Universität der Bundeswehr München

MATERIAL FLOW CONTROL ON SHOP FLOORS IN JOB SHOPS: RELEASE, ROUTING, AND SEQUENCING

by

Tao Zhang

Full copy of the thesis approved by the Department of Computer Science
of the Bundeswehr University Munich for obtaining the doctoral degree

Doktor der Naturwissenschaften (Dr. rer. nat.)

Advisor: 1. Prof. Dr. rer. nat Oliver Rose
2. Prof. Dr. rer. nat. Stefan Pickl

The doctoral thesis was submitted at the Bundeswehr University Munich on 12.06.2018 and accepted by the Department of Computer Science on 23.08.2018. The oral examination took place on 07.11.2018.

November 2018

Abstract

Material flows in manufacturing are the movements of materials through a defined route in a plant for producing a final product. Material flow control (MFC) oversees the movements of all materials. Our study focuses on the release, routing, and sequencing processes. The release process decides if a job will be released while the workload changes. The routing process specifies which machine a job will go to when the job is ready. The sequencing process determines which job will be processed first when machines become idle. These processes are sequential decision-making processes in which decisions are made in a sequence and aim to optimize a long-term objective. Due to the randomness and complexity of the material flows, these decisions are usually made by some decision rules. However, these rules often lack an overall view of the system, which in turn lead to an unstable performance.

In our study, each decision-making process is an alternative selection procedure in which a priority value is computed for each possible alternative, and the alternative with the highest priority is chosen. The priority values are subject to the constraint from the sequential decision-making, i.e., all decisions made based on the priorities result in the good long-term performance. Apparently, the priority value is a function of the concerned alternative and the current and future states of the material flows. Because simulation is always the first choice to make predictions and analyses of the complex system while machine learning dedicates itself to finding knowledge from raw data, our basic idea is drawn to calculate the priority values by the machine learning according to the current and future information generated from the simulation. Three simulation-based methods are proposed, including a simulation try-then-decide method (STTD), an intelligent method based on the simulation try-then-decide method (INT1), and another intelligent method based on Markov decision process (INT2). Lots of techniques from the machine learning are utilized, such as clustering, neural networks, reinforcement learning, and so on. Because the methods highly depend on the simulation, an agent-based simulator for the material flows is developed first, which is also used to evaluate the methods at last.

The three methods are employed to a sample manufacturing line and compare with each other as well as some decision rules. 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.

Contents

1 Introduction

Manufacturing is always accompanied by material flows in which materials are moved from one place to another place or transformed from one form into another form. To control the manufacturing is essentially the control of the material flows. Lots of familiar issues are within the scope of the material flow control, such as production scheduling (Pinedo 2012, Baker 1974) which in a narrow sense controls sequences of the materials flowing through resources, inventory control (Tersine and Tersine 1994) which decides timing and quantity of the materials flowing into storages, routing (Ibaraki and Katoh 1988) which directs the materials to the next place or resource, material release (Graves, Konopka, and Milne 1995) which determines timing and quantity of the materials flowing from the storages into the shop floors, and so on. Obviously, the material flow control is multidisciplinary as it covers multiple domains spanning across manufacturing engineering, industrial engineering, operations research, and computer science, in particular, mathematical optimization and decision science. Because most manufacturing systems are very complicated and highly stochastic, it is very difficult to make an optimal plan or schedule in advance by the mathematical optimization to facilitate the material flow control. In our study, the material flow control turns out to be sequential decision-making processes in which decisions are made in an unpredictable sequence, and the goal is not local optima but long-term global optima. Moreover, to comply with requirements for the future smart manufacturing that the resources and materials must be able to make their own decisions, the study is finally going to focus on the decentralized real-time sequential decision-making in the material flow control without any beforehand plans.

1.1 Background and Context

1.1.1 Material Flow in Manufacturing

Material flow in manufacturing is the movement of materials through a defined route in a plant to produce a final product. Production and transportation are two main aspects of the material flow. The production is a chemical or physical transformation of the materials to new materials. According to the number of inputs and outputs, there are four types of transformation: one-to-one transformation, one-to-many transformation (e.g., decomposition process), many-to-one transformation (e.g., assembly process), and many-to-many transformation. In the plant, materials can be held in the following places: storages, machines, and buffers in front of/behind the machines. The movement of the materials is the transportation among these places. There are two types of transportation: point-to-point transportation and pass-by transportation. For the point-to-point transpiration, vehicles carry materials directly to their destination without any stops. For the pass-by transportation, vehicles follow a given route and stop at certain places to load/unload materials. The types of production and transportation have significant influences on the material flow. They determine the characteristics of the material flow.

Materials in manufacturing can be grouped into six types: raw materials, semi-products, by-products, final products, reworking products, and scraps. Lots of raw materials are required for producing a final product. At last, not only the final product is produced, but also some by-products and scraps are obtained. The semi-products are the temporary forms of materials. They will be transformed into the final products or other semi-products. The reworking products are the final products or the semi-products which failed in their qualities and will be reworked somewhere. Thus, the material flow in manufacturing can be cut into four parts: main material flow (final products and semi-products), by-product flow, scrape flow, and reworking flow. These partial flows influence one another. Only all of them flow smoothly, the production can go well. The main material flow is the most important one in the material flow control. There are three typical forms: line flow(a), tree flow(b) and network flow(c) (shown in Figure 1.1 adopting the point-to-point transportation). The line flow contains only the one-to-one transformation. One semi-product is transformed into only one new semi/final product. The tree flow comprises the many-to-one transformation. Several semi-products can be transformed to one new semi/final product. The network flow contains the many-to-many transformation.

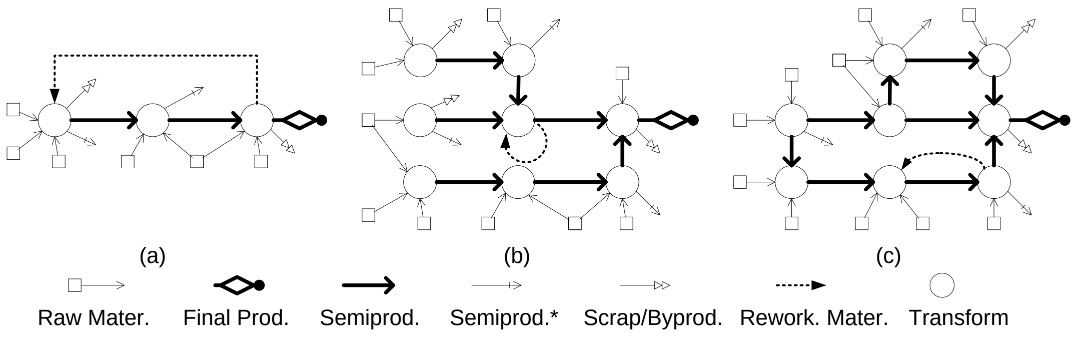


Figure 1.1: Types of material flow

1.1.2 Material Flow and Material Flow Control in Job Shops

The job-shop is a process type in which small volumes of a variety of products are made. In job shops, the main material flow can be any one of the three types, which depends on the structure of the process flow. Different products have different flows. The transportation can be either the point-to-point type or the pass-by transportation. Each transformation of materials is in conjunction with a processing operation on a machine, and each movement of materials is a transportation operation. All operations to finish a final product make up a job. The job and the operation are fundamental units in the material flow control rather than the materials. Detailed characteristics of the material flows in job shops will be addressed in Section 4.1 when we create a material flow model.

Material flow control (MFC) is the kernel of shop floor control (SFC). All the other functions of the SFC, e.g., WIP (work in process) tracking and quality control, is based on the MFC. The SFC is at the bottom of the enterprise resource planning platform. The higher level of the SFC is a production planning system including material requirement planning and capacity requirement planning. The production planning system generates production orders and specifies each order a due date according to a master production schedule. The generated orders are put into an order pool and wait for being released to the shop floor. Each order usually includes several jobs. The SFC decides if the shop floor accepts the released orders or not. Jobs in the accepted orders are put into a job pool. Then, the MFC is in charge of the job releasing and the movements of all materials involved in the jobs on the shop floor.

1.1.3 Decisions on Material Flow Control in Job Shops

The material flow control answers the following questions: 1) whether or not to move materials; 2) which materials to move; 3) where to move the materials to; 4) when to move the materials; 5) how to move the materials. To answer these questions, we must consider the location of the material. A different location of materials raises a different issue surrounding each question. Table 1.1 shows most issues of each question except 4) and 5). The question that when to move the materials is mostly a planning or scheduling problem which is out of the scope of the real-time control. So we do not list it on the table. The question how to move the materials usually refers to decisions on the batch size of each movement and is overlapping with question 1). We do not put it on the table either.

Moving raw materials from storage for the first operation implies the start of a new job. The number of jobs being carried out in a shop represents the workload of the shop. The main issue of the SFC is to balance the workload against the effective capacity of the shop. So controlling the movements of raw materials coming out from storage is the most important way to deal with the balance issue. This problem is usually called as a job release problem. Once the workload changes, the decisions on whether to release a certain amount of jobs into the shop should be made.

No matter how we release jobs, materials may have to wait for a machine before being processed due to bottlenecks, machine failures, and other interruptions. A queue may often appear in front of the machine. In this case, the problem that in which sequence the waiting materials will be processed or which one should be processed first becomes very important. To process materials on a machine denotes an operation. So the problem is an operation sequencing problem. Similarly, the processed materials may form a queue behind the machine to wait for transporters, and we have to decide which one should be transported first. We call this problem as a transportation sequencing problem. The problem is critical in some shops which have inadequate capacity for transportation.

Table 1.1: Decisions on material flow control in job shops

Place of materials	*Decision point*	*Whether to move*	*Which one to move*	*Where to move*
storage (Raw mat.)	the workload changes	Job release	-	Routing (transporter)
buffer before	the machine becomes free	Batching (process)	Sequencing (process)	-
machine	rush materials arrive	Preemption	-	-
	the machine breaks down	-	-	Rerouting (machine)
buffer behind	a transporter arrives	Batching (transport)	Sequencing (transport)	Routing (transporter)
transporter	materials are loaded	-	-	Routing (machine)
	the transporter breaks down	-	-	Rerouting (transporter)

Once a piece of materials is loaded on a transporter, another decision that where the materials should be carried to has to be made. This decision-making is a routing problem. The routing problem exists in both process design phase and material flow control phase. In the process design phase, the routing problem is to determine how to produce a product. At last, an operation path is assigned to the product. Each operation may be carried out on several machines which may have different capacities. In the material flow control phase, the routing assigns one of the related available machines to each operation. Sometimes the routing also makes changes on the operation path, such as changing operation sequence and adding/removing operations into/from the operation path. It makes the routing problem more complicated. In practice, the changes on the operation path are forbidden.

There are still many other decisions which need to be settled in the material flow. If a machine processes materials in a certain size of the batch, we call the machine as a batch processing machine. While the machine becomes free and batching is not ready, a decision should be made that whether to move the batches out the buffer and start the process or just wait for more materials. While a material arrives and the machine is waiting for batching at that time, the decision also needs to be made. This is an operation batching problem. It aims to process as many as possible materials in a batch but without waiting too long for batching. The similar problem happens on the transportation. A transporter usually can carry multiple materials. To wait or to start transportation should be determined while a transporter arrives or a piece of material comes to the transporter. When rush materials arrive at a busy machine, we have to decide whether to stop the current job and to process the rush materials. This is a preemption problem. Moreover, if there are multiple transporters which are capable of carrying the materials, we should choose one transporter. This is a transporter routing problem. If the machine or transporter breaks down, the materials on them have to be assigned a new machine or transporter. These are rerouting problems.

1.1.4 Objectives of Material Flow Control in Job Shops

The objectives of the shop floor control are to increase throughput, reduce cost and finally maximize the profit: profit = throughput*(price-cost). The material flow control, as the kernel of the shop floor control, must accomplish these objectives first of all.

In the material flow control, the way to increase the throughput is to choose a proper job release rate. Here we assume that the accepted orders are always enough. If the material flow has a fixed capacity and is no-delay, i.e., machines and transporters cannot be idle while operations are waiting in front of them, the throughput only depends on the job release rate regardless of sequencing, routing and so on. When the release rate is slower than the effective capacity of the shop, the throughput is equal to the release rate; when the release rate is faster than the effective capacity, the throughput can only reach the effective capacity. Figure 1.2 (a) shows while the release rate keeps increasing, what will happen on the throughput, WIP level, and average cycle time. We can see that as the release rate increases the throughput raises linearly until reaching the effective capacity. After that, the throughput will keep at a steady level. After the release rate reaches the capacity, the WIP level and cycle time ascend dramatically. Apparently the release rate cannot exceed the capacity; otherwise, the profit will decrease because the throughput does not change while the WIP holding cost increases. Theoretically, the profit will be maximized while the release rate equals the effective capacity. In practice, the capacity changes a lot due to some interruptions, such as machine breakdowns, rush jobs, unexpected processing times and so on. The release rate should change and adapt to the new capacity dynamically.

The way to reduce the cost by the material flow control is to reduce the WIP level to cut the holding cost. If we scale values of the WIP level and cycle time in Figure 1.2 (a), we get Figure (b) where the WIP level is increasing slowly and the cycle time is nearly constant before the release rate reaches the capacity. If no interruptions occur in this range, the WIP level is contributed only by jobs being processed on machines and jobs being moved by transporters. No jobs are waiting for machines and transporters. Thus no sequencing problem occurs in this range. Making good decisions on the routing and batching can shorten the cycle time in this case. According to the Little' s Law Throughput=WIP/Cycle Time, the WIP level can be reduced by decreasing the cycle time while the throughput stays constant. We can make a profit from the WIP reduction using routing and batching processes. For a fixed release rate, there is the lowest WIP level. The routing and batching procedure can reduce the level to the lowest if the job shop is small and simple enough. However, it is impossible to find such a simple shop. The lowest WIP level is always an ideal. Also, if the interruptions happen very often, queues usually appear. The jobs waiting in the queues is also contributed to the WIP level. In this case, sequencing can also lower the WIP level.

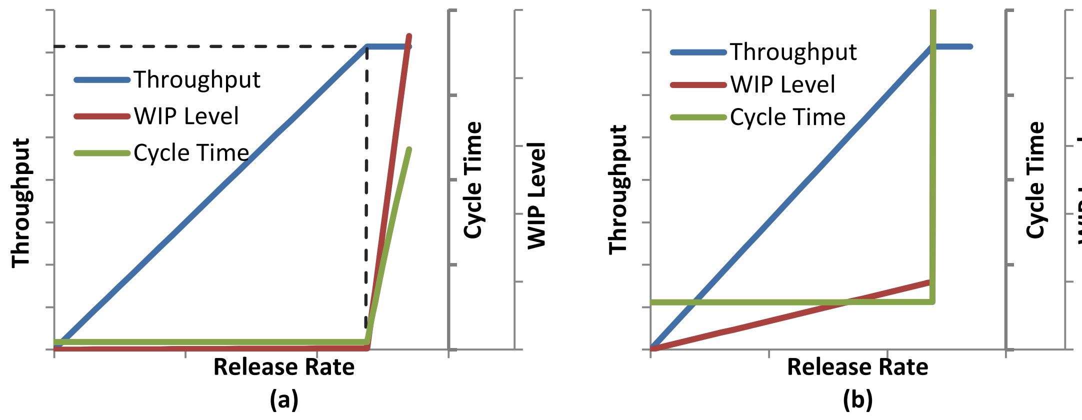


Figure 1.2: Relationship between throughput and release rate

Except for the objectives of the shop floor control, the material flow control has its objectives. The main objective is to make sure the material flow goes smoothly and rhythmically. The smooth means no job jam and no flood of jobs. It can be described by the stability of the WIP level. The rhythm is related to the job release and the cycle time. Because multiple products are produced at the same time, a release sequence of products should be made by the cycle time, such as A-B-A-C-A-B-A-C. The jobs will be released in this sequence under certain tempo repeatedly. If the material flow goes in this way, it will be rhythmical. The stability of the cycle time is the foundation of the rhythmical flow. If the release rate keeps steady and does not exceed the capacity, the stability of the cycle time is equivalent to the stability of the WIP level. The variation of the WIP level and cycle time is caused mainly by the interruptions in the shops. The sequencing procedure can be used to stabilize the cycle time and the WIP level. If the interruptions last a long time, the only way to stabilize the WIP level and cycle time is to adjust the release rate. The sequencing procedure only has tiny influence in this case. WIP reduction and WIP stabilization seem like two contrary problems. Essentially, they are the same. The WIP reduction tries to lower the level to the lowest value. The WIP stabilization tries to pull an unexpected level back to a normal level. Usually, the normal level is always expected to be the lowest level.

There is still a very important objective: meeting due dates. These typically come from one of two sources: directly from the customer or the requirements of successor shops. If delays happen often, customers will not order products from the shop anymore. If let the successor shops wait too long, it might also lose lots of money. The due dates are specified by the production planning system. The order release procedure is the first stage to meet the due dates. The second stage is carried out by the job release procedure in the material flow control. The sequencing, routing and batching procedures are responsible for the last stage. The importance of each stage to the due date satisfaction is decreasing stage by stage. So in the material flow control, the job release is the most important method of meeting the due dates. Other procedures have fewer effects on the due date satisfaction. Also, in the material flow control, there is another type of due dates: the operational due date. The due date is assigned to each operation. If each job has a due date, the operational due date can be calculated backward according to the job' s due date. The due date objective is distributed to each operation. The operational due date will be utilized in the sequencing procedure. Sometimes even there is no due date for each job, an operational due date calculated from the release date is also assigned to each operation. Essentially it aims to stabilize the cycle time.

There are also many other objectives of the material flow control. For example, if machines in the shop, especially at bottlenecks, are very expensive, we hope to use them thoroughly. The objective is utilization maximization. This objective is equal to the throughput maximization. The job release procedure plays an important role to achieve the objective. If some jobs and their release date are given, we want to finish them as early as possible. To minimize makespan will be the objective. The sequencing, routing and batching will take responsibility.

1.1.5 Approaches to Material Flow Control in Job Shops

As we mentioned before, the material flow control is a decision-making process. Job release decides if a job will be released while the workload changed. The alternatives are yes and no. Sequencing determines which job will be processed or transported first. The alternatives are jobs in the buffer. Routing decides which machine or transporter the job will go to. The available machines and transporters are alternatives. Batching decides whether or not to wait for a job. We should answer yes or no. There are three types of approach to the decision-making: experiences & data approach, mathematical approach, and simulation approach.

To the experiences & data approach, we assume the knowledge contained in the experiences and data are right and are capable of making decisions. We learn the decision-making knowledge from the experiences and dig useful information from the data. At last all information help us to make decisions. In the mathematical approach, the decision-making processes are considered as an optimization problem and mostly a combinational optimization problem. An optimization model is built, and lots of algorithms can be used to solve the optimization model. For the simulation approach, a simulation model of the material flow is created. Simply a try-then-decide method is used. The alternatives will be tried by the simulation in advance, and the best one will be selected according to the simulation results.

The experiences & data approach is low-risk and practical. It can make good feasible decisions. However, the best situation is that it performs as good as the experiences. We cannot get benefits from it to improve the material flow control. Theoretically, the mathematical model can obtain an optimal solution. However, because of the complexity of the problems it usually can get only a near-optimal solution and sometimes only a feasible solution. The simulation approach can forecast the future of the material flow. However, the accuracy of the results highly depends on the simulation model.

1.2 Scope and Approach to the Research

1.2.1 Scope of the Research

As we analyzed before, so many types of decisions need to be made in the material flow control in job shops. However, the capacity of the transportation tools is usually adequate so that the transportation-related decision-making can be ignored. Also, the operation batching problem can be included in the sequencing problem by simply adding a null alternative before deciding which one should be processed. We also assume that once a machine starts to process a job, it cannot be interrupted. Thus there is no preemption problem.

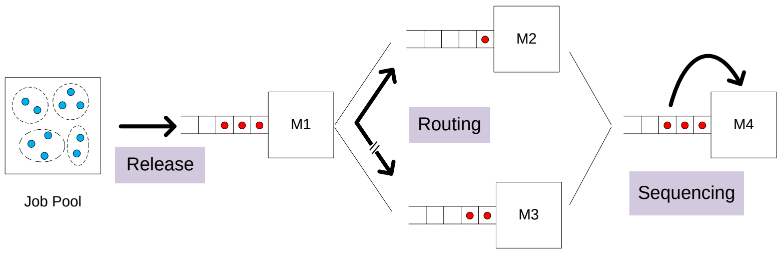


Figure 1.3: Release, routing, and sequencing problems

Moreover, we assume that if a machine breaks down, the job on the machine will be put back in the front buffer and wait for reprocessing. So there is no rerouting problem. Thus, job release, operation sequencing, and machine routing, shown in Figure 1.3, are the most critical decisions in the material flow control. In our study, we considered these three types of decision-making. The sequencing and routing aim to the reduction of the cycle time and WIP level. The decision-making on the job release tries to meet the throughput constraints.

1.2.2 Approach to the Research

The release, routing, and sequencing problems are sequential decision-making problems. The goal of the sequential decision-making is that the decisions we made together can result in a good long-term performance. This means we must know the influence of the decisions on the long-term performance. On the other hand, the decision-making is the selection of the best alternative from possible alternatives. It can be described as a priority calculation problem. For each alternative, a priority value is computed. The alternative with the highest priority is selected. Obviously, the priority value is a function of the concerned alternative and the current state of the material flow. Thus if we find the function, the problems are solved.

Considered the goal of the sequential decision-making, the priority value must represent the influence of the decisions on the long-term performance. So the function must be able to analyze the current situation and predict the future after a decision is taken. Moreover, inputs of the function are a high-dimensional dataset which contains features extracted from the alternative and the state of the material flow. The relationship between the inputs and the output, i.e., the priority value of the alternative, is nonlinear. Therefore, it is very hard to obtain a mathematical function. Because simulation is always the first choice to make predictions and analyses of the complex system while machine learning dedicates itself to finding knowledge from raw data, especially data mapping, our basic idea is drawn to calculate the priority values by the machine learning according to the current and future information generated from the simulation. We proposed three simulation-based approaches to calculate the priority values of alternatives, including

- Simulation try-then-decide method (STTD),
- Intelligent method based on STTD,
- And intelligent method based on Markov decision process (MDP).

1.3 Goals and Significance of the Research

1.3.1 Goals of the Research

Based on the general idea of our three approaches, we divided our study into five stages (shown in Figure 1.4): agent-based simulation, simulation of material flow, simulation try-then-decide method, intelligent method based on the STTD, and intelligent method based on the MDB. The former stage supports the next stage. Different goals are established to a different stage.

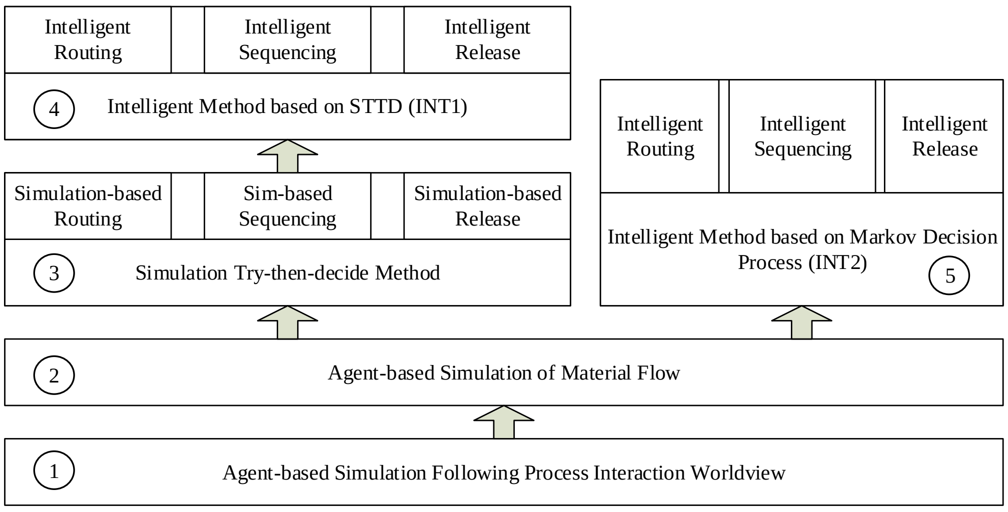

Intelligent Routing
Intelligent Sequencing
Intelligent Release
4 Intelligent Method based on STTD (INT1)
Simulation-based Routing
Sim-based Sequencing
Simulation-based Release
3 Simulation Try-then-decide Method
Intelligent Routing
Intelligent Sequencing
Intelligent Release
Intelligent Method based on Markov Decision Process (INT2) 5
2 Agent-based Simulation of Material Flow
1 Agent-based Simulation Following Process Interaction Worldview

Figure 1.4: Five stages of the study

The goals of the agent-based simulation (stage 1) are,

- Address a formal procedure of the agent-based simulation with the process-interaction worldview (ABS&PIW);
- Develop a framework for the ABS&PIW involving general behaviors and communications among agents;
- Compare the framework to other frameworks.

The goals of the simulation of material flow (stage 2) are,

- Develop an agent-based simulator for the material flow in job shops using the framework above;
- Include key features of the material flow in the simulation model, such as setup, breakdown, preventive maintenance, reworking flow, and so on;
- Include common dispatching rules and release policies in the simulator, such as FIFO, SPT, EDD, CONWIP, CONINT, Starvation Avoidance, and so on;
- Design a format to describe and store the information of the material flow and the results of the simulation; provide interfaces to connect to other simulators.

The goals of the simulation try-then-decide method (stage 3) are:

- Address a scenario of simulation try-then-decide method in the material flow control; The scenario must include alternatives, environment, simulation runs, and evaluation;

- Answer three questions about the simulation runs: how long the simulation runs; which base-rule should be selected; how to evaluate decisions according to the simulation results;
- Implement the simulation try-then-decide method for the material flow control by the simulator we developed before, which can make decisions about the release, routing, and sequencing.

The goals of the intelligent method based on the STTD (stage 4) are:

- Develop a data collector obtaining data from the simulation which uses the STTD method to make decisions;
- Analyze and reorganize factors upon which the decisions on the release, routing, and sequencing are made;
- Cluster the states of the material flow into several patterns;
- For each pattern create a data-driven model to map the relationships between the factors and the selection of alternatives;
- Implement an intelligent decision-making system for the material flow control.

The goals of the intelligent method based-on the MDP (stage 5) are:

- Create a MDP model for the release, routing, and sequencing problems;
- Map the relationship between the value of action and the state-action pair to a data-driven model;
- Solve the MDP model by the simulation-based batch-model Q-learning algorithm.

1.3.2 Significance of the Research

Though many goals listed above have been more or less achieved by other researchers, we still put some new features on them and try to improve their advantages and meanwhile to avoid their disadvantages. There are also many new opinions and methods in our study. The significance of this study lies in the following aspects. The research is the first to:

- Combine the simulation and machine learning to solve the sequential decision-making problems in the material flow control; The data needed by the machine learning comes from the simulation;
- Convert the release, routing, and sequencing problems to decision-making problems in which a priority value is calculated for each alternative and the decision maker select an alternative according to the values. Especially, the release problem becomes a simple yes-or-no decision-making process;

- Propose three methods to calculate the priority values; They are simulation try-then-decide method (STTD), intelligent method based on the STTD, and intelligent method based on Markov decision process;
- Present a purely decentralized decision-making system for the material flow control. For each type of product, a release decision maker is created; for each machine, a sequencing decision maker is created; for each machine group, a routing decision maker is created.

1.4 Overview of Dissertation

The dissertation is structured in the sequence of the study stages. Chapter 1 is the introduction to our research. Some basic issues of the material flow control including concepts, decisions, objectives, and approaches are introduced. The scopes, approaches, goals, and significance of our research are also addressed here. Chapter 2 is the state-of-the-art. Lots of papers about the release, routing, and sequencing problems are reviewed and summarized regarding the scope, approach, and advantages & disadvantages. Except for Chapter 1, 2 and 8, at the end of each other chapters, an experiment related to the approach proposed in the chapter is reported. In Chapter 3 an approach to introducing the process-interaction worldview into the agent-based simulation is proposed. A framework is developed by using multi-threading and synchronization technology. In Chapter 4, a simulator for the material flow in job shops is developed within the framework. The simulator contains release agents, machine group agents, and job agents. The simulation-based try-then-decide method is carried out in Chapter 5 by using the simulator. A general scenario for the simulation try-then-decide method is addressed first. And then it was used to solve the routing, sequencing, and release problems. In Chapter 6, the intelligent approach based on the data from the simulation try-then-decide method is introduced into the decision-making processes. According to the simulation data, the states of the material flow are divided into several patterns. A neural network is created for each pattern to map the relationships between the state and the selection of the alternatives. In Chapter 7, a different method of calculating the priority values for alternatives is presented. The method is based on the Markov decision process (MDP) which aims to find out an optimal action-value function to take the best action. The same data-driven model presented in Chapter 6 is used to map the relationship between the value of action and the state-action pair. The model is solved by a simulation-based Q-learning algorithm. The thesis is concluded in Chapter 8, and a plan for the future works is given at the end.

2 Literature Review

Lots of literature related to the decentralized real-time job release, routing, and sequencing are reviewed in this chapter. To remove ambiguities, we clarify and define some common terms at the beginning. An in-depth discussion is given at last.

2.1 Basic Terms

Even though there are lots of related literature, they usually use different terms to express the same problems or approaches. There is also some related literature which uses the same term but refers to the different problems. So, at the beginning of this chapter, we discuss some terms that frequently appear in the literature and explore the relationships among these terms.

2.1.1 Planning, Scheduling, and Control

In the business dictionary, **planning** is a basic management function involving the formulation of one or more detailed plans to achieve certain objectives. The plan is a set of decisions about how to do something in the future. **Control** is also a management function and means measuring actual performance and taking corrective action aimed at achieving certain objectives. **Scheduling** is to assign each task some resources (routing) and determine the sequence of tasks being carried out on the resources (sequencing) so as to achieve certain objectives under some constraints.

We can see that all of them are optimization problems. The scheduling problem will be a planning problem if we create a schedule in advance. The scheduling problem can also be a control problem if the routing and sequencing decisions are made in real time. The job release problem is not included in the scheduling problem. Because it is an interface between the production planning and the production scheduling in practice. Similarly, the release is a planning problem if we make a release plan; If we make the release decision in real time, the release is a control problem. From another point of view, they all can be considered as a decision-making problem. The planning makes decisions in advance while the control makes decisions in real time. The release, routing, and sequencing can make decisions either in advance or in real time.

Most of the researchers considered the release, routing, and sequencing as a planning problem. Our study concerns the control problem about the release, routing, and sequencing. We do not make any schedules and plans. There is also another term " **dispatching**" in practice. It usually refers to the real-time sequencing problem.

2.1.2 Predictive, Proactive and Reactive Scheduling

Many approaches to scheduling and rescheduling considering the presence of uncertainties in the shop floor have been proposed in the literature. They can be classified into five categories: predictive scheduling, proactive scheduling, reactive scheduling, predictive-reactive scheduling, and proactive-reactive scheduling.

The **predictive scheduling** makes a schedule in advance without considering any disruptions. The predictive schedule is less robust(Busch et al. 2007). The **proactive scheduling** takes into account possible disruptions while constructing the original predictive schedule. This allows making the predictive schedule more robust(Beck and Wilson 2007). The **reactive scheduling** is based on up-to-date information regarding the state of the system. No predictive schedule is made, and the decisions are made locally in real time. These approaches are used when the level of disturbances is always important. Due to the constraints on the response time of the reactive algorithm, one cannot expect for an optimal or near-optimal decision(Sabuncuoglu and Bayı z 2000). In the **predictive-reactive scheduling**, a predictive schedule is generated without considering possible perturbations. Then, a reactive algorithm is used to maintain the feasibility of the schedule and improve its performances(Yang and Geunes 2008). In the **proactive-reactive scheduling**, a predictive schedule is generated by the proactive algorithm. Then, a reactive algorithm plays its role(Aloulou and Portmann 2005).

The predictive and proactive scheduling are planning problems. Sometimes, we call them **offline/non-real-time scheduling**. The reactive scheduling problem is a complete control problem. The predictive-reactive and proactive-reactive scheduling are controlling problems based on a given plan. Sometimes we call them **repair-based reactive scheduling**. The reactive-related scheduling is also known as **online/real-time scheduling**. Also, these six concepts can be easily broadened and introduced to any other decision-making processes, e.g., the release problem. Our work concerns only the reactive decision-making for the release, routing, and sequencing problems.

2.1.3 Order Release and Job Release

The order in our study refers to the manufacturing order generated by the production planning system. Each order may contain several jobs. The orders are released at some points of time determined by the order release procedure which also is called the order review/release procedure (ORR). In practice, the released orders are usually accepted at a certain time, such as at the beginning of a shift or a day. Thus, a job pool is formed. The job release is responsible for releasing the jobs in the job pool to the shop floor. The order release is mainly dependent on the aggregate capacity of the shop while the job release pays attention to the even more detailed workload. If the release orders are accepted immediately and each order has only one job, no job pool is formed. The order release problem is equivalent to the job release problem in that case.

2.1.4 Resource Allocation and Routing

Resource allocation is in charge of allocating required resources to jobs. A job may need several resources to finish one of its operations. The resource allocation is a wider concept than the routing we considered. Resource-constrained scheduling is very similar to the resource allocation. It includes both the sequencing problem and the resource allocation problem. The routing problem in our study refers to the selection of a machine for an operation in jobs. We assume that each operation just needs one resource, i.e., one machine. Usually, during the capacity requirement planning, a machine has already been specified for each operation before orders are released. Thus, the routing problem is mostly not considered in the material flow control phase. However, for the shops with multiple identical machines which are located in different places, the routing problem cannot be ignored. Although there is no difference among these machines during the capacity requirement planning, in the material flow control they are different from one another because of differences in transportation times. So, in this case, the routing should be involved in the material flow control to specify one of the identical machines.

2.2 Related Works to the Real-time Job Release

Job release, which decides when to release each job into the shop floor, is a primary function of the material flow control. It has significant influences on the performance of the materials flow control. To the real-time job release, the decision must be made from a policy rather than a release plan. If we change our view on the problem, the job release can also be another problem that decides which jobs should be released at a given point of time. We call the given point of time a decision point. The decision points can be generated either by a periodic method or by event-oriented methods. For the periodic method, the decision is made at certain time intervals. For the event-oriented methods, the decision is made while some events occur. According to the information considered in the decision-making, the job release can be classified into three types: due-date-based job release, workload-based job release, and bottleneck-based release. A shop may just use one of them always or dynamically select one of them according to the situation. It is also possible to consider more than one type of information with different weights on each type of the information.

Most previous research studies implicitly ignore the releasing function. These studies assume that jobs are released immediately to the shop as they randomly arrive; while this may reflect the reality of some production systems, in most cases the assumption does not hold true. Practitioners give several reasons for not releasing jobs as they are received(Mahmoodi, Dooley, and Starr 1990). First, the required material may not be available, and the pre-production activities may not be completed. Second, parts released to the shop too early may be damaged and result in excessive work-in-process (WIP) inventory. Third, jobs released to the shop long before they are needed may compete for resources (e.g., machines) with more urgent jobs and thus interfere with the timely progress of those jobs. Finally, controlled job releasing provides shop floor a means of maintaining the balance between the load on the floor and the capacity of the shop.

2.2.1 Due Date-based Job Release

The due date-based job release calculates a release date for each job at decision point according to the due dates and current cycle time estimates, and creates a release time window based on the release date. If the decision point is within the release time window, the related job will be released. The decision is usually made at certain time intervals.

There are three common ways to estimate the cycle time. One assumes that the cycle time is proportional to its sum of processing times (Conway et al. 1967). They try to find the coefficient of the sum of processing times. One just uses the average cycle time obtained from the historical data of the shop floor. The last one believes that the cycle time does not only depend on the sum of the processing times, but also the situation of the shop floor at the decision point. Mahmoodi, Dooley, and Starr (1990) used regression analysis method to find the relationship between the total waiting time and the number of jobs on the concerned job’ s route. The simulation collects the data pair they needed. The obtained regression function is used to determine the due date of a job at a decision point, and the parameter of the function is the number of jobs on the job’ s route at the decision point. The cycle time equals the total waiting time plus the sum of processing times. Ragatz and Mabert (1988) calculate the cycle time according to the number of operations and the number of jobs on the concerned job’ s route. Two coefficients are set for these two numbers.

2.2.2 Workload-based Job Release

The workload is the amount of work that has to be done. It can be measured by the number of in-process jobs or the total processing time of in-process jobs. For the workload-based job release, the decision is made while the workload changed. A releasable job list is included in the method. The list stores the releasable jobs at the decision points in certain priority sequence. The following are three subtypes of the workload-based job release.

2.2.2.1 Workload-based job release at the shop floor level

The workload is measured at the shop floor level by terms of the number of in-process jobs or total processing time of in-process jobs in the shop at the decision point. A workload norm is set for the entire shop floor. While a job is finished, the first job in the list will be considered first. If releasing the first job does not cause that the workload exceeds the norm, the job will be released, and the workload will be updated. Otherwise, the job will continue to be kept in the pool. After that, the second job will be considered.

The remaining can be done in the same manner. If the workload exceeds the predetermined workload norm, the rest jobs will not be considered, and the release procedure ends.

Framinan, Gonzá lez, and Ruiz-Usano (2006) developed a dynamic method to determine the norm (card number). The norm is adjusted dynamically according to the throughput of the shop floor at that moment. If the throughput is less than the target throughput, the norm will increase; otherwise, the norm will decrease. The lower and upper bounds on the norm restrict the adjustment.

2.2.2.2 Workload-based job release at the product level

The workload is measured at the product level by terms of the number of in-process jobs or total processing time of in-process jobs, which belong to the same product, in the shop at the decision points. A workload norm is set for each product. For each product, a releasable job list is created as well. While a job belonging to a product is finished, the jobs in the related list will be considered from the first to the last. If releasing a job does not cause that the workloads of the product exceed its norm, the job will be released.

With a fixed number of Kanbans (norms) dedicated to each product, Ryan and Vorasayan (2005) use a nonlinear program to evaluate and optimize the allocation of Kanbans to product types. In numerical examples, the allocations identified are similar to those obtained by exhaustive enumeration with simulation but frequently differ significantly from a naive allocation according to demand rates. A variant of the model that minimizes the total work-in-process to achieve specified throughput targets yields results like a previous heuristic method.

2.2.2.3 Workload-based job release at the machine level

The workload is measured at the machine level by terms of the number of jobs or total processing time of jobs, which are/will be processed on the machine or are waiting for the machine, in the shop at the decision points. A workload norm is set for each machine. While an operation is finished on a machine, the jobs in the list which will visit the influenced machines will be considered in the original sequence. If releasing a job does not cause that the workloads of all related machines exceed their norm, the job will be released.

Land and Gaalman (1998) give a general procedure for the workload-based job release. The procedure considers (1) the workload situation on the shop floor in combination with the workload contribution of the jobs, and (2) the relative urgency of the job. The release procedure is built up in two phases, sequencing and selecting. The jobs in the pool are sequenced in order of a planned release date to determine their relative urgency. Urgent jobs have a higher probability to be released. The selecting phase, choosing jobs that obey the workload norms, is responsible for the load balancing function. The term load balancing refers to maintaining a constant direct load level for each machine, which speeds up the throughput in the first place.

These three methods can also be used for the periodic release. At each decision point, no matter which level we are, if the workload is less than the norm, the same procedure will be used. If the workload is greater than the norm, we just skip this period.

2.2.3 Bottleneck-based Job Release

The bottleneck-based job release is a special case of the workload-based job release at the machine level. It focuses only on the workloads of bottlenecks. The bottleneck principle is converted into a manufacturing control method.

Akhavan-Tabatabaei and Salazar (2011) propose a procedure based on trial and error to find the best values for the WIP threshold (norm) at bottlenecks in different cases. The procedure begins with running the simulation model for no policy case with norm =0 and recording the resulting cycle time and throughput. Then this step is repeated through norm value increment of one and increasing the value of arrival rate in such a way that the effective arrival rate of cases with policy remains very close to that of the no policy case. The iterations stop when no significant improvement in the cycle time is observed.

2.3 Related Works to the Real-time Sequencing

2.3.1 Dispatching Rules

Dispatching rules are a very common method for the real-time sequencing. The dispatching rules specify a priority value for each alternative job according to the information of the job and the shop floor. The job with the highest priority will be selected. If the priority depends on static information, the dispatching rules are static; otherwise, the dispatching rules are dynamic. If the priority is obtained from the information of the job and the current machine, the dispatching rules are local dispatching rules. If the priority is related to the information of the shop floor, the dispatching rules are global dispatching rules. The studies on the dispatching rules focus on three fields: creating new rules for complex environments; combining old rules to improve the performance; selecting rules which adapt to a specific environment of the shop floor.

2.3.1.1 Create new rules for complex environment

Even though lots of dispatching rules have been developed, the step for creating new rules never stops. Because to date the research still did not find out any dispatching rules that can be used in any environment and always results in a good performance. Also, more and more complex environments will be considered as the relative simple environments have been studied. The variation of the environment relies on the complex objectives, special constraints, and other more specific configurations. Thus, once a new environment appears, the old rules may not adapt to it. There is a need to develop new rules for it.

Chen and Matis (2013) created a dispatching rule called the Weight Biased Modified RRrule (WBMR). The rule minimizes the mean tardiness of weighted jobs in a job shop. It is a significant extension of the RRrule in that it has linear complexity and considers weighted jobs. The rule allows for biasing of the schedule towards meeting the deadline of high priority jobs through the tuning of a single parameter, where such an effect is quantified by evaluating tardiness at different truncation thresholds. Numerical testing demonstrates the ability of the WBMR to outperform other traditional rules at various congestion and due-date tightness levels. Jayamohan and Rajendran (2004) propose dispatching rules by explicitly considering different weights or penalties for flow time and tardiness of a job. Many measures of performance related to weighted flow time and weighted tardiness of jobs are considered. Chiang and Fu (2012) propose a dispatching rule for lot scheduling in wafer fabs, focusing on three due date-based objectives: on-time delivery rate, mean tardiness, and maximum tardiness. The rule implements good principles in existing rules by means of (1) an urgency function for a single lot, (2) a priority index function considering total urgency of multiple waiting lots, (3) a due date extension procedure for dealing with tardy lots, and (4) a lot filtering procedure for selecting urgent lots. Piplani and Wetjens (2007) give some background of manufacturing system flexibility, including its measurements and present quantifiable measures of flexibility and discuss parts dispatching based on entropic measures of part routing flexibilities. Two rules for the parts dispatching, namely "least reduction in entropy" and "least relative reduction in entropy," are presented. It is proposed that to take advantage of system flexibility, parts dispatching rules based on flexibility measures should be used, especially in systems with high breakdown rates. An extensive simulation study is conducted to evaluate the performance of the dispatching rules. The simulation study confirms that entropy-based dispatching rules outperform traditional dispatching.

2.3.1.2 Combine old rules to improve the performance

In the real world, objectives are often more complicated. For example, an objective may be a combination of several basic objectives or a function of time and the set of jobs waiting for processing. Most of the dispatching rules target on optimizing a specific objective, and that may not perform well on other objectives. For instance, the Shortest Process Time (SPT) is known that has good performance in minimizing mean cycle time of jobs in a single machine environment, but it results in poor performance for variance of cycle time and due-date-related objectives. On the other hand, the Earliest Due Date (EDD) has good performance in minimizing mean tardiness of jobs in a single machine environment, but it does not perform well for cycle time-related objectives because it does not take information of processing time into account. Hence, a dispatching rule that focuses on single objective may not work well for the realistic objectives. A solution to address this problem is using composite dispatching rules.

Tzafestas and Triantafyllakis (1994) developed a new adaptively weighted combinatorial dispatching (AWCD) rule which is appropriate for complex scheduling problems with multiple and conflicting criteria. This rule identifies the constraints that are tightest, and adaptively estimates the tightness of each constraint. Valente and Schaller (2012) consider the single machine scheduling problem with weighted quadratic tardiness costs. Several efficient dispatching rules are proposed. These include existing heuristics for the linear problem, as well as procedures suitably adapted to the quadratic objective function. Also, both forward and backward scheduling procedures are considered. The computational results show that the heuristics that specifically take into account the quadratic objective significantly outperform their linear counterparts. Also, the backward scheduling approach proves to be superior, and the difference in performance is even more noticeable for the harder instances. The best of the backward scheduling heuristics are both quite efficient and effective. Indeed, this procedure can quickly generate a schedule even for large instances. Also, its relative deviation from the optimum is usually rather low, and it performs adequately even for the more difficult instances. Tay and Ho (2008) solved the multi-objective flexible job-shop problems by using dispatching rules discovered through genetic programming. They evaluated and employed suitable parameter and operator spaces for evolving composite dispatching rules using genetic programming, with an aim towards greater scalability and flexibility. Experimental results show that composite dispatching rules generated by the genetic programming framework outperform the single dispatching rules and composite dispatching rules selected from literature over five large validation sets concerning minimum makespan, mean tardiness, and mean flow time objectives. Further results on sensitivity to changes (in coefficient values and terminals among the evolved rules) indicate that their designs are robust.

2.3.2 Dispatching Rule Selection

There are two types of the rule selection: online and offline. Given an environment and selecting a better rule is the offline rule selection. Because the environment is always stochastic, one rule for all situations of the environment is not enough. A mechanism to dynamically select a rule according to the situation of the environment is the online rule selection.

2.3.2.1 Offline rule selection

The offline rule selection is based on the evaluation of all alternative rules. The rule resulting in the best performance will be selected. Simulation is always a very popular way to evaluate the rules.

Abou-Ali and Shouman (2004) present a study of the effect of dynamic and static dispatching strategies on dynamically planned and unplanned FMS. The proposed simulation model comprised eight machines, storage buffer areas, receiving area, and three robots and pallets. Parts enter and leave the FMS at load/unload stations and transfer between machine centers by available trucks. Based on some specific assumptions, 12 different dispatching strategies were considered. A simulation run was made for each strategy, where the design parameters were systematically changed. The analysis of the obtained results showed that an overall improvement could be achieved for dynamic dispatching than that rendered by static dispatching. Baykasoğ lu and Özbakı r (2010) study the reactive scheduling problems in a stochastic manufacturing environment. Specifically, they test several scheduling policies under machine breakdowns in a classical job shop system. In addition, they measure the effect of system size and type of work allocation (uniform and bottleneck) on the system performance. The performance of the system is measured by the mean tardiness and makespan criteria. They also investigate a partial scheduling scheme under both deterministic and stochastic environments for several system configurations. Lin, Chiu, and Tsai (2008) use analytical network process (ANP) method to construct a dispatching model based on the characteristics of all the production facilities on-site (such as the utilization of bottleneck machines), in order to explore the relationship among various performance indicators and correlation between performance indicators and the dispatching rules. The paper aims to analyze the production dispatching issues of wafer fabs in an effective and systematic approach, to provide an on-site dispatching analysis model that takes into consideration production characteristics and indicator adjustments. The paper finds that the most optimal dispatching method for ANP dispatch model is EDD dispatching method, followed by LS dispatching method. FIFO dispatching method yields the worst performance. The ANP dispatching assess model proposed in this paper can surely serve as an analytical architecture for decision makers to evaluate production dispatching models of multiple production indicators in the future. Pickardt et al. (2013) propose a two-stage hyper-heuristic for the generation of a set of work center-specific dispatching rules. The approach combines a genetic programming (GP) algorithm that evolves a composite rule from basic job attributes with an evolutionary algorithm (EA) that searches for a good assignment of rules to work centers. The hyper-heuristic is tested against its two components and rules from the literature on a complex dynamic job shop problem from semiconductor manufacturing. Results show that all three hyper-heuristics can generate (sets of) rules that achieve a significantly lower mean weighted tardiness than any of the benchmark rules. Moreover, the two-stage approach proves to outperform the GP and EA hyper-heuristic as it optimizes on two different heuristic search spaces that appear to tap different optimization potentials. The resulting rule sets are also robust to most changes in the operating conditions. Petroni and Rizzi (2002) present a fuzzy logic based tool intended to rank flow shop dispatching rules under multiple performance criteria. This tool is detailed concerning a significant industrial case of a major company operating in the boilermaker industry. The results show that the approach is robust and effective in providing practical guidance to scheduling practitioners in choosing priorities dispatching rules when there are multiple objectives.

2.3.2.2 Online rule selection

The online rule selection tries to find out the relationship between the situation of the environment and the rule performing better in the situation. Lots of technologies from the artificial intelligence are introduced, such as Q-learning, neural networks, self-organizing, and so on.

Korytkowski, Wiś niewski, and Rymaszewski (2013) developed an evolutionary simulation-based heuristic to construct near-optimal solutions for dispatching rule allocation. The heuristic is easy to use and gives managers a useful tool for testing a configuration that can minimize certain performance measures. The optimization heuristics are used to determine priority strategies so as to maximize the performance of a complex manufacturing system with a large number of different products, along with an overtime that changes with a mix of different process types, including assembly and disassembly operations and with different types of internal and external disturbances. Modeling is carried out using discrete-event simulation. Wang and Usher (2004) use a single machine agent employing the Q-learning algorithm to develop a decision-making policy on selecting the appropriate dispatching rule from among three given dispatching rules. The system objective is to minimize mean tardiness. The paper presents a factorial experiment design for studying the settings used to apply Q-learning to the single machine dispatching rule selection problem. This study not only investigates the main effects of this Q-learning application but also provides recommendations for factor settings and useful guidelines for future applications of Q-learning to agent-based production scheduling. Mouelhi-Chibani and Pierreval (2010) proposed a new approach based on neural networks (NN) to select in real time the most suited DR as a resource becomes available. The selection is made in accordance with the current system state and the workshop operating condition parameters. Contrary to the few learning approaches presented in the literature to select scheduling heuristics, no training set is needed. The NN parameters are determined through simulation optimization. Shiue, Guh, and Lee (2011) proposed an intelligent multi-controller that consists of three main mechanisms: (1) a simulation-based training example generation mechanism, (2) a data preprocessing mechanism, and (3) a self-organizing map (SOM)-based MSR selection mechanism. The results reveal that over a long period this approach provides better system performance based on various performance criteria than the system performance of the machine learning-based RTS based on the SSR approach for two different types of manufacturing systems (FMS and FAB). Hence, the proposed intelligent multi-controller approach is efficient enough to be incorporated into the operation of an RTS system. Lee (2008) proposed a fuzzy rule-based system for an adaptive scheduling, which dynamically selects and applies the most suitable strategy according to the current state of the scheduling environment. The adaptive scheduling problem is generally considered as a classification task since the performance of the adaptive scheduling system depends on the effectiveness of the mapping knowledge between system states and the best rules for the states. A rule base for this mapping is built and evolved by the proposed fuzzy dynamic learning classifier based on the training data cumulated by a simulation method. Distributed fuzzy sets approach, which uses multiple fuzzy numbers simultaneously, is used to recognize the system states. The developed fuzzy rules may readily be interpreted, adopted and, when necessary, modified by human experts. An application of the proposed method to a job-dispatching problem in a hypothetical flexible manufacturing system (FMS) shows that the method can develop more effective and robust rules than the traditional job-dispatching rules and a neural network approach.

2.3.3 Cooperative Sequencing

El-Bouri (2012) proposed a cooperative sequencing method. The Cooperative Dispatching is a real-time scheduling methodology, which consults downstream machines before making a job dispatching decision on any given machine. The paper proposes such an approach for minimizing the mean tardiness in a dynamic flow shop where new jobs arrive continuously, at random points in time, throughout the production cycle. Cooperative Dispatching is based on the idea that individual machines act self-interestedly, with the objective of optimizing their local performance criteria. A consulted machine attempts to influence upstream dispatching decisions in a manner that promotes its ability to minimize its total local tardiness. A machine's influence in the dispatching decision depends on current congestion and due-date tightness levels in the shop. A multiple regression model is proposed to help determine the weight, and a consulted machine's preferences will carry in the dispatching decision. Conflicting demands from the different machines are resolved by a minimum regret decision procedure, which aims to minimize the aggregate deviation from the consulted machines' preferences. The winning candidate that ultimately emerges from this procedure is the job that is dispatched. A comparative analysis to evaluate the performance of cooperative dispatching, compared to six other dispatching rules that are commonly favored for tardiness-based criteria, is performed by means of simulation, using randomly generated test problems. Computational results indicate that Cooperative Dispatching outperforms the other dispatching rules, across a broad range of flow shop congestion and due-date tightness levels.

2.3.4 Intelligent Sequencing

The intelligent sequencing can be treated as a complex dispatching rule which considers lots of local and global information. The relationship between the information and the priority of the concerned job is usually nonlinear.

Olafsson and Li (2010) learn new scheduling rules from existing schedules using data mining techniques. However, direct data mining of scheduling data can at best mimic existing scheduling practices. They, therefore, propose a novel two-phase approach for learning, where they first learn which part of the data correspond to best scheduling practices and then use this data and decision tree induction to learn new and previously unknown dispatching rules. The numerical results indicate that the newly learned rules can be a significant improvement upon the underlying scheduling rules, thus going beyond mimicking existing practice. Xanthopoulos et al. (2013) proposed two approaches for dynamic scheduling. One is a Reinforcement Learning-based, and one is based on Fuzzy Logic and multi-objective evolutionary optimization. The performance of the two scheduling approaches is tested against the performance of 15 dispatching rules in four simulation scenarios with the different workload and due date pressure conditions. The scheduling methods are compared regarding Pareto optimal-oriented metrics, as well as regarding minimizing mean earliness and mean tardiness independently. The experimental results demonstrate the merits of the proposed methods. Shahzad and Mebarki (2012) presented a data mining based approach to discover previously unknown priority dispatching rules for job shop scheduling problem. The approach is based on seeking the knowledge that is assumed to be embedded in the efficient solutions provided by the optimization module built using tabu search. The objective is to discover the scheduling concepts using data mining and hence to obtain a set of rules capable of approximating the efficient solutions for a job shop scheduling problem (JSSP). A data mining-based scheduling framework is presented and implemented for a job shop problem with maximum lateness as the scheduling objective. Chen (2011) constructed a self-adaptive agent-based fuzzy-neural system to enhance the performance of scheduling jobs in a wafer fabrication factory. The system integrates dispatching, performance evaluation and reporting, and scheduling policy optimization. Unlike in the past studies, a single pre-determined scheduling algorithm is used for all agents, in this study every agent develops and modifies its scheduling algorithm to adapt it to the local conditions. To evaluate the effectiveness of the proposed methodology and to make a comparison with some existing approaches, production simulation is also applied in this study to generate some test data. According to experimental results, the self-adaptive agent-based fuzzy-neural system did improve the performance of scheduling jobs in the simulated wafer fabrication factory, especially concerning the average cycle time and cycle time standard deviation.

2.4 Related Works to the Decentralized Material Flow Control

2.4.1 Decentralized Decision-making

Decentralized decision-making is a process where the decision-making authority is distributed throughout a larger group. It also connotes a higher authority given to lower level functionaries, executives, and workers. This can be in any organization of any size, from a governmental authority to a corporation. The decisions arising from a process of decentralized decision-making are the functional results of group intelligence and crowd wisdom. Decentralized decision-making is often favorable, not only because it can speed up the computation considerably, but also because it can result in more robust and flexible solutions.

The multi-agent system is usually used to address the decentralized decision-making system. Each agent denotes an individual decision maker and has its own goal. There are three types of the multi-agent-based decision-making: adaptive multi-agent decision-making, cooperative multi-agent decision-making, and competitive multi-agent decision-making.

For the adaptive multi-agent decision-making, the agents adapt to the environment by means of learning knowledge from the environment. The decisions they made are dependent on the knowledge they learned. For the cooperative multi-agent decision-making, the agents have the same goal. The decisions are made through the cooperation among them. The cooperative sequencing mentioned in Section 2.3.3 is an example. For the competitive multi-agent decision-making, the agents have conflict goals. The final decision is the results from the competition among them. The market mechanism is introduced in the competition processes, such as biding, negotiation, and so on.

2.4.2 Multi-agent System for the Shop Floor Control

Manufacturing has faced significant changes during the last years, namely the move from a local economy towards a global and competitive economy, with markets demanding for highly customized products of high quality at lower costs, and with short life cycles. In this environment, manufacturing enterprises, to remain competitive, must respond closely to customer demands by improving their flexibility and agility, while maintaining their productivity and quality. Dynamic response to emergence is becoming a key issue in manufacturing field because traditional manufacturing control systems are built upon rigid control architectures, which cannot respond efficiently and effectively to dynamic changes. In these circumstances, the current challenge is to develop manufacturing control systems that exhibit intelligence, robustness, and adaptation to the environment changes and disturbances. The introduction of multi-agent systems and holonic manufacturing system paradigms address these requirements, bringing the advantages of modularity, decentralization, autonomy, scalability, and re-usability (Leitã o 2009).

Baker (1998) described various multi-agent architectures, including the heterarchical architecture. The paper reviews the claimed advantages for multi-agent heterarchies and describes the types of factories that could use this architecture. It surveys the three common types of factory control algorithms: dispatching algorithms, scheduling algorithms, and pull algorithms and describes how all common factory control algorithms used in industry can be implemented in a multi-agent heterarchy.

2.4.3 Decentralized Job Release

The job release can be addressed in the cooperative multi-agent system. Whether to release a job at a decision point can be decided by the cooperation of related machine agents. The problem also can be described in the competitive multi-agent system. The decision that which jobs should be released results from the competition among the job agents. The problem, moreover, can be stated in the adaptive multi-agent system. The decision that which jobs should be released is made by a release agent who has learned enough knowledge.

Rossi and Lödding (2012) proposed a Decentralized WIP Oriented Manufacturing Control (DEWIP). The approach is based on the cooperative multi-agent system. A centralized PPC system generates a list of urgent orders based on information about the market and/or customers' orders. The orders in the list are released for production via the decentralized WIP control loops between the production's workstations. DEWIP releases orders for each operation separately. The release decision is made based on the WIP on the next workstation. The corresponding request for a 'go-ahead' from the next workstations creates an information flow in the same direction as the material flow. The decision about whether the order can be processed and the receipt of this 'go-ahead' however result in information flow opposing the material flow.

2.4.4 Decentralized Routing

Similarly, the routing can also be described as the three types of multi-agent-based decision-making. The routing problem can be illustrated as the competition for the concerned job agent between the alternative machine agents. The problem also can be described as that the job agent selects routes by means of its knowledge.

Csá ji and Monostori (2008) investigated stochastic resource allocation problems with scarce, reusable resources and non-preemptive, time-dependent, interconnected tasks. The proposed approach is a natural generalization of several standard resource management problems, such as scheduling and transportation problems. First, reactive solutions are considered and defined as control policies of suitably reformulated Markov decision processes (MDPs). They argue that this reformulation has several favorable properties, such as it has finite state and action spaces, and it is periodic. Hence all policies are proper, and the space of control policies can be safely restricted. Next, approximate dynamic programming (ADP) methods, such as fitted Q-learning, are suggested for computing an efficient control policy. In order to compactly maintain the cost-to-go function, two representations are studied: hash tables and support vector regression. Several additional improvements, such as the application of limited-look ahead rollout algorithms in the initial phases; action space decomposition; task clustering and distributed sampling are investigated, too. This approach uses an adaptive agent to make the decision.

2.4.5 Decentralized Sequencing

The sequencing problem can be decentralized by means of cooperation between downstream machine agents. Whether a job is dispatched relies on the cooperation results. The problem can also be expressed as several job agents compete for a machine or a machine agent uses its sequencing knowledge to select the job.

Liu and Sycara (1997) present a multi-agent problem-solving model and an effective coordination technique for job shop sequencing problem. The model involves a group of agents; each agent is associated with either a job or a resource. A solution to a sequencing problem is the result of coordinated conflict resolution in the iterative and asynchronous multi-agent decision-making process. Madureira (2005) modeled a Manufacturing System by means of a Multi-Agent Systems, where each agent may represent a processing entity. This work has an objective to deal with the complex problem of Dynamic Scheduling in Manufacturing Systems. He wanted to prove that a good global solution for a scheduling problem may emerge from a community of machine agents solving their schedules locally and cooperating with other machine agents that share some relations between the operations/jobs. The proposed approach is in line with reality and away from the approaches that deal with static and classic or basic Job-Shop scheduling problems. In fact, in the real world, where problems are essential of dynamic and stochastic nature, the traditional methods or algorithms are of very little use. This is the case with most algorithms for solving the so-called static scheduling problem for different settings of both single and multi-machine systems arrangements. This reality, motivated him to concentrate on tools, which could deal with such dynamic, disturbed scheduling problems, for multi-machine manufacturing settings, even though, due to the complexity of these problems, optimal solutions may not be possible to find. Miyashita (1998) proposed a new integrated architecture for distributed planning and scheduling that exploits constraints for problem decomposition and coordination. The goal is to develop an efficient method to solve densely constrained planning/scheduling problems in a distributed manner without sacrificing solution quality. A prototype system (CAMPS) was implemented, in which a set of intelligent agents try to coordinate their actions for ‘ satisfying’ planning/scheduling results by handling several intra- and inter-agent constraints. The repair-based methodology for distributed planning/scheduling is described, together with the constraint-based mechanism of dynamic coalition formation among agents.

Gabel and Riedmiller (2007a) adopt an alternative view on production scheduling problems by modeling the job-shop sequencing problems as multi-agent reinforcement learning problems. In fact, they interpret job-shop sequencing problems as sequential decision processes and attach to each resource an adaptive agent that makes its job dispatching decisions independently of the other agents and improves its dispatching behavior by trial and error employing a reinforcement learning algorithm. The utilization of concurrently and independently learning agents requires special care in the design of the reinforcement learning algorithm to be applied. Therefore, they develop a novel multi-agent learning algorithm that combines data-efficient batch-mode reinforcement learning, neural network-based value function approximation, and the use of an optimistic inter-agent coordination scheme. The evaluation of the learning framework focuses on numerous established Operations Research benchmark problems and shows that the approach can compete very well with alternative solution methods.

2.5 Discussions

For the decision-making problem, theoretically, the more information that is considered, the better decision that is made and the wider scope the approach will have. However, from above review, most studies consider very limited information. For example, for the real-time job release, they considered either due dates or workloads. However, in reality, the due dates, workloads and even more information should be involved. The real-time routing and sequencing also face the similar problem. Because of the limitation on the considered information, the approaches work only in some specific situations. Different approaches must be developed to fit different situations. Like the rule selection approach in the sequencing problem, we should select rules according to the situation. Obviously, if we can consider all information which influences the decision-making, the approach will be applicable to all situations. Thus, how to organize and utilize all information to make decisions is an important issue. Unluckily, no literature study on it. Moreover, some information which influences the decision-making may be hard to quantize. We must find another way to consider them.

Besides the information considered, the future prediction is also very important in the decision-making. The predicted future can help us to make good decisions. Especially, if we can predict what will happen in future after a decision is executed, the decision-making will become wiser. Nevertheless, lots of studies are even not aware of it. For example, the cooperative sequencing focused only on the current situation while most of the intelligent sequencing methods considered only the historical data. The release policies and dispatching rules may be able to see a very short future, like look ahead rule, but they do not know what the long-term influence of a single decision is. No literature introduces how to predict what will happen in future after a decision is executed and how to use the predicted information.

Back to the problems that we are going to solve, i.e., the real-time job release, routing, and sequencing. We also did some reviews on the predictive job release, routing, and sequencing. Because they are out of the scope of the study, we do not note them in the thesis. Based on this unmentioned literature, we seldom find the works that combine these three types of decision-making together to carry out their study. The reason probably is that the mixed predictive problem is too complicated. Even for the real-time problems, no one studies them together. These three decisions have a coherent relation between each other. Better performance may be obtained using coordination of the three decisions. From the review of the real-time and decentralized decision-making, we find it will be easier to combine them in a decentralized real-time manner.

Furthermore, when we develop any system for the manufacturing shops, we must make sure the system is also suitable for the future manufacturing industry. Nowadays, smart factories are the trend of the manufacturing industry. The main principles of the smart factories are decentralization and intelligence, which means each resource and each piece of materials can learn and make decisions. Thus, the decentralized intelligent material flow control is required. However, most literature does not consider it.

All in all, based on above discussion, the study will consider more information in the decision-making and try to organize the information and find a proper way to use the information. In the study, the simulation will be used to predict the future. Job release, routing, and sequencing problems will be solved together. At last, a decentralized intelligent real-time material flow control system will be developed.

3 Agent-based Simulation with Process-interaction Worldview

Simulation is the foundation of our approaches. Because we need more flexible ways to initialize the simulation model and control the simulation progress, we decide to develop a simulator from scratch. In this chapter, in the beginning, we explain how we use the simulation in our approaches and why we select the agent-based simulation (ABS). After analyzing the agent-based simulation, we realize that most research focuses on the agent-based modeling, but only a little research pays attention to the simulation. The reason is that the agent-based model (ABM) can run directly in a real-time manner by communicating with each other and the running of the ABM is already one kind of simulation. Obviously, the 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 introduce a process-interaction worldview (PIW) originated in the discrete event simulation to the ABS. The remaining parts of this chapter will give a detailed description of the method for combining the agent-based simulation and the PIW. A framework of the agent-based simulation upon this method is also depicted. Finally, the method is validated by applying in a simple queue system and compared with the normal ABS with a timescale.

3.1 Role and Necessity of Simulation

Our approaches we will state later are heavily dependent on the simulation. In the simulation try-then-decide method (STTD), the simulation is used both to evaluate alternatives and to emulate the environment. In the intelligent method based on the STTD, the simulation generates data for us to build data-driven models. In the intelligent method based on Markov decision process, the simulation is used to explore the state space and return rewards. All these usages require very flexible ways to control the simulation process, like programmatically generating and initializing simulation models, and communication among different simulation runs. Also, we want to observe every detail, e.g., concurrent events, while the simulation is running. The existing simulators fail to meet these requirements. Thus, it is necessary to develop a simulator by ourselves. Discrete event simulation (Fishman 1978) and agent-based simulation (Macal and North 2010) are two options for us. The reason why we select the agent-based simulation has two points. The first is that it is a natural method which describes a system composed of real-world entities and makes the model seem closer to reality. The second is its flexibility. The agent-based simulation is easy to extend the system as a whole by adding more agents to an agent-based model, and also provides a framework for tuning the complexity of an individual agent or group of agents by elaborating on their behaviors, degree of rationality, interaction rules, or abilities to learn and evolve(Bonabeau 2002). This can be done, for example, by adding different behavioral algorithms to an agent's code. In addition, the agent-based simulation will facilitate our study. In chapter 6 and 7, we will create a decision maker for each machine and each product, and, at last, these decision makers will be integrated into the simulation model. In this case, it is very easy to integrate the decision-makers to the ABM just by adding decision-making algorithms to the agents' behaviors.

3.2 Issues on the Agent-based Simulation

3.2.1 Agents and Agent-based Simulation

Before introducing agent-based simulation (ABS), we need to discuss the definition of the agent first. There still exists controversy over the definition. One concept of the agent appears in the distributed intelligence in Artificial Intelligence(Sugumaran 2008), in which besides distributed, autonomous, and social features, agents have to be intelligent, such as being able to perceive, learn, and adapt to the environment. Most of the researchers following this concept direct towards the multi-agent system(Lian, Shatz, and He 2009), and some researchers focus on the hardware agent(Tapia et al.) which has been widely used in the robotics(García et al. 2012). Another concept , which is derived from the emergence theory(Namatame 2007) in which a key notion is that simple rules generate complex behaviors, in other words, system properties emerge from its constituent agent interactions(Bonabeau 2002), is very similar to the cellular automata(Wolfram 1994) in which agents are asked to keep simple and short, which is contrary to the first concept because the intelligence certainly makes agents more complex. However, they also have a lot in common, such as autonomy, society, distribution and so on. In most research, the contrary parts of the agents seem to be discarded. Both the intelligence and the simplicity are not given weight, but much attention is paid to the common parts, autonomy, society, and distribution (Váncza and Márkus 2000, Oliva, Panzieri, and Setola 2010, McLane et al. 2011). These research lead to the formation of a new bottom-top modeling method(Macal and North 2010), i.e., agent-based modeling, in which a system is modeled as a collection of autonomous agents. Based on a set of rules each agent individually assesses its situation, makes decisions and may execute various appropriate behaviors for the system it represents(Bonabeau 2002). The agent-based modeling is kind of similar to the object-oriented modeling(Davidsson 2001), but more flexible and more natural to describe the system. There are no strict requirements of intelligence or simplicity for the agents in the agent-based model. Now we define the ABS. The ABS is the process of designing an ABM of a real system and conducting experiments with this model to understand the behavior of the system and evaluating various strategies for the operation of the system(Shannon 1975). At present, the researchers on the ABS mainly focus on two directions, agent-based modeling and its application. The first one is trying to build a standard, and universal framework for the modeling (Botta et al. 2011, Fortino and Russo 2012, Grimm, Berger, and Bastiansen 2006), Autonomy, society, and distribution features of agent involve in. Another one focuses on creating domain agents through studying their attributes and behaviors (Chen 2012, Rebaudo and Dangles , Cao and Chen 2012). Both have provided lots of approaches and mechanisms for agent-based modeling and application in practice.

3.2.2 Simulation in the ABS

Based on the literature review, the current studies on the ABS focus on the ABM, and few researchers pay attention to the simulation in the ABS. Probably because the ABM can run directly in a real-time way(Macal and North 2010), in which the running of the ABM is a simulation run, and there is no necessity to study the simulation separately. However, this type of simulation misses some important contents in the computer simulation such as the worldview in the simulation, also referred to as a simulation strategy. Therefore, many researchers studying on the computer simulation, especially the discrete event simulation, cannot help asking where the simulation is in the ABS. The so-called simulation in the ABS is a real-time simulation in which the simulation time equals the real time. However, the ABS is less efficient when being used in a non-real-time system. In some research, the ABS is speeded up by giving a timescale under the condition of synchrony. There are two ways to achieve the synchronization: conservative algorithm and optimistic algorithm. The Conservative algorithm keeps the model running in sync exactly, but the optimistic algorithm allows asynchronous phenomenon to occur and then makes it synchronous, such as SimJade does. The SimJade(Pawlaszczyk and Timm 2007) is a synchronization service for the JADE using an optimistic synchronization technique to manage the time in a distributed way. Because the optimistic synchronization techniques allow the asynchronous phenomenon to appear, the agents influenced by the asynchronous phenomenon have to roll back. So, lots of time is consumed by the rollback. In addition, the real-time ABS with a timescale has two features: (1) there is no central time manager, and the agents move on according to their local time (computer clock); (2) The time spent on executing code is counted in the simulation time. In the real-time ABS without a timescale, the time for code execution is very short and can be ignored. On the other hand, in the real world, it also takes time while people make a decision. So, the time for executing the codes exists reasonably in the real-time ABS. However, in the real-time ABS with the time scale, the execution time is enlarged, and errors occur. For example, an agent enters the system and informs other agents it's coming.

Then the agent stays in the system for 10000 s and leaves. Theoretically, the agent stayed in the system for 10000 s. However, in the real-time ABS, the time that the agent stayed in the system is 10000 s plus the time spent on informing other agents (code execution time). If we assume the execution time is 0.0001 s and the time scale is 1, the total time is 10000.0001 s. It is very close to the theoretical value. However, if the time scale is 10^{-8}, the total time is $(10000*10^{-8}+0.0001)/\ 10^{-8}$ =20000 s. This error is very big and cannot be ignored anymore. And the worse thing is that the precision of simulation results will decrease as the timescale increases. We face a tradeoff between the efficiency and the precision. So how to speed up the ABS without losing any precision is a key issue at this stage.

3.2.3 Efficiency of the ABS

If back to the computer simulation again, we can find that the simulation has different efficiencies when different worldviews are used. There are two main types of the worldview, time-driven worldview, and event-driven worldview. In the time-driven worldview, the world progresses as the time is passing with a fixed increment (time step). Correspondingly, the world progresses as some events occur in the event-driven worldview which includes three sub worldviews: event scheduling, activity scanning, and process interaction. Introducing a suitable worldview into the ABS will be a good way to speed up the simulation. But some researchers (Pawlaszczyk and Timm 2007, Davidsson 2001) argue that the simulation worldview violates the autonomy principle of the agents due to the centralization of time handling and sharing. So, most researchers did not study the ABS and the worldview together. Siebers even declares that the discrete event simulation is dead, long live the ABM (Siebers et al. 2010). However, the time handling is only in charge of simulation time which is independent and never affected by the agent. The local clocks in each agent merged into one sharing simulation clock does not intervene in the behavior of agents at all, and the agents still take action autonomously. Therefore, it is possible to introduce the worldview to the ABM.

A little research has already focused on this field and obtained great progress, such as the entity-relationship and agent-oriented-relationship (ER"AOR) (Wagner 2004). Agents, objects, events, and messages are entities in the ER"AOR; the agents and the objects are distinguished; the messages and the events are managed together to control the simulation time. In the ER"AOR, it is natural to partition the simulation system into the environment simulator and some agent simulators. The environment simulator is responsible for advancing simulation time and managing the state of all external (or physical) objects and the external/physical state of each agent; a number of agent simulators are in charge of managing the internal (or mental) state of agents. By means of ER modeling and combination with the discrete event simulation, the ER"AOR has attracted extensive attention. However, we realize that the ER"AOR simulation is not a pure ABS because of the objects in the model. In the pure ABS, the objects which cannot be modeled as agents must belong to certain agents (attributes). In addition, because the conditional events or messages are involved, and the lifecycle of the agent is divided into many activities, the AOR simulation is the ABS with the activity-scanning worldview. As we all know, the activities-scanning worldview is not the most efficient one. It is better to choose a more efficient worldview for the ABS. Now we come to the next question: which worldview is more suitable and efficient for the ABS.

3.2.4 Worldview for the ABS

The event-scheduling worldview focuses on the events that instantaneously transform a system's state and schedule future events (Miller et al. 2004). The advantage of this worldview is that periods of inactivity can be skipped over by jumping the clock from one event time to the next event time. The event-based approach is the most computationally efficient one of the three classical worldviews. The activity-scanning worldview focuses on activities and their preconditions (triggers)(Miller et al. 2004). An activity's preconditions must be satisfied for an activity's operations. This worldview is less efficient than the event-scheduling worldview because it requires a frequent evaluation of conditions. The process-interaction worldview can be considered a combination (hybrid) of the activity-scanning worldview and the event-scheduling worldview (Zeigler, Praehofer, and Kim 2000). It focuses on processes and the entities that flow through the processes and interact with resources (Banks and Carson 1985). The process-interaction worldview is more efficient than the activity-scanning worldview, but it is less efficient than the event scheduling worldview.

From the analysis above, we can see that the process-interaction worldview is the second most efficient worldview. But besides efficiency, the choice of worldview should be made by considering other characteristics such as maintainability, modifiability, reusability, and ease of development. The process-interaction worldview is considered to be a natural way to describe models (Franta and Maly 1977) and is closer to most people's mental model. In addition, the notion of "process" corresponds closely to the lifecycle of the agent and the implementation of the process-interaction worldview is very similar to the agent-based model. Moreover, if we use the event-scheduling worldview or the activity-scanning worldview, the flexibility, maintainability, and modifiability of the ABM will shrink. Therefore, we introduce the process-interaction worldview into the ABS to speed up the simulation.

3.3 Agent-based Simulation with the PIW

In this section, we will introduce an approach (ABS&PIW) which brings the process-interaction worldview into the agent-based simulation. First, we discuss agents in detail including its attributes, behaviors, and messages. And then some concepts in the ABS&PIW are defined. To make the approach more formal and rigorous, we formulate the approach. At last, we discuss the parallelism in the ABS&PIW.

3.3.1 Agents in the Agent-based Simulation

In our study, an agent has some attributes, behaviors, messages, and activation points (see Figure 3.1). The behaviors endow the agents with abilities to make independent decisions, and the messages are medium for their communications. Activation points are designed for the process-interaction worldview.

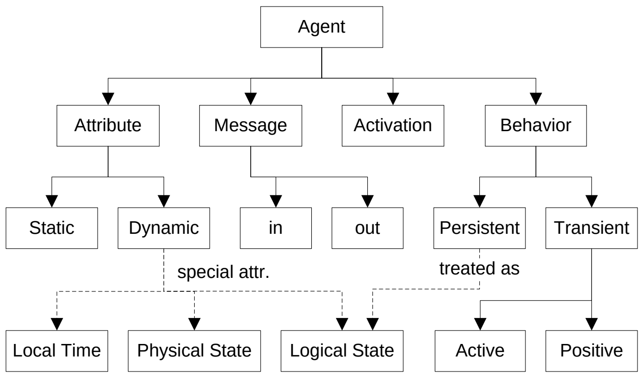


Figure 3.1: Attributes, behaviors, messages, and activations in an agent

3.3.1.1 Attributes

Attributes are characteristics of the agent. An agent's attributes can be static, i.e., not changeable during the simulation, or dynamic, i.e., changeable by behaviors as the simulation progresses. For example, a static attribute is an agent's name; a dynamic attribute is an agent's memory of past interactions. The agent adapts to the environment by changing its attributes. There can be a large number of attributes in an agent, but only attributes related to the goal of the system need to be considered. The agent has three special attributes: local time, physical state, and logical state. The local time is from the inner clock in the agent, and it may be not synchronous with the simulation time. The two states will be defined in Section 3.3.2.1.

3.3.1.2 Behaviors

There are two types of behaviors: persistent behaviors and transient behaviors. The persistent behaviors are equal to activities, and they will change the state of the agent. One persistent behavior is related to one logical state, so the persistent behaviors are treated as logical states. Transient behaviors can be divided into passive behaviors and active behaviors. The passive behaviors are responsible for receiving messages and updating dynamic attributes, and the active behaviors are in charge of generating and sending messages.

3.3.1.3 Messages and Activation

Agents can receive/send messages from/to the other agents. The message has a given format and typically contains sender, receivers, sending time, keywords, and content. The sending time is the local time of the sender agent when the message is sent. The local time is updated with the sending time of the received messages. The concept of activation will be introduced in Section 3.3.2.1.

3.3.2 Concepts in the ABS&PIW

3.3.2.1 Concepts in the ABS&PIW

Firstly, six concepts are given. Delays and activation points come from the process-interaction method but offer some improvements.

Physical state of the agent*: active and blocked, is related to the implementation. If an agent is blocked, it gives up control of the CPU. Otherwise, it occupies the CPU.*

Logical state of the agent *closely connected to the application domain and is the same as the state of the entity. The logical state is a very important dynamic attribute.*

State of the agent-based model*: ready or unready. If the physical states of all agents are blocked, the state is ready. Otherwise, it is unready.*

Straggler message *is a message that its sending time is earlier than the local time of the receiver agent. It means a later message is received before the earlier message. The straggler message will make the simulation wrong and must be avoided. There are two ways to avoid it: a conservative algorithm which does not allow the straggler messages and an optimistic algorithm which allows them and corrects them later.*

Delay *is a period in which the logical state of the agent stays the same. When a delay occurs, the agent will create the next activation point and become blocked.*

Activation point *is a time position where a delay ends. The agent is activated at this point and performs actions until a new delay occurs. An activation point has such a given format including activation time, activation agent, and keywords. There are two types of activation points, conditional and non-conditional. Non-conditional activation points are explicit. In contrast, the conditional activation points are uncertain in which delays of the agents do not end until the agents meet the given condition.*

3.3.2.2 Relationship among some Concepts

The relationship among these concepts is shown in Figure 3.3. The agent is similar to the active entity; the life cycle of the agent is the process of the entity and is made up of a series of activation-delay-activation. The activation point is located at the time an event occurs, and the physical state turns active at this time. The agent responds to the event through a transient behavior. During a delay, an activity is being carried out, and the physical state of the agent turns blocked. The activity is a persistent behavior corresponding to a logical state.

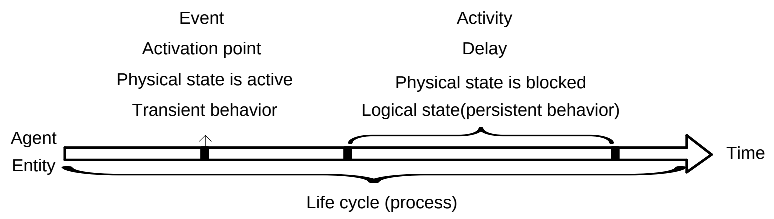


Figure 3.2: Relationship between activation point, event, and so on

3.3.3 Formulation of the ABS&PIW

3.3.3.1 Symbol Definition

t, t_{plan} current and planned simulation time

t_{sa} sampling interval time

s_{mo} state of the ABM(0-ready,1-unready)

u flag of model update (0-no update,1-update)

R_{FAL}, R_{FAL}^0 activations in the future activation list and initial value

R_{CoAL}, R_{CoAL}^0 activations in CoAL and initial value

R_{CuAL} current activations in CuAL

a an agent, $a \in A$,A is a set of all agents

r_a current activation point of the agent a

c_{r_a} type of activation point(1-conditional,0-uncondt.)

g_{r_a} flag of the condition(0-unmeet,1-meet)

R_a a set of activation types from the agent a

m_a a message received by the agent a

M_a^{out} a set of message types sent by the agent a

M_a^{in} a set of message types received by a

τ_{m_a} timestamp of the message m

τ_r activation time (timestamp of the activation point)

τ_a local time of agent a updated by the time stamp τ_{m_a} or τ_r

s_a physical state of the agent a

f_a agent attribute, $f_a \in F_a$

F_a, F_a^0 a set of all attributes and initial value

b agent behavior

B_a^{po} , B_a^{ac} a set of passive and active behaviors

$H_a^1 = RS(r_a, b_a^{po}) = \{h_{r,b}^1 \mid r \in R_a, b \in B_a^{po}\}$

relationship between activations and passive behaviors . If b is related to r, $h_{r,b}^1 = 1$, otherwise $h_{r,b}^1 = 0$. The following relationships have the same rule.

$H_a^2 = RS(m_a^{in}, b_a^{po}) = \{h_{m,b}^2 \mid m \in M_a^{in}, b \in B_a^{po}\}$

$H_a^3 = RS(r_a, b_a^{ac}) = \{h_{r,b}^3 \mid r \in R_a, b \in B_a^{ac}\}$

$H_a^4 = RS(m_a^{in}, b_a^{ac}) = \{h_{m,b}^4 \mid m \in M_a^{in}, b \in B_a^{ac}\}$

$H_a^5 = RS(b_a^{po}, f_a) = \{h_{b,f}^5 \mid b \in B_a^{po}, f \in F_a\}$

$H_a^6 = RS(b_a^{ac}, a', m_a^{out}) = \{h_{b,a',m}^5 \mid, b \in B_a^{ac}, a' \in A, m \in M_a^{out}\}$

$H_a^7 = RS(b_a^{ac}, r_a) = \{h_{b,r}^7 \mid, b \in B_a^{ac}, r \in R_a\}$

3.3.3.2 Four-tuple of the ABS&PIW

We provide a mathematical framework for the ABS with the process-interaction worldview. The simulation is specified as a four-tuple $SIM = (I, TM, ABM, O)$. I is a set of inputs $I = (R_{FAL}^{0}, R_{CoAL}^{0}, \{F_a^0 \mid a \in A\})$ including initial activation points and initial attributes of all agents. O is a set of outputs $O = (\{(t, F_a^t) \mid a \in A, 0 < t < t_{aim}, \mathrm{mod}(t, t_{sa}) = 0\})$, which is made up by attributes of all agents at each sample point. TM is a time manager $TM = (t, t_{plan}, R_{FAL}, R_{CoAL}, R_{CuAL})$ who is in charge of the simulation time and manages all activation points created by agents. The activation points are grouped into three lists: conditional activation list (CoAL), future activation list (FAL), and current activation list (CuAL). FAL and CoAL are direct lists. The activation points coming from the agents are put into them. CuAL is an indirect list and the earliest activation points are moved in from FAL. The planned time t_{plan} is the maximal simulation time. The simulation will end when the simulation time t reaches the planned time. ABM is an agent-based model described as $ABM = (s_{mo}, u, \{(\tau_a, s_a, F_a, B_a^{po}, B_a^{ac}, M_a^{in}, M_a^{out}, R_a, RS_a) \mid a \in A\})$. The model state s_{mo} and the updated flag u are used by the TM to advance the simulation. The simulation clock advances whenever the model state is ready and the model does not update anymore. In this way, we can avoid the straggler messages and ensure that all conditional activation points which meet the corresponding conditions are activated as soon as possible. This is a conservative synchronization algorithm. Local time τ_a and physical state s_a are two special attributes and play a great role in the simulation. We extract them from the attributes and consider separately. The physical state is used to determine the model state. The model state will be ready if the physical states of all agents are blocked. The ABM is different from the general one(Macal and North 2010) which contains only three elements: agents, relationship, and messages. In a complex system, the relationship changes dynamically and there are massive situations. It is difficult to express the relationship among all agents by a two-axis matrix. But for the individual agent, the situations of relationship with other agents are countable. So, in our ABM, the relationship and messages are specified for the individual agents and the target agents with corresponding messages can be got by some simple IF-THEN rules. RS_a is a set of the relationship sets, $RS_a = (H_a^1, H_a^2, H_a^3, H_a^4, H_a^5, H_a^6, H_a^7)$, which contains seven relationships such as the relationship between received messages (in) and passive behaviors, as well as the relationship between active behaviors, target agents and corresponding messages (out).

3.3.3.3 Procedure of the ABS&PIW

The following is a procedure for the simulation. Simulation initialization (1), advancing time (2), and activating agents (3) are executed by the TM. The simulation is initialized with the inputs. Simulation clock t advances according to the time of the earliest activation points. All concurrent activation points, including both current activation points and conditional activation points, are activated at a time.

```
(1) Initialize
    $t = 0$, $R_{FAL} = R^0_{FAL}$, $R_{CoAL} = R^0_{CoAL}$
    $F_a = F^0_a$, $s_a = 0$, where $a \in A$
    $s_{model} = 1$
(2) Advance time
    if $R_{FAL} = \phi$ or $t > t_{plan}$ then simulation ends
    $\tau_{\min} = \min(\{\tau_r \mid r \in R_{FAL}\})$
    $R_{CuAL} = \{r \mid \tau_r = \tau_{\min}, r \in R_{FAL}\}$
    $R_{FAL} = R_{FAL} - R_{CuAL}$
    $t = \tau_{\min}$
(3) Activate
    $R' = \{R_{CoAL}, R_{CuAL}\}$, $R_{CuAL} = \phi$
    if $R' = \phi$ then go to (2)
    $A' = \{a' \mid r_{a'} \in R'\}$
    $S_{A'} = 1$, where $S_{A'} = \{s_{a'} \mid a' \in A'\}$
    TM activates $A'$ at $R'$
(4) When an agent $a' \in A'$ is activated at $r_{a'} \in R'$
    $\tau_{a'} = \tau_{r_{a'}}$
    Get passive behavior $b_1$ by $H^1_{a'}$,
    which satisfies $h^1_{r_{a'},b_1} = 1, h^1_{r_{a'},b_1} \in H^1_{a'}$
    if $c_{r_{a'}} = 1$ then
        $g_{r_{a'}} = g^{new}_{r_{a'}}$, where $g^{new}_{r_{a'}}$ is new one from b1
        If $g_{r_{a'}} = 0$ then $s_{a'} = 0$ and go to (6)
    end if
    if $F'_{a'} \neq \phi$ then $F'_{a'} = F^{n'}_{a'}$,
    where $F^{n'}_{a'}$ are new values calculated by $b_1$ and
        $F'_{a'} = \{f \mid h^5_{b_1,f} = 1, f \in F_{a'}, h^5_{b_1,f} \in H^5_{a'}\}$
    u=1
    Get active behavior $b_2$ by $H^3_{a'}$
    which satisfies $h^3_{r_{a'},b_2} = 1, h^3_{r_{a'},b_2} \in H^3_{a'}$
    if $\exists r'_{a'} \in R_{a'} : h^7_{b_2,r'_a} = 1, h^7_{b_2,r'_a} \in H^7_{a'}$ then
        if $c_{r'_{a'}} = 0$ then $R_{FAL} = R_{FAL} + \{r'_{a'}\}$
        ifn $c_{r'_{a'}} = 1$ then $R_{CoAL} = R_{CoAL} + \{r'_{a'}\}$
    end if
    $(A'', M^{out'}_{a'}) = \{(a'', m) \mid h^6_{b_2,a'',m} = 1, h^6_{b_2,a'',m} \in H^6_{a'}\}$
    if $A'' \neq \phi$ then
        $S_{A''} = 1$, where $S_{A''} = \{s_{a''} \mid a'' \in A''\}$
        send messages $M^{out'}_{a'}$ to $A''$
    end if
    if $c_{r_{a'}} = 1$ then $R_{CoAL} = R_{CoAL} - \{r_{a'}\}$
    $s_{a'} = 0$
    go to (6)
(5) When an agent $a'' \in A''$ receives the message $m_{a''}$ from the agent $a'$
    It is similar to step (4) and just needs the following replacements and to ignore condition activation:
    $a' \to a''$ , $r_{a'} \to m_{a''}$ , $H^1_{a'} \to H^2_{a''}$ , $H^3_{a'} \to H^4_{a''}$
(6) When an agent is blocked
    if $\forall a \in A : s_a = 0$ then $s_{mo} = 0$
    if $s_{mo} = 0$ then
        if $u = 1$ then $u = 0$ and go to (3)
        if $u = 0$ then go to (2)
    end if
```

In steps (4, 5), when an agent becomes active or receives messages, passive behaviors handle the received messages or activations, and update its attributes. Active behaviors create new activation points and communicate with others. Decisions on the timing of advancing the time and quitting repeat of the conditional activations (6) are made by ABM according to the model state and the updated flag whenever the physical state of one agent becomes blocked.

3.3.3.4 Parallelism in the ABS&PIW

In the agent-based simulation, agents run in parallel. After the concurrent activation points are activated simultaneously, associated agents will respond in parallel. The parallelism in ABS is shown in Figure 3.3. To avoid the straggler messages mentioned above, we adopt the conservative synchronization algorithms to ensure the correct local time (see step 6). Even though it cannot fully take advantage of parallelism, it can prevent the straggler messages from appearing at all and save the rollback time spent on the optimistic algorithm.

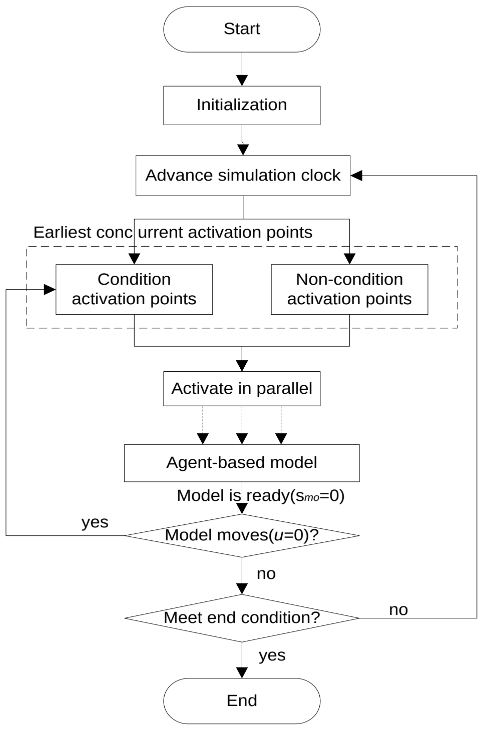


Figure 3.3: Parallelism in the agent-based model

3.4 Development of a Framework for the ABS&PIW

To reuse the code and make the development of the ABS easy, in this section we will give a general idea that how to develop a framework for the proposed approach. We will design the individual agent and the time manager first and then implement them.

Finally, a static structure of the framework is built, and a code sample shows how to use it. The framework is developed in Java programming language.

3.4.1 Design of Individual Agent

The structure of the agent is shown in Figure 3.4. The agent is composed of attributes, an initialization method, a behavior controller, and a message handler.

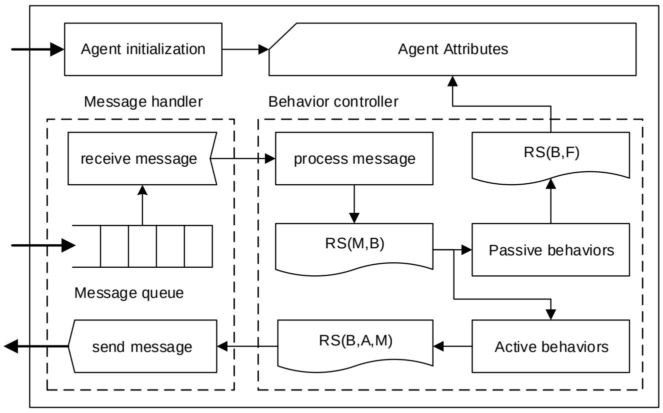


Figure 3.4: Structure of individual agent

3.4.1.1 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.

3.4.1.2 Behavior Controller

The behavior controller decides which behavior to be executed when receiving a message. The decision is made depending on the relationships RS mentioned in Section 3.3.2. Here we just summarize them in three types of relationships: messages (in) with behaviors RS (M, B), behaviors with attributes RS (B, F), and behaviors with other agents and messages (out) RS (B, A, M). The behavior controller can also control behaviors, such as adding behavior, removing behavior, and changing behavior's state according to the environment.

3.4.1.3 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.

3.4.2 Design of Time Manager

In order to keep consistent with the agent-based model, the time manager is also developed as an agent to be responsible for advancing time, activating agents, and managing activation points. The time manager extends the class of agents, and Figure 3.5 shows its structure.

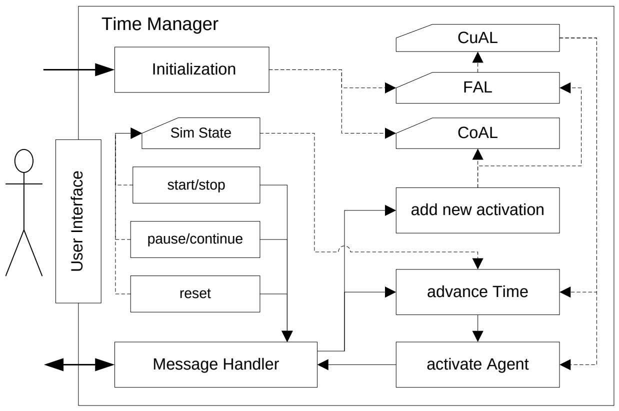


Figure 3.5: Structure of the time manager

3.4.2.1 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.

3.4.2.2 Activation Point Lists in the Time Manager

Three lists of activation points 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.

3.4.3 Design of the Agent-based Model

3.4.3.1 Single Group Agent-based Model

The agent-based model includes the agent environment, the agent manager, and a set of agents (shown in Figure 3.6). 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. The agent manager is in charge of all agents and provides the model state to the time manager. 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.

3.4.3.2 Multi-group Agent-based Model

The agent-based model may also be divided into several groups. Each group has the same structure as the single group model. The time manager must be in one of the groups or a new group. For the multi-group ABM, the environment of a group only influences the agents in the group and the agent manager. Besides the environment of the group, the agent manager can also communicate with managers in other groups, shown in Figure 3.6. The agent managers share one agent manager environment (see Figure 3.7), which enables agents to communicate among the groups.

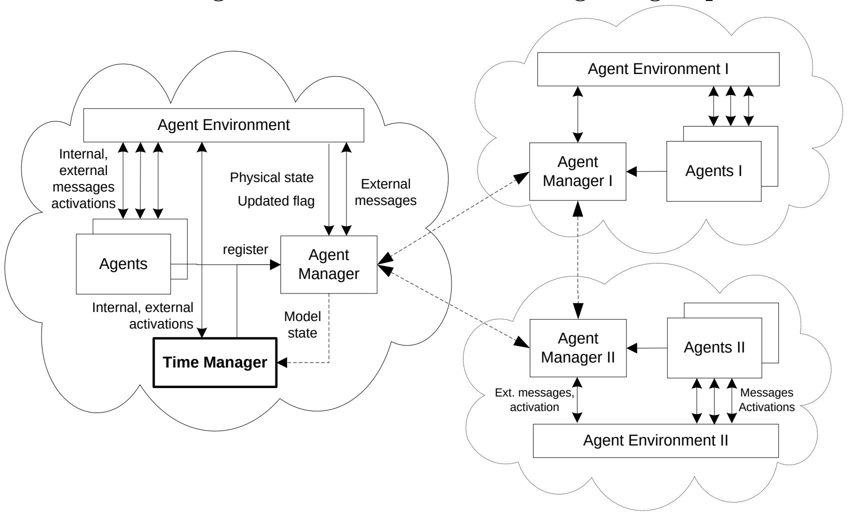


Figure 3.6: Structure of the Agent-based Model

Agents sending a message to an external agent must send a message to the agent manager first. The agent manager will hand the message over to another agent manager containing the target agent. And then the target agent will receive the message from its manager. The time manager communicates with external agents in the same way. The activation points of external agents must go through the local agent manager and the target agent manager who is in the group containing the time manager, and at last, are received by the time manager. The time manager activates an external agent in the same way.

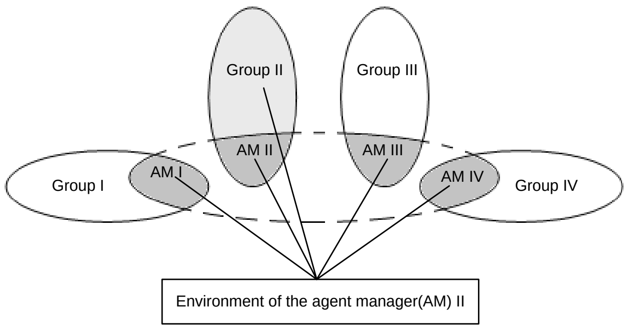


Figure 3.7: The environment of agent managers

3.4.4 Implementation of the agent-based model

3.4.4.1 Implementation of the Agent-based Model

The agents in the model run in parallel. Therefore, an agent-based model is developed using multi-threading and synchronization technology. An agent acts as a thread in the computer to facilitate itself to run in parallel with other agents. To decrease the occupation time of the CPU, we set a synchronous locker for each agent. While a delay occurs in the agent, the thread will be blocked (physical state) by the locker calling method “wait” and gives up control of the CPU. After the delay, the thread becomes active again and starts running when the locker calls method “notify.” The agent environment is implemented as the data shared by the agent threads. Most of the sharing data is the message queues of agents, and the rest are synchronous lockers. The agent manager is also developed as a special agent. To make the relation between the time manager and the agent manager clear, we connect the time manager to the agent manager directly in Figure 3.8. The time manager communicates with the agent manager through the environment too.

3.4.4.2 Implementation of the Communications

Communications among agents are achieved by sharing data between threads. The safety of the data sharing is guaranteed by synchronization technology. The implementation of the communication is shown in Figure 3.8. There are two types of communication, internal and external. The internal way is that the agent directly puts the message into the associated queue according to the receiver name and notifies the receiver agent. If the receiver agent is blocked, it will be activated to receive the message. In an external way, the agent needs to put the message into the message queue of its manager first, and then the manager reads it and pushes the message into the target agent manager's queue. The target agent manager dispatches the message to the receiver's queue and notifies it. The receiver agent will read the message from its queue.

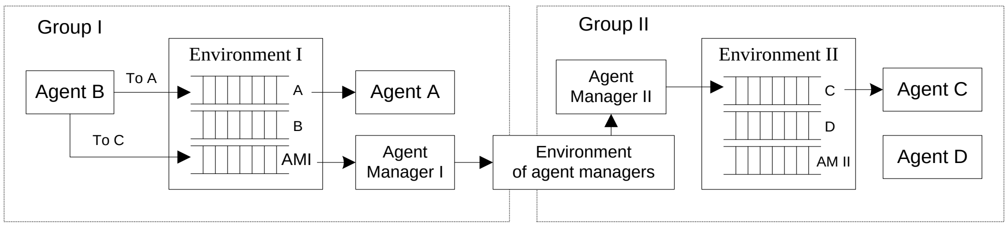


Figure 3.8: Implementation of the agent communication

3.4.5 Static Structure and Sample Code of the Framework

Figure 3.9 shows a static structure of the framework package. The time manager, agent environment, and basic communications are already considered in the framework. It is not necessary for users to consider them again while developing the simulator. What the user needs to do are agent abstractions and definitions, such as the attributes F, behaviors B, messages M, and relationships RS. A code sample which gives a very brief idea how to use the framework for the single group ABS can be found in appendix A.1.

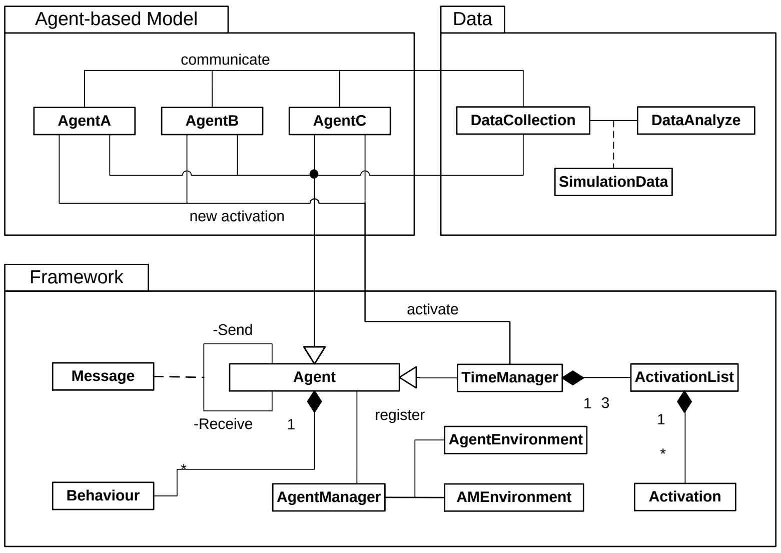


Figure 3.9: Static structure of the framework

3.5 Experiments

Queuing system $M / M^r / 1$ with a batch service is one of the classical discrete event systems. We use it to validate the proposed approach. An ABM of the system is built for the queue system, and the simulation result is compared with the theoretical value. We also compare the efficiency of the approach with the real-time ABS by using the built model.

3.5.1 Queuing System $M / M^r / 1$

The queuing system $M / M^r / 1$, shown in Figure 3.10, consists of an infinite population of customers, an infinite queue with FCFS (First Come First Serve) dispatching rule, and one batch server. The batch server provides service in batches (of size r) for arrived customers based on the rule. Customers who arrive and find the server busy join in the queue. Customers in a batch start service at the same time and depart together after served. The interarrival time and service time follow exponential distributions.

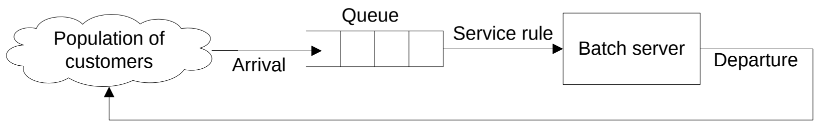


Figure 3.10: A Queuing system

3.5.2 Agent-based Model of the $M / M^r / 1$ Queuing System

Three types of agent are abstracted from the queuing system: customer source, customer, and server. Because a server is considered as an active entity, activation points of the whole system are simplified to two types: customer arrival and service completion. We build a single group ABM for the queuing system. The customer source generates customers according to the iterarrival time. The behaviors of the customers are requesting service, joining the queue to wait, accepting service, and leaving the system. The behaviors of the server include handling customer messages and providing service. The queue is part of the server agent. After a customer is served, the queue will use the given rule to choose new customers to begin the service. An interaction fragment among the three types of agents and the time manager is shown in Figure 3.11.

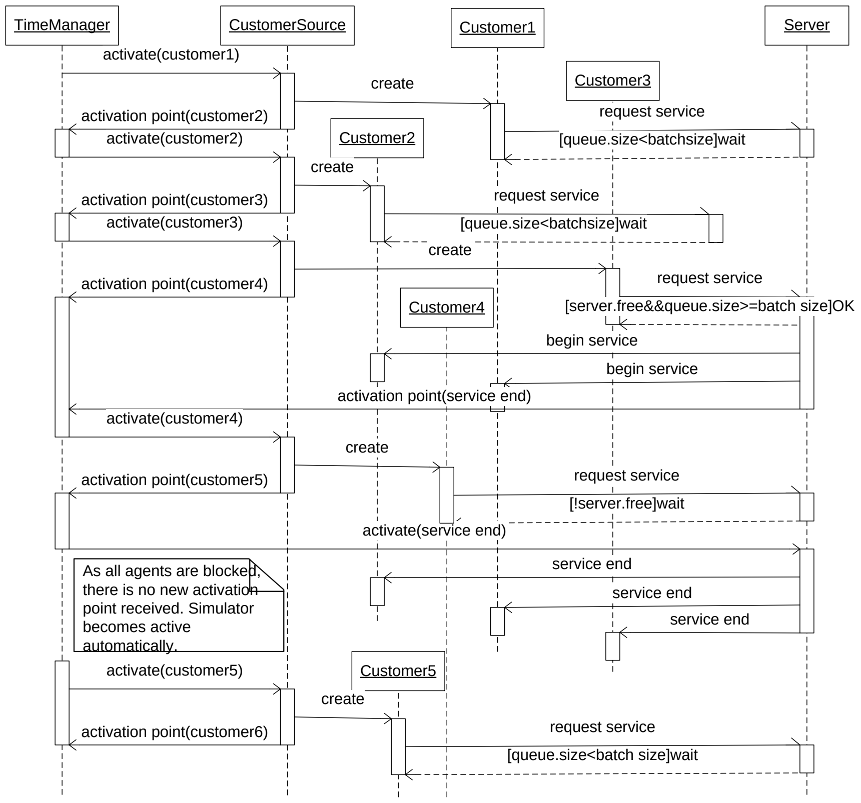

TimeManager
CustomerSource
Customer1
Server
activate(customer1)
create
Customer3
request service
activation point(customer2)
activate(customer2)
Customer2
[queue.size<batchsize]wait
create
activation point(customer3)
request service
activate(customer3)
[queue.size<batchsize]wait
create
activation point(customer4)
request service
Customer4
[server.free&&queue.size>=batch size]OK
begin service
begin service
activation point(service end)
activate(customer4)
create
activation point(customer5)
request service
[!server.free]wait
activate(service end)
service end
As all agents are blocked, there is no new activation point received. Simulator becomes active automatically.
service end
service end
Customer5
activate(customer5)
create
activation point(customer6)
request service
[queue.size<batch size]wait

Figure 3.11: Interaction fragment among agents of a $M / M^r / 1$ queuing system

3.5.3 Correctness Verification

Assuming that customers arrive one by one; arrival rate λ is 9.76 per hour; the service batch r is 3; service rate μ is 5 per hour. Measures of performance in the steady state can be calculated with the following formulas(Gelenbe and Pujolle 1987).

The probability that n customers are in the system,

$$p_n = \begin{cases} (1-s_0^{-n-1})/r & 0 \le n < r \\ \rho(s_0-1)s_0^{r-n-1} & r \le n \end{cases}$$

where $\rho = \lambda/(r\mu)$, s0 satisfies $|s_0| > 1$ and $r\rho s_0^{r+1} - (1+r\rho)s_0^r + 1 = 0$

The average number of customers in the system,

$$\overline{N} = (r-1)/2 + 1/(s_0-1)$$

The average number of customers in the queue,

$$\overline{N}' = (r-1)/2 + 1/(s_0-1) - r\rho$$

The average waiting time of customers,

$$\overline{W} = (r-1)/(2\lambda) + 1/(r\mu(s_0-1))$$

The average time of customers in the system,

$$\overline{T} = (r-1)/(2\lambda) + 1/(r\mu(s_0-1)) + 1/\mu$$

We assume that the system will be in the steady state after 1000 days, so the simulator runs 1000 days. Table 3.1 is the comparison between theoretical values and statistical values from simulation results. The comparison result shows that the simulation results match the theoretical results very well.

Table 3.1: Comparison between theoretical result and simulation result

Results	p_0	$p(n>0)$	$\overline{N}'$	$\overline{N}$	$\overline{W}$ (min)	$\overline{T}$ (min)
Theoretical	0.067	0.933	3.05	5.00	0.37	0.57
ABS&PIW	0.067	0.933	3.01	4.96	0.31	0.51

3.5.4 Comparison with the Real-time ABS (Time Scale)

In order to prove the less efficiency and precision, a brief simulator for the real-time ABS with timescale is developed by using JADE (Java Agent Development Environment). In JADE, the messages are not in sync sent and received (asynchronous communication). To achieve the simulation with the timescale, we improved it to the synchronous communication. After improvements, an agent (1) who just sends a message (a) will move on only after the message (a) is received and handled by its receiver agent (2). If the receiver (2) needs to send another message (b) in the message (a)'s handling process, the agent (1) has to wait for the receiver (2) until its message (b) is received and handled by another receiver. In addition, when a delay occurs in the agent, the agent will be blocked until the delay ends. The timescale is used in such delays to decrease the delay time so as to speed up the simulation.

We still use the queuing system but with the constant arrival rate (3 per hour) and service rate (3 per hour) to avoid the stochastic influences. The theoretical values can be got easily, shown in Table 3.2. The simulation runs ten days, and the simulation results from our approach and the real-time ABS are shown in Table 3.2 too.

Table 3.2: Comparisons with real-time ABS (time scale)

Results	p_0	$p(n>0)$	$\overline{N}'$	$\overline{N}$	$\overline{W}$ (min)	$\overline{T}$ (min)	Time spent (s)
Theoretical	0.000	1.000	1.00	4.00	20.0	80.0	-
ABS&PIW	0.000	1.000	1.00	4.00	20.0	80.0	0.374
ABS (scale 70000)	0.002	0.998	1.07	3.20	24.6	86.4	12.300
ABS (scale 500000)	0.041	0.959	1.05	1.85	147.5	300.4	1.743

We can see that the results of our approach are the same as the theoretical values. It takes only 0.374 seconds (the configuration of hardware is Intel i3-330M 2.13GHz

CPU and 2GB memory). However, for the ABS with the timescale, the error is very big, and it also took the longer time.

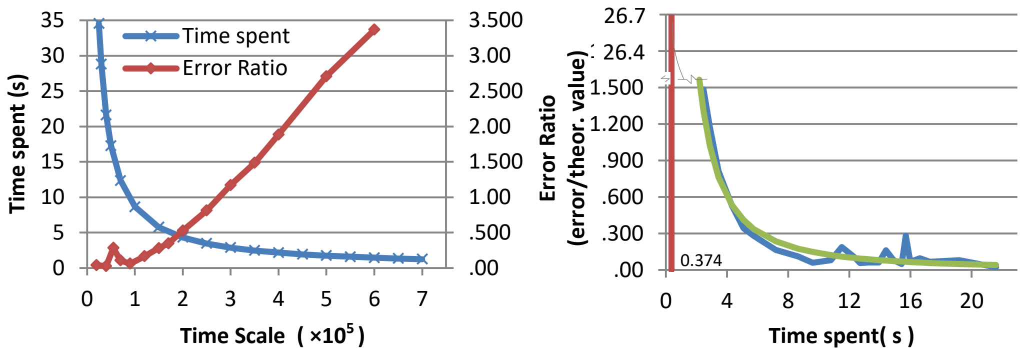


Figure 3.12: The relation between the scale and the error in the real-time ABS

An additional experiment is carried out to analyze the relationship between the time scale and the error. An error ratio $e.r. = \left|\overline{T}_{theo.} - \overline{T}_{sim}\right| / \overline{T}_{theo.}$ is used to denote the error from the real-time ABS. The results in Figure 3.12 show that: the more scale, the shorter time but, the greater error. If the simulation finishes in 0.374 seconds the same as the time spent in the ABS&PIW, the error ratio will be 26.7 which cannot be accepted obviously. Mostly, only the error ratio below 0.1 is acceptable. In this case, the time scale must be smaller than 70000. From Table 3.2, we can find that while the time scale is 70000, not only the error is greater, but also the time is longer than the ABS&PIW.

3.6 Summary

Because the PIW is more natural and closer to the mental model, it is combined with the ABS to speed up the simulation. The ABS&PIW approach is proposed on the basis of the agent-based model. We provide a four-tuple $SIM = (I, TM, ABM, O)$ with elements, including 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. A conservative algorithm is adopted to avoid straggler messages. According to the formula, the framework is developed by using multi-threading and synchronization technology. The result from an application to the queuing system $M/M^r/1$ shows the validity of the proposed approach. Comparing the efficiency with the real-time ABS, it performs more efficiently. Besides the advantages mentioned above, the ABM can be naturally combined with the process-interaction worldview. The flexibility, maintainability, and modifiability of the ABM are also enhanced in this way.

4 Agent-based Simulation of Material Flow in Job Shops

On the basis of the framework presented in the previous chapter, a simulator for the material flow is developed. Because communication functions and simulation controllers have been included in the framework, agent abstraction, behavior definition, cooperation design, and data analysis are the main tasks of developing the simulator. Thus, in this chapter we will answer the following questions: who are the agents in the material flow; what do they do; and how do they cooperate with one another. At last, the simulator will be used to solve a job release problem in a wafer FAB. This will show a traditional and most used way to solve the real-time decision-making problems by the offline simulation in the manufacturing. The method will also be compared with our other three methods in the following three chapters lately.

4.1 Analysis of Material Flow in Job Shops

4.1.1 Job Shop Analysis

A critical part of any manufacturing is the process flow (StudyMode.com 2013). The material flow is a visual expression of the process flow. The process flow consists of a series of steps which determine how a product is manufactured. The structure of the process flow determines how facilities will be laid out, how the working methods and the technology used, how the resources needed, and how efficient the process is. The process flow in job shops represents a general structure, which is characterized by manufacturing one or few quantity of products designed and produced as the specification of customers within prefixed time and cost. A job shop comprises general purpose machines and highly skilled operators arranged into different departments. Each job demands unique technical requirements and processing on machines in a certain sequence (Kumar and Kumar S. Anil 2009). Because of general purpose machines and facilities, a variety of products can be produced. However, this also leads to some limitations of job shop production: higher cost due to frequent setup changes; a higher level of inventory at all levels and hence higher inventory cost; and complicated production planning and material flow control.

4.1.2 Job Analysis

A job is some tasks that need to be done in order to produce one product. After product and process design, these tasks are usually grouped into several successive operations. Each operation finishes one type of material transformation. For a simple process, the operations usually make up a chain (the line type of the flow in Section 1.1.1). If the process is complex, the operations usually form a tree or a network. Because the chain and the tree are only two special cases of the network, in our study, we consider the operations of each job form a network (shown in Figure 4.1). Some raw materials and semi-products are input into an operation, and by using some resources, the input materials are transformed to some other types of materials. The output materials often include some semi-products and sometimes scrap. If the operation is the last one, the final product is also on the output list.

It is common that multiple resources are involved in one operation. Each resource has its function for the operation. So, the operation is made up of activities of all involved resources. These activities must meet some precedence constraints which are more flexible than the precedence constraints on the operations. For example, one activity can only start after a certain time of another activity' s starting, or several activities must start at the same time. The processing time of each activity can be either resource-dependent or resource-independent. Loading and unloading activities connect the materials with the operation. One operation may have more than one loading and unloading activity. In other words, the input materials are not loaded together, and the output materials are not produced at the same time either.

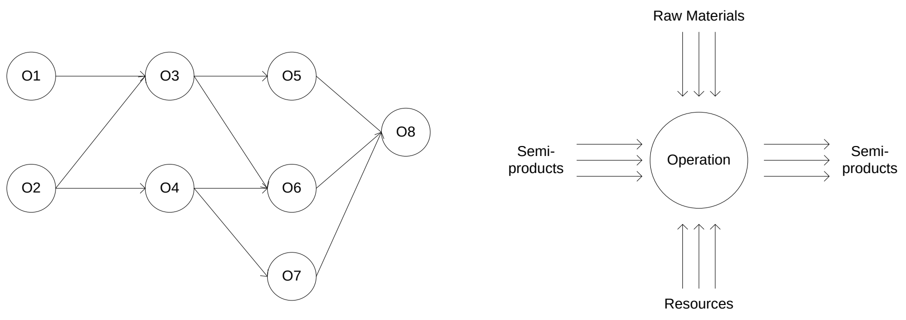
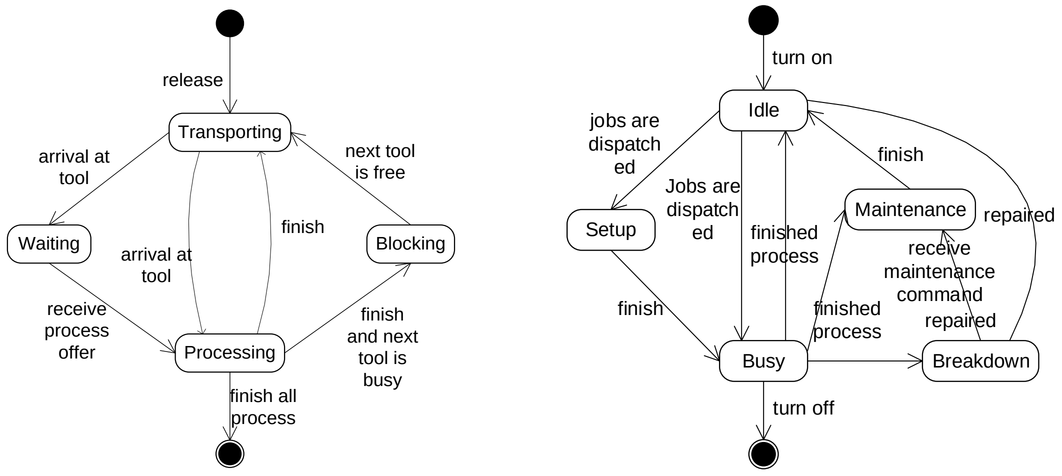


Figure 4.1: Operation and operation networks in a job

Once a job starts, the involved materials will be transported between two machines or between one machine and the related storage. The transportation will take a certain amount of time which is up to the distance, the transportation tool used, and so on. When the required materials arrive at the machine, they may wait before the machine if the machine is occupied or broken, or the materials have to wait for batching. The sequencing and batching procedures aim to handle the waiting time. The materials will be processed by several resources specified by the activities in the operations. The processing time may be deterministic or may be stochastic. After being processed, the output materials may be blocked. The blocked state means when the related operation has finished, but the job still stays on the machine for some reasons. Figure 4.2 (a) shows the static state chart of a job. In a real material flow, jobs have lots of configurations, such as job priority, time-critical operations, rework, reentrant and so on. Different jobs may have different priorities in the material flow. For example, some rush jobs may have very high priority. The time-critical operation is an operation that may have to be carried out in a certain time after its predecessor. A job that needs to be processed on the same machine several times is a reentrant job. If the semi-product or final product is of poor quality, the corresponding job may need to go through a rework loop.



(a) (b)

Figure 4.2: State charts of the job (a) and machine (b)

4.1.3 Resource Analysis

There are two main types of resources in the material flow: machines and workforces. Usually, on the shop floor, the workforces are sufficient. In our study, we consider only the machines. The machine usually needs setup before starting a job or a new type of job. There are three types of setup: sequence-dependent setup (related to both current jobs and previous jobs), sequence-independent setup (related to current jobs), and product-unrelated setup. In order to improve machines' reliability, preventive maintenances are carried out in the certain time interval. Some maintenance may not have an impact on processing jobs, and the machines can still process job during the maintenance; some may reduce the capacity of the machines; some may need the machines to stop work. In our study, we consider only the last situation, i.e., during the maintenances, the machines cannot process jobs. Because usually some routine inspections are taken during the maintenance, the time variances are not too big. We assume the maintenance time is deterministic. Even though the maintenances are carried out, the breakdown still cannot be avoided, and we need to repair the machines.

There are three situations during the repair: the machines can still work without any impact; the machines' capacity is reduced, and the machines cannot work anymore. The breakdown occurs totally stochastically. The repair time may be determined or may be stochastic. The main cause of uncertainty in the material flow is due to the unpredictable machine downtime. Figure 4.2 (b) is the state chart of a machine. In addition, a machine may be a single processing machine or a batch processing machine. The single processing machine can only process one job at a time while the batch processing machine can process more. All jobs in a batch start and finish processing at the same time. Machines with the same function, which are usually located in the same place in the shop, make up a machine group and can be treated as a whole in the material flow control. If machines in the same group are dedicated to different products, the machines group will be separated. If the machines are not in the same place, they will be treated separately too.

4.2 Agents in the Material Flow

4.2.1 Agents in the Material Flow

Agent-based modeling enables us to model the material flow more realistically and systematically comparing to other modeling methods, like discrete event simulation modeling, Petri nets, and so on. In chapter 3 we know that an agent-based model is composed of agents and their relationships. Thus we should determine who the agents are in the material flow first. Obviously, material plays a leading role in the material flow. 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 to the material flow control. We usually focus on jobs which organize all materials that one product needs in a serial of operations. Besides, machines and transporters also make up a large proportion of the material flow. Because we assume that the transportation capacity is infinite, the transporters are out of scope. The transportation in our study is only 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 them. 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 material flow, this valve is a job release procedure. The release procedure is what we will concentrate rather than the job pools because the method of forming the job pools is out of the scope of the material flow control. On the basis of analysis above, we list three types of entities concerned in the material flow: the jobs, machine groups, and release procedures. We will model these three entities as agents in the agent-based model: job agent, machine agent, and release agent. In this section, we will describe how to create them on the basis of the framework, including defining their attributes and behaviors, designing communications and cooperation, and clarifying the simulation-related delays and activation.

4.2.2 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 based on a release policy which is given in the product data. The buffer is located behind the release agent (see Figure 4.3). If the corresponding buffer is full in the first operations, the released job cannot start the operatons. It will be blocked and stay in the release buffer until the buffers 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.

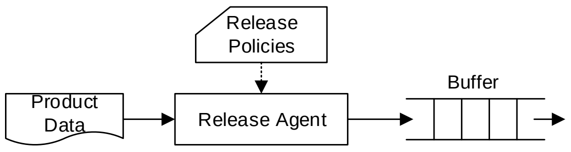


Figure 4.3 the release agent with one buffer

4.2.2.1 Product Data in the Release Agent

At the beginning of the simulation, the simulator reads the 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 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.

4.2.2.2 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 target WIP level is given in advance. If the actual WIP level is less than the target WIP level, a new job is released. One interval time is still needed to release jobs before the shop reaches the target WIP. In the AS policy, a target buffer size is set for a bottleneck machine group. When the buffer size of the bottleneck is bigger than the target value, the releasing stops. When the buffer size is less than the target, new jobs are released. The quantity of the jobs which will be released is the difference between the buffer size and the target value.

4.2.2.3 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 being accepted, the job sends unblocked message to the release agent. The job will be removed from the buffer and moved to the first operation.

4.2.3 Job Agent

The job agent is a temporary entity. After being 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.

4.2.3.1 Behaviors and Lifecycle 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 4.4. 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.

4.2.3.2 Delays and Activations in the Job Agent

There are four types of delays related to the logical states: 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.

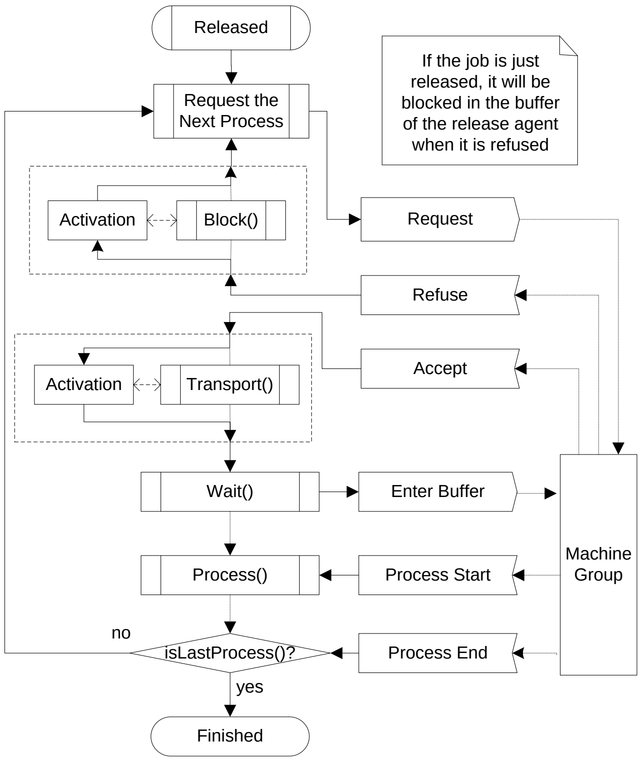


Figure 4.4 Lifecycle of a job agent

4.2.4 Machine Group Agent

The machine group agent is a permanent and active entity. It includes one buffer and several machines (see Figure 4.5). The machines have the same function and share the buffer.

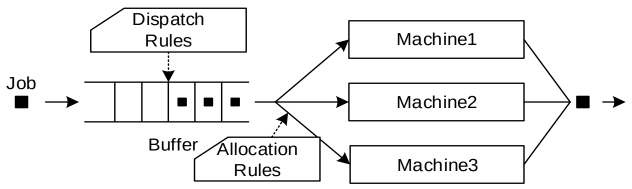


Figure 4.5 Structure of the machine group

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 job will either be transported to the next machine group or be blocked on the current machine.

4.2.4.1 Behaviors of Machine Group Agent

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

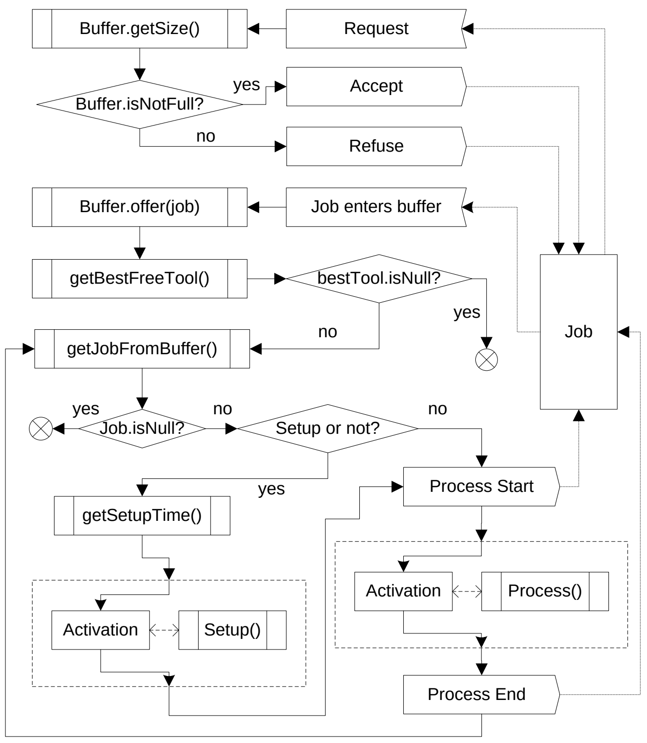


Figure 4.6 Behavior flow of a machine group agent

When a job enters a buffer, or a machine just finishes one job, the machine group will start 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 4.6 shows the behavior flow of the machine group agents.

4.2.4.2 Dispatch and Allocation Rules in Machine Group Agent

A dispatch rule is a criterion for determining the jobs' priorities 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. 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 (Earlies 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.

4.2.4.3 Delays and Activations in Machine Group Agent

There are four types of delays related to the logical states of 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 the 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.

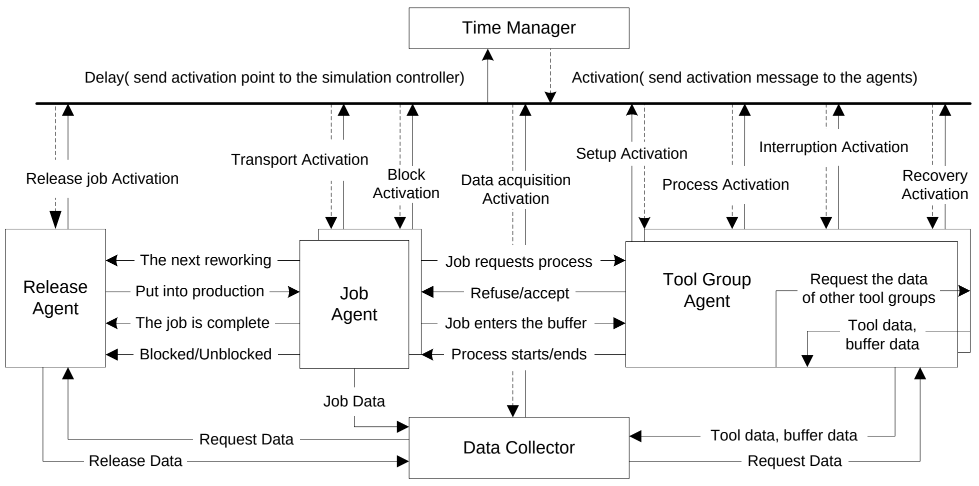


Figure 4.7 communications among the agents and the time manager

4.2.5 Communication among the Agents

We summarize the communication among the agents in Figure 4.7. It also includes the delays and activations conveying between the time manager and the agents.

4.3 Agent-based Simulation of the Material Flow

4.3.1 Overview of the Simulation

The agent-based simulation of the material flow consists of a simulator, data about the material flow, and performance measures (see Figure 4.8). The Data about the material flow are inputs, and the performance measures are outputs.

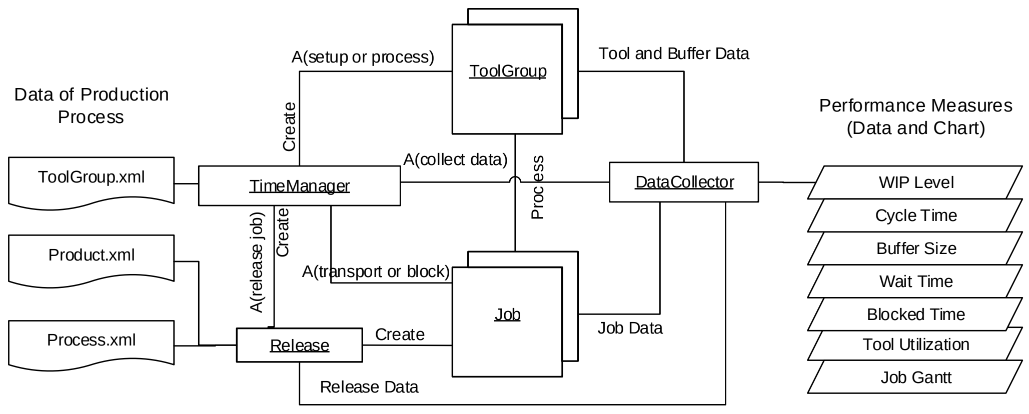


Figure 4.8 Agent-based simulation of the material flow

4.3.2 Data of the Material Flow

We classify the data of the material flow into three parts: machine group data, product data, and process data, and then adopt three XML files 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 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.

4.3.3 Simulator for the Material Flow

The agent-based model (including release agents, machine group agents, and job agents), data collector and time manager make up the simulator for the material flow.

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.

4.3.4 Performance Measures

The performance measures include WIP level, cycle time, buffer size, waiting time (by machine group), blocking 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 material flow and can also be provided for the optimization or control algorithms to achieve the goals.

4.3.5 Static Structure of the Simulator

The static structure of the simulator (see Figure 4.9) has three layers: framework layer, agent layer, and production characteristics layer.

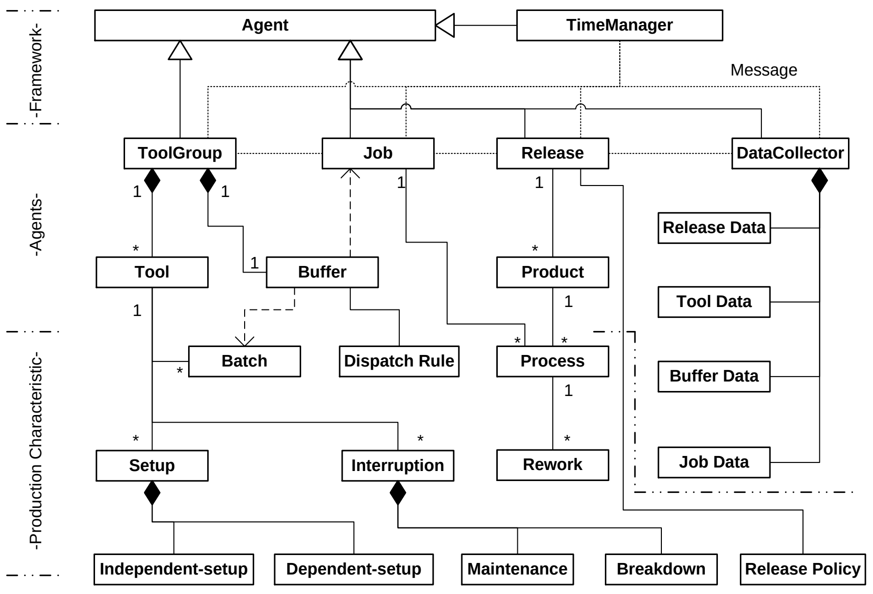


Figure 4.9 Static structure of simulator

The framework layer provides the agent base class and the time manager. The agent layer contains the agents which are abstracted from the material flow, and these agents extend the base class of the agent in the framework. The data collector, which is also an agent, is in charge of data collection from machines, buffers, jobs, and release agents.

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.

4.4 Experiments

As mentioned in Chapter 2, developing decision rules and using simulation to evaluate and improve the rules are the general way to solve the job release, routing, and sequencing problems. The method is carried out simply by designing and running some scenarios with different rules or different parameters of the rules. At last the rule or the parameters are determined by the performance computed from the simulation results. Since the simulation model has been created, in this section, we will use an example to show this general way to solve a job release problem. The routing and sequencing problems can be solved in a very similar way. The method will be compared with our other approaches in the following three chapters.

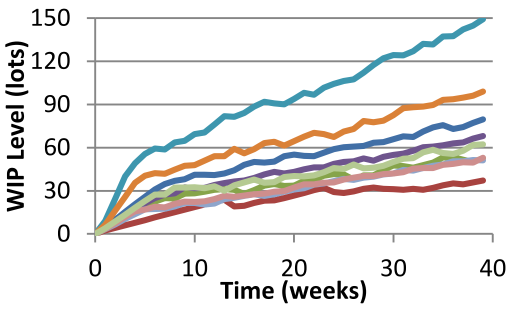
4.4.1 Problem Description

Wafer FAB has complicated material flows and includes all characteristics mentioned above. The developed simulator is applied to a wafer FAB Mimac6 (Measurement and Improvement of Manufacturing Capacity (Fowler and Robinson 1995)) with some random modifications. The Mimac6 has 93 machine groups (46 machine groups are batch processing machines; six machine groups need set up before processing jobs) and produces nine products with nine process flows and nine different throughputs. Average operation number of the processing flows is nearly 300. Distribution of the interval between two interrupts and distribution of the recovery time are specified for each machine group.

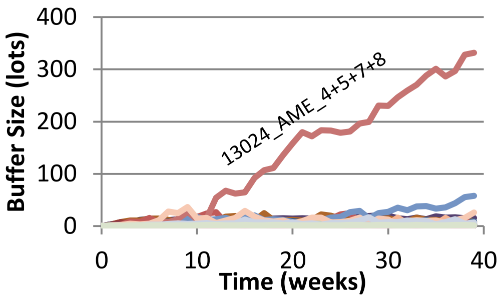
We evaluate the material flows using the release policy CONINT and dispatch rule FIFO. The interarrival time of each product are calculated on the basis of their target throughputs, shown in Table 4.1. The simulator runs 40 weeks. The results are shown in Figure 4.10. We can find that the FAB is totally unstable. From (a), we can see that the WIP levels of all products are increasing. The reason can be found from (b) according to the buffer size of the machine groups: there is a bottleneck (13024˙AME˙4+5+7+8) in this case. After the simulation starts for a while, the utilization ratio of the bottleneck goes up to 100% shown in (c).

Table 4.1: Product target throughputs and release intervals

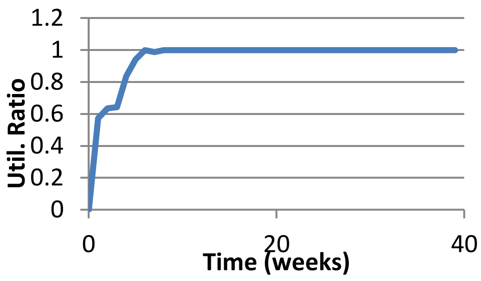
Product	B5C	B6HF	C4PH	C5F	C5P	C5PA	C6N3	C6N2	OX2
Target (lots/year)	287	94	199	241	802	508	184	213	248
Interval (hour)	30	93	44	36	11	17	48	41	35



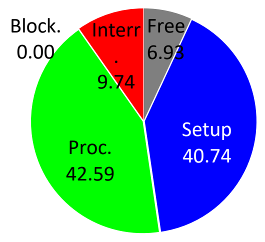
(a)



(b)



(c)



(d)

Figure 4.10: Problem description (a) WIP levels of 9 products (b) Buffer sizes of machine groups (c) Utilization ratio of the bottleneck (d) Utilization ratio of the bottleneck in detail

The dispatch rules are incapable of improving the material flows in this case. What we will do is to find a better release policy and related parameters by means of the simulation. The goal is to maximize the utilization of the bottleneck as well as to keep WIP at a lower level. In the following sections, we will evaluate the three release policies which are preset in the simulator, and try to find optimal parameters for each policy. We also choose the best parameter for each policy according to the simulation results. At last, we compare the three policies with the chosen parameters and decide the best one.

In addition, we need to mention that the maximal WIP level is 665 lots. Plus, the 93 machine group agents, total 758 agents are running in parallel. The simulator works well and spends 1.3 minutes in this case (CPU: AMD Phenom™ II 2.6GHz, Memory: 8.0GB). In one of the following case (see Figure 4.13), the maximal WIP level is 2715 lots, and the simulator with 2808 agents runs 1.8 minutes and works well too. So the efficiency of the simulator is acceptable for offline analysis of the material flow.

4.4.2 Simulation with Three Release Policies

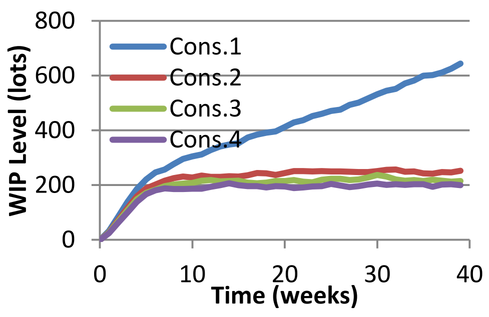
4.4.2.1 Constant Time Interval

The easy way to decrease and stabilize the WIP level is to increase the release time intervals. We carry out four cases with different time intervals to determine the best intervals. In Table 4.2, the intervals are specified in a scale form related to the intervals in Table 4.1. The intervals increase proportionally.

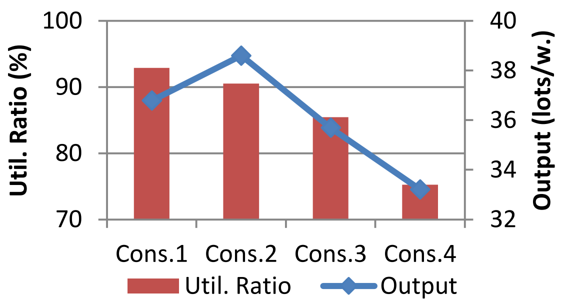
Table 4.2 Cases of simulation with different release intervals

Case	Cons.1	Cons.2	Cons.3	Cons.4
Time interval (increases by)	0%	10%	20%	30%

Figure 4.11 shows the simulation results. The total WIP level decreases as the intervals increase (from a). The utilization ratio goes down when the intervals increase. The throughput goes up when the intervals increase by 10% and then decreases (from b). Even though the case Cons.1 has the highest utilization ratio of the bottleneck, but the total WIP level is the highest one, and the throughput is lower than the case Cons.2. So the case Cons.2 is the best one.



(a)



(b)

Figure 4.11 Comparison of the cases with different release intervals (a) Total WIP level of the products; (b) Utilization ratio of the bottleneck and total throughput

4.4.2.2 Constant WIP Level

The parameters of the CONWIP policy are the target WIP levels. From the case Cons.2, we can get initial target WIP levels of the products for the CONWIP policy shown in Table 4.3. On the basis of the initial target WIP levels, six cases are designed shown in Table 4.4. The target WIP levels increase or decrease according to the given scales.

Table 4.3 Initial target WIP levels from the case Cons.2

Product	B5C	B6HF	C4PH	C5F	C5P	C5PA	C6N3	C6N2	OX2
Target WIP (lots)	28	14	22	21	47	34	15	16	23

Table 4.4 Cases of simulation with different target WIP level

Case	C. WIP1	C. WIP2	C. WIP3	C. WIP4	C. WIP5	C. WIP6
Target WIP Scale	×0.9	×1.0	×1.1	×1.2	×1.3	×1.4

Figure 4.12 is the simulation results. The total WIP levels become steady at the target levels after a while when the simulation starts. The utilization ratio of the bottleneck is increasing while the target WIP levels increase. The throughput is increasing too, but in case C. WIP6 the throughput declines. That is because a new bottleneck appears in this case. So we choose the case C. WIP5.

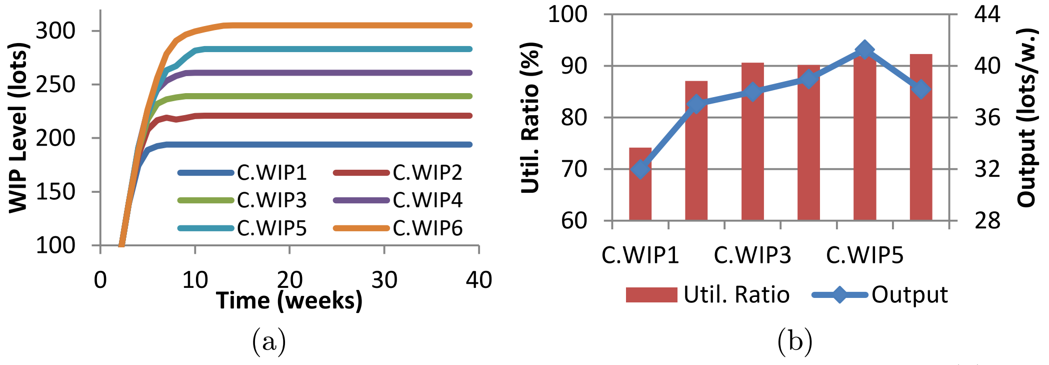
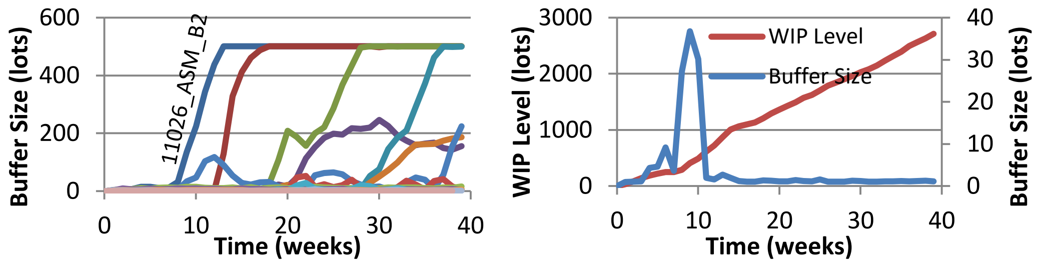


Figure 4.12 Comparisons of the cases with different target WIP levels (a) Total WIP level of the products (b) Utilization ratio of the bottleneck and total throughput

4.4.2.3 Avoiding Starvation

Firstly, we set the machine group 13024˙AME˙4+5+7+8 as the first level bottleneck with the initial target buffer size, i.e., 35 lots, which is getting from the case C.WIP5. The results are shown in Figure 4.13. Even though the buffer size of the bottleneck decreases, many new bottlenecks appear and the total WIP level increases more. So we set the first appeared new bottleneck 11026˙ASM˙B2 as the second level bottleneck with the initial target buffer size, i.e., ten lots, which is obtained from the case C.WIP5 too. The simulation results with these two levels AS policy show that no any other new bottlenecks appear. So two is the final level number of the AS policy.

Six cases with different target buffer sizes (see Table 4.5) are carried out. The simulation results show in Figure 4.14. As the target buffer size of the first level bottleneck increases, the total WIP level is increasing, and the utilization ratio of the first level bottleneck begins with the increase and then declines. As the target buffer size of the second level bottleneck increases, the total WIP level, the throughput, and the utilization ratio of the first level bottleneck varies irregularly. Even though the case A.S.4 gets the highest utilization ratio of the first level bottleneck, the throughput is lower than the case A.S.3 which achieves the second highest utilization. So the case A.S.3 is the best one.

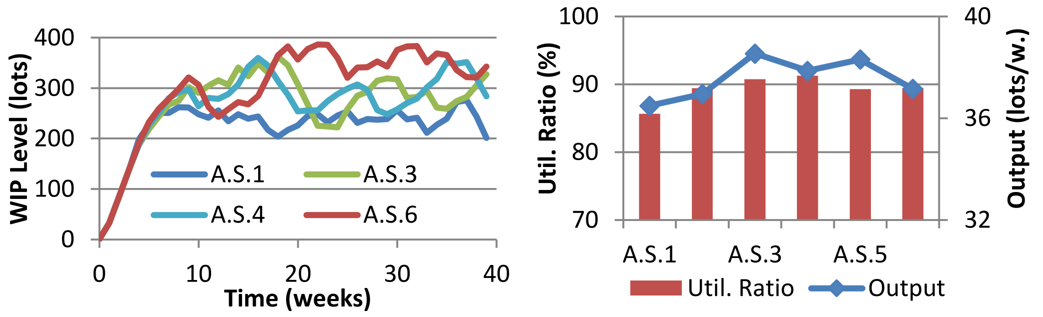


(a) (b)

Figure 4.13 Simulation results using the avoiding starvation policy with one bottleneck (a) Buffer size of machine groups (b) Buffer size of the bottleneck (13024˙AME˙4+5+7+8) and total WIP level

Table 4.5 Cases of simulation with different target buffer size at the bottlenecks

Case	A.S.1	A.S.2	A.S.3	A.S.4	A.S.5	A.S.6
1st Level (3024) /lots	5	25	35	35	35	45
2nd Level (11026) /lots	10	10	10	15	20	10



(a) (b)

Figure 4.14 Comparisons of the cases with different target buffer sizes (a) Total WIP level of the products; (b) Utilization ratio of the first level bottleneck and total throughput

4.4.2.4 Comparison of Three Release Policies

In this section, we compare three policies above with the chosen parameters. The results are in Figure 4.15. The total WIP level in the case C.WIP5 is the steadiest one. The case C.WIP5 also obtains the highest utilization ratio of the bottleneck and the largest throughput. The case Cons.2 has the lowest total WIP level, the shortest cycle time (of the product C5PA), and the smallest buffer size of the bottleneck (13024˙AME˙4+5+7+8), and its total WIP level is steadier than the case A.S.3. But the utilization ratio of the bottleneck and the throughput is lower than the case C.WIP5. The case A.S. performs worst, in which the total WIP level, the cycle time, and the buffer size vary a lot and the throughput is the lowest. So we choose the case C.WIP5 as the best release policy for the Mimac6.

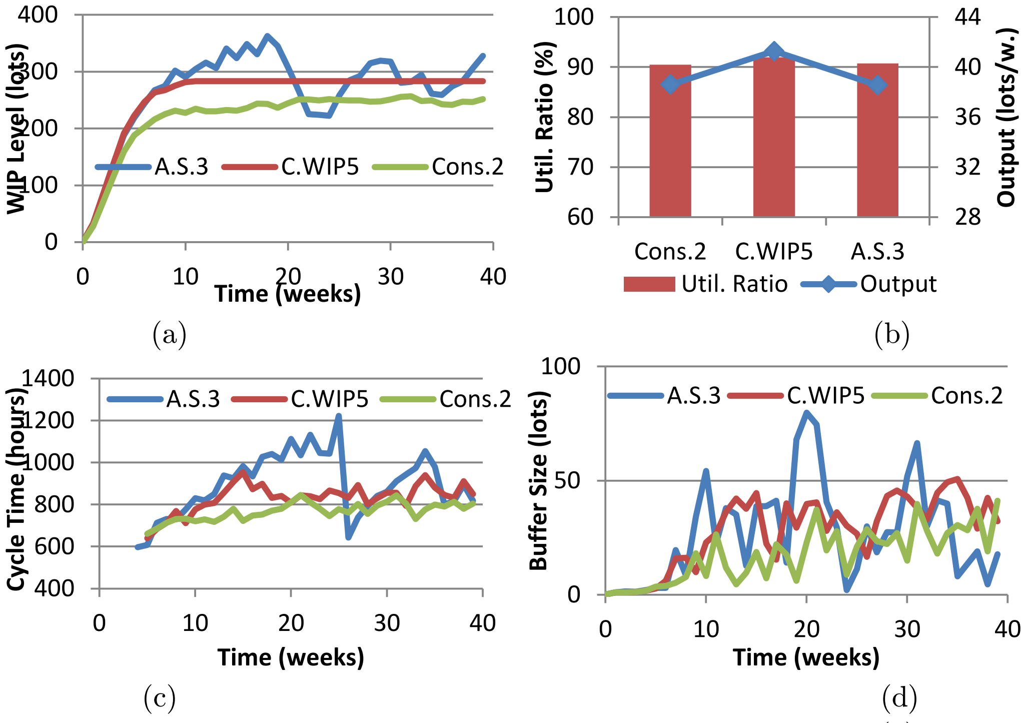


Figure 4.15 Comparisons of three best cases from three case groups (a) Total WIP level of the products (b) Utilization ratio of the bottleneck and total throughput (c) Cycle time of the product C5PA (d) Buffer size of the bottleneck (13024˙AME˙4+5+7+8)

4.5 Summary

On the basis of the framework, an agent-based model of the material flow including release agents, machine group agents, and job agents is built. The behaviors and interactions among the agents are explicit defined. A large variety of dispatching and allocation rules and release policies are preset in the model as well. The agent-based model, the data collector, and the time manager make up the simulator for the material flow. Information of the material flow is stored in XML files. Applying the simulator to a wafer FAB model, we found the best release policy for the FAB according to the simulation results. The application demonstrates the general way to solve the job release problem by using the simulation. This will make a strong contrast with our other approaches lately. In the coming chapters, we will give the agents abilities to learn how to make decisions on the release, routing, and sequencing.

5 Simulation Try-then-decide Method for Release, Routing, and Sequencing

As we mentioned before, the material flow control is a decision-making process. The job release decides if a job will be released while the workload changed. The alternatives are yes and no. The sequencing decides which job will be processed first. The alternatives are jobs in the buffer. The routing decides which machine the job will go to. The available machines are alternatives. These three types of decisions are made throughout the manufacturing. The decisions make up sequential decisions in which decisions that made now have both immediate and long-term effects, and determine the later decision-making. The later decisions influence the performance of the earlier decisions. In this chapter, a simulation try-then-decide method (STTD) is introduced to solve such sequential decision-making problems. The method will be compared with the decision rules studied in the previous chapter.

5.1 Sequential Decision-making in Release, Routing, and Sequencing

5.1.1 Sequential Decision-making(Littman 1996)

A sequential decision-making system includes a decision maker and its environment that the decision maker interacts with. The thing the decision maker interacts with, comprising everything outside the decision maker, is called the environment. The problem addressed is, given a complete and correct model of the environmental dynamics and a goal structure, to find an optimal way to behave (to make decisions). The behavior maps the state of the environment to the action choice. The action, i.e., the decision-making process, is usually turned to a selection of the best alternative from possible given alternatives. The decision maker observes the environment and carries out the proper action according to the state of the environment. The action results in some changes in the environment. Thus the decision maker will take another proper action again to adapt to the new state. The process is shown in Figure 5.1. The behavior usually follows a policy which tells the decision maker how to act according to the state of the environment. The policy may be a plan which specifies an action for each possible state. The policy can also be an equation in which the action is a function of the state.

The task of the sequential decision-making is to produce an optimal policy which can maximize a long-term performance.

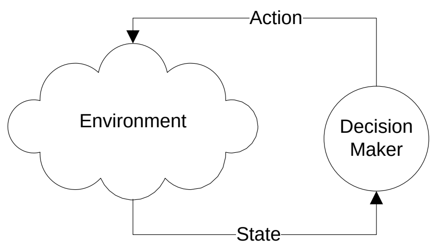


Figure 5.1: A decision maker interacts with its environment

5.1.2 Sequential decision-making in Release, Routing, and Sequencing

For our problem, the environment is the material flow. The decision maker is a scheduler who makes the release, routing, and sequencing decisions during the manufacturing according to the state of the material flow. The decisions will be executed by the operators. The decisions are made only at some discrete events. We call the time that the events occur as decision points. The release decisions are made while the workload of the related product changed, i.e., a job for producing the product is finished on a machine. The sequencing decisions are made while one of the following conditions is met: 1) while a machine becomes idle and jobs are waiting for it; 2) while a job arrives at a machine and the machine is empty; 3) while a machine has been repaired or maintained. The routing decisions are made 1) after a job is finished on a machine; 2) after a job is released.

We aim to develop a plugin for MES to help or replace the real scheduler. To develop the plugin, firstly a model of the material flow is necessary. Chapter 4 exactly does what we need now. The agent-based simulation model can be used directly without any changes here. Secondly, a decision maker should be included and makes decisions by utilizing the model of the material flow.

The decision-making in the release, routing, and sequencing problems is a selection process which selects the best alternative from all possible alternatives. On the other hand, we can treat the problems as priority calculation problems. For each alternative, we calculate a priority. The alternative with the highest priority will be selected. The scheme is shown in Figure 5.2. The priority calculation may utilize the model of the material flow directly or use it indirectly. In an indirect way, the model is used to build a new model to calculate the priority value. In our study, we use three methods to calculate the priority values for the alternatives. In this chapter, we introduce the first method, i.e., STTD which uses the model of material flow directly. The other two methods in Chapter 6 and Chapter 7 will use the model indirectly.

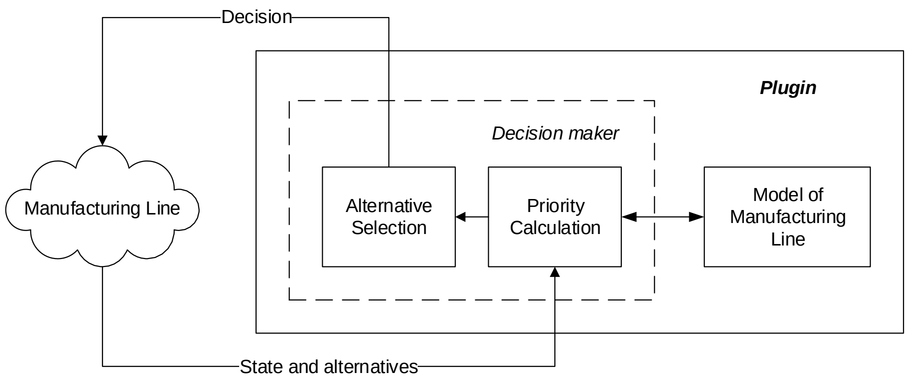


Figure 5.2: Sequential decision-making in release, routing, and sequencing

5.1.3 Decentralized Sequential Decision-making

In our problem, there are three types of decision-making. If we use only one decision maker to make all decisions, the decision maker will be too complicated. We divide the task of the decision-making into some small tasks. For each type of product, a release decision maker is created. The release decision makers make the release decisions for the related product while the workload is changed. For each machine group, one sequencing decision maker is created. The sequencing decision makers make the sequencing decision for the related machine while a job arrived or the machine becomes idle. For each job, a routing decision maker is built. The routing decision makers are responsible for making the routing decision after the related job is finished on one machine. Once a decision needs to be made in the material flow, the corresponding decision maker will start work. All decision makers have the same right to utilize the model of the material flow.

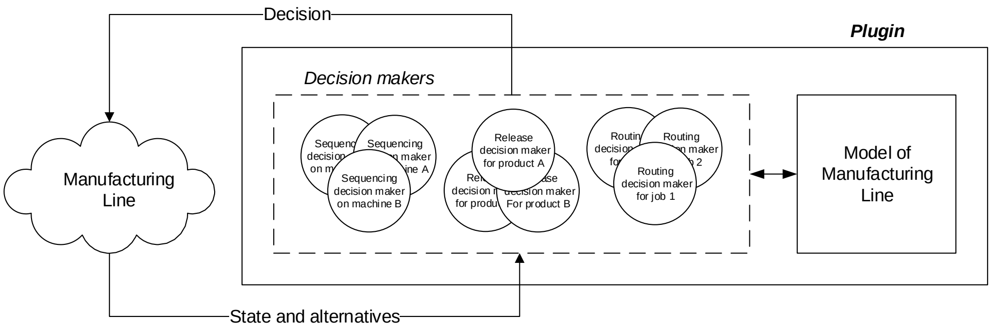

Decision
Plugin
Decision makers
Manufacturing Line
Sequencing decision maker on machine A
Sequencing decision maker on machine B
Release decision maker for product A
Release decision maker For product B
Routing decision maker for job 2
Routing decision maker for job 1
Model of Manufacturing Line
State and alternatives

Figure 5.3: Decentralized sequential decision-making

The decentralized sequential decision-making has several advantages. Firstly, it is flexible. For example, if we want to add another type of decision-making into the plugin, we do not need to change too much on the origin plugin. Moreover, if the manufacturing line changed, e.g., adding one machine, it is also easy to add a decision maker. Secondly, the decision-making problems can be simplified. For example, in the sequencing problem, if we create each machine a decision maker, the position of the machine in the layout of the manufacturing line can be ignored while we calculate the priority values. Thirdly, it can facilitate the evaluation of our approaches. Because we will use the agent-based simulation to evaluate the approaches, each decision maker exactly connects to one agent in the agent-based model. We just need put the decision makers into the related agents and run the simulation. Figure 5.3 shows the scheme of the decentralized sequential decision-making.

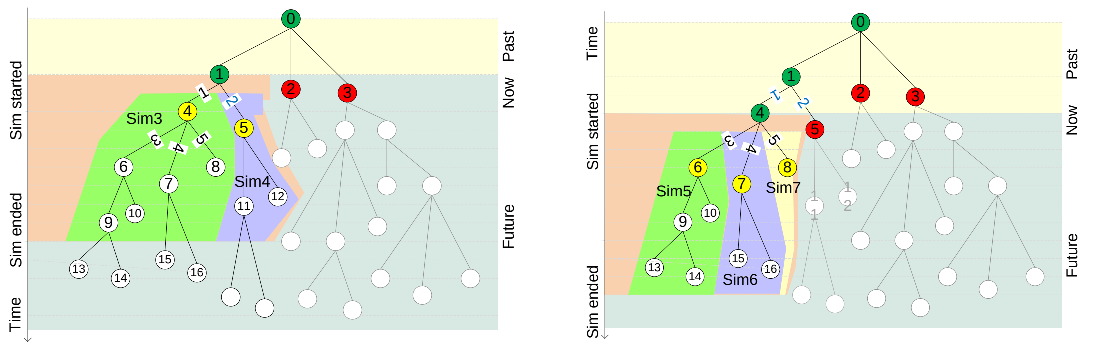
5.2 Simulation Try-then-decide Method

The biggest problem with the decision-making is that we do not know what will happen in future after the decision is executed, due to the complexity and randomness of the system. The simulation try-then-decide method uses the simulation to predict the future after one decision is carried out. It uses the predicted future information to make decisions. The method can be stated as follows: once we need to make a decision, the simulation will evaluate each alternative; for each alternative, the simulation runs once (determined model) or several times (stochastic model); the selection of an alternative is dependent on the evaluation of the simulation results.

5.2.1 Branching Tree with a Time Axis

First, we present the method in a branching tree with a time axis, shown in Figure 5.4(a). A node denotes a decision point. A branch represents a selection of one alternative. The corresponding value of nodes on the time axis is the time when the decisions need to be made. A path from the root node to the last node is related to an active schedule. As time goes on, the tree will become bigger and bigger because of the exponential explosion. For example, ten decisions need to be made one by one, and each decision has two alternatives. The number of the paths will be 2^{10}. In the real case, the decisions are even huge and not fixed. Thus, it is impossible to enumerate all paths and evaluate them. Our method only focuses on the partial paths starting from the decision point and ending at a given time.

We use the sequencing problem to give a general idea of our approach. In Figure 5.4(a) we assume that current time is at decision point 1 and we need to select one job from two. Thus, the simulation sim3 and sim4 will start respectively from the decision point and end at a given time. The sim3 will simulate the future situation while we select job 1. The sim4 deals with the selection of job 2. The results of the sim3 and sim4 will be used to evaluate the selection of job 1 and job 2. Moreover, we can see that in both sim3 and sim4, there are also lots of decision points. For these decision points, we specify a **base-rule** to select jobs. We assume that the selection of job 1 performs best, so we select job 1. When time advances to the decision point 4, shown in Figure 5.4(b), we have to select one job from three again. Thus, the simulation sim5, sim6, and sim7 will start. The rest work will be done in the same manner. Note that the tree is only for demonstrations here and it cannot be determined in advance due to uncertainties in the system.



(a)

(b)

Figure 5.4: A branching tree with a time axis (a) before the time advances (a) after the time advances

5.2.2 Four-tuple of STTD Scheme

Now we give a more general description of our approach in a four-tuple *STTD= (S, A, Sim, V). S* is the state of the environment at a decision point. *A* is a set of alternatives. *Sim* is the simulation. We call the simulation as alternative simulation. *V* is the evaluation. Before we start the simulations, the simulation model is initialized with the environment at a decision point. For a selection of each alternative, the simulation runs once or several times. The alternative will be selected according to the evaluation of the simulation results. In addition, scheduled events, e.g., starting maintenance on one machine, are also concerned. These events will be directly put on the event list of the alternative simulation and will occur during the simulation run.

For the alternative simulation and the evaluation, there are four key questions: 1) how many times do the alternative simulations run; 2) how long does each alternative simulation run; 3) how to select the base-rule; 4) How to evaluate the selection according to the simulation results.

For question 1), usually, if the simulation model is deterministic, the simulations just run once for each alternative. If the model is stochastic, the simulations need to run many times. The more times the simulations run, the better results we obtain. However, because the machines/jobs are waiting when we make the decision, we should make our decision as soon as possible in order to maximize the utilization or minimize the cycle time. The number of times can be calculated according to the time allowance in the real system.

For question 2), theoretically, the simulation ends when the effect of current decision disappears. But it is hard to determine this time. Usually, the decision influences only the jobs which are completed in a short period after the decision is executed. Thus, the period of simulation is decided by the number of completed jobs n after the decision point, $n = \kappa WIP$, where κ is a factor which denotes the length of the period. WIP is the work in process level at the decision point.

The selection of the base-rule will be analyzed in Section 5.3.2. In sequencing problem, the base-rule can be First In First Out (FIFO), Shortest Processing Time (SPT), Longest Processing Time (LPT), and so on. For the routing problem, the base-rue can be the Shortest Queue Length (SQL), Shortest Queue Time (SQT), and so on. For the release problem, the base-rule can be CONINT, CONWIP, and so on.

The evaluation is related to the objective. Here, we give two formulas under two common objectives for the sequencing and routing problems. While the objective is to minimize the cycle time,

$$v = 1 / \sum_{p \in P} (\omega_p \sum_{j \in J_p} c_j / n_{J_p}), \text{ (Eq. 5.1)}$$

where p denotes a product; P is the set of products. ω_p is the weight of product p. j is a job. J_p is the set of completed jobs whose product type is p. c_j is job j' s cycle time. n_{J_p} is the number of jobs in the set J_p.

While the objective is to minimize total weighted tardiness

$$v = 1 / \sum_{p \in P} (\omega_p \sum_{j \in J_p} \max(C_j - d_j, 0) / n_{J_p}), \text{ (Eq. 5.2)}$$

where C_j is job j' s completion time, and d_j is job j' s due date. If the simulation runs many times, the average value of the priorities is adopted.

For the release problem, we have to achieve not only the objective of scheduling but also the objective of the throughput. The priority can be calculated by,

$$v = \alpha \sum_{p \in P} (O_p^T / | O_p - O_p^T |) + \beta v_s, \text{ (Eq. 5.3)}$$

where O_p^T is the target throughput of product p; O_p is the throughput of product p during the simulation; v_s is the normalized priority value considering only the scheduling objective; α, β denote the importance of the meeting throughput and achieving objectives in the priority value.

5.2.3 Alternative Simulation

The agent-based simulation of the material flow proposed in chapter 4 will be used to evaluate each alternative. As we mentioned, a job shop model is stored in XML files.

Once we start to evaluate one alternative, firstly, we create a model from the files and initialize the model with the current environment information. Then we load the model to the simulator. An activation point (event) with the current time stamp is put on the activation list of the time manager. The activation point contains the information of the alternative and the agent who needs to make the decision. The simulation exactly starts at the activation point. Meanwhile, the time manager activates the agent and the agent takes the alternative. Another conditional activation point with some termination conditions is also put into the activation list together at the beginning. During the simulation, the conditions are checked whenever the clock advances. If the conditions are met, the simulation stops. The priority of the alternative is computed from the objective functions in the previous section based on the simulation results. Some example codes showing this procedure can be found in appendix A.2.

5.2.4 Environment Simulation

Before each alternative simulation run starts, the simulation model should be initialized according to the environment information of the real system at decision points. However, it is impossible and impractical to connect to the real system and read the environment information for now. Therefore, in order to evaluate the STTD method, we replace the real system with a simulation model shown in Figure 5.6.

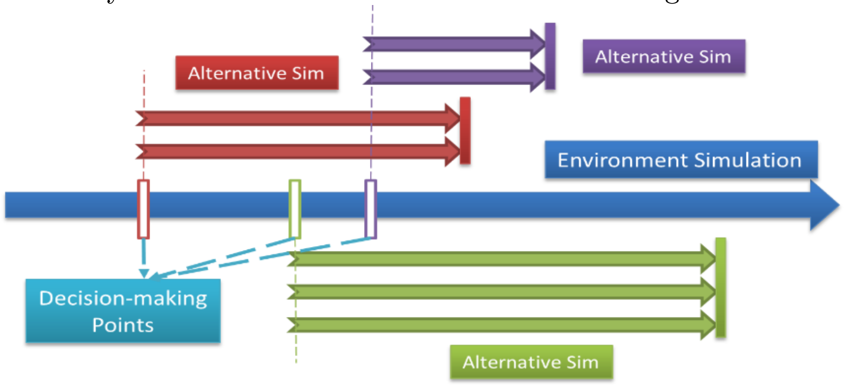


Figure 5.5: Environment simulation and alternative simulations

The simulation is always running like a real system. We call the simulation as environment simulation. In this case, the state of the environment is actually the agent environment, i.e., all agents' state. Once a decision needs to be made, the environment simulation will pause. The decision-making procedure will start. The alternative simulation is initialized according to the current state of the environment simulation. When the decision-making is done, the environment simulation will take the decision and continue (shown in Figure 5.5). The performance of the STTD method can be calculated according to the results of the environment simulation.

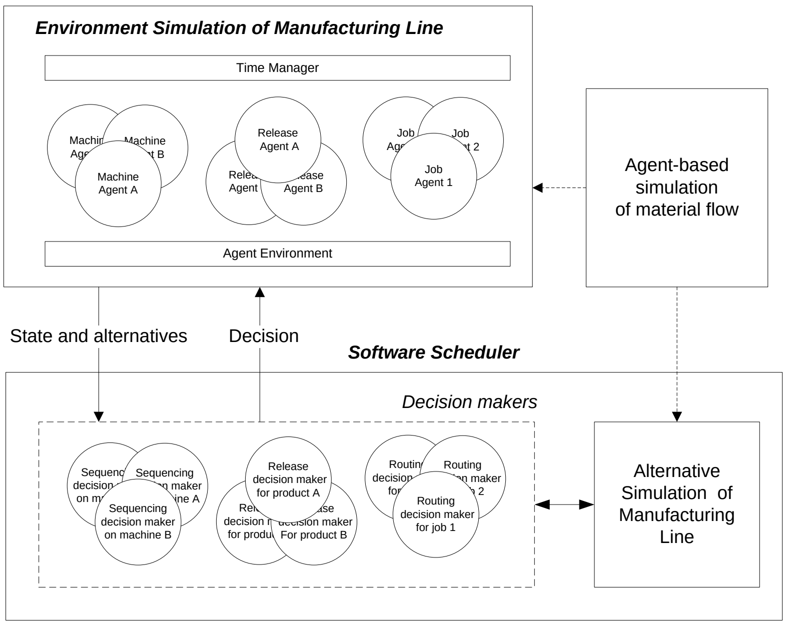

Environment Simulation of Manufacturing Line
Time Manager
Machine Agent B
Machine Agent A
Release Agent A
Release Agent B
Job Agent 2
Job Agent 1
Agent Environment
Agent-based simulation of material flow
State and alternatives
Decision
Software Scheduler
Decision makers
Sequencing decision maker on machine A
Sequencing decision maker on machine B
Release decision maker for product A
Release decision maker For product B
Routing decision maker for job 2
Routing decision maker for job 1
Alternative Simulation of Manufacturing Line

Figure 5.6: Replace the manufacturing system with the simulation

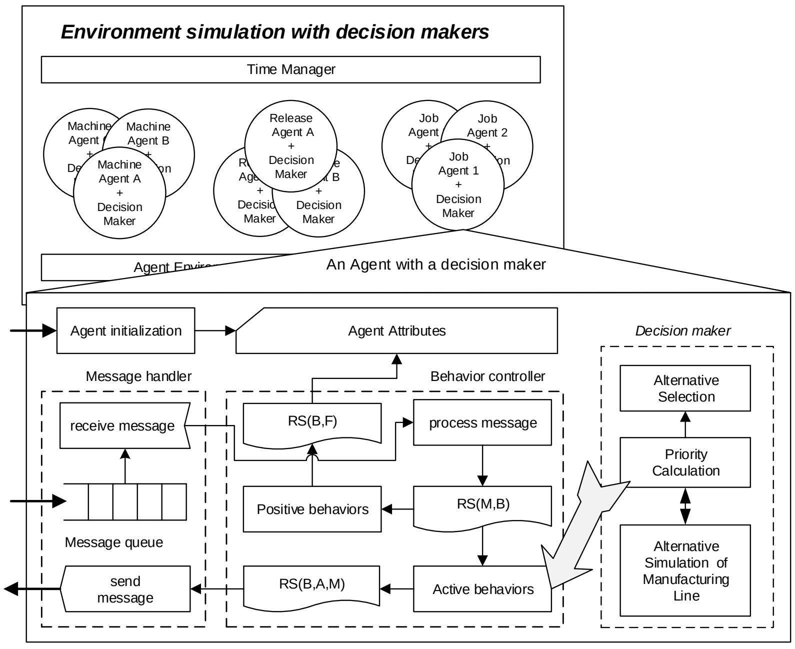

Environment simulation with decision makers
Time Manager
Machine Agent B + Decision Maker
Machine Agent A + Decision Maker
Release Agent A + Decision Maker
Release Agent B + Decision Maker
Job Agent 2 + Decision Maker
Job Agent 1 + Decision Maker
Agent Environment
An Agent with a decision maker
Agent initialization
Agent Attributes
Decision maker
Message handler
Behavior controller
receive message
RS(B,F)
process message
Alternative Selection
Priority Calculation
Positive behaviors
RS(M,B)
Message queue
send message
RS(B,A,M)
Active behaviors
Alternative Simulation of Manufacturing Line

Figure 5.7: An agent with a decision maker

To simplify the evaluation process, we do not develop an independent software scheduler but put the decision makers into their appropriate agents in the environment simulation of the material flow. Each agent has one decision maker. The agent with the decision maker is illustrated in Figure 5.7. We replace the default decision-making behaviors, i.e., making decisions by the decision rules, with new behaviors, i.e., making decisions by the decision makers. This approach will also be used to evaluate the other two decision-making approaches in Chapter 6 and 7.

5.3 Experiments

5.3.1 A Sample Manufacturing Line

The STTD method is applied in a manufacturing system. The system 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 objective is to minimize the cycle time.

Table 5.1: Machines and their functions

Machine	M1	M2	M3	M4	M5
Function	function1 function3	function2 function3	function1	funciton2	function1 function2

Each machine has multiple functions shown in Table 5.1. Machine requirement of each operation is specified by the function shown in Table 5.2. For example, operation Oa1 requires one machine which has function 1. Thus, Oa1 can be processed on machines M1, M3, or M5.

Table 5.2: Products' processing flows and machine requirements

Product	Processing Flow
Pa	Oa1(function1)- Oa2(function2)- Oa3(function3)
Pb	Ob1(function3)- Ob2(function1)- Ob3(function2)

The processing time of an operation is dependent on the machine which processes the operation. All processing times are given in Table 5.3. The minus symbol means that the operation cannot be processed on the related machines.

Table 5.3: Processing times (minutes) of operations on related machines

	M1	M2	M3	M4	M5
Oa1	10	-	10	-	11
Oa2	-	40	-	42	35
Oa3	59	61	-	-	-

Ob1	56	56	-	-	-
Ob2	27	-	28	-	22
Ob3	-	18	-	18	14

The transportation time between two machines is given in Table 5.4. The time spent on a trip may be different from the time spent on the return trip.

Table 5.4: Transportation time (minutes)

To From	M1	M2	M3	M4	M5
M1	0	12	17	4	17
M2	6	0	21	13	24
M3	6	5	0	17	27
M4	13	25	3	0	21
M5	3	13	22	19	0

The setup time on a machine is dependent on the previous operation and current operation. If these two operations are the same type, the machine does not set up. Otherwise, the machine has to set up. As an example, the setup times of machine M1 is given in Table 5.5.

Table 5.5: Setup times on machine M1

Next Pre	Oa1	Oa2	Oa3	Ob1	Ob2	Ob3
Oa1	0	-	15	6	15	-
Oa2	-	-	-	-	-	-
Oa3	6	-	0	2	12	-
Ob1	16	-	9	0	25	-
Ob2	29	-	3	11	0	-
Ob3	-	-	-	-	-	-

5.3.2 Base-rule Analysis

In the alternative simulations, the decision is made by a base decision rule. Different base-rules may result in different simulation results, thereby obtaining different priority values which influence the decision-making directly. Now we carry out some experiments to analyze the influences of the base-rules on the performance of the STTD method. For each experiment, the environment simulation runs for one week and ten replications. The statistical results are the average values over the replications. The decision in the environment simulation is made by the STTD method using one concerned base-rule.

For the release problem, the base-rule can be either CONINT or CONWIP. Thus the environment simulation runs twice. The routing decisions in the environment and alternative simulations are made by the allocation rule SQL. The sequencing decisions in the environment and alternative simulations are made by the dispatching rule FIFO. The experiment results are shown in Table 5.6 and Figure 5.8. The base-rule influences not only the average cycle time but also the throughput because both are considered in the equation of the priority calculation in the STTD-release method. We can see that the STTD-release method using any base-rules almost meets the throughput requirements. The method using the base-rule CONINT performs better on the average cycle time than the one using CONWIP does. Thus, in the latter study, we adopt CONINT base-rule in the STTD-release method.

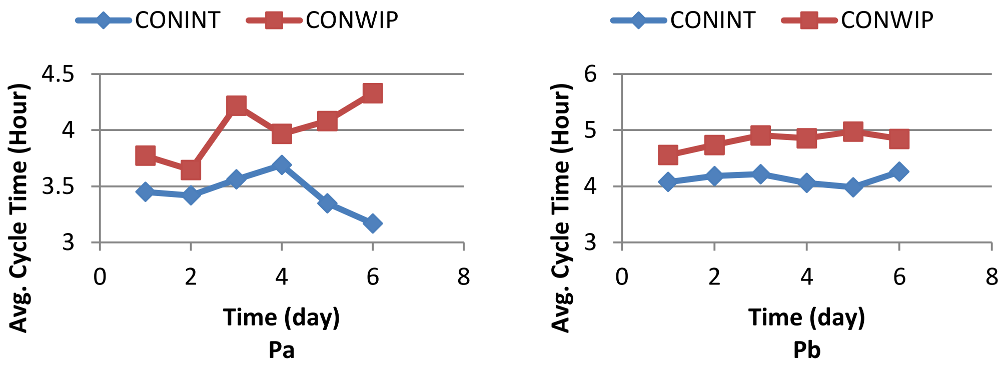


Figure 5.8: Influence of the base-rule on the performance of the STTD-release method

Table 5.6: Throughputs and cycle times obtained by the STTD-release method under different base-rules

Base-rule	Throughput		Cycle Time		
	Pa	Pb	Pa	Pb	Summary
CONINT	109	193	3.44	4.21	3.93
CONWIP	108	191	4.03	4.80	4.53

For the routing problem, we give two base-rules to experiment: SQL and SQT. The release decisions in the environment simulation and alternative simulations are made by the release policy CONINT. The sequencing decisions are made by the dispatching rule FIFO. The experiment results are shown in Figure 5.9 and Table 5.7. The performances of these two base-rules are nearly the same.

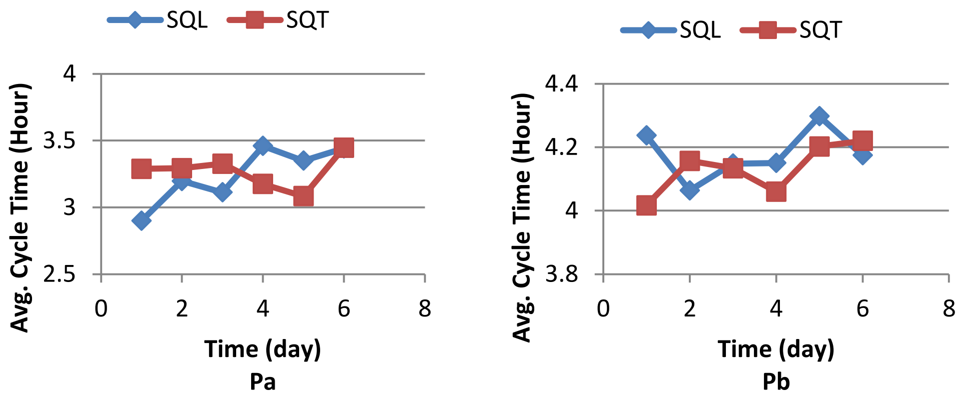


Figure 5.9: Influence of the base-rule on the performance of the STTD-routing method

Table 5.7: Average cycle times obtained by the STTD-routing method under different base-rules

Base-rule / Product	SQL	SQT
Pa	3.20	3.22
Pb	4.18	4.13
Summary	3.84	3.81

For the sequencing problem, five dispatching rules are concerned to be the base-rule in the experiment, including FIFO, SPT, LPT, LRPT (longest remaining processing time), and SRPT (shortest remaining processing time). The release decisions are made by the release policy CONINT in the environment simulation and alternative simulations while the routing decisions are made according to the allocation rule SQL. As Table 5.8 and Figure 5.10 indicate, the STTD method using LPT as the base-rule performs best.

Table 5.8: Average cycle times obtained by the STTD-sequencing method under different base-rules

Base-rule / Product	FIFO	LRPT	LPT	SRPT	SPT
Pa	3.54	3.63	3.59	3.65	3.58
Pb	4.43	4.3	4.26	4.3	4.42
Summary	4.12	4.07	4.03	4.08	4.13

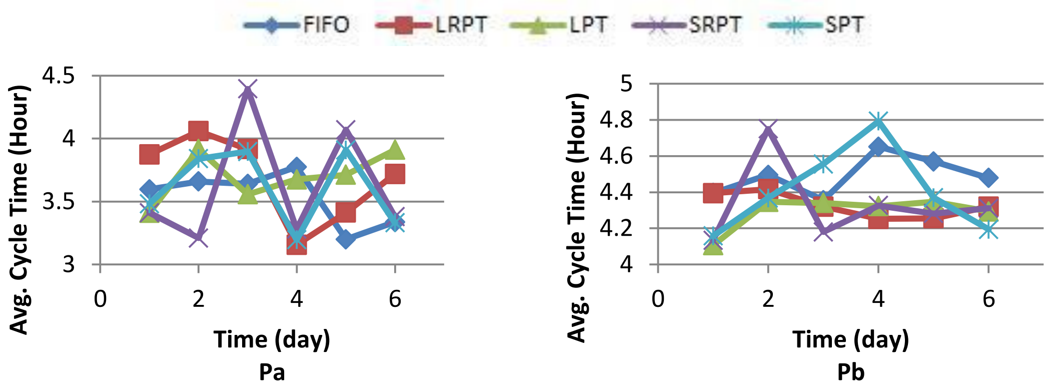


Figure 5.10: Influence of the base-rule on the performance of the STTD-sequencing method

We also tried some other decision rules, such as starvation avoidance policy for the STTD-release method, SPT (shortest processing time) for the STTD-routing method, and so on. However, these decision rules perform badly when they are used independently, not to mention being used as the base-rule. Thus, for the selection of the base-rule, the principle is to select a rule which at least performs not too badly when it is used independently to make the decisions.

5.3.3 Duration Analysis

The duration of the alternative simulation is decided by the number of jobs which are finished during the simulation. When the given numbers of jobs are finished, the alternative simulation ends. The number n is k times of the WIP level at the decision point, $n = \kappa WIP$. We analyze the duration' influence on the performance here.

For the release problem, we focus on two terms: the scheduling performance and the throughput. Figure 5.11 shows the throughput curves under different k values.

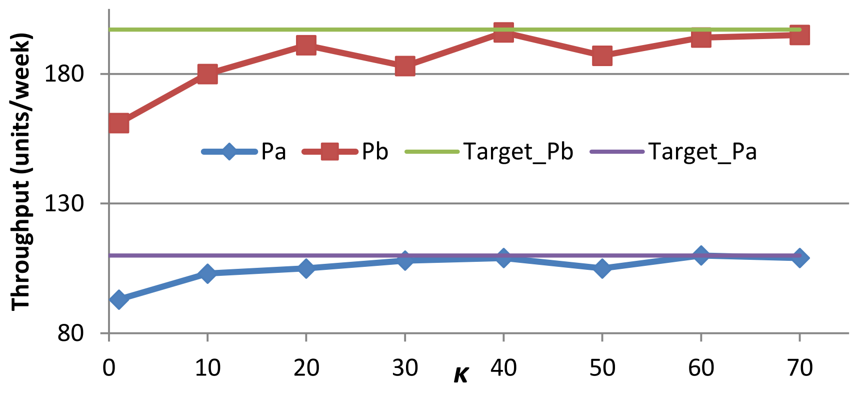


Figure 5.11: Influence of the duration on the throughputs in the STTD-release method

We can see that if the values are too small, the throughputs cannot reach the target throughputs. The reason is that if the simulation duration is too short, the throughputs we get from the simulation results are too rough and the priority values calculated from the throughput are also not accurate. Therefore, the decision we made is not the best, sometimes even worse. After k increases to 20, the throughputs are varying near the targets.

Figure 5.12 shows the average cycle times under different k values. The average cycle times keep increasing as the duration increases before k is 20. Because during this period, the throughputs increase too. It means the release rates are growing. Thus the cycle times are growing. To select a proper value for k, first, it should ensure the target throughputs. On the basis of meeting this condition, it should also minimize the cycle times. In this case, we set k to 40.

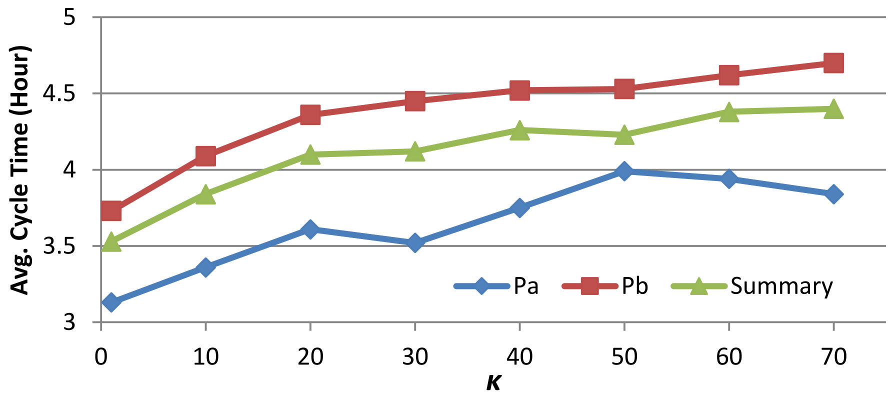


Figure 5.12: Influence of the duration on the average cycle times in the STTD-release method

For the routing problem, if k is greater than 1, the average cycle times are nearly steady, i.e., k does not influence the performance. Thus to save time, we set k to 3.

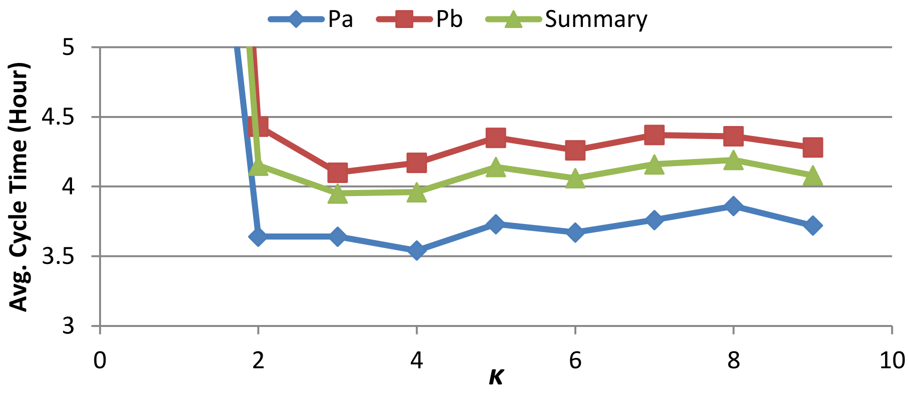


Figure 5.13: Influence of the simulation duration on the performance of the STTD-routing method

For the sequencing problem, the curves of the average cycle times are like valleys. Thus we set k to the value at which the average cycle times are minimal.

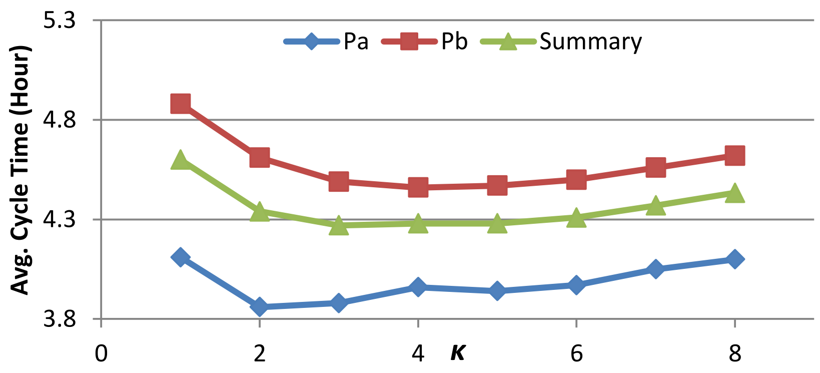


Figure 5.14: Influence of the duration on the performance of the STTD-sequencing method

From the results of all these experiments, we cannot find any general ways to determine the duration. Therefore, the only way is to carry out experiments to find it, like what we did in this section.

5.3.4 Comparisons with Decision Rules

In the two foregoing sections, we determined the base-rules and the duration of the alternative simulations. Now we compare the STTD method with some decision rules. First, we compare the STTD-release method with the release policies CONINT and CONWIP. The routing decisions are made by the allocation rule SQL. The sequencing decisions are made by the dispatching rule FIFO. The results are shown in Figure 5.13 and Table 5.9. The CONINT policy can meet the throughput requirements exactly while the CONWIP cannot (see the throughput of product Pb). The STTD can almost meet the throughput requirements. For the average cycle time, the STTD method performs best.

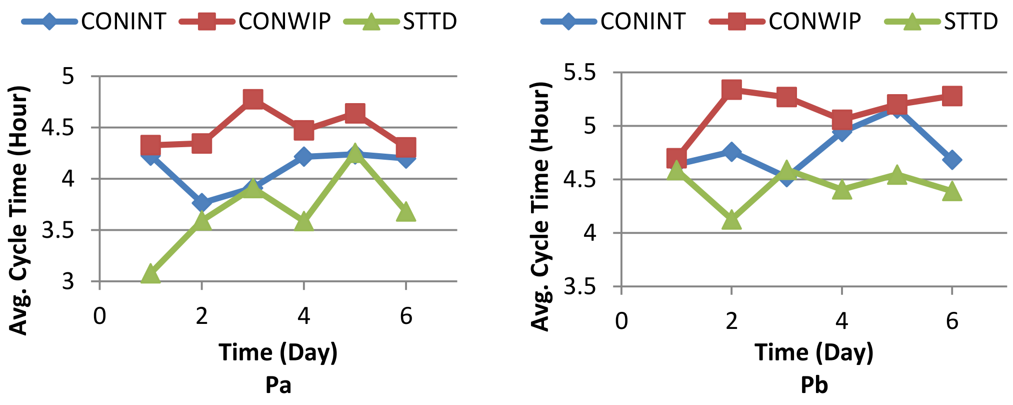


Figure 5.15: Comparison of the STTD-release method with some decision rules

Table 5.9: Throughputs and average cycle times obtained by using release control

Approach	Throughput		Cycle Time		
	Pa	Pb	Pa	Pb	Summary
CONINT	110	197	4.03	4.78	4.52
CONWIP	111	189	4.43	5.14	4.89
STTD	109	196	3.75	4.52	4.26

For the routing problem, we compare the STTD-routing method with the allocation rules SQL and SQT. The release decisions are made by the release policy CONINT. The sequencing decisions are made by the dispatching rule FIFO. Figure 5.16 and Table 5.10 show the results. Obviously, the STTD performs best.

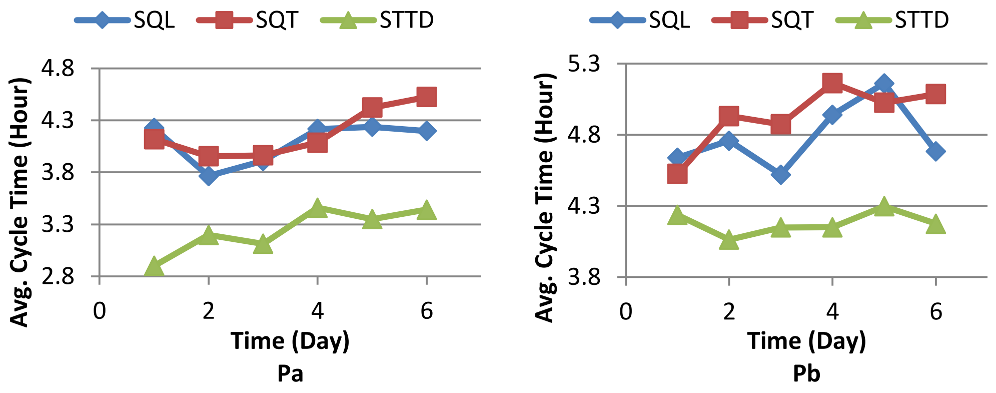


Figure 5.16: Comparison of the STTD-routing method with some decision rules

Table 5.10: Average cycle times obtained by using routing control

Product \ Approach	SQL	SQT	STTD
Pa	4.09	4.08	3.24
Pb	4.78	4.87	4.18
Summary	4.52	4.60	3.84

For the sequencing problem, we compare the STTD-sequencing method with the dispatch rules FIFO and SPT. The release decisions are made by the release policy CONINT, and the routing decisions are made by the allocation rule SQL. The results are shown in Figure 5.17 and Table 5.11. The STTD method still performs best.

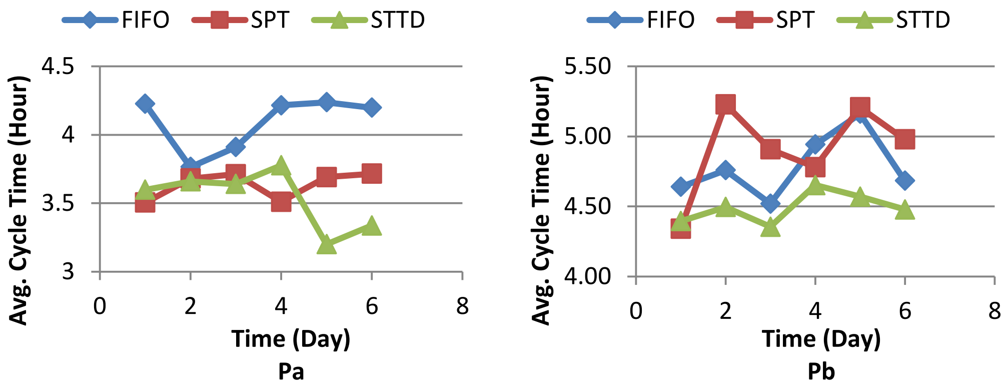


Figure 5.17: Comparison the of STTD-sequencing method with some decision rules

Table 5.11: Average cycle times obtained by using sequencing control

Approach / Product	FIFO	SPT	STTD
Pa	4.09	3.64	3.53
Pb	4.78	4.91	4.49
Summary	4.52	4.46	4.12

5.4 Summary

The release, routing, and sequencing problems are sequential decision-making problems. The decision-making is the selection of the best one from some alternatives. The STTD method calculates each alternative a priority value. The alternative with the highest priority will be selected. The alternative simulations are used to predict the future after an alternative is taken and the priority value is calculated from the predicted future information. In order to evaluate the STTD method, the material flow system is replaced with an environment simulation. The decisions are executed in the environment simulation. The results from the environment simulation are used to evaluate the STTD method. At last comparing with many other decision rules, the STTD method always performs best in the release, routing, and sequencing problems.

6 Intelligent Release, Routing, and Sequencing Based on STTD Method

In the preceding chapter, we use the simulation try-then-decide method to obtain the priority values of alternatives when we make decisions. The experiments prove the excellence of the STTD performance. However, because we have to run an alternative simulation for each alternative, it is kind of time-consuming. Especially for some systems which have high real-time requirements, the STTD method shows its shortages. In this chapter, we will introduce a new approach to calculate the priority values of the alternatives. The approach manages to learn knowledge from the STTD method and find out the relationship between the priority value and the factors which influence the priority. Once we know the relationship, the decision can be made quickly.

6.1 A Scheme for Intelligent Release, Routing, and Sequencing

6.1.1 Basic Idea

The basic idea is shown in Figure 6.1. The simulation is replaced with an intelligent approach which is a data-driven model trained by the dataset obtained from the simulation. The intelligent approach is expected to be more efficient than the simulation.

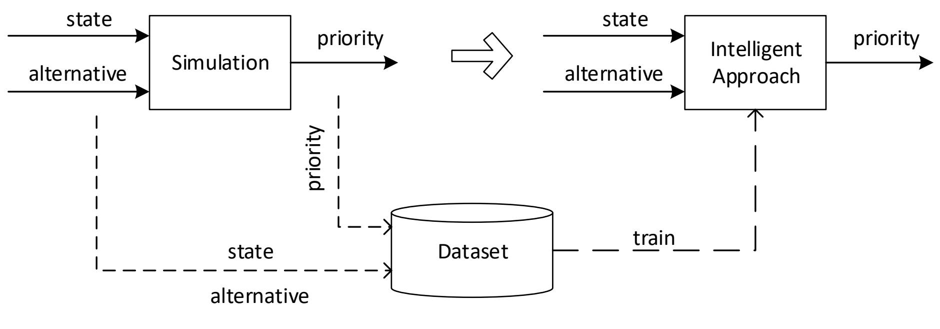


Figure 6.1 Basic idea

6.1.2 Global Factors and Local Factors

The priority of an alternative is not only dependent on the alternative itself, but also the state of the environment when we make the decision. The properties of the alternative influence its priority directly while the state of the environment has an indirect impact on the priority. We call the factors depicting the state as the global factors and the factors depicting the alternative' properties as the local factors. The priority value depends on these two types of factors and can be described as follows,

$$q = Q(G_{local}, G_{global}) .$$

The global factors set the weights of local factors first. The weights of local factors denote the importance of each local factor for decision-making. For example, in the sequencing problem, with the same objective the processing time dominates the value in one state, but in another state, the remaining processing time plays the most important role. Sometimes it is possible that in one state the shorter processing time leads to a higher value, but in another state, the longer processing time results in a higher value. These phenomena depend on the global factors. The relations between local and global factors are presented below.

$$A = Q''(G_{global}) = \{a_1, a_2, a_3, \ldots\}$$
$$q = Q'(AG_{local}) = Q'(a_1 g^1_{local}, a_2 g^2_{local}, a_3 g^3_{local}, \ldots)$$

where A is the weight vector. Different values of the global factors result in different A vectors. The problem is reformulated to determine the functions Q' and Q''. However, Q'' is hard to know. Thus, we use a clustering method to replace function Q''. According to the values of the global factors, the state of the material flow is divided into several patterns. For each pattern, G_{global} varies slightly and A is nearly constant. Therefore, the global factors can be ignored in the same pattern. Consequently, the function Q' is divided into the appropriate subfunctions too. Each pattern has one relevant subfunction to calculate the value. Because A is constant, it can be removed from function Q'. For pattern *i*,

$$q = Q_i^{'}(AG_{local}) \approx Q_i^{'}(G_{local}) .$$

Because G_{global}, G_{local} and q can be easily obtained from the STTD method, The function Q' can be built from these data.

6.1.3 Schemes for Data-driven Model

On the basis of the previous analysis, we create a data-driven model to map the relationship between the priority value and the alternative-state pair. The input of the model is the alternative-state pair. The output is the priority value of the alternative.

The model includes five components: data preprocessing, pattern recognition, function Q' selection, pattern pool, and function pool. The data preprocessing reorganizes the local factors and the global factors. The pattern pool is made of all state patterns which are distinguished by the patterns' centroid. The pattern recognition decides which pattern the current state belongs to in the pattern pool according to the distances from current values of global factors to each centroid. The state will fit the pattern whose centroid is closest to the current values of the global factors. As mentioned above, for each pattern we create one function to map the relationship between the local factors and the priority value. The pattern and the function Q' have a one-to-one relationship. All functions make up the function pool. Thus the appropriate function is selected from the function pool by the function selection according to the pattern of the current state. The selected function is used to compute the priority value according to the local factors. While making a decision, for each alternative we use the model to calculate its priority value. The alternative with the greatest value will be taken. The process flow is shown in Figure 6.2. Obviously, the most important thing is to build the appropriate pattern pool and function pool

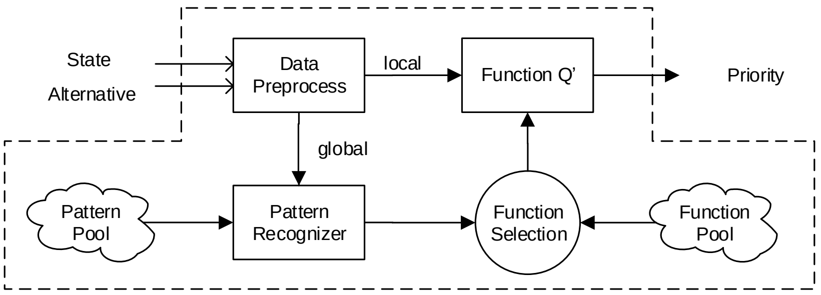


Figure 6.2: Data-driven model for calculating the priority value of the alternative

6.1.4 Decentralized Intelligent Release, Routing, and Sequencing

Recalling the decentralized sequential decision-making in Section 5.1.3, for each type of product a release decision maker is created; for each machine, a sequencing decision maker is created; for each job a routing decision maker is created. For this approach, we still adopt the decentralized manner. Thus, lots of decision makers are developed the same as before, and each decision maker has its own data-driven model for calculating the priority values (see Figure 6.3). However, there are still some slight differences. The current approach is a data-driven method. We need the data to implement the method. If we still create each job a routing decision maker, as we did in the STTD method, this means we have to build a data-driven model for each job. However, jobs are temporary entities and the time that the jobs stay in the system is very short. We have not enough data for each job. Thus we connect the routing decision makers to the machine group agent.

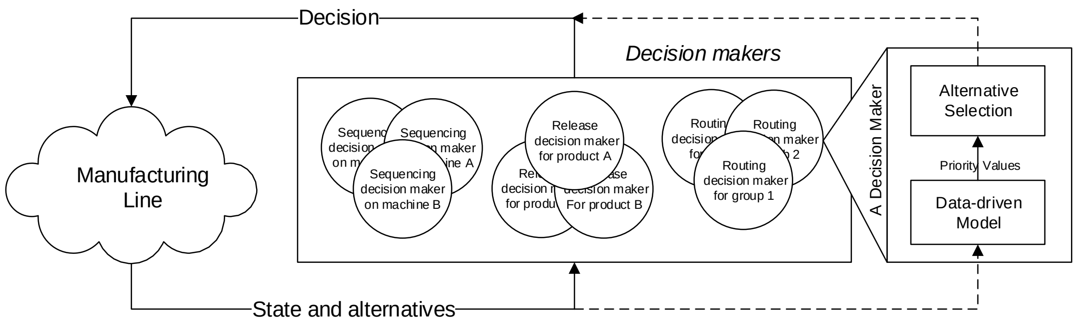

Decision
Decision makers
Manufacturing Line
Sequencing decision maker on machine A
Sequencing decision maker on machine B
Release decision maker for product A
Release decision maker For product B
Routing decision maker for group 2
Routing decision maker for group 1
A Decision Maker
Alternative Selection
Priority Values
Data-driven Model
State and alternatives

Figure 6.3: Decentralized intelligent release, routing, and sequencing

6.2 Data Acquisition and Preprocess

6.2.1 Data Acquisition

Both the pattern pool and function pool are data-driven models. Thus the first task is to collect data we need. The data we need include the alternative, state of the environment, and the priority value of the alternative. We collect these data from the simulation in which the decisions are made by the STTD method. Thus we can learn knowledge from the STTD. When a decision needs to be made in the simulation, the alternative and the state are collected first. The priority value of the alternative is also collected after the STTD method gets it. In order to obtain more general dataset, in the simulation, the decision is made by using a ε-greedy method derived from the function Q'. The ε-greedy method allows us to select an alternative with certain probability ε without considering the priority. Because the simulation is an agent-based simulation and each agent (except job agents) connects to one decision maker, we create a dataset for each agent. The data will be stored in the corresponding dataset which will be used to build its data-driven model, i.e., pattern pool and function pool. The procedure is as follows.

```
    Let $Y_i = \phi$ for all agents $i \in D$, where D is a set of agents in the model
except job agents
    Initialize simulation model randomly
    Start simulation
    Whenever the simulation enters a decision point in agent $i$ Do
        Pause simulation
        Let $s \leftarrow$ current state, $A(s) \leftarrow$ alternative set
        For each alternative $a, a \in A(s)$
```

```
            Calculate priority value for alternative $a$, $q = STTD(s, A(s))$
            $Y_i = Y_i \cup \{(s, a, q)\}$, where q is the priority value of alternative $a$
        Agent $i$ takes an alternative which has the highest priority or randomly
        Continue simulation
    Till the simulation ends then do
        For each agent $i$, build its pattern pool and network pool using dataset $Y_i$
```

6.2.2 Data Preprocess

The state of environment we obtained from the agent-based simulation is the states of all agents. A generalized definition of an agent's state is a set including values of all its dynamic attributes. A huge amount of attributes can be found in the agents, but only a few attributes are what we concerned. Moreover, sometimes a compound attribute may be more meaningful than the elementary attributes. So we are going to filter and reorganize the attributes in this section. The followings are some elementary attributes selected from the release, job, and machine agent.

Attributes of the release agent are,

- Local time
- Starting time to release
- Release end time
- Total job number
- Released job number
- State (blocked/unblocked)
- State starting time
- Expected ending time of the state

Attributes of the job agent are,
- Local time,
- Released time,
- Current operation ID,
- Current machine ID,
- Priority,
- Due date,
- Product configuration,

- Job state,
- State starting time,
- Expected ending time of state.

Attributes of the machine agent are,

- Local time
- Setup configuration
- Batch configuration
- Machine state
- State starting time
- Expected ending time of state

These attributes have only a little meaning in the material flow control. We should aggregate or reorganize them and generate more meaningful factors. After reorganization, based on the experiments the following factors are built in the state of the environment,

- Number of all unavailable machines (breakdown and maintenance),
- Mean queue length of all machines,
- Mean total processing time of jobs in the queues of all machines,
- Mean total waiting time of jobs in the queues of all machines,
- Work in process level,
- Mean progress ratio of all in-process jobs ($ratio = n_{finished\,step} / n_{total\,step}$),
- Number of unavailable machines in each machine group,
- Mean queue length in front of each machine group,
- Mean total processing time of jobs in the queue in front of each machine group,
- Mean total waiting time of jobs in the queue in front of each machine group,
- Work in process level of each product,
- Mean progress ratio of jobs grouped by product type,
- Current theoretical production capacity of each product.

The first six factors are the overall state of the material flow. The following four factors are related to each machine group. The rest three factors refer to different products. The number of factors is $6+4n_M+3n_P$, where n_M and n_P are the number of machine groups and the number of products respectively. In the last factor, the theoretical production capacity is the capacity while only the concerned product is produced in the material flow.

While we make a sequencing decision, the action is the selection of one job from the queue before a machine. The factors of an alternative are,

- Processing time for the job on the machine,
- Setup time of the machine before processing the job,

- Waiting time of the job,
- Remaining step number of the job,
- Total step number of the job,
- Sum of the job’ s processing time at remaining steps,
- Raw processing time of the job,
- Priority of the job,
- Due date for the job.

While we make a routing decision, the action is the selection of one machine from all possible machines to process a job. The factors of an alternative are,

- Utilization of the machine,
- Queue length of the machine,
- Total processing time of jobs in the queue before the machine,
- Processing time for the job on the machine,
- Setup time of the machine before processing the job,
- Transport time from current machine to the selected machine.

While we make a release decision, the action is to decide the number of jobs which will be released. The factors of an alternative are,

- Number of jobs that will be released,
- Number of jobs that have not been released,
- Average release interval *(releaseEndTime-currentTime)/ remainingJobNumber*,
- Work in process level of the concerned product
- Current theoretical production capacity of the concerned product.

6.3 Global State Clustering and Recognition

This section will discuss how to cluster the global factor data and classify the state of material flow into several patterns. Because the amount of global factors is in direct proportion to the number of machines and jobs, the clustering may be a considerable effort. Thus, the dimension should be reduced before the clustering.

6.3.1 Laplacian Eigenmaps for Dimensionality Reduction

Lots of algorithms to dimensionality reduction have been developed, such as Principal components analysis (PCA) (Jolliffe 1986), kernel PCA (Mika et al. 1999, Smola and Schölkopf 1998), locally linear embedding (LLE)(Roweis and Saul 2000), Isomap(Tenenbaum 1998), Laplacian Eigenmaps (LEM)(Belkin and Niyogi 2003), and so on. Laplacian Eigenmaps find a low-dimensional data representation by preserving local properties of the manifold(Belkin and Niyogi 2002). In Laplacian Eigenmaps, the local properties are based on the pairwise distances between near neighbors. Laplacian Eigenmaps compute a low-dimensional representation of the data in which the distances between a data point and its *n* nearest neighbors are minimized. This is done in a weighted manner, i.e., the distance in the low-dimensional data representation between a data point and its first nearest neighbor contributes more to the cost function than the distance between the data point and its second nearest neighbor. The Laplacian eigenmaps is selected in our study. Because of the following: 1) The core algorithm is straightforward. It has a few local computations and one sparse eigenvalue problem. The solution reflects the intrinsic geometric structure of the manifold. 2) The framework of analysis presented makes explicit use of connections to interpret dimensionality-reduction algorithms geometrically. 3) The locality-preserving character makes it relatively insensitive to outliers and noise.

In the study, the Euclidean distance is selected to be the similarity function. The similarity graph is built by the n-nearest neighbor method. The normalized graph Laplacian is used. The application of the Laplacian Eigenmaps to the study is given here.

6.3.1.1 Building the Similarity Graph

We treat each record of global factor data as a data point. The collected data is represented by a matrix X_{global}. Rows of X_{global} correspond to points *x*, columns correspond to global factors *g*, *N* is the number of points needed to be clustered; *M* is the number of global factors. The *n*-nearest neighbor method is used to build the similarity graph *W*. x_i is connected to x_j if x_j is one of the *n*-nearest neighbors. In this case, $w_{i,j}$ is the Euclidean distance between points x_j and x_i,

$$w_{i,j} = \sqrt{\sum_{k=1}^{M} (g_{i,k} - g_{j,k})^2} .$$

If x_j is not in the *n*-nearest neighbors of x_i, $w_{i,j} = 0$.

$$X_{global} = \begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_N \end{array} \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,M} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,M} \\ \vdots & \vdots & \vdots & \vdots \\ g_{N,1} & g_{N,2} & \cdots & g_{N,M} \end{bmatrix} \quad W = \begin{array}{c|cccc} & x_1 & x_2 & \cdots & x_N \\ \hline x_1 & 0 & w_{1,2} & \cdots & w_{1,N} \\ x_2 & w_{2,1} & 0 & \cdots & w_{2,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & w_{N,1} & w_{N,2} & \cdots & 0 \end{array}$$

6.3.1.2 Reducing Dimension

The dimensionality reduction is achieved by calculating eigenvectors of a Laplacian matrix *L*, $L = D - W$, where *D* is a diagonal matrix and $d_{i,i} = \sum_{j=1}^{N} w_{j,i}$. Then we compute the first M^- generalized eigenvectors $u_1, u_2, \ldots, u_{M^-}$ corresponding to the first M^- smallest eigenvalues of the generalized eigenproblem $Lu = \lambda Du$. Let *V* be the matrix containing M^- eigenvectors as columns. The matrix *V* is the result of the dimensionality reduction.

$$V = \begin{array}{c} \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & \cdots & u_{M^-} \end{array} \\ \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,M^-} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,M^-} \\ \vdots & \vdots & \vdots & \vdots \\ v_{N,1} & v_{N,2} & \cdots & v_{N,M^-} \end{bmatrix} \end{array}$$

6.3.1.3 Clustering and Centroid Calculation

Now the *M*-dimension global factor data X_{global} has been reduced to M^- dimension data *V*. Using any classical clustering methods, such as *K*-means method, to cluster $V = \{V_1, V_2, \ldots, V_k\}$, in which each row denotes one point, into *k* classes. The data X_{global} is divided into *k* classes, $X_{global} = \{X_1, X_2, \ldots, X_k\}$, where $X_j = \{x_i \mid v_{i,*} \in V_j\}$; $v_{i,*}$ means the *i*-th row of *V*. Then we calculate centroids *C* of classes according to the global factor data in the corresponding class, $C = \{C_1, C_2, \ldots, C_k\}$. The centroids make up the pattern pool and will be used in the pattern recognition. A point belongs to class *i* if the distance between the points and the centroids C_i is the shortest.

6.3.2 K-means Clustering on Low Dimensional Data

Thousands of clustering algorithms have been proposed in the literature in many different scientific disciplines(Jain 2010). Single-link(Sneath and Sokal 1973), complete-link(King 1967), and minimum-variance(Ward Jr 1963, Murtagh 1983) are some very common hierarchical algorithms. K-means. (MacQueen 1967), ISODATA(Ball and Hall 1965), and dynamic clustering(Diday 1973) are some partitional algorithms. The *k*-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. It will be used in the study, because its time complexity is O(*nml*) and space complexity is O(*k*+*h*), where *n* is the size of the dataset, *k* is the number of clusters, and *l* is the number of iterations. The underlying idea of the *k*-means algorithm is to try to find clusters that minimize the within-cluster variance and maximize the between-cluster variance because the total variance is fixed. To determine the cluster number and the initial clusters are the two most important tasks.

6.3.2.1 Cluster Number

The elbow method, which tries different cluster number and observes the value $F = \mathit{within\,group\,error} \,/\, \mathit{between\,group\,error}$, is used to determine the cluster number. The within-group error is the sum of distances from each point to its centroid. The between-group error is the sum of average distances from each point to the other centroids. A curve can be drawn according to the F value and the cluster number, shown in Figure 6.4. On the curve, we can find a point of inflection after which the F value decreases very slowly. The cluster number related to the point is the cluster number we will use in the k-means algorithm.

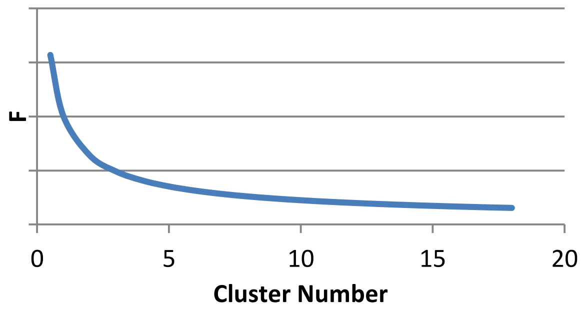


Figure 6.4: Elbow method to determine the cluster number

6.3.2.2 Initial centroids

The initial centroids are obtained by a particular procedure. The first centroid is selected randomly from the dataset. Then from the dataset, we select the second centroid which is the furthest from the first centroid. After that, we choose the third centroid which is the furthest from the first and second centroids, i.e., the maximal sum of distances from the concerned point to the first and second centroid. The other centroids will be selected in the same manner. In order to eliminate the influence of initial centroids, the k-means algorithm usually runs many times with different initial centroids. The result from the best run will be selected.

6.4 Neural network

In the study, neural networks are introduced to perform the function Q'. A neural network (Haykin 2009) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. In more practical terms neural networks are non-linear statistical data modeling or decision-making tools. Feedforward networks can be used for any kind of input to output mapping. A feed-forward network with one hidden layer and enough neurons in the hidden layers can fit any finite input-output mapping problem. Here, we use it to map the relationship between the local factor and the priorities of the alternatives. Backpropagation algorithm(Hecht-Nielsen 1988, Rumelhart, Hinton, and Williams 1988) is the most used training method for the feedforward neural networks. There are considerable methods to accelerate the convergence of the algorithm, such as varying the learning rate(Jacobs 1988), using momentum(Phansalkar and Sastry 1994) and rescaling variables(Tollenaere 1990, Rigler, Irvine, and Vogl 1991). In our study, the Marquardt algorithm for nonlinear least squares is introduced into the backpropagation algorithm.

6.4.1 Neural Network Structure

Three-layer feedforward networks are introduced into the model. The inputs *X* are the local factors, and the output *Y* is the priority. We use the tangent sigmoid transfer function $f_h(x) = 1/(1+e^{-x})$ in the hidden layer and linear function $f_o(x) = x$ in the output layer. The sum of square errors (SSE) is defined to evaluate the training process.

$$SSE(w) = \sum_{n=1}^{N} e_n \text{ , where } e_n = \frac{1}{2}\sum_{k=1}^{K}(y_{n,k} - o_{n,k})^2 ,$$

$$o_{n,k} = f_o(\sum_{j=1}^{J} w_{j,k}h_{n,j} + w_{0,k}) , \; h_{n,j} = f_h(\sum_{m=1}^{M} w_{m,j}x_{n,m} + w_{0,j}) .$$

where *n* is the index of training data, *N* is the data number. e_n denotes the error of the *n*-th training data. The symbols *k*, *j*, *m* denote the node in the output layer, in the hidden layer, and in the input layer respectively; $y_{n,k}$ is the target output of the node *k* in the output layer. $o_{n,k}$ is the actual output. $h_{n,j}$ is the node *j*' s output. $x_{n,m}$ is the data of the node *m* in the input layer. $w_{0,j}$ and $w_{0,k}$ are biases of the node *j* in the hidden layer and the node *k* in the output layer. $w_{j,k}$ is the weight between the node *j* in the hidden layer and the node *k* in the output layer.

6.4.2 Neural Network Training

We adopt Levenberg-Marquardt algorithm to train the networks in batches. The Levenberg-Marquardt algorithm combines the steepest descent method and the Gauss-Newton algorithm. It inherits the speed advantage of the Gauss-Newton algorithm and the stability of the steepest descent method. A mathematical description of the LM neural network training algorithm has been presented by Hagan and Menhaj (1994). The formula of updating weights and biases is

$$w_{s+1} = w_s - (Jac_s^T Jac_s + \lambda I)^{-1} Jac_s^T E_s$$

where I is an identity matrix, λ is the damping factor. The damping factor λ is adjusted at each epoch and guides towards the optimization process. If reduction of E is rapid, a smaller value can be used, bringing the algorithm closer to the Gauss-Newton algorithm, whereas if an epoch gives an insufficient reduction in the residue, λ can be increased, providing a step closer to the gradient descent direction. Jacobian matrix *Jac* is shown as follows. It is a N-by-W matrix, where N is the number of training patterns, and W is the total number of weights. It can be computed by using the chain rule of calculus and the first derivatives of the transfer functions.

In addition, the early stopping technique is used to avoid over-fitting. The data is divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will typically decrease during the initial phase of training. However, when the network begins to overfit the data, the error on the validation set will typically begin to rise. When the times of validation error increasing reach a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned. The third subset is the testing set. When the training is done, the network is tested by using this set.

$$\begin{array}{c} \qquad\qquad \textit{Hidden Layer} \qquad\qquad\qquad \textit{Output Layer} \\ Jac = \begin{bmatrix} \partial e_1/\partial w_{0,j} & \cdots & \partial e_1/\partial w_{i,j} & \partial e_1/\partial w_{0,k} & \cdots & \partial e_1/\partial w_{j,k} \\ \partial e_2/\partial w_{0,j} & \cdots & \partial e_2/\partial w_{i,j} & \partial e_2/\partial w_{0,k} & \cdots & \partial e_2/\partial w_{j,k} \\ \vdots & & \vdots & \vdots & & \vdots \\ \partial e_N/\partial w_{0,j} & \cdots & \partial e_N/\partial w_{i,j} & \partial e_N/\partial w_{0,k} & \cdots & \partial e_N/\partial w_{j,k} \end{bmatrix}, \quad E = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}. \end{array}$$

In Section 6.3, the global factor data is divided into k classes. Consequently, the local factor data and the priority data are grouped into k groups. Based on each group of local factor data and the priority data, one neural network is trained. These networks make up the function pool mentioned in Section 6.1.3.

6.4.3 Parameter Determination using Ant Colony Algorithm

Till now, there are not any theories for calculating the number of the nodes at the hidden layer. Through the experiments, we realized that the performance of the network is sensitive to the number of the hidden nodes. There are also many other parameters of the network which need to be determined before the training, such as iterate number, maximal failure number, initial damping factor, the factor for changing the damping factor and so on. How to find out the best combination of these parameters directly determines the networks' performance. In this section, we introduce the ant colony algorithm to calculate the parameters.

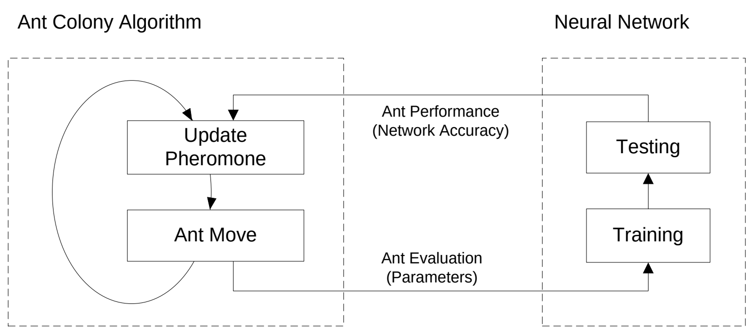


Figure 6.5: Ant colony algorithm determines the parameters for the neural networks

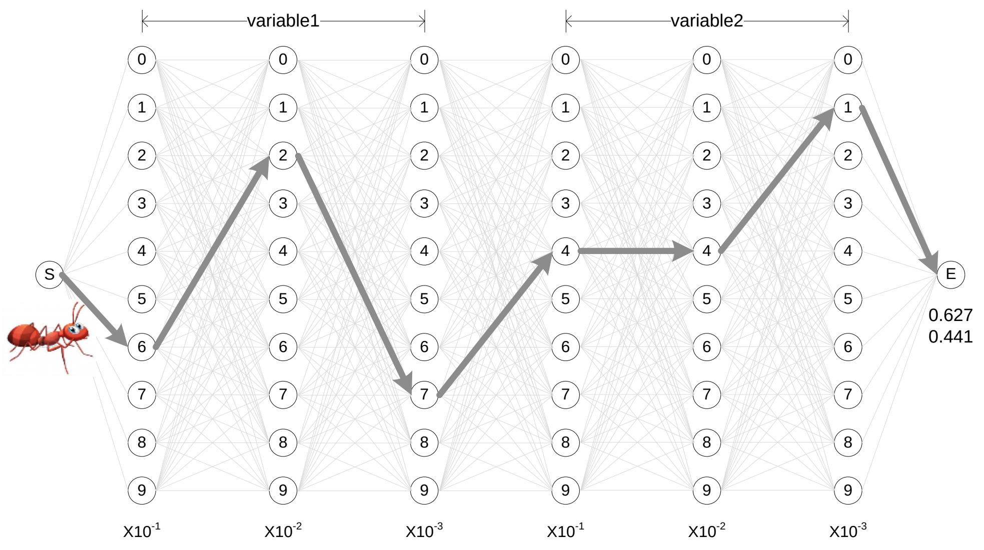


Figure 6.6: Discretization of the continuous problem for the ant colony algorithm

Determination of the parameters is an optimization problem. The objective is to maximize the accuracy of the neural network. And the decision variables are the parameters we have to determine. The accuracy of the neural network comes from the test results by using the test dataset. Figure 6.5 shows the paradigm of the approach. All parameters make up an ant. The ants are evaluated by training and testing the network with the parameters that the ants represent. Because the parameters are continuous, but the ant colony algorithm is only suitable for a discrete model, the continuous parameters have to be converted to discrete variables first (Liao et al. 2014). We normalize all variables to [0, 1) and specify for each variable a number of digits after the decimal point. Each decimal place can be one digit from 0 to 9. Thus a network can be created as shown in Figure 6.6. The start node and the end node have no real meaning. Each of other columns represents a decimal place of one variable. Figure 6.6 shows an example having two variables and three decimal digits for each variable. Thus the continuous problem turns into a discrete routing problem in the network.

Ants will run on the network and leave the pheromone on the sides. The pheromone can also be evaporated as time goes on. The side that the ant will choose depends on the volume of the pheromone on the sides. The probability to select one side is directly proportional to the volume of the pheromone (roulette wheel selection). The path that an ant goes through denotes a solution. We update the pheromones only after the ant finishes the run. The increased pheromone level is related to the evaluation results of the ant. The evaporation rate of pheromones is fixed. When more and more ants go through the network, there will be one path with the very high pheromone level, and most of the ants run on the path. This path will be the optimal solution. In our approach, the parameters are normalized according to their lower and upper bound. In order to speed up the convergence, we form batches of ants. For each batch, we select several best ants and update the pheromones according to these ants' performances. The pheromone update is performed as follows.

$$\lambda = (1-\rho)\lambda_0 + f(r_{i,k}^{sim}, \kappa_{i,k}^{sim})Q,$$

where λ and λ_0 are the new and old value of the pheromone. ρ is the evaporation rate. Q denotes the importance of the objective.

6.5 Experiments

We still use the manufacturing line mentioned in the previous chapter to carry out experiments. The simulation is used to evaluate the approaches. The simulation runs for one week. Because during the simulation, the simulation will stop and wait for the decision makers' result, the whole time that the simulation spends can reflect the efficiency of the approaches. Here we compare the method (INT1) with some decision rules and the STTD method. First, we compare the INT1-release method with release policy CONINT and CONWIP, and the STTD-release method. The routing decisions are made by the allocation rule SQL. The sequencing decisions are made by the dispatching rule FIFO. The results are shown in Figure 6.7 and Table 6.1. For the average cycle time, the STTD method performs best. However, it needs much time. The INT1-release method is the second best one and needs just very short time.

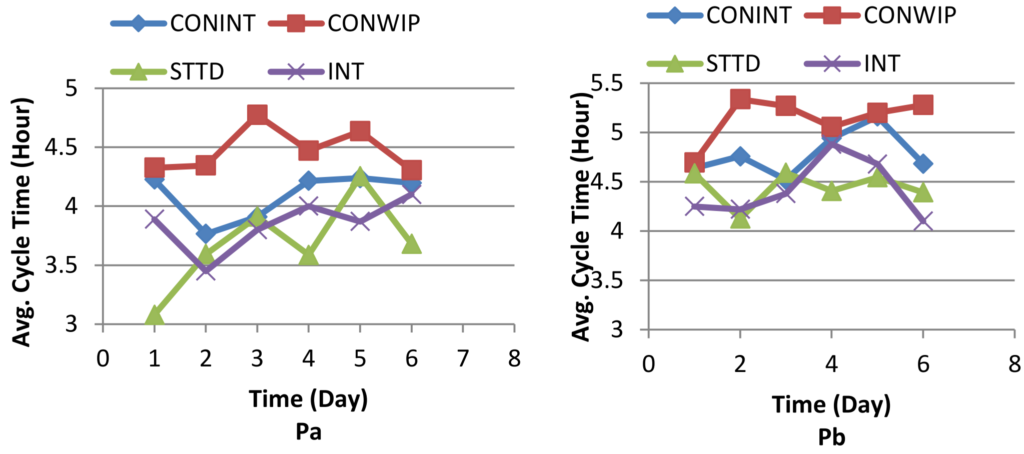


Figure 6.7: Comparison the INT1-release method with other approaches

Table 6.1: Throughputs and average cycle times obtained by using release control

Approach	Throughput		Cycle Time (Hour)			Run Time (s)
	Pa	Pb	Pa	Pb	Summary	
CONINT	110	197	4.03	4.78	4.52	1.1
CONWIP	111	189	4.43	5.14	4.89	1.2
STTD	109	196	3.75	4.52	4.26	215.3
INT1	111	195	3.85	4.41	4.21	1.8

For the routing problem, we compare the INT1-routing method with the allocation rules SQL, SQT, and the STTD method. The release decisions are made by the release policy CONINT. The sequencing decisions are made by the dispatching rule FIFO. Figure 6.8 and Table 6.2 show the results. We can get the same conclusion. The INT1-routing method takes very short time while its performance is better than the decision rules.

Table 6.2: Average cycle times obtained by using routing control

Items / Approach		SQL	SQT	STTD	INT1
Cycle Time (Hour)	Pa	4.09	4.08	3.24	3.51
	Pb	4.78	4.87	4.18	4.37
	Sum.	4.52	4.60	3.84	4.07
Run Time (s)		1.0	1.2	167.9	1.5

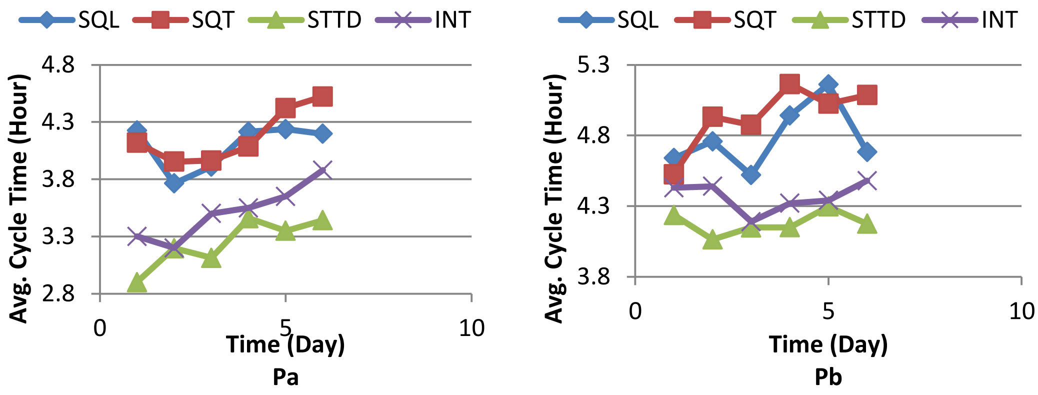


Figure 6.8: Comparison the INT1-routing method with other approaches

For the sequencing problem, we compare the INT1-sequencing method with the dispatch rules FIFO, SPT, and the STTD-sequencing method. The release decisions are made by the release policy CONINT, and the routing decisions are made by the allocation rule SQL. The results are shown in Figure 6.9 and Table 6.3. We can get the same conclusion for the INT1-sequencing method as the INT1-release and INT1-routing methods.

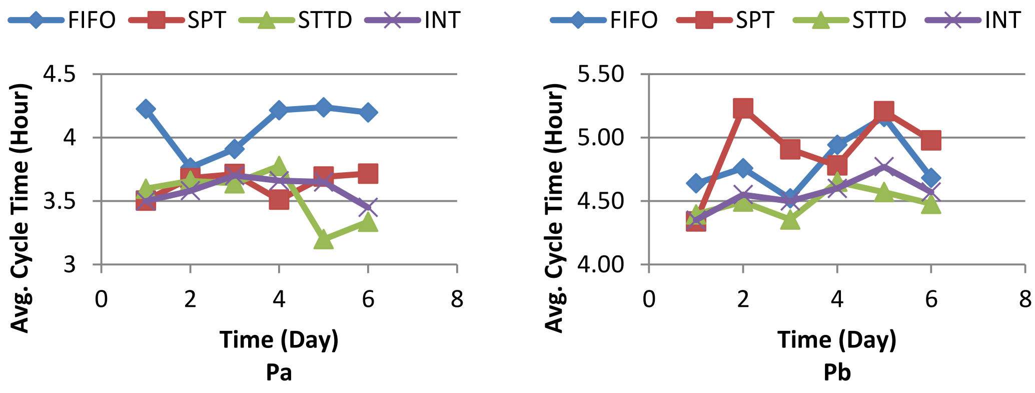


Figure 6.9: Comparison the INT1-sequencing method with other approaches

Table 6.3: Average cycle times obtained by using sequencing control

Items \ Approach		FIFO	SPT	STTD	INT1
Cycle Time (Hour)	Pa	4.09	3.64	3.53	3.58
	Pb	4.78	4.91	4.49	4.56
	Summary	4.52	4.46	4.12	4.21
Run Time (s)		1.0	1.3	86.8	1.4

6.6 Summary

A data-driven model is introduced to calculate the priority values for alternatives. 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 factors influence the priority value of the alternative: global factors and local factors. The environment 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 function pool. While making the decision, the decision maker determines the pattern of the current environment according to the pattern pool and selects one corresponding neural network from the function pool. The neural network will calculate the priority for each alternative according to the local factors. Compared to other approaches, the approach always performs better than decision rules. Even though it performs not as well as the STTD method, it spends just very short time. Therefore, it can be used in the system which has high requirements on the efficiency of the approach.

7 Intelligent Release, Routing, and Sequencing Based on Markov Decision Process

In this chapter, we present a different method of calculating the priority values for alternatives. The method is based on the Markov decision process (MDP) (Puterman 2014) which aims to find out an optimal action-value function to take the best action. The concepts in the MDP are quite similar to the concepts appearing in Chapter 5 and 6. The actions in the MDP are the selections of the best alternatives. The value of an action equals the priority value of the alternative, which is determined by both the state of the environment and the action itself. Reinforcement Learning (Sutton and Barto 1998) provides some good approaches to solve the MDPs problem, such as temporal difference learning (Tesauro 1992, 1995) and Q-learning(Watkins and Dayan 1992, Rummery and Niranjan 1994). We still use the same data-driven model presented in Chapter 6 to map the relationship between the value of action and the state-action pair. The data-driven model is built from the data generated from a simulation-based Q-learning algorithm. Most concepts and notations in this chapter come from Sutton and Barto's book. The initial idea is inspired by Gabel and Riedmiller (2008a). They introduced the reinforcement learning to a job-shop scheduling problem, i.e., the sequencing problem. For more information about Gabel and Riedmiller's study, refer to (Gabel and Riedmiller 2012, 2008b, 2007b). We extend the study to the release and routing problems and combine the algorithms with the simulation.

7.1 Markov Decision Process

7.1.1 Introduction to Markov Decision Process

A general Markov decision process has two components: a decision maker and its environment. To be more specific, the Markov decision process is a five-tuple $M = < T, S, A(s), P(s'|s,a), R(s'|s,a) >$, where T is a set of decision points; S is a set of all possible states of the environment; $A(s)$ is a set of possible actions (alternatives) while the state is $s, s \in S$; $P(s'|s,a)$ is a set of the probabilities that the state of the environment changes from state s to state s' after taking action a, which satisfies $\sum_{s' \in S} p(s'|s,a) = 1$. $R(s'|s,a)$ is a set of rewards that the decision maker obtains after taking action *a* and the state of the environment changes from state *s* to state *s'*. The decision maker and environment interact at decision points *T*. At each decision point $t, t \in T$, the decision maker receives the environment's state t_{sa} , and on that basis, the decision maker selects an action $a, a \in A(s)$. One step later, in part as a consequence of its action, the decision maker receives a numerical reward $r_{t+1}, r_{t+1} \in \Re$ and finds itself in a new state $s', s' \in S$. At each decision point, the decision maker implements mappings from states to probabilities of selecting each possible action. This mapping is called the policy and is denoted by π_t , where $\pi_t(s,a)$ is the probability that the decision maker takes action *a* in state s at *t*-th decision point.

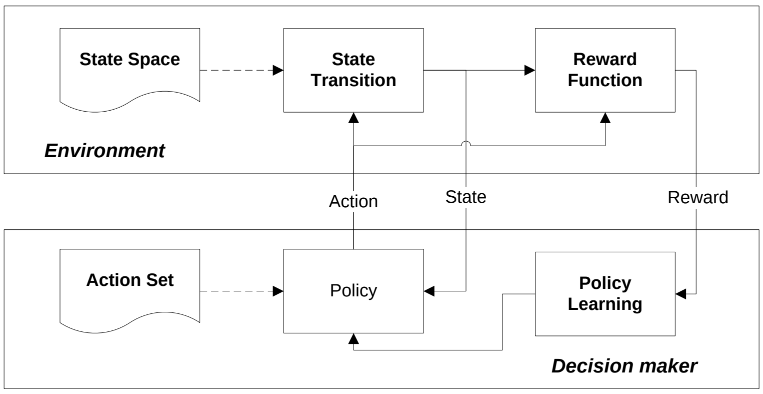


Figure 7.1: Markov decision process

Informally, the decision maker's goal is to maximize the total amount of rewards it receives. This means maximizing not immediate reward, but the cumulative reward in the long run. At each decision point, a reward is a simple number $r_t \in \Re$. In general, we seek to maximize the expected reward defined as some specific function of the reward sequence. A widespread way is that the decision maker tries to select actions so that the sum of the discounted rewards it receives in the future is maximized. The expected discounted reward is,

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where r is a parameter, $0 \leq \gamma \leq 1$, called the discount rate.

7.1.2 Value Function and Optimal Value Function

Estimation of how good it is to perform a given action in a given state (or how good it is for the decision maker to be in a given state) is usually based on some value functions. The notion of "how good" here is defined in terms of future rewards that can be expected. The rewards the decision maker can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular policies(Sutton and Barto 1998).

Recall that policy, π, is a mapping from each state, $s \in S$, and action, $a \in A(s)$, to the probability $\pi(s,a)$ of taking action a when in state s. Informally, the value of a state s under a policy π, represented by $V^{\pi}(s)$, is the expected reward. For MDPs, we can define $V^{\pi}(s)$ formally as

$$V^{\pi}(s) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\}.$$

We call the function $V^{\pi}(s)$ the state-value function for policy π. Similarly, we define the value of taking action a in state s under a policy π, denoted by $Q^{\pi}(s,a)$, as the expected reward starting from s, taking action a, and thereafter following policy π,

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\}.$$

We call the function $Q^{\pi}(s,a)$ the action-value function for policy π. In order to obtain the value functions, we describe these two value functions in the form of Bellman equation. To solve the Bellman equation, we can obtain the value functions for policy π. The Bellman equation for the state value function is defined below, which will be used in the policy iteration algorithm in the following section.

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} p(s' \mid s,a)[r(s' \mid s,a) + \gamma V^{\pi}(s')]$$

The aim of a decision maker is to find the optimal policy. There is always at least one policy that is better or equal than all other policies. This policy is called the optimal policy. There is only one optimal-value-function, which is the base for an optimal policy. An optimal state-value function is, denoted by $V^*(s)$, and defined as,

$$V^*(s) = \max_{\pi} V^{\pi}(s), \text{ for all } s \in S.$$

Among all policies, there is also one that is at least equal or better than the others using the optimal action-value-function. We define an optimal action-value function,

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \text{ for all } s \in S, a \in A(s).$$

Thus, we can rewrite Q^* in terms of V^* as follows:

$$Q^*(s,a) = E\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\}.$$

The optimal value of a state must equal the expected reward for the best action from that state. Thus, the optimal state-value function can be defined as,

$$V^*(s) = \max_{a \in A(s)} Q^{\pi^*}(s,a) = \max_{a \in A(s)} \sum_{s'} p(s' \mid s,a)[r(s' \mid s,a) + \gamma V^*(s')].$$

This is the Bellman equation for the optimal state value function. This equation will be used in the value iteration algorithm in the following section. We can see that in the equation the optimal values have been independent of the policy. To solve this Bellman equation, we can obtain the optimal value function. Once we have the optimal value function, it is effortless to get the optimal policy.

7.1.3 Algorithms for Solving Markov Decision Process

Policy iteration and value iteration are two standard methods of computing an optimal MDP policy. Policy iteration starts with an arbitrary policy π_0 (an approximation to the optimal policy which works best) and iteratively improves it. It carries out the following algorithm.

```
Step 1 Initialization
        $V(s) \in \Re, \pi(s) \in A(s)$ arbitrarily for all $s \in S$
Step 2 Policy Evaluation
    Repeat
        $\Delta \leftarrow 0$
        For each $s \in S$
            $v \leftarrow V(s)$
            $V(s) \leftarrow \sum_{s'} p(s' | s, \pi(s))[r(s' | s, \pi(s)) + \gamma V(s')]$
            $\Delta \leftarrow \max(\Delta, | v - V(s) |)$
        End
    Until $\Delta < \theta$
Step 3 Policy Improvement
    $policy - stable \leftarrow true$
    For each $s \in S$
        $b \leftarrow \pi(s)$
        $\pi(s) = \arg\max_{a \in A(s)} \sum_{s'} p(s' | s, a)[r(s' | s, a) + \gamma V(s')]$
        If $b \neq \pi(s)$ then $policy - stable \leftarrow false$
    End
    If policy-stable, then stop; else go to Step 2
```

In fact, the step of the policy evaluation in the policy iteration algorithm can be truncated in several ways without losing the convergence guarantees. One particular important case is when the policy evaluation is stopped after just one sweep (one backup of each state). This algorithm is called value iteration. The algorithm is shown as follows.

```
Initialize V arbitrarily, e.g., $V(s)=0$, for all $s \in S \cup \{s^f\}$
Loop
    $\Delta \leftarrow 0$
    For each $s \in S$
        $v = V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} p(s'|s,a)[r(s'|s,a)+\gamma V(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    End
Until $\Delta < \theta$ ( $s\_time\_\min = s\_time$ is a small positive number)
Output a deterministic policy, $\pi$, so that
    $\pi(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r(s'|s,a)+\gamma V(s')]$
```

Both two algorithms have to iterate over the state space and need the transition probabilities to be given. However, for most problems, the state space is infinite or very huge, and it is impossible or tough to acquire transition probabilities. Thus, we introduce another algorithm, called Q-learning, to solve the problem. Q-learning is a model-free reinforcement learning technique, which means we do not need to know details of the model, like the state space and transition probability. Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP). It works by learning an action-value function that ultimately gives the expected reward of taking a given action in a given state and following the optimal policy thereafter. The algorithm is presented as follows.

```
Initialize $Q(s, a)$ arbitrarily, e.g., $Q(s,a)=0$, for all $s \in S \cup \{s^f\}, a \in A(s)$, let
$i$=0
While $i$ < Max iteration number, do
    Initialize current state $s$
    While $s \neq s^f$, do
        Take action $a$ using a policy derived from Q (e.g., $\varepsilon -$greedy)
        Observe new state $s'$ , $r(s'|s,a)$, and action set $A(s')$ in state $s'$
            $Q(s,a) \leftarrow (1-\alpha)Q(s,a)+\alpha[r(s'|s,a)+\gamma \max_{a' \in A(s')} Q(s',a')]$
        $s \leftarrow s'$
    End
    $i$=$i$+1
End
Output a deterministic policy, $\pi$, so that
    $\pi(s) = \arg\max_a Q(s,a)$
```

7.1.4 Markov Decision Process with Multiple Decision Makers

In the real world, one Markov decision process usually has various decision makers shown in Figure 7.2. The decision makers make their own decisions and influence the environment. They have their own action sets and policies and may interact with each other. In the environment, each decision maker may have its own transition and reward functions. Their local goals are to maximize their cumulative rewards, in the long run, to achieve a global goal together. The local goals may conflict with each other even though they have the same global goal.

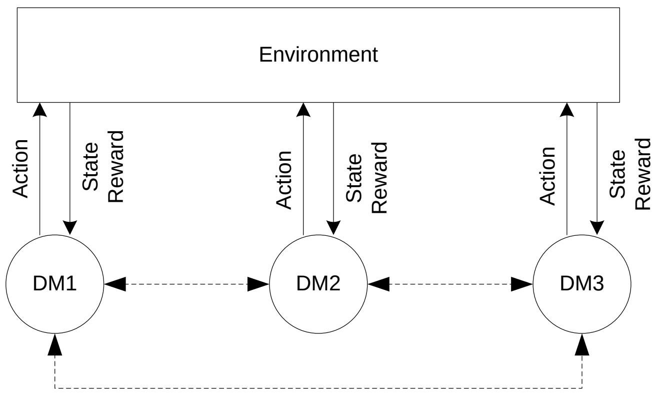


Figure 7.2: Markov decision process with multiple decision makers

7.2 MDP model of the Release, Routing, and Sequencing

In this section, we will create an MDP model for the release, routing and sequencing problems with multiple decision makers. The task is to determine the decision makers and define the five-tuple for the problems, including the decision points, the state space, the action sets, the transition procedure, and the reward function.

7.2.1 Multiple Decision Makers

Similar to Chapter 6, for each type of product a release decision maker is created; for each machine group, a sequencing decision maker and a routing decision maker are created.

7.2.2 Decision Points

Points of referred time at which decisions are made are defined as decision points. The set of decision points *T, a* subset of the non-negative real line may be classified in two ways: either a discrete set or a continuum, and either a finite or an infinite set. In our study the decision point set is discrete, i.e., the decisions are made when certain discrete events occur. It can be either finitely or infinitely dependent on the type of the material flow. For example, some material flows never stop because the manufacturing orders come successively; some material flows will stop after finishing all current orders and wait for the next orders.

7.2.3 State Space

The state here refers to the state of the environment. The state space can be either discrete or continuous. It also may be either finite or infinite. In our model, the state space is discrete and finite but is huge. In order to facilitate the introduction to the transition procedure, the state space will be precisely defined here.

The state space is defined as a set $S=\{s\}$ for all possible state *s*. Because we use the agent-based model to represent the environment, the state of the environment is the integration of all agents' states. A state *s* of the environment is,

$$s=\{\{s_j \mid j\in J\},\{s_l \mid l\in L\},\{s_m \mid m\in M\}\},$$

where s_j is the state of job agent *j*; *J* is a set of job agents. s_l is the state of release agent *l*; *L* is a set of release agents. s_m is the state of machine agent *m*; *M* is a set of machine agents. Recall the states of the machine we addressed in Chapter 4, here we give each state an id, denoted by ms_id , $ms\_id \in \{0,1,2,3,4\}$. The relation is shown as follows: $0 \leftarrow FREE$, $1 \leftarrow SETUP$, $2 \leftarrow BUSY$, $3 \leftarrow BREAKDOWN$, $4 \leftarrow MAINTAINENCE$. Let s_time denote the remaining time that the state will hold. Thus,

$$s_m=\{ms\_id, s\_time\}.$$

In addition to the state ID and the remaining time that the state will last, a job' s state includes its position and progress. The position describes which machine or buffer or transport line the job can be found on. The progress indicates which step the job reaches. In order to simplify the model, we use only the machine ID to describe the position. If the job is on a transport line or in a buffer, we specify the machine that the job will go instead of the transport line and buffer. Thus,

$$s_j=\{js\_id, s\_time, m\_id, o\_id\},$$

where $m\_id \in \{1,2,3,...\}$ is the id of the specified machine; $o\_id \in \{1,2,3,...\}$ is the current operation id of the job. $js\_id \in \{-1,0,1,2,3,...\}$ is the state id of the job. If the state id of a job is -1, it means the job has been released. If the state id is 0, it means the job has been finished. Others are normal states that we mentioned in Chapter 4. The relation is shown as follows: $-1 \leftarrow UNRELEASED, 0 \leftarrow FINISHED$, $1 \leftarrow TRANSPORTING, 2 \leftarrow WAITING$ $3 \leftarrow PROCESSING, 4 \leftarrow BLOCKED$.

In each release agent, there is a counter which counts the number of released jobs. The state of the release agent is the current value of the counter and the remaining time to next release, $s_l = \{rj\_num, s\_time\}$, where rj_num is the number of the released job. Note that even though the remaining times s_time are continuous variables, in Section 7.2.5 we will know that the remaining times are updated discretely, so they are also discrete.

7.2.4 Action set

Action set $A(s)$ is the set of actions that can be taken in state s. In our model, the set is finite and discrete. The size of the set is relatively small. The action is actually a selection of one decision from some alternatives. For each type of decision makers, they have their alternatives, i.e., an action set. For the release decision makers, the action set is,

$$A_l(s) = \{0, 1, 2, ..., \max\_num_l\},$$

where the elements mean the number of jobs that will be released. $\max\_num_l$ denotes the maximal release number at a time, $\max\_num_l = WIP_{\max} - WIP$. WIP is the work-in-process level of the product that the decision maker is in charge of. $WIP_{\max}$ is the maximal WIP level of the product, which can be obtained according to the configuration of buffers in front of the related machines. This action set varies at different decision points due to the variable WIP level.

For the sequencing decision makers, they will select one job from the buffer to process at a decision point. The action set at the decision point is

$$A_m(s) = \{null, \{j \mid j \in J_m\}\} ,$$

where J_m is used to define the jobs waiting in the buffer in front of the machine *m* at the decision point. *Null* means it will select nothing and keep itself free even though jobs are waiting for it. This selection is quite useful if the machine is a batch processing machine. In this way, the batching problem is included in the sequencing problem. J_m changes at the decision points, thus the action set is also variable.

For the routing decision makers, after a job is finished they will select one machine from a set of machines that can carry out the next operation *o* of the job. The action set at the decision point is

$$A_j(s) = \{null, \{m \mid m \in M_{j,o}\}\} ,$$

where $M_{j,o}$ is a set of machines which can perform operation *o* in the job *j*. If the selection is null, the job will continue staying on current machine and block the machine (because we assume that there is no buffer behind the machine). The practical meaning of the null selection is that "waiting a moment and observing the environment, then making the decision."

From the definitions of the state and the action sets, we can see that, given a state, it is possible to check if any decision makers have decisions to make in the state and it is also possible to obtain action sets dynamically from the state. The procedure is described as follows.

Procedure I: Given a state, find out all decision makers who have decisions to make and generate the appropriate action sets. We call the decision makers active decision makers.

We scan all decision makers. For a sequencing decision maker on machine *m*, if the state of *m* is FREE, $ms\_id = 0$ and there is at least one job whose position is on the machine *m*, $m\_id = ID(m)$, and the state of the job is WAITING, $s\_id = 2$, the sequencing decision maker will have to make a sequencing decision. J_m appearing in action set $A_m(s)$ will be the set of the jobs which meet the conditions above. Similarly, if the state of the machine is *3* or *4* and the remaining time $s\_time = 0$, the decision maker also needs to make a decision. A routing decision maker will make a routing decision to select a machine for job *j*, if the state of job *j* $js\_id = 3$, and the remaining time of the state $s\_time = 0$. The action set for the current operation o in job j is always constant no matter if the state of the machines in the set is FREE or not. We assume that even if a machine is broken, the job can still go there and wait in the buffer in front of the machine, because it may result in a good performance. For a release decision maker, if there are jobs whose operation id is the last id and the state is PROCESSING, $js\_id = 3$, and the remaining time of the state $s\_time = 0$, the release decision maker has to make a release decision. $\max\_num_l$ will be updated according to current WIP level of the related product.

In addition, given a state of the environment, we may find that more than one active decision-makers, i.e., several decision makers may need to make decisions in the state at the same time. In this case, we give a sequence for the decision makers and let the decision makers make decisions in such order. The sequence is related to the scheduling objective.

7.2.5 Transition Procedure

The transition function is described by $P(s'|s,a), where\ s,s' \in S, a \in A(s)$. *P* is a set of the probabilities that the state of the environment changes from state *s* to state *s'* after taking action *a*. In our model, after an action is taken, the new state is also random. For example, when a sequencing decision maker selects a job to process, the state of the job agent will be updated as follows, $js\_id = 3$ and $s\_time = proc\_time_{j,0}$, where $proc\_time_{j,o}$ is the processing time of operation o in job j on the machine. Because the processing time, following specific distribution function, is stochastic, s_time is stochastic too. Thus, the state of the job $s_j = \{js\_id, s\_time, m\_id, o\_id\}$ is random.

Except for the remaining time, other elements are determined. The probability that the state changes to a new state with certain remaining time is the probability that the remaining time is generated following the distribution function.

Because the state space is quite huge and it is impossible to enumerate all possible states and create P, Q-learning will be used in our approach. We just need to sample the next new state from the current state rather than calculating the transition probabilities. A state transition procedure can be specified for each type of decision makers.

Procedure II: Given a state and an action that one decision maker takes, randomly generate the next new state. Only the relevant agents in the agent-based model are considered.

After a routing decision maker takes one routing action a (a machine) for a job, both the state of the job and the state of the previous machine have to be updated. For the job agent, the state is updated as follows: $js\_id = 1$, $o\_id = o\_id + 1$, $m\_id = ID(a)$, $s\_time = trans\_time_{m \to a}$. For the machine agent, $ms\_id = 0$, $s\_time = \infty$. Because the machine becomes free, the machine needs to make sequencing decision. If the job agent selects null, the job is blocked. In this case, only the job agent needs to be updated, $js\_id = 4$, $s\_time = -1$.

After a sequencing decision maker takes one sequencing action a (a job), the state of both the job and the current machine have to be updated. For the job agent, the state is updated as follows: $js\_id = 3$, $s\_time = proc\_time_{j,o}$. For the machine agent, $ms\_id = 1$, $s\_time = proc\_time_{j,o}$. If the machine agent selects null, nothing will be changed.

After a release agent takes one release action a (number of jobs that will be released), both the state of the release agent and the states of the released jobs need to be updated. For the release agent, the released job number num is updated, $rj\_num = rj\_num + a$; For all released jobs, the state IDs change from -1 to 1, $js\_id = 1$, and we set the current operation id, $o\_id = 1$. Then each released job agent has to make a routing decision.

In our model, an initial state can be either empty or generated randomly. In order to find an optimal policy, the iteration always starts from the initial state. After that, we examine if there are any active decision makers. If so, the active decision makers will make decisions successively and update the states of the corresponding agents. Then we will examine again if in the new state there are still active decision makers, the active decision makers make decisions too. The loop ends when there are no more active decision makers. Now the problem is how to let the environment move to next state. In this situation, we carry out Procedure III, shown as follows, to generate the next new state. And then we examine the decision agents again. If the state space is finite, the iteration will end when the state enters the final state.

Procedure III: Given a state and Procedure I cannot find any active decision-makers in the state, generate the next new state.

We scan all agents' state and find out the agent which has the minimal s_time, where $s\_time \neq -1 \, or \, \infty$. We let $s\_time\_\min = s\_time$, $s\_time = 0$, and update other agents' remaining times $s\_time = s\_time - s\_time\_\min$. In this way, the environment moves to a new state.

7.2.6 Reward Function

The decision maker always learns to maximize its reward. If we want it to do something for us, we must provide rewards for it. When the rewards are maximized, the decision maker will also achieve our goals. It is thus critical that the rewards we set up truly indicate what we want to be accomplished. In particular, the reward is a way of communicating with the decision maker what we want it to achieve, not how we want it to achieve.

A crucial precondition to enable the decision maker to learn to make sophisticated scheduling decisions is that the reward function coincides with the overall objective of scheduling. In our model, the reward, $r(s'|s,a) = (1-\kappa)r_a + \kappa\Delta r_{s'}$, includes two parts, where $0 < \kappa < 1$ indicates the relative importance of the parts. The first part is the reward for the action selection. The second part is the reward for the state sojourning in a period Δ, where Δ is the time between two successive decision points. The first part is only dependent on the new state s' and the scheduling objective. If the objective is related to the due date of jobs, r_a is the average time slack of all in-process jobs in state s'. If the objective is associated with the completion time of jobs, r_a is the reciprocal of the number of jobs which are waiting in buffers in state s'. For the release decision maker, besides factors above, the reward also considers the current average release rate to ensure to meet the throughput constraints.

7.3 Simulation-based Q-learning Algorithm for Solving RRSMDP Model

The algorithms for solving the model aim at generating for each decision maker a policy which maps the relation between states of the environment and actions which will be taken. The value iteration and policy iteration, given in Section 7.1.3, output the actions being taken in all possible states of the environment. They backup values for all states. However, in our model, even though the state space is finite, it is still too large to enumerate all the states. In addition, the output that we need is a parameterized policy for each decision maker rather than the results of the policy, i.e., actions we take. In other words, we try to find each decision maker a function which calculate the value of action according to the action-state pair. The decision maker takes one action from the action set based on the values of the actions. Therefore, in our model, the Q-learning algorithm is changed to fit our problem. The data-driven model presented in Chapter 6 maps the value of the action to the state-action pair

7.3.1 Improved Q-learning Algorithm in the Multi-agent Environment

Procedure I, II and III are used to explore the state space from an initial state to a final state. The period from an initial state to the final is considered as an episode. The iteration times we give are the number of the episodes. If the number of the episodes is big enough, more and more state will be visited. The more states that the algorithm involves, the better policy we will obtain. We specify each decision maker a data-driven model Q_i, i.e., action-value function, which will be updated after the decision maker takes action. The algorithm is given as follows:

```
Input: initial state $s^0$, iteration times $N$ and exploration rate $\varepsilon$
Initialize data-driven model $Q_i$ randomly, for each decision maker $i \in D$, where D is a set of all decision makers
Let index=0
A: Let $s = s^0$, previous state $s_i^- = null$, previous action $a_i^- = null$ for all agent $i \in D$
B: Find out all active decision makers $D' \subseteq D$ in state $s$ using Procedure I
If $|D'| = 0$ then go to Line C
For each active decision agent $i, i \in D_i$
  $s_i \leftarrow s$, $A(s_i) \leftarrow current\ action\ set$
      If $s_i^- \neq null$ then
      $Q_i(s_i^-, a_i^-) \leftarrow (1-\alpha)Q_i(s_i^-, a_i^-) + \alpha[r(s_i \mid s_i^-, a_i^-) + \gamma \max_{a \in A(s_i)} Q_i(s_i, a_i)]$

      Update data-driven model $Q_i = UpdateValueFunc(s_i^-, a_i^-, Q_i(s_i^-, a_i^-))$
  End
  Select an action $a_i$ by exploiting $Q_i$ greedily according to
      $a_i = \arg\max_{a' \in A(s_i)} Q_d(s_i, a')$ or select $a_i$ randomly $a_i \in A(s_i)$ with probability $\varepsilon$
  $s_i^- \leftarrow s_i$, $a_i^- \leftarrow a_i$
End
```

```
    Execute actions $\{a_i\}$ that the decision makes just selected
    Update states of the influenced agents using Procedure II, we get new state $s'$
    Let $s = s'$, go to Line B
    C: Move to the next state $s'$ using Procedure III, let $s = s'$
        If $s = s^f$ then
            index=index+1
            If $index<N$ then go to Line A else the algorithm ends
    Else
            Go to Line B
    End
```

Given a state, we use Procedure I to find all active decision makers. After that, for each active decision maker, the data-driven model will be updated according to the previous action it took and the previous state in which the previous action is taken. Then, it selects an action by using a ε-greedy method derived from the value function. The ε-greedy method allows us to enter a new state with certain probability ε without considering the value function, to explore the state space. When all active decision makers have updated their value function and selected their actions, the selected actions will be executed in a specific sequence. After that, Procedure II will update the state of the environment, i.e., states of the influenced agents. The procedure I and II will repeat until Procedure I cannot find any active decision makers. Then Procedure III moves the environment to the next state, and Procedure I and II start again. When the environment enters the final state, we give a new initial state again and repeat all steps above. The algorithm ends when it reaches maximal iteration times given before.

7.3.2 Batch-mode Q-learning Algorithm in the Multi-agent Environment

In the previous algorithm, the data-driven models are updated every time immediately. Usually, batch-mode training can yield improvements regarding learning speed and performance. In this section, a batch-model value iteration algorithm is addressed. We create a training pattern set X for each decision maker. Once a decision maker made decisions, the related pattern data will be put into its training pattern set. When an episode finishes, the data-driven model of each decision maker will be trained using the training pattern sets. After that, the sets are emptied, and a new episode starts.

```
Input: initial state $s^0$, iteration times $N$ and exploration rate $\varepsilon$
Initialize data-driven model $Q_i$ randomly, for each decision maker $i \in D$, where D is a set of all decision makers
Let index=0
A: Let $s = s^0$, previous state $s_i^- = null$, previous action $a_i^- = null$, dataset $X_i = \phi$ for all agent $i \in D$
B: Find out all active decision makers $D' \subseteq D$ in state $s$ using Procedure I
If $|D'| = 0$ then go to Line C
For each active decision agent $i, i \in D'$
  $s_i \leftarrow s$, $A(s_i) \leftarrow current\ action\ set$
        If $s_i^- \neq null$ then
        $Q_i(s_i^-, a_i^-) \leftarrow r(s_i \mid s_i^-, a_i^-) + \gamma \max_{a \in A(s_i)} Q_i(s_i, a_i)$
        $X_i = X_i \cup \{(s_i^-, a_i^-, Q_i(s_i^-, a_i^-))\}$
  End
  Select an action $a_i$ by exploiting $Q_i$ greedily according to
        $a_i = \arg\max_{a' \in A(s_i)} Q_d(s_i, a')$ or select $a_i$ randomly $a_i \in A(s_i)$ with probability $\varepsilon$
  $s_i^- \leftarrow s_i$, $a_i^- \leftarrow a_i$
End
Execute actions $\{a_i\}$ that the decision makes just selected
Update states of the influenced agents using Procedure II, we get new state $s'$
Let $s = s'$, go to Line B
C: Move to the next state $s'$ using Procedure III, let $s = s'$
    If $s = s^f$ then
        For each decision maker $i$, $i \in D$
              Update data-driven model $Q_i = UpdateValueFunc(X_i)$
        index=index+1
        If $index<N$ then go to Line A else the algorithm ends
  Else
        Go to Line B
  End
```

7.3.3 Simulation-based Batch-mode Q-learning Algorithm

As we mentioned before, three procedures are used to explore the state from a large number of states. Once a decision-maker takes an action, some agents in the environment model will be influenced. We have to update not only the state of the connected agent but also the states of the influenced agents. The changes in the state of the affected agents may lead the relevant decision-makers to take some other actions too. A reaction chain exists in this case. In addition, every time the environment enters a new state, we have to use Procedure I to find out all active decision makers. Once the action chain ends, we have to use Procedure III to move to a new state. Things seem to be too complicated.

But if we put the decision makers into their related agent as before we did, we can find that all of these procedures have been implemented in the agent-based simulation model we created before. Updating other agents' states can be achieved by communications among them. What the procedure III did is exactly to advance and synchronize the time of all agents in the simulation. The simulation can capture all events including the events at which the decisions are made. So there is no need to find out the active decision makers as Procedure II did. Therefore, we can use the simulation model in the Q-learning algorithm. In practice, the transition procedure is more complicated than what we addressed in Section 7.2.5. If all features are considered, the transition will lead the MDP model unreadable. On the contrary, the agent-based simulation model can easily involve most of the features. Thus we use the simulation model in the Q-learning algorithm. The combination of them is shown as follows.

```
    Input: iteration times $N$ and exploration rate $\varepsilon$
    Initialize $Q_i$ randomly, for each agent $i \in D$, $D$ is a set of all agents except job
agents
    Let index=0
    A: Initialize simulation model randomly
    Start simulation
    Let $X_i = \phi$, previous state $s_i^- = null$, previous action $a_i^- = null$ for all agent
$i \in D$
    Whenever the simulation enters a decision point in agent $i$ do the following,
        Pause simulation
        $s_i \leftarrow current\ state$, $A(s_i) \leftarrow current\ action\ set$
     If $s_i^- \neq null$ then
            $Q_i(s_i^-, a_i^-) \leftarrow r(s_i \mid s_i^-, a_i^-) + \gamma \max_{a \in A(s_i)} Q_i(s_i, a_i)$
            $X_i = X_i \cup \{(s_i^-, a_i^-, Q_i(s_i^-, a_i^-))\}$
    End
        Agent $i$ takes action $a_i$ by exploiting $Q_i$ greedily according to
            $a_i = \arg\max_{a' \in A(s_i)} Q_i(s_i, a')$ or select $a_i$ randomly $a_i \in A(s_i)$ with
probability $\varepsilon$
        $s_i^- \leftarrow s_i$, $a_i^- \leftarrow a_i$
        Continue simulation
    Till the simulation ends then do
        For each agent $i$,
            update data-driven model $Q_i = UpdateValueFunc(X_i)$
        Let index=index+1
        If index<$N$, then go to Line A
```

In each agent, a training dataset and a data-driven model representing the value function are created. Once a decision-making event occurs, the simulation will pause and wait for the related agent to make the decision. The agent will use the data-driven model to make the decision and also communicate with the influenced agents which will update their state according to the messages they received. The old state, old action, and the new state are stored in the dataset. After that, the simulation continues. Each simulation run is an episode as we mentioned before. When the agents receive the signal about the simulation ends, they will start the training process for the data-driven model.

7.4 Experiments

In this section, the algorithm in Section 7.3.3 will be evaluated. We still use the manufacturing line mentioned in Chapter 5 to carry out experiments. The simulations are used to evaluate the approaches. The simulation runs for one week. Because during the simulation, the simulation will stop and wait for the decision makers' results, the whole time that the simulation spends can reflect the efficiency of the approach.

7.4.1 Convergence

Due to the generalization and extrapolation abilities of neural networks, unpredictable changes at different places in the state-action space can be built(François-Lavet, Fonteneau, and Ernst 2015). It is known that errors may be propagated and that this may even become unstable. It cannot be guaranteed that the current estimation for the accumulated rewards always underestimates the optimal reward, and therefore convergence is not assured (Tsitsiklis and Van Roy 1997, Gordon 1999). Convergence may be slow or even unreliable with the neural networks(Riedmiller 2005). Moreover, in the multi-agent Markov decision process, the convergence is still an open question(Abtahi and Meybodi 2008). Thus, in the study, the convergence should be verified experimentally.

In the algorithm in Section 7.3.3, all agents are trained together after each iteration (one simulation run). Thus, after each iteration, we record the integrated average cycle time and draw a curve. The curve will depict the convergence of the algorithm. Figure 7.3 is the obtained convergence curve in which the convergence is not obvious. But from the trending line, we can see that the average cycle time is going down slowly and at last almost stays steady.

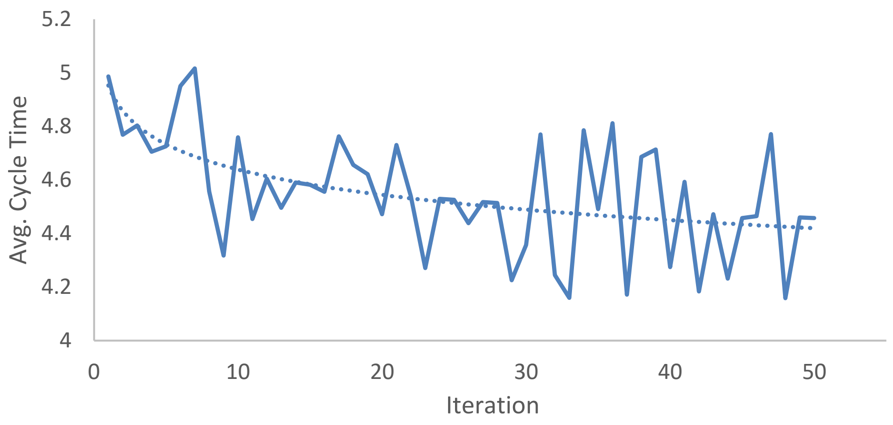


Figure 7.3 Convergence curve of the algorithm for the sequencing problem

7.4.2 Comparison

Here we compare the intelligent method (INT2) in this chapter with some decision rules, the STTD method, and the intelligent method (INT1) in the previous chapters. First, we compare the INT2-release method with the release policies CONINT and CONWIP, the STTD-release method, and the INT1-release method.

Table 7.1: Throughputs and average cycle times obtained by using release control

Approach	Throughput		Cycle Time (Hour)			Run Time (s)
	Pa	Pb	Pa	Pb	Summary	
CONINT	110	197	4.03	4.78	4.52	1.1
CONWIP	111	189	4.43	5.14	4.89	1.2
STTD	109	196	3.75	4.52	4.26	215.3
INT1	111	195	3.85	4.41	4.21	1.8
INT2	108	190	3.71	4.53	4.23	1.7

The routing decisions are made by the allocation rule SQL. The sequencing decisions are made by the dispatching rule FIFO. The results are shown in Figure 7.4 and Table 7.1. We can see that the INT2-release method performs better than the release policies and similar to the STTD-release method and the INT1-release method.

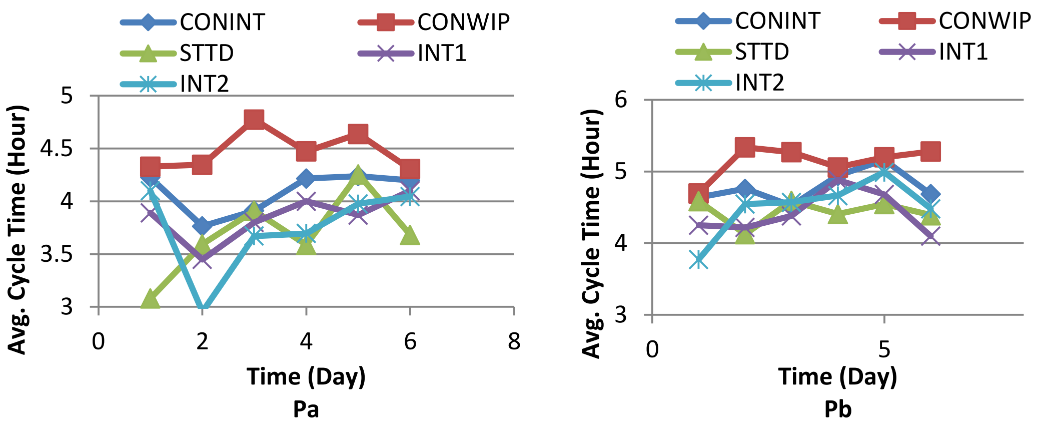


Figure 7.4: Comparison between the INT2-release method and other approaches

For the routing problem, we compare the INT2-routing method with the allocation rules SQL and SQT, the STTD method, and the INT1-routing method. The release decisions are made by the release policy CONINT. The sequencing decisions are made by the dispatching rule FIFO. Figure 7.5 and Table 7.2 show the results. We can see that the INT2- routing method performs better than the allocation rules but worse than the STTD- routing method and the INT1- routing method.

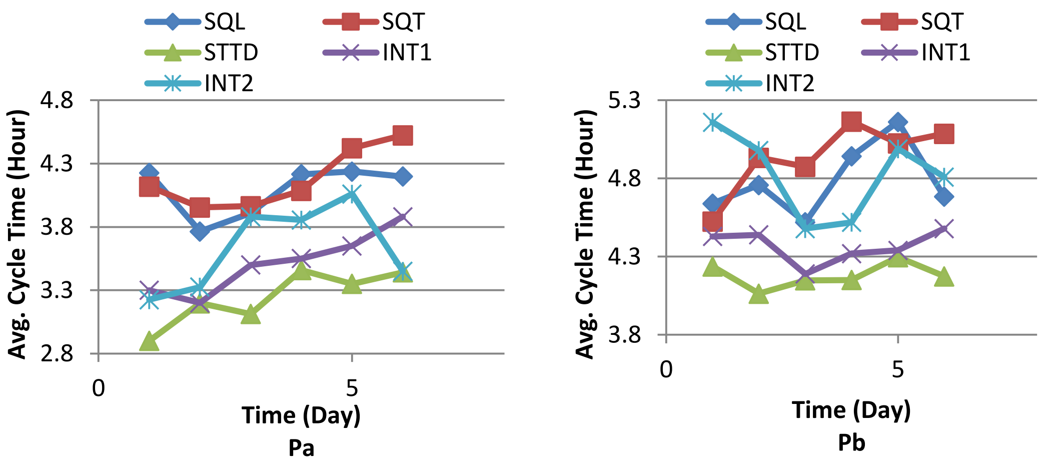


Figure 7.5: Comparison between the INT2-routing method and other approaches

Table 7.2: Average cycle times obtained by using routing control

Items \ Approach		SQL	SQT	STTD	INT1	INT2
Cycle Time (Hour)	Pa	4.09	4.08	3.24	3.51	3.64
	Pb	4.78	4.87	4.18	4.37	4.80
	Sum.	4.52	4.60	3.84	4.07	4.38
Run Time (s)		1.0	1.2	167.9	1.5	1.9

For the sequencing problem, we compare the INT2-sequencing method with the dispatch rules FIFO and SPT, the STTD-sequencing method, and the INT1-sequencing method. The release decisions are made by the release policy CONINT, and the routing decisions are made by the allocation rule SQL. The results are shown in Figure 7.6 and Table 7.3. We can see that the INT2- sequencing method performs similar to the dispatch rules but worse than the STTD- sequencing method and the INT1- sequencing method.

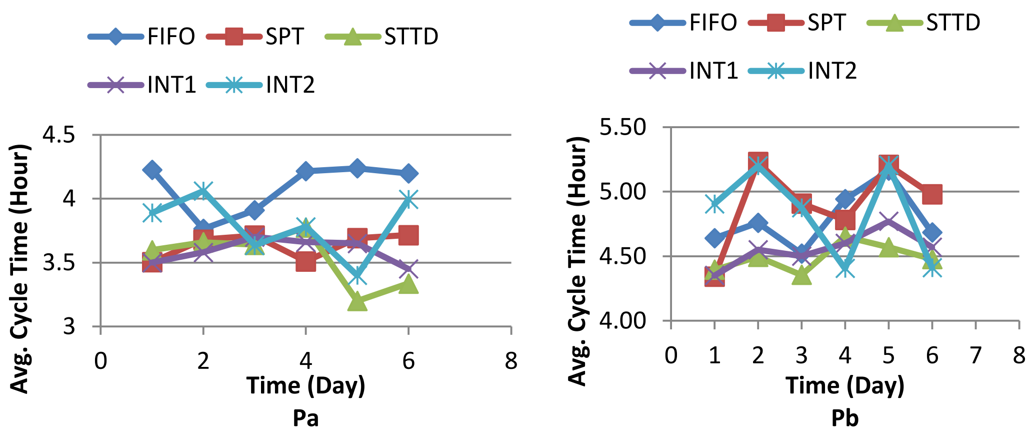


Figure 7.6: Comparison between the INT-sequencing method and other approaches

Table 7.3: Average cycle times obtained by using sequencing control

Items \ Approach		FIFO	SPT	STTD	INT1	INT2
Cycle Time (Hour)	Pa	4.09	3.64	3.53	3.58	3.73
	Pb	4.78	4.91	4.49	4.56	4.88
	Summary	4.52	4.46	4.12	4.21	4.46
Run Time (s)		1.0	1.3	86.8	1.4	1.5

7.5 Summary

The release, routing, and sequencing problems are modeled as the Markov decision processes 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 is introduced to solve the problem. It explores the state space by the simulation. Each simulation run is one iteration. The data-driven models are updated and improved gradually after each iteration. We compare the approach with other approaches; the results show that it performs unstable. Sometimes it performs better than the STTD method; occasionally it performs even worse than a simple rule. Though there are still lots of works to do on this approach, the study already shows the possibility of using the reinforcement learning to solve the problems.

8 Conclusions and Perspectives

8.1 Conclusions

The release, routing, and sequencing problems are sequential decision-making problems. The decision-making is the selection of the best alternative from possible alternatives. In other words, it is a priority calculation problem. For each alternative, a priority value is computed. The alternative with the highest priority is selected. We proposed three approaches to calculate the priority values of alternatives, including the simulation try-then-decide method (STTD), the intelligent method based on the simulation try-then-decide method (INT1), and the intelligent method based on Markov decision process (INT2).

Because the methods highly depend on the simulation, we developed an agent-based simulator for the material flow. To speed up ABS, we introduced worldviews from the discrete event simulation into the ABS. Compared to other worldviews, the process-interaction worldview is more natural and closer to the mental model ,and we took it into the ABS. The result from an application to the queuing system $M / M^r / 1$ shows the validity of the proposed approach. It performs more efficiently than the real-time ABS (timescale). The ABM and the process-interaction worldview are inextricably linked. The flexibility, maintainability, and modifiability of the ABM are also enhanced in this way. A framework for the ABS with the process-interaction worldview is developed for further uses.

Within the framework, an agent-based model of the material flow including release agents, machine group agents, and job agents is built. The agent-based model, the data collector, and the time manager make up the simulator for the material flow. In the model, the release agents make release decisions according to the release policy, the routing and sequencing decisions are made by the machine and machine group agents according to the priority rules. It is also very easy to replace these rules with the decision makers we created later in our three methods. Thus, the simulation can also evaluate our methods. Applying the simulator to a wafer FAB model, we made a good release policy for the FAB according to the experimental results.

Based on the simulator, the STTD method, a pure simulation approach, is proposed. It uses the alternative simulation to predict the future after an alternative is taken and select alternatives according to the future information from the simulation. The most important innovation is the usage of the base-rule in the alternative simulation. The base-rule avoids the exponential explosion of the number of the alternative simulations. To evaluate the STTD method, we replace the material flow 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. At last, we compare the STTD method with many decision rules; and it always performs best in the release, routing, and sequencing problems.

The INT1 method combines the experiences & data approach with the simulation approach. A data-driven model is introduced to calculate the priority values for alternatives. 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 alternative: global factors and local factors. The states of the material flow are 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 function 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 function pool. The neural network will calculate the priority for each alternative according to the local factors. Compared to other approaches, the approach always performs better than the decision rules. Even though it performs not as well as the STTD method, it spends just very short time. Therefore, it can be used in the system which has high requirements on the efficiency.

The INT2 method combines the experiences & data approach, mathematical approach, and simulation approach. It models the release, routing, and 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. From the comparison with other approaches, the results show that this approach performs unsteadily. Occasionally it performs worse than a simple rule, but sometimes it can perform even better than the STTD method. Thus it is still well worth doing more works on this approach.

Furthermore, these three methods are the decentralized decision-making approaches. For each machine, we created a sequencing decision maker to deal with the machine' s sequencing problems. It can remove the influences of the machines' position in the shop' s layout on the decision-making process. We also create a release decision maker for each type of product to decide if we release a job while the workload changed. It simplifies the decision-making process. If we create only one decision maker for all products, it has to decide which type of product to release besides whether or not to release. For the STTD method, for each job, there is a routing decision maker who is in charge of making routing decisions. For the other two methods, a routing decision maker connecting to a group of machines is created and responsible for the routing decisions.

All in all, it is believed that the study and its findings provide a solid foundation for the research community in developing further research activities in this field.

8.2 Perspectives

As this research field is relatively young, the study has thrown up many questions in need of further investigation.

- The initial idea is to study the release, routing, and sequencing problems together to achieve an even better performance of the material flows. However, from some other experiments, we realized that the performance is even worse when they are carried out together (These experiments are not reported in the dissertation) comparing to the cases in which these three problems are solved separately. In the future works, the reason must be figured out.
- Once the above issue is settled, more details of the material flows should involve in the study, such as the workforces and the transportation. Consequently, more real-time decision-making problems should be studied together, such as workforce scheduling, inventory control, and so on.
- For the simulation, because all three methods in the study are based on it, further emphasis might be put on the development of a general framework for the simulation to serve and evaluate the decision-making. It is also worth having a model of a real material flow system for further validations and evaluations.
- For the STTD method, it runs very slowly now. In the future work, the alternative simulations can run on distributed machines in parallel. Moreover, there is only one level alternative simulation now, i.e., in the alternative simulation, the decisions are made by the base-rules. We can try two level alternative simulations to reduce the influence of the base-rule. In the first level, the decisions are made by the second level simulations in which the decisions are made according to the base-rule. Also, we need to point out that the method is evaluated by the environment simulation which uses the same model as the alternative simulation model. However, if the method is tested in the real system, the alternative simulation model may not be the same as the real system. The performance may not be good anymore. Thus, in the future test, the environment simulation model should be a little bit different from the alternative simulation model.
- For the INT1 method, the most important thing is the data reorganization, especially the state of the environment. We should extract more meaningful features from the global factors to cluster the environment patterns. Because lots of machine learning techniques are used in the method, we should find an even better way to optimize the parameters of these techniques or try other similar techniques. After the data preparation, the method has three steps: dimension

reduction, clustering, and learning. We need to examine how do these three steps affect each other, for example, how the number of the reduced dimensions affects the clustering and how the number of the clusters affects the learning.

- For the INT2 method, the problem we are trying to solve now is too complicated. It is very difficult to analyze the details of the method using such a complicated case. It would be helpful, for example, for future researchers to examine some elementary cases. This might enable us to derive some laws for the complicated cases. For example, we can create a sequencing problem on a single machine. Then we can analyze the influences of the state definition and reward calculation. We can also use a two-machine-sequencing problem to analyze the multi-agent MDP (two agents) to see how their learning procedures interact and if the separate learning is better than the learning together. At last, the knowledge we learned from the simple cases can be applied in the complicated cases.

The ultimate goal of the study is to apply the solution on a real shop floor and integrate our system with the manufacturing execution system(MES). Each entity, like one piece of material, one machine, and so on, will make their own decisions. The study will be the most critical part of the smart factories.

At last, it needs to be noted that the proposed three methods are quite general and can solve not only the material flow control problem but also some other sequential decision-making processes which can be easily modeled and emulated.

Appendix Code Samples

A.1 Agent-based Simulation with process-interaction worldview

Below is a code sample which gives a very brief idea how to use the framework for a single group ABS.

```
using AgentBasedSimFramwork;
public static void main (string[]   args){
   AgentEnvironment ae=new AgentEnvironment();
   AgentManager am=new AgentManager(ae);
   TimeManager sim=new TimeManager();
   am.register(sim);
   …
   CustomAgent ca_1 =new CustomAgent (“CA_1”,
null);
   am.register(ca_1);
   CustomAgent ca_2 =new CustomAgent (“CA_2”,
null);
   am.register(ca_2);
   …
   Activation firstAct=new Activation(0,” CA_1”
,””);
   sim.initialize (firstAct);
   sim.start () ;

}
```

```
public class CustomAgent extends  Agent{
    public CustomAgent(string name, Object[]
args){
      super.Agent(name,args);
      addBehaviour(new SimBehaviour());
    …
    }
    …//add other behaviors
    private class SimBehaviour extends  Behavior{
        public void action()      {
              Message msg=receive();
              if(msg==null) return;
                  //activation code here
                  //send new activation point
                   sendActivationPoint(new
Actiation(…));
              return;}
        }
}
```

A.2 Alternative Evaluation in the STTD Method

The following function shows how to evaluate one alternative by the simulation.

```
double evaluate (Environment envy, Alternative alt)
{
       Model mod= Model.fromFile(“ExampleShop.xml”);
```

```
        mod.setBaseRule(Rule.SPT);
        mod.init(env);
        Simulation sim=new Simulation();
        sim.loadModel(mod);
        sim.setEndCondition(EndCondition.JobNumber);
        sim.setFirstEvent(alt.createEvent());
        sim.setRunTimes(50);
        sim.start();
        Evaluation eva=new Evaluation();
        eva.setPriorityEquation(…);
        double v= eva.getPriority(sim.getResults());
        return v;
 }
```

References

Abou-Ali, M. G., and M. A. Shouman. 2004. "Effect of Dynamic and Static Dispatching Strategies on Dynamically Planned and Unplanned Fms." *Journal of Materials Processing Technology* 148 (1):132-138.

Abtahi, F., and M. R. Meybodi. 2008. "Solving Multi-Agent Markov Decision Processes Using Learning Automata." In *Proceedings of the Intelligent Systems and Informatics, 2008. SISY 2008. 6th International Symposium on*, edited by, 1-6: IEEE.

Akhavan-Tabatabaei, R., and C. F. R. Salazar. 2011. "Effective Wip Dependent Lot Release Policies: A Discrete Event Simulation Approach." In *Proceedings of the Simulation Conference (WSC), Proceedings of the 2011 Winter*, edited by, 1971-1980.

Aloulou, M., and M.-C. Portmann. 2005. "An Efficient Proactive-Reactive Scheduling Approach to Hedge against Shop Floor Disturbances." In *Multidisciplinary Scheduling: Theory and Applications*, edited by Graham Kendall, EdmundK Burke, Sanja Petrovic and Michel Gendreau, 223-246. Springer US.

Baker, A. D. 1998. "A Survey of Factory Control Algorithms That Can Be Implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling, and Pull." *Journal of Manufacturing Systems* 17 (4):297-320.

Baker, K. R. 1974. *Introduction to Sequencing and Scheduling*: Wiley.

Ball, G. H., and D. J. Hall. 1965. Isodata, a Novel Method of Data Analysis and Pattern Classification. Stanford research inst Menlo Park CA.

Banks, J., and J. S. Carson. 1985. "Process-Interaction Simulation Languages." *SIMULATION* 44 (5):225-234.

Baykasoğlu, A., and L. Özbakır. 2010. "Analyzing the Effect of Dispatching Rules on the Scheduling Performance through Grammar Based Flexible Scheduling System." *International Journal of Production Economics* 124 (2):369-381.

Beck, J. C., and N. Wilson. 2007. "Proactive Algorithms for Job Shop Scheduling with Probabilistic Durations." *J. Artif. Int. Res.* 28 (1):183-232.

Belkin, M., and P. Niyogi. 2002. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering." In *Proceedings of the Advances in neural information processing systems*, edited by, 585-591.

Belkin, M., and P. Niyogi. 2003. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural computation* 15 (6):1373-1396.

Bonabeau, E. 2002. "Agent-Based Modeling: Methods and Techniques for Simulating Human Systems." *Proceedings of the National Academy of Sciences of the United States of America* 99 (Suppl 3):7280-7287.

Botta, N., A. Mandel, C. Ionescu, M. Hofmann, D. Lincke, S. Schupp, and C. Jaeger. 2011. "A Functional Framework for Agent-Based Models of Exchange." *Applied Mathematics and Computation* 218 (8):4025-4040.

Busch, J., J. Oldenburg, M. Santos, A. Cruse, and W. Marquardt. 2007. "Dynamic Predictive Scheduling of Operational Strategies for Continuous Processes Using Mixed-Logic Dynamic Optimization." *Computers & Chemical Engineering* 31 (5–6):574-587.

Cao, Y., and X.-h. Chen. 2012. "An Agent-Based Simulation Model of Enterprises Financial Distress for the Enterprise of Different Life Cycle Stage." *Simulation Modelling Practice and Theory* 20 (1):70-88.

Chen, B., and T. I. Matis. 2013. "A Flexible Dispatching Rule for Minimizing Tardiness in Job Shop Scheduling." *International Journal of Production Economics* 141 (1):360-365.

Chen, L. 2012. "Agent-Based Modeling in Urban and Architectural Research: A Brief Literature Review." *Frontiers of Architectural Research* 1 (2):166-177.

Chen, T. 2011. "A Self-Adaptive Agent-Based Fuzzy-Neural Scheduling System for a Wafer Fabrication Factory." *Expert Systems with Applications* 38 (6):7158-7168.

Chiang, T.-C., and L.-C. Fu. 2012. "Rule-Based Scheduling in Wafer Fabrication with Due Date-Based Objectives." *Computers & Operations Research* 39 (11):2820-2835.

Csáji, B. C., and L. Monostori. 2008. "Adaptive Stochastic Resource Control: A Machine Learning Approach." *J. Artif. Intell. Res.(JAIR)* 32:453-486.

Davidsson, P. 2001. "Multi Agent Based Simulation: Beyond Social Simulation." Proceedings of the Second International Workshop on Multi-Agent-Based Simulation-Revised and Additional Papers.

Diday, E. 1973. "The Dynamic Clusters Method in Nonhierarchical Clustering." *International Journal of Computer & Information Sciences* 2 (1):61-88.

El-Bouri, A. 2012. "A Cooperative Dispatching Approach for Minimizing Mean Tardiness in a Dynamic Flowshop." *Computers & Operations Research* 39 (7):1305-1314.

Fishman, G. S. 1978. "Principles of Discrete Event Simulation.[Book Review]."

Fortino, G., and W. Russo. 2012. "Eldameth: An Agent-Oriented Methodology for Simulation-Based Prototyping of Distributed Agent Systems." *Information and Software Technology* 54 (6):608-624.

Fowler, J., and J. Robinson. 1995. "Measurement and Improvement of Manufacturing Capacity (Mimac) Designed Experiment Report." *SEMATECH Technology transfer*

Framinan, J. M., P. L. González, and R. Ruiz-Usano. 2006. "Dynamic Card Controlling in a Conwip System." *International Journal of Production Economics* 99 (1–2):102-116.

François-Lavet, V., R. Fonteneau, and D. Ernst. 2015. "How to Discount Deep Reinforcement Learning: Towards New Dynamic Strategies." *arXiv preprint arXiv:1512.02011*

Franta, W. R., and K. Maly. 1977. "An Efficient Data Structure for the Simulation Event Set." *Commun. ACM* 20 (8):596-602.

Gabel, T., and M. Riedmiller. 2007a. " Adaptive Reactive Job-Shop Scheduling with Learning Agents." In *Proceedings of the International Journal of Information Technology and Intelligent Computing*, edited by: Citeseer.

Gabel, T., and M. Riedmiller. 2007b. "Scaling Adaptive Agent-Based Reactive Job-Shop Scheduling to Large-Scale Problems." In *Proceedings of the Computational Intelligence in Scheduling, 2007. SCIS'07. IEEE Symposium on*, edited by, 259-266: IEEE.

Gabel, T., and M. Riedmiller. 2008a. "Adaptive Reactive Job-Shop Scheduling with Reinforcement Learning Agents." *International Journal of Information Technology and Intelligent Computing* 24 (4)

Gabel, T., and M. Riedmiller. 2008b. "Joint Equilibrium Policy Search for Multi-Agent Scheduling Problems." In *Proceedings of the German Conference on Multiagent System Technologies*, edited by, 61-72: Springer.

Gabel, T., and M. Riedmiller. 2012. "Distributed Policy Search Reinforcement Learning for Job-Shop Scheduling Tasks." *International Journal of production research* 50 (1):41-61.

García, C., P. F. Cárdenas, L. J. Puglisi, and R. Saltaren. 2012. "Design and Modeling of the Multi-Agent Robotic System: Smart." *Robotics and Autonomous Systems* 60 (2):143-153.

Gelenbe, E., and G. Pujolle. 1987. *Introduction to Queueing Networks*. Edited by a. d vols. Vol. c, *B*. New York: John Wiley &Sons.

Gordon, G. J. 1999. "Approximate Solutions to Markov Decision Processes." *Robotics Institute*:228.

Graves, R. J., J. M. Konopka, and R. J. Milne. 1995. "Literature Review of Material Flow Control Mechanisms." *Production Planning & Control* 6 (5):395-403.

Grimm, V., U. Berger, and F. Bastiansen. 2006. "A Standard Protocol for Describing Individual-Based and Agent-Based Models." *Ecological Modelling* 198 (1–2):115-126.

Hagan, M. T., and M. B. Menhaj. 1994. "Training Feedforward Networks with the Marquardt Algorithm." *IEEE Transactions on Neural Networks* 5 (6):989-993.

Haykin, S. S. 2009. *Neural Networks and Learning Machines*. 3rd ed. New York: Pearson Prentice Hall.

Hecht-Nielsen, R. 1988. "Theory of the Backpropagation Neural Network." *Neural Networks* 1 (Supplement-1):445-448.

Ibaraki, T., and N. Katoh. 1988. *Resource Allocation Problems: Algorithmic Approaches*: Mit Press.

Jacobs, R. A. 1988. "Increased Rates of Convergence through Learning Rate Adaptation." *Neural networks* 1 (4):295-307.

Jain, A. K. 2010. "Data Clustering: 50 Years Beyond K-Means." *Pattern recognition letters* 31 (8):651-666.

Jayamohan, M. S., and C. Rajendran. 2004. "Development and Analysis of Cost-Based Dispatching Rules for Job Shop Scheduling." *European Journal of Operational Research* 157 (2):307-321.

Jolliffe, I. T. 1986. "Principal Component Analysis and Factor Analysis." In *Principal Component Analysis*, 115-128. Springer.

King, B. 1967. "Step-Wise Clustering Procedures." *Journal of the American Statistical Association* 62 (317):86-101.

Korytkowski, P., T. Wiśniewski, and S. Rymaszewski. 2013. "An Evolutionary Simulation-Based Optimization Approach for Dispatching Scheduling." *Simulation Modelling Practice and Theory* 35 (0):69-85.

Kumar, S. A., and S. N. Kumar S. Anil. 2009. *Operations Management*: New Age International (P) Limited.

Land, M. J., and G. J. Gaalman. 1998. "The Performance of Workload Control Concepts in Job Shops: Improving the Release Method." *International Journal of Production Economics* 56:347-364.

Lee, K. K. 2008. ”Fuzzy Rule Generation for Adaptive Scheduling in a Dynamic Manufacturing Environment.” *Applied Soft Computing* 8 (4):1295-1304.

Leitão, P. 2009. ”Agent-Based Distributed Manufacturing Control: A State-of-the-Art Survey.” *Engineering Applications of Artificial Intelligence* 22 (7):979-991.

Lian, J., S. M. Shatz, and X. He. 2009. ”Flexible Coordinator Design for Modeling Resource Sharing in Multi-Agent Systems.” *Journal of Systems and Software* 82 (10):1709-1729.

Liao, T., T. Stützle, M. A. Montes de Oca, and M. Dorigo. 2014. ”A Unified Ant Colony Optimization Algorithm for Continuous Optimization.” *European Journal of Operational Research* 234 (3):597-609.

Lin, Y.-H., C.-C. Chiu, and C.-H. Tsai. 2008. ”The Study of Applying Anp Model to Assess Dispatching Rules for Wafer Fabrication.” *Expert Systems with Applications* 34 (3):2148-2163.

Littman, M. L. 1996. ”Algorithms for Sequential Decision Making.”

Liu, J.-S., and K. Sycara. 1997. ”Coordination of Multiple Agents for Production Management.” *Annals of Operations Research* 75 (0):235-289.

Macal, C. M., and M. J. North. 2010. ”Tutorial on Agent-Based Modelling and Simulation.” *J of Sim* 4 (3):151-162.

MacQueen, J. 1967. ”Some Methods for Classification and Analysis of Multivariate Observations.” In *Proceedings of the Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, edited by, 281-297: Oakland, CA, USA.

Madureira, A. 2005. ”Multi-Agent System for Dynamic Manufacturing Scheduling Using Meta-Heuristics.” Proceedings of the 4th WSEAS International Conference on Applied Mathematics and Computer Science, Rio de Janeiro, Brazil.

Mahmoodi, F., K. J. Dooley, and P. J. Starr. 1990. ”An Evaluation of Order Releasing and Due Date Assignment Heuristics in a Cellular Manufacturing System.” *Journal of Operations Management* 9 (4):548-573.

McLane, A. J., C. Semeniuk, G. J. McDermid, and D. J. Marceau. 2011. ”The Role of Agent-Based Models in Wildlife Ecology and Management.” *Ecological Modelling* 222 (8):1544-1556.

Mika, S., B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. 1999. ”Kernel Pca and De-Noising in Feature Spaces.” In *Proceedings of the Advances in neural information processing systems*, edited by, 536-542.

Miller, J. A., G. T. Baramidze, A. P. Sheth, and P. A. Fishwick. 2004. "Investigating Ontologies for Simulation Modeling." Proceedings of the 37th annual symposium on Simulation.

Miyashita, K. 1998. "Camps: A Constraint-Based Architecturefor Multiagent Planning and Scheduling." *Journal of Intelligent Manufacturing* 9 (2):147-154.

Mouelhi-Chibani, W., and H. Pierreval. 2010. "Training a Neural Network to Select Dispatching Rules in Real Time." *Computers & Industrial Engineering* 58 (2):249-256.

Murtagh, F. 1983. "A Survey of Recent Advances in Hierarchical Clustering Algorithms." *The Computer Journal* 26 (4):354-359.

Namatame, A. 2007. "Collective Intelligence of Networked Agents." In *Emergent Intelligence of Networked Agents*, edited by Akira Namatame, Satoshi Kurihara and Hideyuki Nakashima, 159-176. Berlin Heidelberg: Springer

Olafsson, S., and X. Li. 2010. "Learning Effective New Single Machine Dispatching Rules from Optimal Scheduling Data." *International Journal of Production Economics* 128 (1):118-126.

Oliva, G., S. Panzieri, and R. Setola. 2010. "Agent-Based Input–Output Interdependency Model." *International Journal of Critical Infrastructure Protection* 3 (2):76-82.

Pawlaszczyk, D., and I. J. Timm. 2007. "A Hybrid Time Management Approach to Agent-Based Simulation." Proceedings of the 29th annual German conference on Artificial intelligence, Bremen, Germany.

Petroni, A., and A. Rizzi. 2002. "A Fuzzy Logic Based Methodology to Rank Shop Floor Dispatching Rules." *International Journal of Production Economics* 76 (1):99-108.

Phansalkar, V., and P. Sastry. 1994. "Analysis of the Back-Propagation Algorithm with Momentum." *IEEE Transactions on Neural Networks* 5 (3):505-506.

Pickardt, C. W., T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter. 2013. "Evolutionary Generation of Dispatching Rule Sets for Complex Dynamic Scheduling Problems." *International Journal of Production Economics* 145 (1):67-77.

Pinedo, M. 2012. *Scheduling*: Springer.

Piplani, R., and D. Wetjens. 2007. "Evaluation of Entropy-Based Dispatching in Flexible Manufacturing Systems." *European Journal of Operational Research* 176 (1):317-331.

Puterman, M. L. 2014. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*: Wiley.

Rebaudo, F., and O. Dangles. "An Agent-Based Modeling Framework for Integrated Pest Management Dissemination Programs." *Environmental Modelling &amp; Software* (0)

Riedmiller, M. 2005. "Neural Fitted Q Iteration-First Experiences with a Data Efficient Neural Reinforcement Learning Method." In *Proceedings of the ECML*, edited by, 317-328: Springer.

Rigler, A., J. Irvine, and T. P. Vogl. 1991. "Rescaling of Variables in Back Propagation Learning." *Neural Networks* 4 (2):225-229.

Rossi, R., and H. Lödding. 2012. *Handbook of Manufacturing Control: Fundamentals, Description, Configuration*: Springer.

Roweis, S. T., and L. K. Saul. 2000. "Nonlinear Dimensionality Reduction by Locally Linear Embedding." *science* 290 (5500):2323-2326.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1988. "Learning Representations by Back-Propagating Errors." *Cognitive modeling* 5 (3):1.

Rummery, G. A., and M. Niranjan. 1994. *On-Line Q-Learning Using Connectionist Systems*. Vol. 37: University of Cambridge, Department of Engineering.

Ryan, S. M., and J. Vorasayan. 2005. "Allocating Work in Process in a Multiple-Product Conwip System with Lost Sales." *International Journal of Production Research* 43 (2):223-246.

Sabuncuoglu, I., and M. Bayız. 2000. "Analysis of Reactive Scheduling Problems in a Job Shop Environment." *European Journal of Operational Research* 126 (3):567-586.

Shahzad, A., and N. Mebarki. 2012. "Data Mining Based Job Dispatching Using Hybrid Simulation-Optimization Approach for Shop Scheduling Problem." *Engineering Applications of Artificial Intelligence* 25 (6):1173-1181.

Shannon, R. E. 1975. *Systems Simulation: The Art and Science*: Prentice-Hall.

Shiue, Y.-R., R.-S. Guh, and K.-C. Lee. 2011. "Study of Som-Based Intelligent Multi-Controller for Real-Time Scheduling." *Applied Soft Computing* 11 (8):4569-4580.

Siebers, P. O., C. M. Macal, J. Garnett, D. Buxton, and M. Pidd. 2010. "Discrete-Event Simulation Is Dead, Long Live Agent-Based Simulation!" *J of Sim* 4 (3):204-210.

Smola, A. J., and B. Schölkopf. 1998. *Learning with Kernels*: GMD-Forschungszentrum Informationstechnik.

Sneath, P. H., and R. R. Sokal. 1973. *Numerical Taxonomy. The Principles and Practice of Numerical Classification.*

StudyMode.com. 2013. "Process Flow." Accessed July 16. http://www.studymode.com/essays/Process-Flow-1925050.html.

Sugumaran, V. 2008. *Distributed Artificial Intelligence, Agent Technology, and Collaborative Applications*: Information Science Reference.

Sutton, R. S., and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge, London: MIT Press.

Tapia, D. I., J. A. Fraile, S. Rodríguez, R. S. Alonso, and J. M. Corchado. "Integrating Hardware Agents into an Enhanced Multi-Agent Architecture for Ambient Intelligence Systems." *Information Sciences* (0)

Tay, J. C., and N. B. Ho. 2008. "Evolving Dispatching Rules Using Genetic Programming for Solving Multi-Objective Flexible Job-Shop Problems." *Computers & Industrial Engineering* 54 (3):453-473.

Tenenbaum, J. B. 1998. "Mapping a Manifold of Perceptual Observations." In *Proceedings of the Advances in neural information processing systems*, edited by, 682-688.

Tersine, R. J., and R. J. Tersine. 1994. "Principles of Inventory and Materials Management."

Tesauro, G. 1992. "Practical Issues in Temporal Difference Learning." In *Proceedings of the Advances in neural information processing systems*, edited by, 259-266.

Tesauro, G. 1995. "Temporal Difference Learning and Td-Gammon." *Communications of the ACM* 38 (3):58-68.

Tollenaere, T. 1990. "Supersab: Fast Adaptive Back Propagation with Good Scaling Properties." *Neural networks* 3 (5):561-573.

Tsitsiklis, J. N., and B. Van Roy. 1997. "Analysis of Temporal-Diffference Learning with Function Approximation." In *Proceedings of the Advances in neural information processing systems*, edited by, 1075-1081.

Tzafestas, S., and A. Triantafyllakis. 1994. "A New Adaptively Weighted Combinatorial Dispatching Rule for Complex Scheduling Problems." *Computer Integrated Manufacturing Systems* 7 (1):7-15.

Valente, J. M. S., and J. E. Schaller. 2012. "Dispatching Heuristics for the Single Machine Weighted Quadratic Tardiness Scheduling Problem." *Computers & Operations Research* 39 (9):2223-2231.

Váncza, J., and A. Márkus. 2000. "An Agent Model for Incentive-Based Production Scheduling." *Computers in Industry* 43 (2):173-187.

Wagner, G. 2004. "Aor Modelling and Simulation: Towards a General Architecture for Agent-Based Discrete Event Simulation." In *Agent-Oriented Information*

Systems, edited by Paolo Giorgini, Brian Henderson-Sellers and Michael Winikoff, 174-188. Springer Berlin / Heidelberg.

Wang, Y.-C., and J. M. Usher. 2004. "Learning Policies for Single Machine Job Dispatching." *Robotics and Computer-Integrated Manufacturing* 20 (6):553-562.

Ward Jr, J. H. 1963. "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American statistical association* 58 (301):236-244.

Watkins, C. J., and P. Dayan. 1992. "Q-Learning." *Machine learning* 8 (3-4):279-292.

Wolfram, S. 1994. *Cellular Automata and Complexity*: Addison-Wesley Pub. Co.

Xanthopoulos, A. S., D. E. Koulouriotis, V. D. Tourassis, and D. M. Emiris. 2013. "Intelligent Controllers for Bi-Objective Dynamic Scheduling on a Single Machine with Sequence-Dependent Setups." *Applied Soft Computing* 13 (12):4704-4717.

Yang, B., and J. Geunes. 2008. "Predictive–Reactive Scheduling on a Single Resource with Uncertain Future Jobs." *European Journal of Operational Research* 189 (3):1267-1283.

Zeigler, B., H. Praehofer, and T. Kim. 2000. *Theory of Modeling and Simulation, Second Edition*: Academic Press.