make up the simulator for the job shop production. Applying the simulator to an example job shop, we solved the real-time sequencing problem by three simulation-based approaches we have proposed. The application and experiment results indicate that the overall idea of simulation-based learning and decision-making is feasible in the job shops.

References

[1] Lasi, H., et al., *Industry 4.0.* Business & Information Systems Engineering, 2014. **6**(4): p. 239-242.

[2] Jazdi, N. *Cyber physical systems in the context of Industry 4.0.* in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*. 2014.

[3] Wortmann, F. and K. Flüchter, *Internet of Things.* Business & Information Systems Engineering, 2015. **57**(3): p. 221-224.

[4] Hermann, M., T. Pentek, and B. Otto. *Design Principles for Industrie 4.0 Scenarios*. in *2016 49th Hawaii International Conference on System Sciences (HICSS)*. 2016.

[5] Xie, S., T. Zhang, and O. Rose, *Agent-based Simulation with Process-interaction Worldview*, in *ASIM 2018 24. Symposium Simulationstechnik*. 2018: Hamburg.

[6] Zhang, T. and O. Rose, *A framework for agent-oriented parallel simulation of discrete event systems*, in *Proceedings of the Winter Simulation Conference*. 2012, Winter Simulation Conference: Berlin, Germany. p. 1-2.

[7] Zhang, T. and O. Rose. *Intelligent dispatching in dynamic stochastic job shops.* in *2013 Winter Simulations Conference (WSC)*. 2013.

[8] Zhang, T. and O. Rose, *Simulation-based Dispatching in Job Shops*, in *ASIM 2014 22. Symposium Simulationstechnik*. 2014: Berlin.

[9] Zhang, T., S. Xie, and O. Rose. *Real-time job shop scheduling based on simulation and Markov decision processes*. in *2017 Winter Simulation Conference (WSC)*. 2017.