12-13-2003

Application of Reinforcement Learning to Multi-Agent Production Scheduling

Yi-chi Wang

APPLICATION OF REINFORCEMENT LEARNING TO MULTI-AGENT PRODUCTION SCHEDULING

By

Yi-Chi Wang

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Industrial Engineering
in the Department of Industrial Engineering

Mississippi State, Mississippi

December 2003

APPLICATION OF REINFORCEMENT LEARNING TO MULTI-AGENT

PRODUCTION SCHEDULING

By

Yi-Chi Wang

Approved:

John M. Usher Professor of Industrial Engineering (Director of Dissertation)	Royce O. Bowden Professor of Industrial Engineering (Committee Member)
William N. Smyer Associate Professor of Industrial Engineering (Committee Member)	Steven R. Daniewicz Associate Professor of Mechanical Engineering (Committee Member)
A. Wayne Bennett Dean of the College of Engineering	Stanely F. Bullington Professor of Industrial Engineering Graduate Coordinator in the Department of Industrial Engineering (Committee Member)

Name: Yi-Chi Wang

Date of Degree: December 13, 2003

Institution: Mississippi State University

Major Field: Engineering (Industrial Engineering)

Major Professor: Dr. John M. Usher

Title of Study: APPLICATION OF REINFORCEMENT LEARNING TO MULTI-AGENT PRODUCTION SCHEDULING

Pages in Study: 114

Candidate for Degree of Doctor of Philosophy

Reinforcement learning (RL) has received attention in recent years from agent-based researchers because it can be applied to problems where autonomous agents learn to select proper actions for achieving their goals based on interactions with their environment. Each time an agent performs an action, the environment's response, as indicated by its new state, is used by the agent to reward or penalize its action. The agent's goal is to maximize the total amount of reward it receives over the long run. Although there have been several successful examples demonstrating the usefulness of RL, its application to manufacturing systems has not been fully explored. The objective of this research is to develop a set of guidelines for applying the Q-learning algorithm to enable an individual agent to develop a decision making policy for use in agent-based production scheduling applications such as dispatching rule selection and job routing.

For the dispatching rule selection problem, a single machine agent employs the Q-learning algorithm to develop a decision-making policy on selecting the appropriate dispatching rule from among three given dispatching rules. In the job routing problem, a simulated job shop system is used for examining the implementation of the Q-learning algorithm for use by job agents when making routing decisions in such an environment. Two factorial experiment designs for studying the settings used to apply Q-learning to the single machine dispatching rule selection problem and the job routing problem are carried out. 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.

DEDICATION

This dissertation is dedicated to my mother and the memory of my father.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor, Dr. John Usher, who has been a friend, a counselor, and an excellent role model of a hard worker, for his patience, guidance, and encouragement during my graduate program. Appreciation is also expressed to Dr. Stanely Bullington, Dr. Royce Bowden, Dr. William Smyer, and Dr. Steven Daniewicz for serving as my dissertation committee members and providing their invaluable time, advice, and guidance. I wish to thank my family for believing in me and supporting me for all these years. Without their encouragement and patience this endeavor would not have been possible.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1. Manufacturing Scheduling

A long and profitable life is every enterprise's goal. For a manufacturing enterprise, to maintain profitability requires that they continually excel in converting raw materials into value-added products that meet the customers' needs. This conversion procedure consists of a set of complicated and interrelated activities such as designing, planning, production, inventory control, quality assurance, etc. To remain competitive in the market, manufacturers must focus on continually improving their processes. Production scheduling that translates the detailed process plans into the shop floor schedule is one of the most important processes in manufacturing systems. A good production schedule can provide such benefits as increased shop throughput, enhanced customer satisfaction, lower inventory levels, and increased utilization of resources. Therefore, there is a great need for good scheduling strategies.

Scheduling problems essentially involve completing a set of jobs with a limited number of manufacturing resources under a number of constraints to optimize a particular objective function. These problems are known to be hard and usually belong to the NP-complete class of problems (Morton and Pentico, 1993; Pinedo, 1995). Research in production scheduling has been conducted for many decades and a large number of algorithms and heuristics have been developed for various scheduling problems. A

scheduling problem consists of three components: a machine environment, specific job characteristics, and one or more optimality criteria (Brucker, 2001). The machine environment represents the type of the manufacturing system that will execute the developed schedule. The manufacturing system may be a job shop system, flexible manufacturing system (FMS), cellular manufacturing system, transfer line, etc. Job characteristics represent such factors as the number of operations, the precedence relations among operations, and the possibility of preemption (whether the job can be split). Optimality criteria are the objectives to pursue when scheduling the jobs. Common objectives include minimizing makespan, mean flow time, mean lateness, the number of tardy jobs, and mean tardiness. All the three components mentioned above specify the variety and complexity of each scheduling problem.

A scheduling problem may be comprised of two sub-problems: job routing and job sequencing problems. A job routing problem involves assigning the operations of jobs to the specific machines. Such problems result from the allowance of routing flexibility. Routing flexibility depends on the capability of the machines. A versatile machine is capable of performing different operations. The versatility of the various machines in a shop essentially supports the possibility for the existence of alternative process plans for a job. Routing flexibility is a key issue that has increasingly attracted attention in modern manufacturing systems. A FMS, which consists of a set of computer numerically controlled machines (CNC) linked with an automated material handling system, is a computerized system that is able to produce mid-volume and mid-variety products with high levels of efficiency. The FMS provides routing flexibility due to the capability of NC machines. Once the route of a job is specified, decision makers must

determine the production sequence of the jobs awaiting their next process in the machine queue. A simple approach to such problems is to adopt dispatching rules. A dispatching rule is a priority rule used to determine the order in which the jobs waiting in the machine queue are to be processed as soon as a machine becomes available. Dispatching rules are useful for finding a reasonably good schedule. The dispatching rules are attractive because of their simplicity and ease of implementation. A variety of dispatching rules have been proposed in recent decades, with Panwalkar and Iskander (1977) identifying the existence of more than 100 distinct rules. Scheduling in industry may require meeting several objectives simultaneously. However, a dispatching rule often favors one performance measure only at the expense of other performance measures. In addition, the manufacturing environment usually changes over time. Therefore, the specific dispatching rule employed in such a dynamic environment should be free to change as well.

One of the most notoriously difficult systems for the scheduling community is the job shop system. The strategy of a job shop is based on producing a wide variety of products in very low volumes. Producing such variable products requires different sequences. In a traditional job shop layout, machines are functionally grouped together. For the case of an actual shop floor, uncertainties (i.e., machine breakdowns, material or tool shortages, transportation delays, etc.) complicate the scheduling problem making it more difficult to solve. Therefore, several assumptions are usually made to simplify the problem (i.e., resources are always available, all the jobs are known in advance, all the operation processing times are known and constant, transportation times are ignored, etc.). However, application of too many such assumptions may result in the treatment of

scheduling problems that would be considered unrealistic. That is why the job shop scheduling problems have attracted so much attention over many decades.

1.2. Agent-Based Approach

Due to the structural rigidity of classical centralized control architectures in manufacturing, the decentralized (or heterarchical) control structure has drawn more attention (Crowe and Stahlman, 1995; Dilts et al., 1991; Duffie and Prabhu, 1994). One of the most important properties of the heterarchical structure is that the decision-making responsibilities are fully distributed to each component of the system. Each component is autonomous and possesses local knowledge that is sufficient to accomplish its own task. The task that a single component is unable to finish alone may require the cooperation of a cluster of components. Communication is a means of establishing such cooperation between the autonomous components. Under the guidance of such a control architecture, the requirements of the next generation of manufacturing systems, such as good fault-tolerance, ease of reconfigurability and adaptability, and agility, can be achieved (Shaw and Norrie, 1999).

In recent years, a new paradigm called agent technology has been widely recognized as a promising paradigm for developing software applications able to support complex tasks. From the perspective of a software application, an agent can be viewed as a computational module that is able to act autonomously to achieve its goal (Weiss, 1999; Brenner et al., 1998; Shen et al., 2000). Wooldridge and Jennings defined an intelligent agent as a hardware or software-based computer system with the properties such as autonomy, social ability, reactivity and pro-activeness (Murch and Johnson, 1998). The idea of agent-based approaches has also offered a promising solution for controlling

future manufacturing systems requiring flexibility, reliability, adaptability, and reconfigurability. Agent technology fits naturally into the decentralized control structure for manufacturing systems because the autonomous component can easily be represented by an agent that is defined as an autonomous, pro-active element with the capability to communicate with other agents (Weiss, 1999). In fact, agents can be used to represent physical shop-floor components such as parts, machines, tools, and even human beings. Under the application of multi-agent systems, each agent is in charge of information collection, data storage, and decision-making for the corresponding shop floor component. A popular scheme to achieve cooperation among autonomous agents is through the negotiation-based contract-net protocol (Smith, 1980). The contract-net protocol provides the advantage of real-time information exchange, making it suitable for shop floor scheduling and control.

1.3. Reinforcement Learning

One significant issue for improving an autonomous agent's capability is that of how to enhance the agent's intelligence. Learning is one mechanism that could provide the ability for an agent to increase its intelligence while in operation. Developed in the early 1990s, reinforcement learning (RL) has generated a lot of interest from the research community. As opposed to the popular approach of supervised learning whereby an agent learns from examples provided by a knowledgeable external supervisor (Weiss, 1999), reinforcement learning requires that the agent learn by directly interacting with the system (its environment) and responding to the receipt of rewards or penalties based on the impact each action has on the system. Although there have been several RL applications demonstrating the usefulness of RL (Sutton and Barto, 1999; Mahadevan

and Kaelbing, 1996), its application to manufacturing systems has not been fully explored.

1.4. Problem Statement

This study proposed the use of an agent-based approach for handling a dynamic job-shop scheduling problem. Every customer order consists of a batch of identical parts with each part comprised of a set number of features defined by the customer. Each feature requires at least one operation. Routing flexibility is considered here by providing alternative processing routes to produce the same product. These alternatives are taken into account in the process plan and arise due to the availability of multiple machine types for processing a specific operation.

Two types of agents are used in the system: job agents and machine cell agents. Each job agent representing a specific job is in charge of determining proper operation routing by negotiating with the cell agents that have the potential to finish the operations. Each machine cell agent represents one machine cell that may be comprised of one (or more than one) identical machine. All the machines in the same cell share the same buffer. Each machine cell agent determines the next job (from the buffer) for processing when any machine in the cell is available. That is, the job agents are responsible for solving the routing problem, while the cell agents work out the sequencing problem.

In this study, job routes are dynamically determined through negotiation between job and machine cell agents. A contract net-based mechanism is implemented for agent negotiation. On the other hand, Dispatching rules (DR) are employed to solve the job sequencing problems. However, no single DR can be really dominant across all possible scenarios (Chiu and Yih, 1995; Kouiss et al., 1997; Pierrval and Mebarki, 1997;

Subramaniam et al., 2000). Employing an appropriate DR should depend on the real-time shop circumstances. Therefore, the sequencing problem in this study is actually becoming a DR selection problem. This research is concerned with investigating the application of a reinforcement learning (RL) approach proposed for training job agents to learn a good policy for dynamic making routing decisions and for training machine cell agents to learn a good policy for selecting an appropriate dispatching rule. To apply RL in this study, the following issues must be dealt with:

1. How to specify the states, actions, and penalties and rewards?
2. How do various state determination criteria affect learning performance?
3. How do the parameters of the RL approach impact learning performance?
4. How do various reward functions affect learning performance?

Currently, implementing multi-agent systems in dynamic scheduling is still a highly popular research area. Performance of the agent-based approaches not only relies on the cooperation among the agents but the capability of the agents. In this research, enhancing the agent's capability in terms of making good decisions will significantly benefit applying agent technology to complex dynamic scheduling problems.

1.5. Objective of the Research

The overall goal of this research is to develop a set of guidelines (or recommendations) for applying the Q-learning algorithm to enable an individual agent to develop a decision making policy for use in production scheduling applications such as dispatching rule selection and job routing. The focus of the study is specific to agent-based systems employed in dynamic job shop environments. Suresh and Chaudhuri (1993) surveyed the approaches for the dynamic scheduling problems and identified some

essential characteristics of a good scheduling system. According to their survey, a good scheduling system should be efficient in terms of meeting due dates and reducing cost, generate schedules using actual information from the current environment, and provide flexibility to react to disruptions in an efficient and timely manner. Agent-based approaches seem promising for building a good scheduling system. Currently, implementing multi-agent systems in dynamic scheduling is still one of the most active research areas. In this study, a multi-agent heterarchical system is developed for solving complex production scheduling problems.

Applications of RL techniques to manufacturing systems have not been thoroughly explored yet. The proposed study investigates how Q-learning algorithm can be used by job agents to construct policies for making real-time routing decisions and by machine agents to discover a policy for selecting a proper DR. At present, most of the agent-based research focuses on the issues of negotiation and cooperation among agents. Addressing learning in a multi-agent environment can help agents improve both their performance and that of the system as well (Shen et al. 2000). RL requires that the agent learn by directly interacting with its environment and receive rewards or penalties based on the impact each of its actions has on the system. Therefore, RL may provide an on-line learning capability for individual agents. The successful application of the Q-learning algorithm to agent-based scheduling problems in this research will provide researchers with additional knowledge on the application of RL techniques to agent-based manufacturing systems.

The next chapter provides a review of the literature that introduces related research work providing more details about manufacturing control structures, traditional

approaches and agent-based approaches to dynamic scheduling problems, and applications of reinforcement learning to manufacturing systems. The methodologies of this research will be described in Chapter 3 and Chapter 4. Chapter 5 consists of the experimental results and discussion. The conclusions are presented in Chapter 6.

CHAPTER II

LITERATURE REVIEW

2.1. Control Structures of Agent-Based Manufacturing Systems

The control architecture employed in manufacturing systems plays an important role in defining the interactions among the manufacturing components because it identifies the decision-making responsibilities of each system component. The earliest control architecture is the centralized structure. The characteristics of the centralized control architecture is that there exists only one central computer performing all the information processing functions and maintaining global databases to record all the activities of the system. The centralized control architecture simplifies optimization since it holds all global information in a single control unit. The overall system status can be obtained by accessing the single control unit. As well, communication overhead is low in such a system. These advantages of centralized control structures are tarnished because of a complete reliance on the fault tolerance of a single central computer. As the size of the manufacturing system grows and becomes more complicated, the speed of response may be degraded due to the limited capability of the central computer.

To resolve the deficiencies of the centralized structure, the load on the central computer must be distributed. One approach employs a hierarchical control structure consisting of a small number of layers (usually three to five). The upper-level layers have more authority and responsibility for decision-making than the lower-level layers. The

structure defines rigid master/slave relationships between components on one layer and those below and above and each component in the hierarchy is only able to only communicate with these components. Command information flows top-down, and feedback information flows bottom-up. All the components in the system are assumed to possess deterministic behavior.

The hierarchical control structure became popular in manufacturing starting in early 1980s and was supported by such efforts as that of NIST's AMRF (Jones and McLean, 1986). Although achieving global optimization may be possible with this type of control structure, such systems may not be sensitive to the unexpected events (e.g., machine breakdown, rush orders, etc.) in the manufacturing environment because information exchanges between system components are not very efficient. For example, the information of each lower-level component must pass through an upper-level controller to reach another lower-level component. In addition, use of the rigid hierarchical structure makes it difficult to modify or extend the existing system. Therefore, the hierarchical control structure is unable to handle the expansion and frequent reconfiguration needs required of future manufacturing systems (Maturana et al., 1999).

In order to overcome the weaknesses of the hierarchical architecture, a heterarchical (decentralized) control approach has been recommended for future manufacturing systems (Duffie and Prabhu, 1994). It is a completely decentralized structure containing no supervisor level where the decision making responsibilities are fully distributed to each component of the system. Each component is autonomous and possesses sufficient local knowledge to accomplish its own task. A task that a single

component is unable to finish alone may require the cooperation of a cluster of components. Communication is the key for achieving cooperation between the autonomous components.

Crowe and Stahlman (1995) point out that the overall system complexity and supervisory costs can be reduced when using heterarchical control structures. They also state that system maintenance and modification is simplified for such systems compared to hierarchical control. Okubo et al. (2000) compared the abilities of distributed and centralized production control systems on response time, planning scope, and progressive accuracy. Progressive accuracy is the difference between the prescribed plan and the results from actual production. The larger the differences between estimated and actual processing times are, the longer the lead time will be. Okubo et al. found that a decentralized system allows a larger gap (poorer accuracy) than a centralized system. Their simulation results showed that a distributed control system enables a shorter response time, narrower planning scope, and higher progressive accuracy than a centralized control system. However, when the system is under a heavy load the centralized control system provides shorter lead-times than decentralized control because the centralized system controls the WIP level with a more global perspective . One of the major inherent defects of the heterarchical control structure is poor global optimization (Dilts et al., 1991). This problem results from the high autonomy of the individual components that do not possess a global perspective. Resolving this defect requires a robust mechanism to support cooperation between the autonomous components. It is believed that the benefits that a decentralized control architecture provides include fault-tolerance, ease of reconfigurability and adaptability, and local autonomy, and thereby,

fulfill the requirements of future manufacturing systems (Dilts et al., 1991; Shen and Norrie, 1999). Table 2.1 provides a summary of control architrctures.

Table 2.1. Summary of Control architectures

Architecture	Features	Advantages	Disadvantages
Centralized	• Single control unit	• Global optimization • Easy to access to global information	• Heavy load on the central control unit • Poor fault-tolerance
Hierarchical	• Master/slave relationship • Commands flow top-down. • Feedbacks flow bottom-up	• Possible global optimization • Good predictability	• Poor scalability • Poor reconfigurability • Poor adaptability • Heavier load for higher level components
Heterarchical	• Local Autonomy • Peer to peer communication • Cooperation	• Reduced complexity • Good fault-tolerance • Good scalability • Good reconfigurability • Good adaptability	• Poor global optimization • Poor predictability

2.2. Dynamic Job Shop Scheduling Problems

As was introduced in the previous chapter, the variety, complexity, and scope of a scheduling problem is determined by the machine environment, specific job characteristics, and performance criteria. A review of dynamic job shop scheduling problems reveals that a variety of problem assumptions have been employed in the various research studies. Therefore, it is impossible to directly compare the strategies for these scheduling problems. In general, manufacturing scheduling problems can be classified into routing problems and sequencing problems. The next two sub-sections provide a review of the literature for these two problems.

2.2.1 Job Routing problems

In the context of this study, a job is considered to be a job order consisting of a batch of identical discrete engineered parts. Each part requires the service of one or more machines in order to complete the processing necessary to satisfy the order. A job routing problem results from the allowance of flexibility in the routing of a job through the shop. Routing flexibility of a manufacturing system can be defined as the ability to manufacture a product by alternative routes (Das, 1996). Lin and Solberg (1991) identified four types of routing flexibility based on the availability of alternative machines for an operation, alternative operations for a feature, and alternative operations sequences for a job. For the case of no routing flexibility, a job is completed using a fixed sequence of operations and each operation must be processed on a specific machine. There are no alternative machines capable of performing the same operation. For the fixed sequencing type, the operations of a job must be performed in a fixed sequence, but there can be more than one machine capable of processing any given operation. This case is extended in third type, flexible sequencing, where alternative sequences of the operations are permitted. The last type is flexibly processing where alternative sequences are permitted whereby alternative operations may be available for machining each feature and alternative machines employed to perform the selected operation. The comparison of these four types of routing flexibility is shown in Table 2.2.

Lin and Solberg (1991) compared different cases of these four types of routing flexibility and concluded that the flexible processing case is always superior to the other three cases. Chan (2001) used Taguchi experimental design techniques to study the effects of different levels of routing flexibility on the performance of a FMS. In his study,

routing flexibility is defined as a measure of the average number of choices of a machine that an individual part can choose. He found that increasing routing flexibility doesn't guarantee an improvement in system performance. Chan concluded routing flexibility with a measure of 2 (meaning that on average, each job has two options of which machine to use for its next operation) provided the best system performance under the measures of makespan and flow time.

Table 2.2. Types of Routing Flexibility

	No Flexibility	Fixed Sequencing	Flexible Sequencing	Flexible Processing
Alternative M/C for an operation	No	Yes	Yes	Yes
Alternative operation for a feature	No	No	No	Yes
Operation sequence of a job	Fixed	Fixed	Flexible	Flexible

2.2.1.1 Heuristics

Choi and Malstrom (1988) evaluated the performance of traditional scheduling rules using a simulation of an FMS system constructed using data from a real FMS. The rules evaluated consist of seven job dispatching rules and four machine selection rules creating a total of 28 combinations. Each combination was evaluated by six performance criteria. Their simulation results indicated that the WINQ (the least work in queue in terms of processing time) was the best machine selection rule.

Ro and Kim (1990) proposed three machine selection heuristics (ARD, ARP, and ARPD). The ARD rule is a rule to select the machine that has the shortest time composed of a sum of travel time, queuing time, and processing time. Use of the ARP rule requires that routes be determined by a linear programming (LP) model whose objective is to minimize makespan. Implementation of the ARP rule requires that the LP model be

solved whenever a new job arrival or a machine breakdowns. The ARPD rule is a combination of ARD and ARP. Initially, the routes are determined by solving the LP model, but if the primary machine (from LP solution) is busy, a machine is selected based on the ARD rule. Ro and Kim compared their three heuristics with two other heuristics (NAR and WINQ). The NAR is a rule to select the route with the minimum total processing time (no alternative routes are permitted). From their simulation results, ARD gave the best results in four performance measures (makespan, mean flow time, mean tardiness, and maximum tardiness) except for system utilization. They also found that ARD, APRD, and WINQ were significantly better than ARP and NAR in every performance measure.

Yao and Pei (1990) proposed another definition for the measure of routing flexibility. Their measure of routing flexibility was called “entropy”. The entropy measure takes into account the number of all the immediate next operations, the alternative machines for each of these operations, and the reliability of these machines. Yao and Pei then proposed a heuristic approach called “least reduction in entropy” (LRE), which consists of a machine selection rule and a job selection rule on the basis of incurring the least reduction in entropy. They compared LRE with SPT using a simulated four-machine production system. Their results showed that LRE either outperforms or is as good as SPT in the measures of makespan and machine utilization.

Shmilovici and Maimon (1992) compared three routing heuristics, fixed priorities (FP), least reduction in entropy (LRE), and minimum flow resistance (MFR), and analyzed the computational complexity of these three heuristics. According to their experimental results, FP was easy to implement and required less computational effort,

LRE was not as effective as reported by Yao and Pei (1990), and MFR outperformed the other heuristics in terms of throughput but required more expense due to increased buffer size. They also found that controlling buffer size had a significant impact to the system throughput for any of the three heuristics.

Chandra and Talavage (1991) developed a heuristic dispatching system for FMS. In their system, a part after completing an operation is not routed to a specific machine, but is sent to a global buffer. The routing decisions are not made by the parts, but by the machines. Their dispatching mechanism categorizes and selects the jobs based on a pre-defined algorithm. The mechanism was also able to deal with a scheduling problem with multiple objectives. The authors compared their system to the four traditional dispatching rules (SPT, EDD, LSPO, LRS). Their dispatching system consistently outperformed those dispatching rules under various circumstances. They concluded that making decisions with simple commonsense reasoning combining some empirically proven dispatching rules could achieve a significant improvement.

Subramaniam et al. (2000a) proposed three route selection rules: LAC, LAP, and LACP. LAC selects the machine with the lowest average cost of processing every operation in the machine queue. For LAP machine selection is based on the lowest average processing time of every operation in the machine queue. LCAP awards the highest priority to the machine that has the minimum aggregate cost and processing time. Their results found that LAC and LAP rules perform well for the mean cost and mean tardiness performance measures, respectively, while the LACP rule exhibits performance that is between the LAC and LAP rules.

Among the above routing heuristics, WNIQ, ARD, LAC, and LAP are the approaches that are not only able to provide promising results, but also easy to implement in real time. Some of these approaches will be used as benchmarking approaches in this research.

2.2.1.2 Knowledge-Based System

Bowden and Bullington (1996) developed a machine learning system called Genetic Algorithm Rule Discovery System (GARDS) to discover the best control strategies for the dynamic routing problems. GARDS consists of two components: the Unsupervised Learner and the Plan Manager/Evaluator. The Unsupervised Learner component used a rule-based GA (a rule represents a chromosome) to evolve new populations of control strategies. The Plan Manage/Evaluator component connected with the problem domain's simulation model to evaluate the population of the solutions generated by the Unsupervised Learner. The authors demonstrated that GARDS is able to learn effective routing control strategies in a three parallel machine problem as well as a flexible cellular manufacturing system consisting of 13 machines arranged in 4 cells. However, learning in GARDS is long and requires hundreds of simulation runs.

Palmer (1996) developed another learning system called Genetic Algorithm Prototype Learning System (GAPLS). GAPLS is similar to GARDS except that instead of using a rule-based knowledge representation, GAPLS employed prototypes of clusters to represent knowledge. Using prototypes rather than rules in the GA essentially reduces the complexity of the genetic operators used in searching the control knowledge (Palmer, 1996). The author compared GAPLS with GARDS. GAPLS outperformed GARDS by providing a better routing solution as well as a quicker learning speed.

2.2.1.3. Agent-Based Approaches

2.2.1.3.1. Contract Net-Based Approaches

In an agent-based dynamic routing problem, agents are used to represent each resource and job. The job agent associated with a job will announce its requirements for the next operation to those resource agents that have the potential to perform that operation. The resource agents who receive the announcement message will respond with a bid message to the job agent. All the bids submitted for the job's next operation will be evaluated by the job agent based on a set of heuristics and then one resource will be selected and awarded a contract for performing the operation. The above bidding procedure is the core of the contract-net protocol. Bidding schemes based on the contract-net protocol may differ in such aspects as the timing of message exchanges involving announcements and bid collection, information reported within the bid, and the rules used in bid evaluation.

Shaw (1988) employed the contract-net method for dynamic scheduling in cellular manufacturing systems. In his approach, when an operation of a job at a cell is finished, the cell's control unit will make the decision regarding which cell the job should visit next. To do that, the cell's control unit broadcasts the task announcements to the other cell control units. The cell control unit who received a task announcement checks if the required operation is within its capability and submits its estimation on the earliest finishing time (EFT) or shortest processing time (SPT). There is no job agent in this case. Each job's route is determined through the negotiation between the cells. Shaw's experimental results indicated that the bidding scheme with EFT (earliest finishing time) outperformed the bidding scheme with SPT (shortest processing time).

Saad et al. (1997) proposed a contract-net-based heterarchical scheduling approach for flexible manufacturing systems. In their study, two scheduling mechanisms were tested. The first is the Production Reservation (PR) method where all the operations of a job are scheduled completely at the time when it arrives to the system. The other method, referred to as Single Step Production Reservation (SSPR), schedules one operation at a time with the job agent delaying negotiation of its next operation until the current operation is finished. In the contract-net protocol, a job agent selects the machine that can finish processing the required operation first. If at least two alternatives are tied for this criterion, the job agent will choose the machine with fewer jobs in its reservation list. They compared the PR and SSPR approaches with some traditional dispatching rules. Their results showed that PR outperformed the traditional dispatching rules, while SSPR only outperformed PR on average tardiness. However, unexpected events such as machine breakdowns or emergent jobs were not considered in their experiments. Otherwise, SSPR should be able to take the advantage in the face of these uncertainties.

Xue et al. (2001) developed an intelligent optimal scheduling mechanism that uses a constraint-based search mechanism to identify the best sequence to accomplish the required tasks, as well as timing parameter values (the earliest and the latest task finish times). Given the timing parameter values, the agent-based collaborative mechanism was used to generate a production schedule. Their agent-based collaborative mechanism consists of a bidding mechanism and a mediator mechanism. Their bidding mechanism is implemented based on the contract-net protocol. The mediation mechanism is used to coordinate the activities of the relevant agents to improve the scheduling efficiency. In their approach, the manufacturing resources, including facilities and persons are modeled

as agents. Two mediators, facility mediator and personnel mediator, are used to coordinate the activities of the resource agents.

Oulhadj et al. (1998) presented a negotiation strategy similar to the approach of Shaw (1988). The resource agent is responsible for establishing the negotiation with other resource agents in order to select the most appropriate resources to allocate to the specific task operations. The PR method was employed in their study. Oulhadj et al. (1999) extended the contract-net protocol to a multi-contract net protocol. It provided the function of scheduling several tasks simultaneously. Their experimental results showed that the time required to schedule operations with this approach and the run time including scheduling and execution both are linear rather than exponential with the increase of the number of scheduled tasks.

Sousa and Ramos (1996, 1998, 1999) proposed a contract net-based negotiation protocol for scheduling in manufacturing systems. The bid submitted from the resource agent consists of the information concerning the time windows that the resources are free. Selecting bid was based on the resources being able to finish the part before the due date and with more free time intervals. The authors also mentioned about renegotiation phase when a machine malfunctions. However, no further explanation is given on how to deal with the scheduled operations that are affected by this malfunction.

2.2.1.3.2 Market-Based Approaches

The other agent negotiation approach called market-like approach is very similar to the contract-net protocol except currency is used for bid evaluation. Each job agent carries some amount of currency and pays the resource agent for processing the operation. In every bidding process, the job agent who is able to offer the highest bid

takes priority of being processed. The agent negotiation strategies in the studies presented below employ a market-based approach.

Lin and Solberg (1992) presented an agent-based shop floor scheduling and control framework based on a market-like model that combined the objective and price mechanisms. In their system, each job agent with its unique set of weighted objectives enters the system with some currency and alternative process plans. To achieve the objectives, job agents will try to fulfill the processing requirements by bargaining with resource agents. Each resource agent sets its charging price based on its status. The part agent tries to minimize the price paid, but the resource agent's goal is to maximize the price charged. Each deal is completed once the part agent and resource agent are mutually committed. One important feature of this market-like mechanism is that the negotiation among agents is invisibly guided by an adjustable price to improve the system performance. Lin and Solbergs' results essentially showed that their system was able to handle unexpected resource failures and part objective changes. Lin and Solberg (1994) later presented a manufacturing simulation system based on the dynamic price mechanism for agent negotiation. The proposed agent-based framework simplifies implementation of different negotiation strategies in manufacturing systems.

Dewan and Joshi (2000, 2001) developed an auction-based scheduling mechanism for a job shop environment. They also used currency as a means for agent negotiation. Their market-like approach differed from Lin and Solbergs' (1992) in using Lagrangian relaxation to decompose the problem formulation. Whenever a machine agent is available, it announces an auction for time slots from the current time to the end of the time horizon. Each job agent will bid for the time slots with the cost that they are willing

to pay. The job agent's goal is to minimize cost, while the machine agent uses the submitted bids for price adjustment. If more than one job demands the same time slot, the price for that slot will increase. The price adjustment and bid calculation continue iteratively until the price converges. The machine agent determines the best bid for the earliest time slot as the next operation. After processing is finished for that operation, the above auction procedure is executed again. Dewan and Joshi (2000) further used the above mechanism to schedule the jobs with different objectives.

Ottaway and Burns (2000) proposed an agent-based negotiation involving a currency scheme. In their model, the amount of currency that a job agent carries is based on the job's objective function, a weighted linear combination of time, cost, and quality. The resources determine the amount of currency to be charged for their production services based on their capabilities and the demand for their services. It is noted that there is an incentive factor for preventing a job from being stuck in the system due to a lack of currency. This factor is used to increase the budgeted funds for the jobs that kept failing in the bidding process. Ottaway and Burns also addressed the importance of using supervisor agents to balance the production load and maximize overall throughput. The supervisor agents essentially played a key role for dynamically switching the system structure between a hierarchy and a heterarchy. Table 2.3. shows a comparison of the agent-based approaches mentioned earlier. The features of the systems considered are defined as:

1. Control structure: (Hi) hierarchy, (He) heterarchy, or (Q) quasi-heterarchy.
2. Negotiation approach: (C) Contract-net protocol, (M) Market-like mechanism, or (O) others)

3. What agent initiates the negotiation process? ((P) Part agent or (R) Resource agents)
4. What agent makes the final decision for each negotiation? ((P) Part agent, (R) Resource agent, (B) Both, or (M) Mediator)
5. How many passes of messages are required for routing a job to a machine? ((S) Single pass or (M) Multiple passes)
6. Decision-making frequency: (PR) PR, or (SS) SSPR.

Table 2.3. Comparison of the Agent-Based Approaches

Agent-related studies	1	2	3	4	5	6
Dewan and Joshi (2000,2001)	He	M	R	R	M	SS
Kpothapall and Deshmukh (1999)	He	M	P	P	S	SS
Lin and Solberg (1992)	He	M	P	P	S	SS
Ouelhadj et al. (1998)	He	C	R	R	S	PR
Ottaway and Burns (2000)	Q	M	P	P	S	SS
Saad et al. (1997)	He	C	P	P	S	PR, SS
Shaw (1988)	He	C	P	P	S	SS
Sousa and Ramos (1996, 1998, 1999)	He	C	P	R	S	PR
Xue et al. (2001)	He	C	R	M	S	PR

2.2.1.3.3. Other Approaches

Cicirello and Smith (2001) proposed an ant colony approach in multi-agent systems in shop floor routing. In their approach, an agent is considered as an ant. When a job is released to the shop floor, it is assigned to an ant to carry it through the shop. There is no direct communication between resources and ants. All communication is carried out indirectly with the pheromone that each ant leaves on the resources that they use. In other words, the ants dynamically make the shop routing decisions through the use of simulated pheromone trails. Their experiment results showed that the ant colony control approach

outperformed the local decision making approaches from the standpoint of global performance. The most complex case in their experiments is a flow shop with four machines and processing only two job types. More complicated experiments need to be conducted to prove the robustness of this approach. Also, implementing this approach requires four parameters. The authors did not clearly explain how to set these parameters.

2.2.2. Job Sequencing Problems

Dynamic job sequencing problems make use of two principal approaches: scheduling/rescheduling and dispatching rules (DR). For the scheduling/rescheduling approach, a schedule is generated for all the given operations in the beginning before a job is released. Rescheduling is triggered in response to some unexpected event or a change in the status of the shop. The computational time and the frequency for scheduling are crucial when employing this approach. A job sequencing problem can be NP-complete and very time-consuming to solve. Scheduling too frequently may result in the delay of actual operations. On the other hand, scheduling infrequently may result in poor system performance due to ignoring some events that may significant impact system status (Sabuncuoglu and Karabuk, 1999).

Scheduling by using dispatching rules is an on-line scheduling approach in which operations are scheduled one at a time. A dispatching rule is concerned with selecting a job from the queue of a particular machine to be processed based on some criteria. This local decision can be made very quickly. Use of dispatching rules is attractive because of their simplicity and ease of implementation. However, the dispatching rules have the following shortcomings:

1. A DR always blindly pursues a single objective. (Chandra and Talavage, 1991) In reality, a set of objectives may be important simultaneously.
2. No single DR can be really dominant across all possible scenarios (Chiu and Yih, 1995; Kouiss et al., 1997; Pierrval and Mebarki, 1997; Subramaniam et al., 2000b).
3. A DR does not take in account the status of the other resources.

2.2.2.1. Rolling Horizon-Based Approaches

In the rolling time horizon approach, a scheduling problem is decomposed into a series of sub-problems by time intervals. The next three sub-sections provide a review of three types of rolling horizon-based approaches.

2.2.2.1.1. Rolling Horizon-Based Approach (by Genetic Algorithm)

A genetic algorithm (GA) is a promising search technique. The algorithm starts with a set of solutions (represented by chromosomes) called a population. Solutions from one population are taken and used to generate a new population (offspring). Solutions from the new population are selected according to their fitness value (the more suitable they are the more chances they will be selected). The selected solutions will be used to generate the next population making use of the two key GA operators: crossover and mutation. The above procedure is repeated until either no significant improvement in the fitness is seen from one generation to the next, or the number of generations created reaches a predefined maximum. GAs have received considerable attention and been widely applied in the area of production scheduling because of their capability of dealing with problems with large search spaces.

Fang and Xi (1997) proposed a periodic and event-driven rolling horizon job shop rescheduling strategy in a dynamic environment. In their study, rescheduling is performed not only periodically but also when some unpredictable events (job arrivals, machine breakdown, machine recovery, and changes of due dates of jobs) happen. In their rescheduling procedure, a GA is employed to make decisions on job routing and EDD is adopted for dispatching jobs in the buffer of each machine. Their results showed that the proposed rescheduling strategy was capable of handling the unexpected events that can not be tackled by use of a static strategy.

Khoo et al. (2000) developed a prototype GA-enhanced multi-objective scheduler for manufacturing systems. Their prototype system was validated to generate near-optimal schedules in well-known deterministic scheduling problems. Moreover, this prototype system also demonstrated its capability of handling a dynamic event such as an unexpected rush order. Jian and Elmaraghy (1997) employed the genetic algorithm to generate an initial schedule for a FMS. In their research, the initial schedule must be modified considering the following four uncertainties: machine breakdown, the arrival of rush orders, increased order priority (change in due dates), and order cancellation. The proposed algorithms can be used in conjunction with the classic dispatching rules such as SPT, EDD, FIFO, etc. Chang and Lo (2001) developed an algorithm for solving job-shop scheduling problems with multiple qualitative (marketing criteria) and quantitative (production criteria) objective functions. Their approach incorporated Tabu search (TS) algorithms and GAs. The proposed rescheduling scheme based on their TS/GA mixture approach was able to handle uncertainties such as rush orders, machine breakdowns, job

cancellations, material shortage, and due-date changes. Their results also showed that the TS/GA mixture approach is superior to the GA alone.

In the GA-based studies that have been mentioned above, all the jobs are defined before scheduling and a GA is used to generate a new schedule responding to the unexpected events such as machine breakdowns and modifications of existing orders. If the jobs are not known in advance, a new schedule for all the jobs in the system is generated by the GA-based system whenever a new job arrives at the system. Lin et al. (1997) proposed a GA-based scheduling system that can be implemented for dynamic job-shop scheduling problems where details of the arriving jobs are not known in advance. Their experiment showed that their GA-based scheduling system outperformed the common dispatching rules under different manufacturing environments for various objectives. Chryssoloris and Subramaniam (2001) proposed a GA-based scheduling method for a dynamic job shop with unreliable machines, flexible job routes, and multiple scheduling criteria. They compared their method with several common dispatching rules by conducting a simulated job shop under varied conditions. Their results showed that the proposed GA method significantly outperformed those common dispatching rules when seeking to minimize mean job tardiness and mean job costs. Rossi and Dini (2000) proposed a scheduling system capable of giving a fast optimal response by using a genetic algorithm to determine the optimal solution. Their scheduler is able to respond to events such as new arrival jobs, failures of feeding system, and machine breakdowns. Rossi and Dini compared their scheduling system with a rule-oriented algorithm selecting the best schedule among a set of common dispatching rules. The

results showed that their system is superior to the rule-oriented algorithm on the measures of makespan and computation time of scheduling.

2.2.2.1.2. Rolling Horizon Approaches (by Dispatching Rules)

The idea of the rolling horizon approach can also be applied for use with a DR selection policy. A set of DRs can be evaluated by using a simulation technique and the best DR is employed for the simulated interval. Ishii and Talavage (1991) proposed a transient-based approach to define the next scheduling interval. This approach adapts the length of the next scheduling interval automatically based on the real-time status of the system. By simulating the system ahead, a dispatching rule can be determined for a short period before it is actually carried out. Once the next scheduling interval is determined, simulation is used again to evaluate each rule. The rule that performed the best is selected as a dispatching rule for the next scheduling interval. Their results showed that the proposed approach improved the performance up to 16.5% against the traditional scheduling algorithm that uses a single dispatching rule for the entire manufacturing period.

Kim and Kim (1994) proposed a simulation-based real-time scheduling mechanism for a FMS. Their scheduling mechanism consists of a simulation model and a real-time control system. The simulation model was used to evaluate 13 dispatching rules and select the best rule the next horizon based on an estimated performance value it generates for each rule (the schedule result). The real-time control system then periodically monitors the shop floor and finds the actual performance value. The selected dispatching rule is used until the difference of the actual and the estimated performance values exceeds a predetermined limit. Jeong and Kim (1998) conducted a further study of

the factors that may influence this real-time scheduling mechanism. They examined variant approaches for determining when to select a new rule. They also tested the impact on the performance by using two simulation models (one includes unknown future disturbance and the other does not). Their results indicated that the performance of the scheduling mechanism was affected by the method of determining the time to select a rule, while not significantly affected by the type of simulation model.

Shafaei and Brunn (1999a) identified the best scheduling rule based on the rolling horizon approach from seven rules recently developed. They used cost as the performance measure in their research. From their simulation results, SPT-C/R is the best dispatching rule over various rescheduling intervals and under different conditions. The results indicated that a scheduling rule requiring more global information does not necessarily provide a better schedule than one that only requires local information. The results also indicated that the length of the rescheduling interval should rely on the due date tightness. For orders with tight due dates, rescheduling more frequently is highly recommended. Shafaei and Brunn (1999b) then continued investigating the robustness of scheduling rules in dynamic and stochastic environments using the rolling time horizon approach. They stated that the robustness of a scheduling approach should be gauged based on its ability to maintain its performance in the presence of uncertainties. In that study, Shafaei and Brunn evaluated the influence of the uncertainties in stochastic processing times and machine breakdown. They concluded that the performance of the scheduling rules in uncertain conditions is very sensitive to the rescheduling policy. That is, to reduce the effects of the uncertainties, frequent rescheduling is a promising approach. Based on the above study, Shafaei and Brunn (2000) found that the

performance of a robust scheduling method not only depends on a frequent rescheduling policy but also on how well the shop load is balanced and controlled. To control and balance the shop load, Shafaei and Brunn found it necessary to integrate the planning (i.e. job release and job routing) and scheduling functions. Finally they proposed a framework employing the SPT-C/R, which showed a good potential in their previous research, with the rolling time approach to integrate the above three functions for dynamically generating robust schedules.

2.2.2.1.3. Rolling Horizon Approaches (by Heuristics)

Sun and Lin (1994) proposed a backward scheduling approach on the basis of the rolling time approach. Their approach in dynamic scheduling was to decompose a dynamic scheduling problem into a series of static scheduling problems. Each static scheduling problem can be dealt with in a specific time window. The scheduling system consists of two modules: order module and scheduling module. The order module is responsible for order acceptance and due-date assignment, while the scheduling module has two functional sub-modules, a boundary condition module and a backward scheduling module. The boundary condition module decomposes the dynamic scheduling problem into a series of static scheduling problems over the rolling time period. The backward scheduling module carries out the backward scheduling approach based on the boundary information given by the boundary condition module. The backward scheduling module not only provides the finished schedule but also determines the job release time. The backward scheduling method is also able to evaluate the alternative due-date assignment for the order module. In each rolling time window, the due-date performance and the inventory cost can be controlled by the backward scheduling

approach. Based on the authors' results, the proposed backward scheduling approach outperformed the forward scheduling approach. The authors addressed the importance of effectively decomposing a scheduling horizon but did not provide any further discussion.

Table 2.4. Summary of Rolling Horizon-Based Approaches

Research	Rolling Horizon Approach	Scheduling method
Chang and Lo (2001)	Event driven	GA for sequencing all the available jobs
Chryssoloris and Subramaniam (2001)	Event driven	GA for sequencing all the available jobs
Fang and Xi (1997)	Periodically and event driven	GA for routing, EDD for dispatching
Ishii and Talavage(1991)	Periodically	Evaluate a set of DRs through simulation and select the best rule for next horizon
Jian and Elmaraghy (1997)	Event driven	GA for sequencing all the available jobs
Khoo et al. (2000)	Event driven	GA for sequencing all the available jobs
Kim and Kim (1994) Jeong and Kim (1998)	Periodically	Evaluate a set of DRs through simulation and select the best rule for next horizon
Lin et al. (1997)	Event driven	GA for sequencing all the available jobs
Rossi and Dini (2000)	Event driven	GA for sequencing all the available jobs
Shadaei and Brunn (1999a, 1999b, 2000)	Periodically	Evaluate a set of DRs through simulation and select the best rule for next horizon
Sun and Lin (1994)	Periodically	Backward scheduling approach

2.2.2.2. Knowledge-Based Scheduling System

As pointed out by Nakasika and Yoshida (1992), an effective real-time scheduling system should require the following characteristics:

1. Rule selection must take into account a variety of real-time information about the manufacturing system.
2. Rule selection must be completed in such a short time that the real operation is not delayed.

However, the rolling horizon approaches mentioned before for dynamically selecting dispatching rules require either performing some computation or running one or more simulations in real time. If the system becomes complex, then the simulation and rule selection procedures may not be finished in time resulting in a delay to the real operation. To overcome this problem, Priore et al. (2001a) recommends using "scheduling knowledge" of the manufacturing system to save time and get a rapid response in a dynamically changing environment. One of the most important issues for developing a knowledge-based system is how to acquire useful knowledge about the manufacturing system for use in real time intelligent decision-making. Machine leaning techniques are the popular tools used to acquire knowledge.

2.2.2.2.1. Inductive Learning

Inductive learning can be defined as the process of inferring the description of a class from the description of individual objects of the class (Shaw et al., 1992). In other word, the inductive learning approach is capable of obtaining general domain knowledge from the specific knowledge provided by domain examples.

Nakasika and Yoshida (1992) proposed a learning scheme for acquiring knowledge concerning real-time switching dispatching rules based on the production system status. In their approach, a set of learning problems (examples) are generated and simulated to search for the best scheduling rules. The simulation results are used to

extract the data that are used as the input of the new inductive learning algorithm proposed in their approach. Finally, a binary decision tree is generated based on the proposed learning algorithm. The results showed that their scheduling system outperformed each of the dispatching rules used as the candidates in their system. Their study identified two problems that need to be addressed. The first is a need to reduce the computation time required to generate the binary decision tree and the second is to explore how to set the various parameter values used in their learning system.

Shaw et al. (1992) proposed a scheduling system called PDS (Pattern-Directed Scheduling) for selecting an appropriate dispatching rule in FMS. In order to select the appropriate dispatching rule, the authors considered due date tightness, relative workload imbalance, job routing flexibility (the average number of alternative machines available for processing a given operation), and limitation on buffer size at individual machines as the key factors that represent the patterns of a FMS. In their approach, a number of simulation experiments were conducted with various dispatching rules under various manufacturing environments. The results of these experiments would then be fed as input to the inductive learning process. This process would then generate a decision tree for use in selecting appropriate dispatching rules. The inductive learning algorithm used here was ID3. This approach provided the capabilities of selecting the appropriate rule and switching between different rules in real time based on changes in the state of the system. Park et al. (1997) employed the inductive learning algorithm C4.5, which is a refinement of the ID3, to improve the performance of the original PDS. They also added a rule refinement mechanism for their new version of the PDS. The new PDS was tested by a real system producing 41 different products on two identical production lines. The results

showed that PDS was superior to any of the candidate dispatching rules applied in PDS. Piramuthu et al. (2000) demonstrated the use of genetic algorithm for generating a knowledge base for sequencing applications of PDS.

Priore et al. (2001b) also built a scheduling system that obtains knowledge by using the inductive learning algorithm C4.5. However, they found that, on some occasions, their system didn't perform as expected because it reacts precipitously to changes in control attributes that may be only transitory. The authors, therefore, developed a mechanism to dampen these transitory scenarios. Their results showed an improvement in mean tardiness of 8% compared to use of the single dispatching rule that performs best when used individually. They also pointed out that the major drawback of their approach is the need to perform a large number of simulations in order to generate sufficient training examples.

2.2.2.2.2. Neural Networks

Sim et al. (1994) developed a neural network approach that incorporates an expert system and applied it to dynamic job shop scheduling. Their artificial neural network is based on the back-propagation neural network model. The expert system reduced the training time for the neural network by allowing sub-networks to be trained separately. The input layer consists of 14 neurons representing various scheduling factors for each job. These neurons include 10 nodes for representing 10 different dispatching rules, three nodes representing three different levels of system load, and one node for representing two different criteria. For each dispatching rule, 5,000 jobs are simulated for 8 different arrival rates and 2 different criteria. The composite rule expert system was developed based on the simulation results and is able to select the best dispatching rule based on the

prevailing workload condition and scheduling criteria. The authors compared their expert neural network system with each of the dispatching rules employed in the system. Their results indicated that the expert system is able to maintain the performance of the best rules across the different arrival rates for both scheduling criteria, a feat that none of the dispatching rules could accomplish.

Liu and Dong (1996) also used simulation results to train a neural network to capture knowledge that can be used to select the most appropriate dispatching rules. The input data for training the network is the operation sequence of each job and the associated processing times that are randomly generated for each operation. The output data is the best dispatching rule coming from the results of the simulation. Liu and Dong showed that the better rules have high probabilities of being selected by their neural network rule selector than the least desirable rules. The authors also pointed out that the rule selector's ability to make a good decision in real time required that the neural network receive sufficient training. However, they had no answers regarding how many simulation runs would be enough to cover all or most of the dispatching conditions in a given shop floor.

2.2.2.2.3. GA-Based Learning

Jahangirian and Conroy (2000) proposed a scheduling framework consisting of two modules, a simulation module and a GA-based learning module. The simulation module with a scheduling knowledge base continues to generate learning examples that comprise the system status, the selected dispatching rule, and the results of these decisions. The learning examples will be transferred to the learning module. The GA in the learning module was employed to refine the old knowledge base. Each rule set is

represented as a chromosome in their study. They tested results on a single machine problem with a number of dynamic events such as machine breakdown. The learned knowledge base outperformed the individual dispatching rules used in their study.

Chiu and Yih (1995) proposed a knowledge-based scheduling system that dynamically selects dispatching rules. In their approach, a genetic algorithm was used to search for good schedules. From the good schedules obtained, inductive learning was used to extract scheduling knowledge. Their experimental results showed that the proposed dynamic scheduling system outperformed the dispatching rules (SPT, SIO, SLACK/RO, and EDD) in the weighted performance measures consisting of makespan, number of tardy jobs, and lateness.

2.2.2.3. Other Approaches

Pierreval and Mebarki (1997) proposed a scheduling strategy for dynamic dispatching rule selection. Whenever a machine is available for the next operation, the pre-defined symptoms must be detected. These symptoms include such conditions as recognition that the tardiness of the WIP is increasing, the machine has too many waiting jobs, or possibly that a job has waited too long. These symptoms become active when some observed variables (e.g., utilization, queue length, waiting time, etc.) exceed some specific threshold values. These thresholds are problem dependent and tuned with a the Hooke and Jeeve's simulation-optimization technique. Their approach was compared with some common dispatching rules on a job shop problem. The results showed significant improvements in the measures of the mean tardiness.

Subramaniam et al. (2000b) proposed an approach of dynamic dispatching rule selection based on the analytic hierarchy process (AHP), which considers the shop

conditions existing at every decision point. In fact, AHP is an approach to help the decision makers to make better decisions in problems involving multiple objectives. The AHP provides a framework that ranks the alternatives based on the decision maker's knowledge and preferences. The results in the article showed that the AHP method is not guaranteed to generate the optimal schedule, but it is superior to the method using single dispatching rule for the measure of makespan.

Ariz (1995) proposed a two level distributed production control system (DPCS) for on-line scheduling in a multi-cell flexible manufacturing system. Each flexible manufacturing cell is independently controlled by its own cell-controller using a two level heuristic procedure. The upper level procedure is used to select parts to be processed in the cell, while the lower level procedure is used to control the part flow within the cell. Their results show that the proposed DPCS is able to achieve high throughput with almost no tardiness. However, this DPCS is governed by a set of control parameters that suit a particular order stream only. The values of these parameters need to be recalibrated whenever there is a change in the order stream.

2.2.2.4. Summary

In the review of various rolling horizon-based approaches, one of the important issues that has not received attention is if the new scheduling policy can be developed in real time. Developing a new scheduling policy for the next horizon may be time-consuming and result in an actual operational delay. This issue can be resolved by using a knowledge–based scheduling system. The knowledge-based system has the advantage of rapidly responding to the environment changes. However, some changes that the existing knowledge bases do not cover may result in a bad or infeasible schedule. For instance, if

the system configurations or objectives are changed, the existing knowledge bases are no longer applicable and it becomes necessary to build new knowledge bases for the system. This is because it is unreasonable to construct a knowledge base that can cover all the possible system conditions. Therefore, updating the knowledge bases in real time for covering a new circumstance will be important. This leads to the motivation for building a knowledge-based system with on-line learning capability.

In all studies about dynamically selecting dispatching rules, all resources follow the same rule selection policy at the same period of time. From the perspective of agent technology, an agent representing a resource is autonomous and therefore may have a different rule selection policy than the others. Kouiss et al. (1997) proposed an approach based on a multi-agent architecture where each resource agent in the system selects, locally and dynamically, the DR that seems most suited to the operating conditions, the production objectives, and the current shop status. The selection of the DR employed by each resource agent is carried out based on the strategy proposed by Pierreval and Mebarki (1997). That is, detecting the pre-defined local symptoms (for resource agent) and DR selection is based on the currently active symptoms. The authors added a supervisory agent for monitoring the system status (i.e. global symptoms for the supervisor agent). The supervisory agent may impose a particular DR for all the resource agents if the global symptom is active. Otherwise, each resource agent can autonomously select the DR from a set of pre-selected DRs based on the status of the resource it represents and the other resource's conditions. However, the authors did not explain what information a resource agent would requests from the other resource agents. Therefore,

research on DR selection by agent-based approaches still has some questions that need to be answered.

2.3. Reinforcement Learning

Reinforcement learning (RL) has received some attention from agent-based researchers because it deals with the problem of how an autonomous agent can learn to select proper actions for achieving its goals through interacting with its environment. In the RL framework, a learning agent must be able to perceive information from its environment. The perceived information is used to determine the current state of the environment. The agent then chooses an action to perform based on the perceived state. The action taken may result in a change in the state of the environment. Based on the new state, there is an immediate reinforcement that is used to reward or penalize the selected action. These interactions between the agent and its environment continue until the agent learns a decision-making strategy that maximizes the total reward. Sutton and Barto (1999) defined four key elements for dealing with the RL problems: a policy, a reward function, a value function and a model of the environment. A policy defines the agent's behavior in a given state. A reward function specifies the overall goal of the agent that guides the agent toward learning to achieve the goal. A value function specifies the value of a state or a state-action pair indicating how good it (the state or the state-action pair) is in the long run. A model of the environment predicts the next state given the current state and a proposed action.

2.3.1. Markov Decision Process

Besides the above four elements, a key assumption in the RL framework is that the definition of the current state used by each agent to make its decision should summarize everything important about the complete sequence of past states leading to it. Some of the information about the complete sequence may be lost, but all that really matters for the future is contained within the current state signal. This is called the Markov property. Therefore, if an environment has the Markov property, then its next state can be predicted given the current state and action. This significant assumption enables the current state to be a good basis for predicting the next state. Under this assumption, the interaction of an agent and its environment can be called a Markov Decision Process (MDP).

2.3.2. Generalization and Function Approximation

For a small RL problem, the estimates of value functions can be represented as a table with one entry for each state or for each state-action pair. However, for a large problem with a large number of states or actions, updating information accurately in such a large table may be a problem. Function approximation is currently a popular method to resolve this issue. Function approximation is an approach generalizing experience from a small subset of examples to develop an approximation over a larger subset. Currently, employing neural networks is the most popular approach for function approximation in large RL problems (Sutton and Barto, 1999).

2.3.3. Exploration and Exploitation

Exploration and exploitation is another important issue in RL problems. Exploration entails the agent trying something that hasn't been done before in order to get more reward, while in exploitation the agent favors actions that were previously taken and rewarded. Exploitation may take advantage of guaranteeing a good expected reward in one play, but exploration provides more opportunities to find the maximum total reward in the long run. One popular approach to deal with this trade-off issue is the e–greedy method. The e–greedy method involves selecting, with probability (1-e), the action with the best value, otherwise, with small probability e, an action is selected randomly.

2.3.4. RL Applications to Manufacturing Systems

Mahadevan et al. (1997b, 1999) developed a new model-free average-reward algorithm called SMART for continuous-time semi-Markov decision processes. They applied the SMART algorithm to the problem of optimal preventative maintenance in a production inventory system. In their system, there was a single machine capable of producing multiple types of products with multiple buffers for storing each of the different products. Whenever a job is finished, the machine needs to decide to either undergo maintenance or start another job. Machine maintenance costs and time are less than repair costs and time. In other words, frequent maintenance may be not economical but machine failures resulting from rare maintenance will require more repair costs and time. In their maintenance problem, the state of the system is a 10-dimensional vector of integers that consists of the numbers of five different products manufactured since the last repair or maintenance and the buffer levels of the five products. They compared the

maintenance policy learned from SMART to two well-known maintenance heuristics. They found that SMART is more flexible than the two heuristics in finding proper maintenance schedules as the costs are varied. Mahadevan and Theocharous (1998) applied SMART to the problem of optimizing a 3-machine transfer line producing a single product type. The system goal is to maximize the throughput of the transfer line while minimizing the Work-In-Process (WIP) inventory and failures. They compared the policy from SMART to the kanban heuristic. Their results showed that the policy learned by SMART requires fewer items in inventory and results in fewer failures than with the Kanban heuristic. Paternina-Arboleda and Das (2001) extended the work of Mahadevan and Teocharous (1998) to deal with a 4-machine serial line and compared SMART to more existing control WIP policies. They examined the system with constant demand rate and Poisson demand rate. Under these two circumstances, SMART outperformed those heuristic policies on average WIP level and average WIP costs.

Zhang and Dietterich (1995) applied RL to a job shop scheduling problem involving the scheduling of the various tasks that must be performed to install and test the payloads placed in the cargo bay of the NASA space shuttle for each mission. The objective of this problem was to schedule a set of tasks without violating any resource constraints while minimizing the total duration. The scheduling approach Zhang and Dietterich employed was an iterative repair-based scheduling method that started with generating a critical path schedule by ignoring the resource constraints and incrementally repairing the schedule to find a shortest conflict-free schedule. In their system, each state is a complete schedule and each action is a schedule modification. They applied the temporal difference algorithm TD(?) (an RL algorithm) to this scheduling problem. After

taking an action to repair the schedule the scheduler receives a negative reward if the new state still contains constraint violations. This reward function essentially forces the scheduler to not only find a conflict-free schedule but to do it in fewer iterations. The performance of the iterative repair-based procedure with a simulated annealing (SA) method was compared with the one using the TD method. Their results showed that one iteration of the method with TD is equivalent to about 1.8 iterations of the method with SA.

Aydin and Ozrtemel (2000) proposed an intelligent agent-based scheduling system in which agents are trained by a new RL algorithm they refer to as Q-? . They employed Q-III to train the resource agents to dynamically select dispatching rules. Their state determination criteria consist of the buffer size of the machine and the mean slack time of the queue. The rewards were generated based on some selection rules obtained from the literature (i.e., SPT is best when the system is overloaded). The thresholds used in the rules for determining the systems status were obtained through trial-and-error procedures. Three dispatching rules: SPT, COVERT, and CR, are available for each resource agent to select for their use. The authors compared the proposed scheduling system trained by their RL mechanism to the above three dispatching rules. Their results showed the RL-scheduling system outperformed the use of each of the three rules individually in mean tardiness for most of the testing cases.

2.3.5. Other Applications of RL

More and more work on practical implementations of RL techniques to different fields has been reported. One of the successful stories about RL applications was Tesauro's TD-Gammon (1995), which was used to play backgammon. TD-Gammon was

developed based on the TD(λ) algorithm and a multi-layer neural network for function approximation. The latest version of the TD-Gammon was able to play the backgammon game close to the level of the best human player in the world. Another famous application was the elevator-dispatching problem. Modern elevator dispatchers are usually designed heuristically. Crites and Barto (1996) applied the Q-learning to a four-elevator, ten-floor system. Each elevator made its own decision independently of the other elevators. There were some constraints placed on the decisions. The system they dealt with had more than 10^{22} states. Like TD-Gammon, Crites and Barto also employed a neural network to represent the action-value function. Their RL-based dispatchers outperformed other existing dispatching heuristics on the customer's average waiting time and average squared waiting time. RL also has been widely applied to robotics motion control. Singh and Bertsekas (1997) used the TD(0) algorithm to find dynamic channel allocation policies in cellular telephone systems. Their study showed that RL with a linear function approximation is able to find better dynamic channel allocation policies than two other existing policies. Sutton (1996) applied a RL algorithm, called the Sarsa algorithm, to controlling the motions of a two-link robot. Mahadevan et al. (1997a) successfully applied RL to navigating a delivery robot around an indoor office environment.

2.4. Summary of Literature Review

The heterarchical control structure is believed to be a promising architecture for the next generation of manufacturing systems. Agent-based approaches can be applied in the implementation of a heterarchical control system. For dynamic job routing problems, most of the existing agent-based approaches focus on the issues of cooperation and

negotiation among autonomous agents. Enhancing the intelligence of an individual agent has not received much attention. For job sequencing problems, DRs are very useful and efficient. Although the dispatching rules do not guarantee an optimal schedule, they usually provide a reasonably good schedule. To use DRs appropriately for sequencing jobs, dynamic rule selection is required since the manufacturing shop status may change over time. A knowledge-based rule selection system can be used to rapidly respond to the changes of the shop status. However, the existing knowledge-based systems have the shortcoming that knowledge is acquired based on the use of off-line machine learning techniques. In addition, every resource selects the rules based on the same knowledge bases at the same period of time. The agent-based approach in which each resource agent has its own knowledge base for DR selection has not been explored yet.

Table 2.5 provides a summary of the assumptions made in previous published research studies. Table 2.6 provides a summary of the characteristics of the problems explored in previous research studies. Based on these results there is an average of eight machines in the system, with the system being able to manufacture twelve different jobs, with each job requiring four operations. Table 2.7 provides a summary of the problem objectives of those same systems. The five most popular objectives used involve minimizing something related to tardiness (mean tardiness, weighted mean tardiness, penalty due to tardiness, etc.), minimizing mean flow time/weighted mean flow time, minimizing mean makespan, minimizing number of tardy jobs, and maximizing profit.

Table 2.5 A Summary of Problem Assumptions in Previous Studies

Previous Research	Research Assumptions							
Authors (Year)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ROUTING PROBLEMS								
Bowden and Bullington (1996)					*		*	*
Chandra and Talavage (1991)			*			*	*	*
Cicirello and Smith (2001)				*	*	*	*	
Dewan and Joshi (2000, 2001)		*	*		*	*	*	*
Krothapalli and Deshmukh (1999)		*			*		*	*
Ottaway and Burns (2000)			*			*	*	
Saad et al. (1997)			*			*		
Shaw (1988)		*	*			*		*
Shmilovici and Maimon (1992)	*		*				*	*
Subramaniam et al. (2000a)			*				*	*
Yao and Pei (1990)	*		*	*			*	*
Xue et al. (2001)	N/A							
DISPATCHING PROBLEMS								
Ariz (1995)				*			*	*
Chang and Lo (2001)	*		*		*		*	*
Chiu and Yih (1995)	*	*		*		*		*
Chryssolouris and Subramaniam (2001)			*	*	*		*	*
Fang and Xi (1997)			*	*			*	
Ishii and Talavage (1991)		*	*	*	*	*	*	*
Jahangirian, M. and Conroy, C. V. (2000)		*	*		*		*	*
Jain and ElMaraghy (1997)	*		*	*			*	*
Khoo et al. (2000)	*	*	*	*	*	*	*	*
Kim and Kim (1994)			*		*			*
Kouiss et al. (1997)		*	*			*	*	*
Lin et al. (1997)			*	*	*	*	*	*
Liu and Dong (1996)		*	*		*	*	*	*
Matsuura et al. (1993)		*	*		*		*	*
Nakasuka and Yoshida (1992)		*	*	*	*	*	*	*
Park et al. (1997)					*		*	*
Pierreval and Mebarki (1997)		*	*			*	*	*
Piramuthu et al. (2000)					*		*	*
Priore et al. (2001)		*	*		*	*	*	*
Rossi and Dini (2000)	*			*	*		*	*
Shaw et al. (1992)					*		*	*
Shafaei and Brunn (1999a)		*	*	*	*	*	*	*
Shafaei and Brunn (1999b)		*	*		*		*	*
Shafaei and Brunn (2000)	*		*	*		*	*	*
Sim et al. (1994)		*	*					
Subramaniam et al. (2000b)	*	*	*	*	*	*	*	*
Sun and Lin (1994)		*	*		*	*	*	*

Table 2.5. (continued).

* Represents the assumption was made in the research.

NOTE:
1. All jobs have been given.
2. Each operation has been pre-assigned to a unique machine type. The operation sequence for each job is fixed (No routing decisions).
3. No Parallel machine clusters.
4. Deterministic set-up and processing times.
5. No reentrant machines.
6. No machine breakdown.
7. Transportation times between machines are ignored.
8. Set-up times are sequence-independent.

Table 2.6. A Summary of the Problem Size of the Examples in Previous Studies

Previous Research	# M/Cs	# Jobs	# Ops.
ROUTING PROBLEMS			
Bowden and Bullington (1996)	11 (4 cells)	3	3
Chandra and Talavage (1991)	10	10	3-7
Cicirello and Smith (2001)	4	2	2
Dewan and Joshi (2000, 2001)	6	80	3
Krothapalli and Deshmukh (1999)	40 (5 cells)	5	5
Ottaway and Burns (2000)	6	16	3 or 6
Saad et al. (1997)	9	N/A	5
Shaw (1988)	N/A	N/A	N/A
Shmilovici and Maimon (1992)	4	1	4
Subramaniam et al. (2000a)	4	20	2-10
Xue et al. (2001)	11	N/A	7
Yao and Pei (1990)	4	1	6
SEQUENCING PROBLEMS			
Ariz (1995)	9 (2 cells)	12	3-5
Chang and Lo (2001)	8	10	4-6
Chiu and Yih (1995)	8	8	2-5
Chryssolouris and Subramaniam (2001)	6	20	2-10
Fang and Xi (1997)	4	3	3 or 4
Ishii and Talavage (1991)	6	6	5 or 6
Jahangirian, M. and Conroy, C. V. (2000)	1	N/A	1
Jain and Elmaraghy (1997)	5	4	3
Khoo et al. (2000)	5	20	N/A
Kim and Kim (1994)	11	N/A	3-6
Kouiss et al. (1997)	4	N/A	2-6
Lin et al. (1997)	5	N/A	5
Liu and Dong (1996)	5	5	1-5
Matsuura et al. (1993)	9	N/A	5
Nakasuka and Yoshida (1992)	3	3	3
Park et al. (1997)	6	N/A	3-5
Pierreval and Mebarki (1997)	4	N/A	2-6
Piramuthu et al. (2000)	6	N/A	3-5
Priore et al. (2001)	4	N/A	1-4
Rossi and Dini (2000)	16	14	1 or 2
Shaw et al. (1992)	8	N/A	1-8
Shafaei and Brunn (1999)	15	N/A	4-15
Shafaei and Brunn (1999)	15	N/A	4-15
Shafaei and Brunn (2000)	4	8	4
Sim et al. (1994)	9	N/A	3-6
Subramaniam et al. (2000b)	10	6	3
Sun and Lin (1994)	10	10	N/A
Average	**8**	**12**	**4**

Table 2.6. (continued).

NOTE:
M/Cs: Number of Machines.
Jobs: Number of Job types.
Ops.: Number of Operations Required for Each Job.

Table 2.7. A Summary of Performance Measures in Previous Studies

Previous Research	Performance Measures										
Authors (Year)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
ROUTING PROBLEMS											
Bowden and Bullington (1996)		*									
Chandra and Talavage (1991)	*		*			*					
Cicirello and Smith (2001)						*					
Dewan and Joshi (2000, 2001)			*								*
Krothapalli and Deshmukh (1999)	*		*					*			
Ottaway and Burns (2000)						*		*		*	
Saad et al. (1997)	*	*	*								
Shaw (1988)	*	*	*				*				
Shmilovici and Maimon (1992)						*					
Subramaniam et al. (2000)			*						*		
Xue et al. (2001)	N/A										
Yao and Pei (1990)					*						
SEQUENCING PROBLEMS											
Ariz (1995)			*			*					
Chang and Lo (2001)					*				*		
Chiu and Yih (1995)		*		*	*						
Chryssolouris and Subramaniam (2001)			*						*		
Fang and Xi (1997)					*						
Ishii and Talavage (1991)	*		*								
Jahangirian, M. and Conroy, C. V. (2000)	*		*								
Jain and ElMaraghy (1997)	*		*					*			
Khoo et al. (2000)			*		*				*		
Kim and Kim (1994)	*		*								
Kouiss et al. (1997)	*		*								
Lin et al. (1997)		*	*	*							
Liu and Dong (1992)	*	*									
Matsuura et al. (1993)					*						
Nakasuka and Yoshida (1992)			*		*						
Park et al. (1997)			*								
Pierreval and Mebarki (1997)	*		*								
Piramuthu et al. (2000)			*								
Priore et al. (2001)			*								
Rossi and Dini (2000)					*						
Shaw et al. (1992)			*								
Shafaei and Brunn (1999a)									*		
Shafaei and Brunn (1999b)									*		
Shafaei and Brunn (2000)			*								
Sim et al. (1994)		*	*								
Subramaniam et al. (2000b)					*						
Sun and Lin (1994)	*		*								

Table 2.7. (continued).

* Represents the performance measures employed in the research.

NOTE:
1. Minimize mean flow time/weighted mean flow time.
2. Minimize percentage/number of tardy jobs.
3. Minimize mean tardiness/ weighted mean tardiness/ conditional mean tardiness/ normalized job tardiness/ penalty due to tardiness.
4. Minimize mean lateness/ weighted mean lateness/ conditional mean lateness/ normalized job lateness.
5. Minimize makespan.
6. Maximize throughput.
7. Minimize average queuing time.
8. Maximize resource utilization.
9. Minimize mean job cost/ maximize profit.
10. Minimize average WIP.
11. Minimize earliness-tardiness.

CHAPTER III

Q-LEARNING FOR SINGLE MACHINE JOB DISPATCHING

3.1. Single Machine Dispatching Rule Selection

To develop a set of recommendations for applying the Q-learning algorithm for machine agents to construct a good policy for DR selection, this research considers conducting an experiment on a single machine dispatching rule selection problem. The single-machine production system contains a single buffer for storing jobs awaiting processing. Jobs arrive continuously according to a Poisson process. Each job consists of only one operation requiring variant processing time and the machine can process only one job at a time. If the machine is idle when a job arrives then the job will start processing immediately, otherwise the job will be sent to the buffer. In this research, selection of the next job from the buffer for processing is conducted based on one of the three dispatching rules, EDD, SPT, and FIFO. The system objective is to minimize the mean tardiness of the finished jobs. The selection of a dispatching rule will be based on the current policy in use by the Q-learning algorithm. The response is the mean tardiness measured after the learning process achieves steady state. The effects of applying the Q-learning technique to the dispatching rule selection problem are examined under various system conditions involving variations in system loading conditions and job due date tightness.

3.2. Q-Learning Algorithm

The original Q-learning algorithm was proposed by Watkins in 1989. The goal of this algorithm is to learn the state-action pair value, Q(s, a), which represents the long-term expected reward for each pair of state and action (denoted by *s* and *a*, respectively). The Q values learned with this algorithm have been proven to converge to the optimal state-action values, Q* (Tesauro, 1995). The optimal state-action values for a system represent the optimal policy that the agent intends to learn. The standard procedure of the Q-Learning algorithm is presented in Fig. 3.1. (Sutton and Barto, 1999):

Step 1. Initialize the Q(s, a) value functions arbitrarily
Step 2. Perceive the current state, s_0
Step 3. Following a certain policy (e.g. e–greedy), select an appropriate action (a) for the given state (s_0)
Step 4. Execute the selected action (a), receive immediate reward (r), and perceive the next state s_1
Step 5. Update the value function as follows:

$$Q(s_0, a) = Q(s_0, a) + a\,[\, r + ? \max_{b} Q(s_1, b) - Q(s_0, a)] \qquad (3\text{-}1)$$

Step 6. Let $s_0 = s_1$
Step 7. Go to step 3 until state s_0 represents a terminal state
Step 8. Repeat steps 2 to 7 for a number of episodes.

Figure 3.1. The Q-Learning Algorithm (Sutton and Barto, 1999)

In Figure 3.1, each iteration of steps 2 through 7 represents a learning cycle, also called an "episode". The parameter, a, is the step-size parameter influencing the learning rate. The parameter, ?, is called the discount-rate parameter, 0 = ? = 1, and impacts the present value of future rewards. The Q(s, a) values can be initialized arbitrarily. If no actions for any specific states are preferred, then when starting the Q-learning procedure all the Q(s, a) values in the policy table can be initialized with the same value. If some

prior knowledge about the benefit of certain actions is available, the agent may prefer taking those actions in the beginning by initializing those Q(s, a) values with larger values than the others. Then these actions will initially be selected. This can shorten the learning period. Step 3 involves the tradeoff of exploration and exploitation and many state-action pair selection methods may be used in this step.

3.3. Factors for Applying Q-learning to Single Machine Dispatching Rule Selection

There are a number of factors that one can manipulate in applying the Q-learning algorithm. The research goal was to determine the significance of the various factors for this application and to provide recommendations for factor settings. The main factors that are investigated include the following:

A. Number of states.

B. The threshold value setting for determining states.

C. Number of ranges for determining reward/penalty.

D. The threshold value setting for determining reward/penalty ranges.

E. Approaches to setting reward/penalty magnitude.

F. Initial Q values in the policy table.

G. Step Size (a).

H. Discount rate (γ).

I. Approaches for exploration and exploitation.

These factors are described in more detail in the subsections that follow.

3.3.1. Factors for Constructing the Policy Table

Factors A and B influence the construction of the rule-selection policy table. In the single-machine system, the learning agent's decision on which dispatch rule to employ for selecting a job from the buffer is based on the status of the system buffer. Several choices are available for defining the buffer's status; these include such measures as the number of the jobs in the buffer, the number of the tardy jobs in the buffer, or the tardiness or lateness of those jobs. For this study, the estimated mean lateness of the number of jobs in the buffer is adopted as the state determination criterion in the policy table. This value was chosen over job tardiness since it is able to distinguish between early jobs (unlike the tardiness measure). When constructing a policy table, the individual states defining the buffer's status have to be associated with specified ranges of possible values. Therefore, defining (A) number of ranges, and the endpoints (thresholds) of (B) the range of values for each state represents those factors that must be considered in the learning algorithm's design.

Given that the agent's decision involves selecting an appropriate dispatching rule, two special conditions need to be considered. The first is when the buffer is empty and the second occurs when there is only one job in the buffer. For the former condition, no dispatching rule is needed to determine what job to process next because there is no job in the buffer. In the latter condition, since there is only one job, no matter what dispatching rule is employed the same job will be selected. The conditions for these two special cases are represented in the policy table using two dummy states. Therefore, only when there are two or more jobs in the buffer does the Q-learning algorithm select one of the three dispatching rules. In order to implement this capability, the system also

maintains a measure of the number of jobs in the buffer, as well as the estimated mean lateness of those jobs.

Factor A defines the number of states (without counting the dummy states) in the policy table and Factor B defines the thresholds values for each state thereby creating the specific range of values. For a given number of states, the range for each state is defined as a multiple (m) of the expected mean processing time (EMPT). At smaller values of m for factor B the system is better able to distinguish differences between jobs at the lower end of the lateness spectrum with jobs that are very late being grouped together in the last interval as it acts as the catchall. As the value of m increases, more intervals are provided for differentiating late jobs, but at the expense of decreased resolution of the other intervals. In this experimental study, m is set to either 1 or 3. Table 3.1 provides an example of a policy table with 10 states (factor A).

Table 3.1. An Example of a 10-state Policy Table

State	State criteria	EDD	SPT	FIFO
Dummy	No job in queue	0	0	0
Dummy	One job in queue	0	0	0
1	mean_lateness < 0	Q(1,1)	Q(1,2)	Q(1,3)
2	0 ≤ mean_lateness < m × EMPT	Q(2,1)	Q(2,2)	Q(2,3)
3	m × EMPT ≤ mean_lateness < 2m × EMPT	Q(3,1)	Q(3,2)	Q(3,3)
4	2m × EMPT ≤ mean_lateness < 3m × EMPT	Q(4,1)	Q(4,2)	Q(4,3)
5	3m × EMPT ≤ mean_lateness < 4m × EMPT	Q(5,1)	Q(5,2)	Q(5,3)
6	4m × EMPT ≤ mean_lateness < 5m × EMPT	Q(6,1)	Q(6,2)	Q(6,3)
7	5m × EMPT ≤ mean_lateness < 6m × EMPT	Q(7,1)	Q(7,2)	Q(7,3)
8	6m × EMPT ≤ mean_lateness < 7m × EMPT	Q(8,1)	Q(8,2)	Q(8,3)
9	7m × EMPT ≤ mean_lateness < 8m × EMPT	Q(9,1)	Q(9,2)	Q(9,3)
10	8m × EMPT ≤ mean_lateness	Q(10,1)	Q(10,2)	Q(10,3)

When running the Q-learning algorithm, if the previous system state corresponds to a dummy state, updating Q(s, a) is unnecessary because the decision of taking next action is made without considering Q-learning algorithm. However, if the previous state is not a dummy state but the new state is one of the two dummy states (i.e., one job in queue), then an update in this situation must be treated differently because the Q(s, a) values for both dummy states is fixed at zero. The agent should still get the reward/penalty for such decisions, so under these circumstances, the Q(s, a) value is updated using the following equation instead of equation (3-1) (In Fig. 3.1.).

$$Q(s_0, a) = Q(s_0, a) + a\ r \qquad (3\text{-}2)$$

3.3.2. Factors for Developing the Reward Function

Factors C, D, and E are concerned with the development of an appropriate reward function. A reward function is guided based on the goal of the learning agent. In this study, the machine agent's goal is to minimize the mean tardiness of the finished jobs. Therefore, a job's tardiness is used to determine the amount of the reward or penalty for the agent's decision (dispatching rule selection). The tardier a job is, the greater the penalties assigned to the learning agent. The agent receives a reward only when the selected job is finished prior to or on its due date (tardiness is non-negative).

Factor C defines the number of ranges for determining the amount of reward/penalty. The use of more ranges in the reward function permits the reward or penalty associated with each decision the agent has made to be expressed more precisely. Using too few ranges results in the system not being able to differentiate between the decisions made by the agent in that the outcomes (measured by tardiness) are not

distinguishable since they lie within the same range and therefore result in the same penalty or reward.

Factor D determines the size of each range, and therefore, with a finite number of ranges, it also defines the overall range the reward function covers. Like factor B, each range is determined using a multiple (*n*) of the expected mean processing time (EMPT), which is also set to either 1 or 3. Similar to factor B, a large value of *n* for factor D permits distinguishing between jobs that are extremely tardy when the system is under heavy loading condition or employing some dispatching rules like SPT.

Factor E impacts the magnitude of the reward and penalty assigned to each range of the reward function. By design the penalty is made to increase linearly across the ranges as job tardiness grows. However, a reward is assigned only in the case which the job tardiness is zero. The question then becomes how much reward should be appropriate with respect to the linearly increasing penalties. In this experimental study, two values of factor E (1 or 10) are used for rewarding job that finish before their due date. The penalties applied to ranges associated with tardy jobs were fixed to permit us to study the influence of various rewards.

Factor E may impact the Q(s, a) values in the policy table. When the system is under heavy loading conditions or jobs are assigned with very tight due dates, most of the jobs will be tardy. The Q(s, a) values in the policy table may be all negative. Under such circumstances (very few early jobs), a decision for an early job is very important because it provides some positive amount (reward) for the Q(s, a) value. Using a larger reward for the decisions resulting in early jobs should more strongly influence the Q(s, a) values. Table 3.2 presents an example of a 10-range reward function.

Table 3.2. An Example of 10-range Reward Function

Range		Reward/Penalty
1	Tardiness = 0	r = 1 (or r = 10)
2	$0 <$ Tardiness $< n \times$ EMPT	r = -1
3	$n \times$ EMPT $\leq$ Tardiness $< 2n \times$ EMPT	r = -2
4	$2n \times$ EMPT $\leq$ Tardiness $< 3n \times$ EMPT	r = -3
5	$3n \times$ EMPT $\leq$ Tardiness $< 4n \times$ EMPT	r = -4
6	$4n \times$ EMPT $\leq$ Tardiness $< 5n \times$ EMPT	r = -5
7	$5n \times$ EMPT $\leq$ Tardiness $< 6n \times$ EMPT	r = -6
8	$6n \times$ EMPT $\leq$ Tardiness $< 7n \times$ EMPT	r = -7
9	$7n \times$ EMPT $\leq$ Tardiness $< 8n \times$ EMPT	r = -8
10	$8n \times$ EMPT $\leq$ Tardiness	r = -9

3.3.3. The Other Factors

When starting the Q-learning algorithm, the values of the state-action pairs, Q(s, a) can be initialized arbitrarily or assigned specific relative values to represent the confidence in favoring each possible alternative. Factor F represents the strategy of setting the initial values of the state-action pairs. In this study, all the values of the state-action pairs are initialized to zero since all the actions for each state are assumed to be an equally valid choice. This approach starts the system from a neutral state assuming no a priori knowledge of which dispatching rule is best to use in any situation. Therefore, the system would be required to learn from scratch. Other possible alternatives might have been to favor the wrong choice or correct choice initially. It is believed that either approach would have only impacted the run time making it take longer or shorter depending on how far off or close the initial values were to the best case.

Factor G is the step-size parameter, a, which is a small positive fraction that influences the learning rate. The value of this factor can be constant or varied from step

to step. In the latter case, the steps become smaller and smaller as learning progresses to assure convergence of Q(s, a) values. With a constant step-size parameter, the Q(s, a) values never completely converge but continue to vary in response to the most recently received rewards. This is more desirable for a non-stationary system (Sutton and Barto, 1999).

Factor H is the discount-rate parameter, γ. As γ approaches zero, the agent is more myopic because it takes immediate reward into account more strongly. On the other hand, as γ approaches 1, the agent will be more farsighted reducing the impact that recent results have on the learned policy.

Factor I concerns the approach for exploration and exploitation. The e–greedy method is adopted in this study. If e is set to 0.1, then 10% of the time the strategy will be to randomly select one of the three dispatching rules independent of their Q(s, a) values, while the other 90% of the time the dispatching rule with the best Q(s, a) value is selected.

Several example systems, such as those illustrated in Sutton and Barto (1999) apply the Q-learning algorithm with settings of a = 0.1, $\gamma = 0.9$, and $\varepsilon = 0.1$. This study uses these same common parameter settings for the three factors G, H, and I across all experimental runs. Table 3.3 summarizes the experimental factors and their levels.

Table 3.3. Experimental Factors and Their Levels

Experimental Factors	Level 1	Level 2
A. Number of states	10 states	20 states
B. Threshold value settings for determining state.	$m = 1$	$m = 3$
C. Number of ranges in reward function	10 ranges	20 ranges
D. Threshold value settings for reward function.	$n = 1$	$n = 3$
E. Reward magnitude	r = 1	r = 10

3.4. Design of Experiment

The purpose of this study is to identify the factors related to the application of the Q-learning algorithm that are significant when used by an agent for learning an appropriate policy for dispatching rule selection. The factors considered in experimentation and their levels are shown in Table 3.3. Testing involved using a simulation of a single-machine with an infinite buffer with no consideration of potential machine failures.

The simulation is conducted under four different sets of system conditions by varying the mean inter-arrival time of jobs to the system and due date tightness. The time between job arrivals to the system follows an exponential distribution with a mean of 8 representing a heavy loading condition and 10 for a light loading condition. The estimated processing times (EPT) of jobs were uniformly distributed between 6 and 8. The resulting mean system utilization is 87.5% under the heavy loading condition and 70% under the light loading condition. The due date of the job was determined based on the following equation:

$$\text{Due Date} = \text{Arrival time} + \text{Allowance factor} \times \text{EPT} \tag{3-3}$$

Due date tightness is controlled by adjusting the allowance factor. In this study, the allowance factor is drawn from the uniform distribution between 1.2 and 1.8, U[1.2, 1.8], for jobs with tight due dates and between 1.7 and 2.3, U[1.7, 2.3], for jobs with loose due dates. The real processing time (RPT) of each job was generated using a normal distribution with a mean of EPT and standard deviation of EPT/10. Given the possibility

that a normal distribution may generate an extreme value, the RPT values were constrained to be within ±3 times the standard deviation.

For each control factor combination setting used in the experiment, the learning horizon was monitored and analyzed to make sure that the learning process had reached steady state. As a result, a horizon of 200,000 job completions was determined as an appropriate run length under all conditions in order to guarantee that learning had reached steady state. After completing these 200,000 jobs as a system warm-up, 300,000 additional jobs are processed by the system and the mean tardiness of these additional jobs is calculated and recorded as a single observation for an experiment. A full factorial (2^5) experiment was conducted with ten replications under each of the four different system conditions (see Table 3.4).

Table 3.4. A full factorial (2^5) experiment is conducted under the following conditions

System Conditions	M/C Utilization	Allowance Factor
Heavy Loading/Tight Due Date (HT)	87.5 %	U[1.2, 1.8]
Heavy Loading/Loose Due Date (HL)	87.5 %	U[1.7, 2.3]
Light Loading/Tight Due Date (LT)	70 %	U[1.2, 1.8]
Light Loading/Loose Due Date (LL)	70 %	U[1.7, 2.3]

CHAPTER IV

Q-LEARNING FOR JOB ROUTING

4.1. Agent-Based Job Shop System

To develop a set of recommendations for applying the Q-learning algorithm to job routing problems, a simulated job shop system is used for examining the implementation of the Q-learning algorithm for use by agents when making routing decisions in such an environment. The control structure in this system is pure heterarchical and no supervisory agents are employed. There are only two types of agents in the system: job agents and machine cell agents. Each machine cell agent represents one machine cell that may be comprised of one (or more than one) identical machine. All the machines in the same cell share a buffer. Each job agent represents a specific job and is in charge of determining proper operation routing by negotiating with specific cell agents that have the potential to finish the operations. The agent negotiation scheme is based on the contract-net protocol.

In this study, every customer order is considered a job and consists of a batch of identical parts with each part comprised of a set number of features defined by the customer. Each feature requires one operation. Routing flexibility is available allowing a job agent to direct the manufacture of a product using alternative processing routes. These alternatives are taken into account in the process plan and arise due to an availability of multiple machine types for processing a specific operation. The following

subsections will detail what each job and cell agent's responsibilities are in the negotiation strategy.

4.1.1. Job Agent

Each job agent carries the process plan for the part it represents and this plan specifies the alternative routes. The job agent initially sends requests for bids to the cell agents that have the capability to process the job's next operation. The request indicates what feature is to be processed next. The job agent may send more than one request to the same cell agent if multiple features satisfy precedence and can be processed on the same machine cell. The cell agents immediately respond with their bids. Each bid contains information regarding the current status of the machine cell such the number of jobs in the buffer and how much work, in terms of the total processing time of the jobs in the buffer. After collecting the bids, the job agent evaluates them and selects one bid for the next operation. The selected bid identifies what operation will be processed on what machine cell next. After identifying the next machine cell, the job is routed there. If all the machines in the cell are busy, the job is placed in the buffer. Whenever a job's current operation is completed, the job agent sends bid requests for the next operation. This bidding procedure continues until all the requested features of a job are finished. We assume that the time delay due to the exchange of messages during negotiation can be ignored compared to the operation processing time.

4.1.2. Machine Cell Agent

Each cell agent is responsible for preparing bids and dispatching the jobs in the buffer to the next available machine in the cell. In this study, the cell agents use job

dispatching rules to select the jobs for processing from the buffer. Because dispatching rule selection is not our focus in this part of the study, FIFO (first in first out) is the only dispatching rule employed in this routing problem.

For bid preparation, the cell agent has knowledge about its capability in terms of what operations can be done at what pace. Using this knowledge, the cell agent is able to estimate the processing time for a bid. The cell agent is also able to detect the current status of its buffer in terms of its size and accumulate the processing times of the jobs in the buffer. With the information supplied by each bid response, in contract net negotiation, the job agent must decide which machine to use for a single operation. In this study, a job agent is able to evaluate the collected bids either based on the routing heuristic (NINQ) in which the machine cell with the fewest number of jobs in its queue is selected or based on the other heuristic (WINQ) where the machine cell with the least total estimated processing time of the jobs in its queue is selected. For both of these heuristics, a tie is broken by random selection.

4.2. Factors for Applying Q-learning to Job Routing

There are a number of factors that could be manipulated in applying the Q-learning algorithm. The research goal was to determine the significance of the various factors for this application. The factors that are investigated include the following:

A. State Determination Criteria.

B. Number of ranges for determining reward/penalty.

C. The threshold value settings for determining reward/penalty ranges.

D. Approaches to setting magnitude of reward/penalty.

E. Initial Q values in the policy table.

F. Step Size (a).

G. Discount rate (γ).

H. Approaches for exploration and exploitation.

Factor E, F, G, and H are the same as was discussed in Chapter 3. The factors for constructing the policy table and the reward functions are relatively different from the ones in Chapter 3 and are therefore described in the subsections that follow.

4.2.1. State Determination Criteria

A policy table is used by an agent to make decisions based on its current state. In the job routing problem, more specifically, the policy table for a job agent is a mapping from the job's current state to possible machines it can select for its next operation. To determine the current state, a job agent may only consider information it currently knows or that provided by the machine agents during negotiations. .

Factor A defines the state determination criteria used to construct the policy table. Three state determination criteria are considered in this study. One possible criterion to use is the type of feature (feature ID) to be created as some point in the part's processing. Table 4.1 presents an example of a policy table developed using the feature ID as the state determination criterion where features 1, 3, and 5 must be machined sequentially to complete the job according to the job's process plan. Using this criterion, two dummy states are needed in the policy table. The first dummy state represents the situation where the job has not started processing yet, while the other represents the situation when the job is complete. The actions, shown in column 4, define which specific machine cells are

available for processing the stated feature. The merit of these possible actions corresponds to the magnitude of the associated Q(s, a) value given in column 3 of the same table.

Table 4.1. A Policy Table Using Feature ID as the State Determination Criterion

	State Criteria	Q Values		Actions	
State	Feature ID	Action 1	Action2	Action 1	Action 2
Dummy	Not Started Processing	0	0	N/A	N/A
1	Processing Feature 1	Q(1, 1)	Q(1, 2)	M/C 1	M/C 3
2	Processing Feature 3	Q(2, 1)	Q(2, 2)	M/C 2	M/C 4
3	Processing Feature 5	Q(3, 1)	Q(3, 2)	M/C 1	M/C 5
Dummy	Completed Processing	0	0	N/A	N/A

Table 4.2. A Policy Table Using Feature ID and NIQ as the State Determination Criteria

	State Criteria		Q Values		Actions	
State	Feature ID	No. of Jobs in Queue	Action 1	Action 2	Action 1	Action 2
Dummy	Not Started Processing		0	0	N/A	N/A
1	1	$NIQ_1 < NIQ_3$	Q(1, 1)	Q(1, 2)	M/C 1	M/C 3
2		$NIQ_1 > NIQ_3$	Q(2, 1)	Q(2, 2)	M/C 1	M/C 3
3		$NIQ_1 = NIQ_3$	Q(3, 1)	Q(3, 2)	M/C 1	M/C 3
4	3	$NIQ_2 < NIQ_4$	Q(4, 1)	Q(4, 2)	M/C 2	M/C 4
5		$NIQ_2 > NIQ_4$	Q(5, 1)	Q(5, 2)	M/C 2	M/C 4
6		$NIQ_2 = NIQ_4$	Q(6, 1)	Q(6, 2)	M/C 2	M/C 4
7	5	$NIQ_1 < NIQ_5$	Q(7, 1)	Q(7, 2)	M/C 1	M/C 5
8		$NIQ_1 > NIQ_5$	Q(8, 1)	Q(8, 2)	M/C 1	M/C 5
9		$NIQ_1 = NIQ_5$	Q(9, 1)	Q(9, 2)	M/C 1	M/C 5
Dummy	Completed Processing		0	0	N/A	N/A

Besides feature ID, information provided by machine cell agents such as the number of jobs in the buffer or total work in the buffer could also be employed as a state determination criterion. An example of a policy table for determining state using both the processing feature and the number of jobs in the buffer is shown in Table 4.2. Assume that a job receives two bids (one from cell 2 and the other from cell 4) for processing

feature 3. Given that NIQ_2 and NIQ_4 denote the number of jobs in the buffer of cell 2 and cell 4, respectively, if NIQ_2 is greater than NIQ_4, the job's new state will be state 5 with possible actions involving the use of machine 2 or machine 4 to process that feature.

With the capability of estimating the processing time of each job, the cell agent is able to estimate the total work (in terms of processing time) represented by the jobs in its buffer. WIQ_i denotes the total estimated processing time of the jobs in the buffer of cell i. Table 4.3 presents an example of a policy table where the state determination criteria is based on both the feature type and the estimated total work in the buffer. For an operation, due to a variety of machine capability, the *WIQ* values provided by various machine cells are hardly the same except when there are no jobs in their buffers. For cases when the *WIQ* values are very close, it is hard to determine which machine cell the job should be routed to because the *WIQ* values are only estimates. To overcome this issue, the relative difference between the two *WIQ* values must exceed some threshold in order for difference to be considered distinct. Suppose that cell x and cell y are able to perform the same operation and each cell responds with its estimate of the total estimated processing time of buffered jobs as WIQ_x and WIQ_y, respectively. If $AWIQ_{xy}$ denotes the average of these two *WIQ* values and $DWIQ_{xy}$ denotes the absolute value of the difference of these two *WIQ* values, then a ratio, $DIFF_{xy}$, indicating the difference of the two WIQ values to their mean value can be defined as the quotient of $DWIQ_{xy}$ and $AWIQ_{xy}$ as follows.

$$AWIQ_{xy} = (WIQ_x + WIQ_y)/2 \quad (4\text{-}1)$$
$$DWIQ_{xy} = | WIQ_x - WIQ_y | \quad (4\text{-}2)$$
$$DIFF_{xy} = DWIQ_{xy} / AWIQ_{xy} \quad (4\text{-}3)$$

To set the threshold value, accuracy of the processing time estimates provided by the machine cell agents must be considered. The more accurate the time estimates, the

smaller the value of the threshold. If no prior knowledge is available regarding the accuracy of the estimates, this threshold value may be set arbitrarily. However, setting too large a value may result in degradation of system performance. We set the threshold value at 10% of the average of the two *WIQ* values. Since the approach used here is unique, there is no prior study or reference regarding how to set this threshold value. A better threshold value setting may be possible. Searching for the best setting for this threshold value is beyond the scope of this research.

Table 4.3. A Policy Table Using Feature ID and WIQ as the State Determination criteria

	State Criteria		Q Values		Actions	
State	Feature ID	Total Work in Queue	Action 1	Action 2	Action 1	Action 2
Dummy	Not Started Processing		0	0	N/A	N/A
1	1	$DIFF_{13} > 0.1$ and $WIQ_1 < WIQ_3$	Q(1, 1)	Q(1, 2)	M/C 1	M/C 3
2		$DIFF_{13} > 0.1$ and $WIQ_1 > WIQ_3$	Q(2, 1)	Q(2, 2)	M/C 1	M/C 3
3		$DIFF_{13} < 0.1$	Q(3, 1)	Q(3, 2)	M/C 1	M/C 3
4	3	$DIFF_{24} > 0.1$ and $WIQ_2 < WIQ_4$	Q(4, 1)	Q(4, 2)	M/C 2	M/C 4
5		$DIFF_{24} > 0.1$ and $WIQ_2 > WIQ_4$	Q(5, 1)	Q(5, 2)	M/C 2	M/C 4
6		$DIFF_{24} < 0.1$	Q(6, 1)	Q(6, 2)	M/C 2	M/C 4
7	5	$DIFF_{15} > 0.1$ and $WIQ_1 < WIQ_5$	Q(7, 1)	Q(7, 2)	M/C 1	M/C 5
8		$DIFF_{13} > 0.1$ and $WIQ_1 > WIQ_5$	Q(8, 1)	Q(8, 2)	M/C 1	M/C 5
9		$DIFF_{15} < 0.1$	Q(9, 1)	Q(9, 2)	M/C 1	M/C 5
Dummy	Completed Processing		0	0	N/A	N/A

To illustrate the use of this measure, suppose that a job agent is making a decision for selecting either machine cell 2 or cell 4 for machining feature 3. Machine cell 2 provided its *WIQ* value as 23.56 and machine cell 4 provided its *WIQ* values as 27.42. The desired computations are now carried out (Equation 4-1, 4-2, and 4-3):

$AWIQ_{24} = (WIQ_2 + WIQ_4)/2 = (23.56 + 27.42)/2 = 25.49$
$DWIQ_{24} = | WIQ_2 - WIQ_4 | = | 23.56 - 27.42 | = 3.86$
$DIFF_{24} = DWIQ_{24} / AWIQ_{24} = 3.86/25.49 = 0.15 > 0.1$

Since $DIFF_{24} > 0.1$ and $WIQ_2 < WIQ_4$, then state 4 is determined based on Table 4.3. If two *WIQ* values are close ($DIFF_{24} < 0.1$), the estimated work in both queues is considered equal. This research examines how use of any of the above three various criteria impact the performance of the routing decisions of job agents.

4.2.2. Factors for Developing the Reward Function

Similar to Factor C, D, and E in Chapter 3, the factors (B, C, and D) are used for developing the reward function. In this study, the system's overall goal is to minimize the mean tardiness of the finished jobs. The objective of each job agent is to finish the required operations before the final due date. The job agent has the knowledge of the estimated processing time for each operation which is calculated as the average of the estimated processing times of the machine cells that are able to perform that operation. Therefore, the job agent is able to estimate the total processing time for completing the job. With the due date and the estimated total processing time, the job agent can determine the allowance factor by using the following equation:

Allowance factor = (due date – arrival time)/ estimated total processing time

Based on the allowance factor, the job agent can determine the intermediate due date for each required operation. The intermediate due date is used by the job agent to check if the corresponding operation to this intermediate due date is behind. . The goal of the job agent is to route the corresponding job to meet the intermediate due date of every required operation. If the machine cell selected by the job agent finishes the operation

before the intermediate due date, the learning agent receives a reward for this action (routing selection); otherwise, the job agent received a penalty.

As with the factors of reward function development in Chapter 3, Factor B defines the number of ranges for determining the amount of reward/penalty, and Factor D determines the size of the interval for each range. Factor E is concern with assigning the magnitude of the reward and penalty to each range of the reward function. As in Chapter 3, we assumed that the machine agent has knowledge about the expected mean processing time (EMPT) of the operations that it is able to perform and uses EMPT as a measure to set the tardiness ranges for the reward function. Table 4.4 presents an example of a 10-range reward function.

Table 4.4. An Example of 10-range Reward Function

Range		Reward/Penalty
1	Tardiness = 0	r = 1 (or r = 10)
2	$0 <$ Tardiness $< n \times$ EMPT	r = -1
3	$n \times$ EMPT $\leq$ Tardiness $< 2n \times$ EMPT	r = -2
4	$2n \times$ EMPT $\leq$ Tardiness $< 3n \times$ EMPT	r = -3
5	$3n \times$ EMPT $\leq$ Tardiness $< 4n \times$ EMPT	r = -4
6	$4n \times$ EMPT $\leq$ Tardiness $< 5n \times$ EMPT	r = -5
7	$5n \times$ EMPT $\leq$ Tardiness $< 6n \times$ EMPT	r = -6
8	$6n \times$ EMPT $\leq$ Tardiness $< 7n \times$ EMPT	r = -7
9	$7n \times$ EMPT $\leq$ Tardiness $< 8n \times$ EMPT	r = -8
10	$8n \times$ EMPT $\leq$ Tardiness	r = -9

4.3. Design of Experiment

The factors considered in experimentation and their levels are shown in Table 4.5. To examine these factors, a simulation of a small-sized job shop system consisting of ten machines is carried out to measure effectiveness of these factors. This simulation model

is programmed in Visual C++ and implemented on a personal computer installed with Intel Pentium 4 2.8GHz CPU. The job shop system is comprised of five types of machine cells, each consisting of from one to three identical machines. There is only one buffer associated with each machine cell. At most seven variant features can be machined in this shop. The system can manufacture only three different jobs (type A, B, and C).

Table 4.5 Experimental Factors and Their Levels

Experimental Factors	Level 1	Level 2	Level 3
A. State Determination Criteria	Feature ID	Feature ID and NIQ	Feature ID and WIQ
B. Number of ranges in reward function	10 ranges	20 ranges	
C. Threshold value settings for reward function.	n = 0.1	n = 0.15	
D. Reward magnitude	r = 1	r = 10	

Table 4.6. Process Plan

Job Type	Required Features (Fixed Sequencing)	Alternative Machine	Alternative Machine
A	1	1	3
	3	2	4
	5	1	5
B	2	4	5
	4	1	3
	7	3	5
C	1	1	3
	2	4	5
	6	2	4

Job arrivals follow a Poisson distribution and the type of job is defined following a uniform distribution. Each job type requires three features and the process plan of each job type is presented in Table 4.6. Job type B and type C are routed using the NINQ routing heuristic where jobs are routed by selecting the machine cell that has the fewest

number of jobs in its buffer. Ties are broken using random selection. Only job type A employs the Q-learning algorithm to learn a routing policy. All machine cell agents employ FIFO to select a job from the buffer for processing. The batch size for each job is uniformly distributed between twenty and seventy units.

Table 4.7. Machine Capability

Machine ID	Number of Machines	Feature ID	Estimated Setup Time	Estimated Processing Time
1	2	1	0.3	0.22
		4	0.1	0.17
		5	0.5	0.31
2	1	3	0.5	0.21
		6	0.3	0.2
3	2	1	0.1	0.17
		4	0.4	0.19
		7	0.7	0.15
4	3	2	0.2	0.15
		3	0.3	0.22
		6	0.5	0.25
5	2	2	0.3	0.17
		5	0.27	0.11
		7	0.4	0.17

Table 4.7 shows the information about machine capability. All the setup and processing times are estimates of experienced engineers. These values were used as the mean values of the normal distribution used to generate the values used in the simulation runs. The following assumptions are also made for the simulation.

1. Each machine can process only one operation at a time.
2. Each job is released to the system immediately after arrival.
3. Individual operations are not preemptable.

4. Set-up and processing times for each operation are not deterministic, but their expected values are available.
5. The job may revisit the same machine cell before completing all its manufacturing steps.
6. No machine breakdown occurs.
7. Transportation times between machines are not considered.
8. Set-up times are sequence-independent.

The simulation is conducted under four different sets of system conditions by varying the mean inter-arrival time of jobs to the system and due date tightness. In the dispatching rule selection problem of Chapter 3, 70% and 87.5% machine utilization was set for light and heavy loading condition, respectively. To make the system conditions consistent, the time between job arrivals to the system follows an exponential distribution with a mean of 3 representing a heavy loading condition and 4 for a light loading condition in this case. Use of these values results in a utilization of the bottleneck machine cell of around 90% under the heavy loading condition and 70% under the light loading condition. The due date of each job is determined based on the following equation:

Due Date = Arrival time + Allowance factor × Total Expected Processing Time

Due date tightness is controlled by adjusting the allowance factor. In this study, an allowance factor is drawn from a uniform distribution between 1.2 and 1.8, U[1.2, 1.8], for jobs with tight due dates and between 1.7 and 2.3, U[1.7, 2.3], for jobs with loose due dates. The total expected processing time of a job is the sum of estimated mean processing times of its required operations. The mean processing time of each operation

is estimated by averaging the estimated processing times (EPT in Table 4.7) provided by different machine cell agents that are able to machine the same feature. The real processing time (RPT) of each operation used in the simulation is generated using a normal distribution with a mean of EPT and standard deviation of EPT/10. Given the possibility that a normal distribution may generate an extreme value, the RPT values were constrained to be within ±3 times the standard deviation.

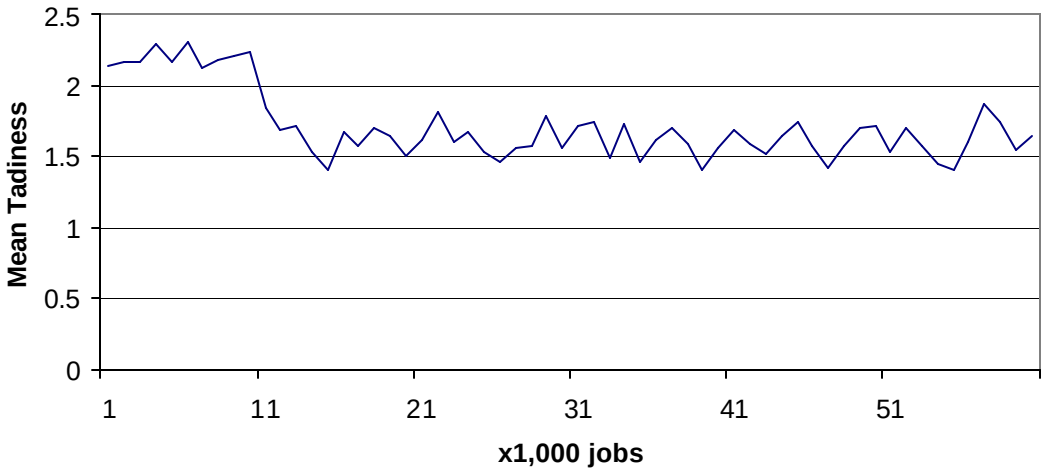

Figure 4.1. An Example of the Observed Learning Progress (with Setting A3_BCd)

For each control factor combination setting used in the experiment, the mean tardiness of every 1000 completed type-A jobs is monitored to determine the sufficient warm-up period for the system. 10,000 type-A job completions was determined as sufficient for a system warm-up period for all four system conditions. All these jobs are routed using the NINQ heuristic during this period. Figure 4.1 presents a plot of the mean tardiness for every 1000 type-A jobs observed for an experimental run. This plot illustrates the change of the mean tardiness as learning progresses. After completing these

10,000 type-A jobs, type-A jobs are routed based on Q-learning algorithm, while job type B and C are routed using the NINQ heuristic. A horizon of the next 20,000 type-A job completions was monitored and determined as an appropriate run length in order to guarantee that learning had reached steady state. After completing these 30,000 type-A jobs, the next 30,000 jobs completion are processed by the system and the mean tardiness of these additional jobs is calculated and recorded as a single observation for an experiment. A 3×2×2×2 factorial experiment was conducted with ten replications under each of the four different system conditions (see Table 4.8).

Table 4.8. A 3×2×2×2 factorial experiment is conducted under the following conditions

System Conditions	Inter-arrival Time	Allowance Factor
Heavy Loading/Tight Due Date (HT)	3	U[1.2, 1.8]
Heavy Loading/Loose Due Date (HL)	3	U[1.7, 2.3]
Light Loading/Tight Due Date (LT)	4	U[1.2, 1.8]
Light Loading/Loose Due Date (LL)	4	U[1.7, 2.3]

CHAPTER V

EXPERIMENTAL RESULTS

5.1. The Single-Machine Dispatching Rule Selection Problem

5.1.1. Experimental Results

Table 5.2 provides a summary of the experimental results for the single-machine dispatching rule selection problem. Each value in this table represents the mean of ten replications for each experimental run involving the factor settings defined in Table 3.3. For each system condition, analysis of variance (ANOVA) is used to identify strong effects and their interactions on a response at a level of significance of 0.05. These significant interactions are presented in Table 5.1.

Table 5.1. Significant Interaction found by ANOVA

System Conditions	Significant Interactions (α = 0.05)							
HT		BCD				ABDE		
HL	A		BDE	CDE				
LT						ABDE	ABCE	BCDE
LL		BCD	BDE	CDE	ADE		ABCE	

The ANOVA results indicated that primarily only various combinations of higher-order interactions were significant with no combination common across all system conditions. To further investigate the best factor level combination, Duncan's multiple range test (at 0.05 level of significance) was applied to each identified

significant interaction. The ANOVA and Duncan's test in this research were conducted by SAS software. For the significant interactions under the HT system condition (high load with tight due dates), Table 5.3 shows the results of Duncan test. In the table, a lowercase letter for a factor represents the level 1 setting for that factor while an uppercase letter indicates the level 2 setting. For each significant interaction, Duncan's test is testing every pair of means for all the possible factor settings. The Duncan grouping letters in the table indicate if there is a significant difference between a pair of means for the factor settings. For example, there is no significant difference between factor setting AbDe and abDe since their corresponding Duncan grouping letters are the same (group A), while there is a significant difference between factor setting abDe and aBDe since their grouping letters are different (abDe – group A, aBDe – group B). In the up-left cell of Table 5.3, the best group for the interaction of control factors A, B, D, and E, found is group J in which all the factors are at level 2. For the significant interaction of control factors B, C, and D, the best group identified by the Duncan test consists of four settings BCD, BCd, bCd, and bcd. There is no significant difference found between these four settings. According to the best groups a common factor level setting (ABCDE) can be concluded and used as the recommended control factor level combination for system condition HT. Going through this same process for the other system conditions, HL, LT, and LL, the results of Duncan tests are presented in Table 5.4, Table 5.5, and Table 5.6, respectively. Table 5.7 summaries the favorable settings found for each of the four system conditions.

Table 5.2. Experimental Results of Single-Machine Dispatching Rule Selection Problem (0: level 1, 1: level 2)

Experiment No.	A	B	C	D	E	HT	HL	LT	LL
1	0	0	0	0	0	20.533	18.444	5.884	4.437
2	0	0	0	0	1	20.463	18.457	5.855	4.458
3	0	0	0	1	0	20.654	18.478	5.921	4.464
4	0	0	0	1	1	20.461	18.453	5.861	4.453
5	0	0	1	0	0	20.529	18.424	5.888	4.403
6	0	0	1	0	1	20.464	18.456	5.858	4.460
7	0	0	1	1	0	20.652	18.469	5.914	4.448
8	0	0	1	1	1	20.459	18.444	5.856	4.454
9	0	1	0	0	0	20.542	18.450	5.891	4.434
10	0	1	0	0	1	20.470	18.463	5.856	4.457
11	0	1	0	1	0	20.584	18.456	5.909	4.460
12	0	1	0	1	1	20.455	18.453	5.860	4.455
13	0	1	1	0	0	20.548	18.434	5.894	4.407
14	0	1	1	0	1	20.470	18.461	5.860	4.462
15	0	1	1	1	0	20.585	18.476	5.907	4.441
16	0	1	1	1	1	20.443	18.445	5.856	4.455
17	1	0	0	0	0	20.532	18.439	5.889	4.426
18	1	0	0	0	1	20.463	18.448	5.853	4.445
19	1	0	0	1	0	20.655	18.464	5.940	4.449
20	1	0	0	1	1	20.460	18.443	5.859	4.442
21	1	0	1	0	0	20.528	18.413	5.884	4.394
22	1	0	1	0	1	20.458	18.447	5.856	4.447
23	1	0	1	1	0	20.657	18.454	5.911	4.435
24	1	0	1	1	1	20.454	18.435	5.853	4.443
25	1	1	0	0	0	20.516	18.447	5.881	4.432
26	1	1	0	0	1	20.454	18.452	5.853	4.444
27	1	1	0	1	0	20.544	18.448	5.895	4.449
28	1	1	0	1	1	20.446	18.442	5.857	4.443
29	1	1	1	0	0	20.524	18.426	5.882	4.409
30	1	1	1	0	1	20.458	18.447	5.857	4.448
31	1	1	1	1	0	20.541	18.438	5.891	4.439
32	1	1	1	1	1	20.434	18.438	5.854	4.443

Table 5.3. The Results of Duncan's Test for System Condition HT (a = 0.05)

Significant Interaction: ABDE

Duncan Grouping	Mean	Factor Setting
A	20.655780	AbDe
A		
A	20.652935	abDe
B	20.584795	aBDe
C	20.544890	aBde
C		
C	20.542550	ABDe
D	20.531015	abde
D		
D	20.530000	Abde
E	20.520050	ABde
F	20.470065	aBdE
G	20.463285	abdE
G		
H G	20.460425	AbdE
H G		
H G	20.460020	abDE
H G		
H G	20.457215	AbDE
H		
H	20.455950	ABdE
I	20.448890	aBDE
J	20.440020	ABDE

Significant Interaction: BCD

Duncan Grouping	Mean	Factor Setting
A	20.55764	bcD
A		
A	20.55534	bCD
B	20.50406	BCD
B		
B	20.49774	BCd
B		
B	20.49749	bcd
B		
B	20.49488	bCd

Recommended Factor Setting: ABCDE

Table 5.4. The Results of Duncan's Test for System Condition HL (a = 0.05)

Significant Interaction: BDE			Significant Interaction: CDE		
Duncan Grouping	Mean	Factor Setting	Duncan Grouping	Mean	Factor Setting
A	18.466173	bDe	A	18.461683	cDe
			A		
B	18.455680	BdE	A	18.459258	CDe
B			A		
B	18.454768	BDe	B A	18.455155	cdE
B			B A		
C B	18.451988	bdE	B A C	18.452513	CdE
C			B C		
C D	18.444490	BDE	B D C	18.447763	cDE
C D			D C		
C D	18.443750	bDE	D C	18.445045	cde
D			D		
D	18.439418	Bde	D	18.440478	CDE
E	18.430098	bde	E	18.424470	Cde
Recommended Factor Setting: AbCde					

Table 5.5. The Results of Duncan's Test for System Condition LT (a = 0.05)

Significant Interaction: ABCE			Significant Interaction: ABDE			Significant Interaction: BCDE		
Duncan Grouping	Mean	Factor Setting	Duncan Grouping	Mean	Factor Setting	Duncan Grouping	Mean	Factor Setting
A	5.914409	Abce	A	5.925640	AbDe	A	5.930400	bcDe
B	5.902350	abce	B	5.917450	abDe	B	5.912690	bCDe
B								
B	5.901245	abCe	C	5.907903	aBDe	C	5.901987	BcDe
B						C		
B	5.900314	aBCe	D	5.892848	ABDe	C	5.898764	BCDe
B			D					
B	5.900240	aBce	D	5.892651	aBde	D	5.888176	BCde
B						D		
B	5.897646	AbCe	E	5.886415	Abde	D	5.886443	Bcde
			E			D		
C	5.888190	ABce	E	5.886145	abde	D	5.886359	bcde
C						D		
C	5.886626	ABCe	F	5.881968	ABde	D	5.886201	bCde
D	5.858200	aBcE	G	5.858280	abDE	E	5.859890	bcDE
D			G			E		
D	5.857725	aBCE	G	5.857963	aBdE	F E	5.858596	BcDE
D			G			F E		
D	5.857550	abcE	G	5.857962	aBDE	F E	5.858298	BCdE
D			G			F E		
D	5.856855	abCE	G	5.856300	AbDE	F E	5.856636	bCdE
D			G			F		
D	5.856210	AbcE	G	5.856125	abdE	F	5.854890	BCDE
D			G			F		
D	5.855462	ABCE	G	5.855523	ABDE	F	5.854775	BcdE
D			G			F		
D	5.855170	ABcE	G	5.855109	ABdE	F	5.854689	bCDE
D			G			F		
D	5.854471	AbCE	G	5.854381	AbdE	F	5.853870	bcdE
Recommended Factor Setting: abcdE, abCdE, abCDE, aBcdE, aBcDE, aBCdE, aBCDE, AbcdE, AbCdE, AbCDE, ABcdE, ABcDE, ABCdE, and ABCDE.								

Table 5.6. The Results of Duncan's Test for System Condition LL (a = 0.05)

Significant Interaction: ABCE

Duncan

Grouping	Mean	Factor Setting
A	4.458381	aBCE
A		
A	4.456660	abCE
A		
A	4.455841	aBcE
A		
A	4.455456	abcE
A		
B A	4.450227	abce
B		
B C	4.447011	aBce
B C		
B C D	4.445651	ABCE
B C D		
B C D	4.444895	AbCE
B C D		
B C D	4.443857	ABcE
B C D		
B C D	4.443710	AbcE
C D		
C D	4.440630	ABce
D		
D	4.437370	Abce
E	4.425476	abCe
E		
E	4.424357	ABCe
E		
E	4.423803	aBCe
F	4.414230	AbCe

Significant Interaction: ADE

Duncan

Grouping	Mean	Factor Setting
A	4.459071	adE
B	4.454097	aDE
B		
B	4.453107	aDe
C	4.446260	AdE
C		
C	4.443005	ADe
C		
C	4.442796	ADE
D	4.420151	ade
E	4.415288	Ade

Significant Interaction: BDE

Duncan

Grouping	Mean	Factor Setting
A	4.452906	BdE
A		
B A	4.452426	bdE
B A		
B A	4.448959	BDE
B A		
B A	4.448652	bDe
B A		
B A	4.447934	bDE
B		
B	4.447460	BDe
C	4.420440	Bde
D	4.414999	bde

Significant Interaction: CDE

Duncan

Grouping	Mean	Factor Setting
A	4.455438	cDe

Significant Interaction: BCD

Duncan

Grouping	Mean	Factor Setting
A	4.451953	bcD

Table 5.6. (continued).

A				A	4.451953	bcD	
A	4.454273	CdE		A			
				A	4.451859	BcD	
B	4.451059	cdE		A			
B			B	A	4.444634	bCD	
B	4.448520	CDE	B	A			
B			B	A	4.444560	BCD	
B	4.448373	cDE	B				
			B		4.441811	Bcd	
C	4.440674	CDe	B				
			B		4.441429	bcd	
D	4.432181	cde					
				C	4.431535	BCd	
E	4.403259	Cde		C			
				C	4.425997	bCd	
Recommended Factor Setting: AbCde							

Table 5.7. Best Factor Level Combinations for Various System Conditions

Conditions	Significant Interactions	Best Factor Level Combinations
HT	ABDE, BCD	ABCDE
HL	A, BDE, CDE	AbCde
LT	ABDE, ABCE, BCDE	abcdE, abCdE, abCDE, aBcdE, aBcDE, aBCdE, aBCDE, AbcdE, AbCdE, AbCDE, ABcdE, ABcDE, ABCdE, and ABCDE.
LL	ABCE, BDE, ADE, CDE, BCD	AbCde

As a basis of another comparison, the performance of the system was determined while operating under each one of the three dispatching rules (EDD, SPT, or FIFO). These results were compared with the Q-learning algorithm using the recommended factor settings. Table 5.8 shows the resulting system performance for each case under each of the four system conditions. Of the three dispatching rules, SPT was identified as the favored rule for system conditions HT, HL, and LT, while EDD outperformed the other two rules for system condition LL. These results align with the scheduling strategy

prescribed by Morton and Pentico (1993) based on their study of several heuristics for a static single-machine problem. They found that to minimize tardiness, one should schedule lightly loaded shops using EDD and schedule heavily loaded shops using WSPT.

Table 5.8. Results of using the individual Dispatching Rules and the Q-learning algorithm.

Dispatching Rules\Conditions	HT	HL	LT	LL
EDD	21.907	19.234	6.031	4.292
SPT	20.298	18.422	5.831	4.499
FIFO	21.966	19.319	6.071	4.354
Q-Learning (Recommended Factor Setting)	20.434 (ABCDE)	18.413 (AbCde)	5.854 (ABCDE)	4.394 (AbCde)
EDD Selection Percentage	4.05%	10.29%	4.99%	15.03%
SPT Selection Percentage	91.66%	79.87%	89.67%	69.65%
FIFO Selection Percentage	4.29%	9.84%	5.34%	15.32%

When the Q-learning algorithm was applied with the recommended factor settings, the learning agent yielded the best performance for one (HL) of the four system conditions. However, in three of the four cases, the resulting policy derived by Q-learning favored the best rule for the condition. It selected the SPT rule 91.7%, 79.9%, and 89.7% of the time for system condition HT, HL, and LT, respectively, but selected SPT only 69.7% of the time for system condition LL. SPT is the best among these three rules for minimizing the number of tardy jobs. (Under system condition LL, the percentage of jobs that reported as tardy using EDD, SPT, and FIFO is 41.6%, 28.1%, and 41.5%, respectively.) However, for minimizing mean tardiness (the measure used here), SPT may cause some jobs with long processing times to be very tardy causing the overall mean tardiness to be worse even though there are only a few tardy jobs. In the reward function, only non-tardy jobs receive a reward, therefore it is not surprising that the

selection percentage of SPT for the LL case is this high. A high selection percentage for SPT means that, most of the time, the Q value representing the action of selecting SPT is larger than the Q values for the other two rules. Given that the performance of the system when employing either EDD or FIFO are nearly the same, it is not surprising that the selection percentages for EDD and FIFO are so close for all the four system conditions. For the LL case, if the reward function is modified to assign larger penalties to the actions causing jobs to be very tardy, then the EDD selection percentage may come out on top.

5.1.2. Discussion

Given prior success at applying Q-learning for the dispatching rule selection problem (Wang and Usher, 2002), this study conducted a factorial experiment for studying the factors important to the design and implementation of the Q-learning algorithm to the single machine dispatching rule selection problem. According to the results in Table 5.7, it is better to design the policy table with more states (control factor A) and the reward function with more ranges (control factor C) independent of the due date tightness when the system is under heavy loading conditions. With the mean lateness of the jobs in the buffer as the state determination, the number of states can be infinite. Then a large amount of memory may be required to build up approximations of the value functions. Although the tabular method (arrays or tables with one entry for each state) in this study is much simpler and easier to implement, the experimental results reveal that more states are better. Therefore, using the function approximation approach instead of the tabular method is suggested.

Based on the experimental results, the ranges for determining the states (control factor B) and penalties (control factor D) should be wider when job due dates are tight. This is because the tight due date setting may result in some jobs being very tardy, particularly when applying SPT as the selection rule. The use of wider ranges (control factor B and D) permits the system agent to better distinguish the different jobs at these higher tardiness levels providing a more accurate identity of the real system status. Also, a reward function that is more able to distinguish between the various levels of the tardy jobs provides more accurate responses regarding the agent's decisions.

Also, under the condition with tight due date jobs, it is better to assign more reward (control factor E) to the action for early jobs. When most of the completed jobs register as tardy, a lot of the Q values in the policy table accumulate and become very large negative values. Hence, the reward magnitude (a positive value) becomes important because it is better able to provide a larger impact when a proper action is selected. The experimental results indicate that the best factor level combinations found for the conditions with loose due dates (system condition HL and LL) are the same and favor more states with narrower ranges for the policy table and likewise for the reward function.

5.2. The Ten Machine Job Routing Problem

5.2.1 Experimental Results

The experimental results for the ten-machine job routing problem are presented in Table 5.10. Each result value in the table represents the mean of ten replications for each experimental run involving the factor settings defined in Table 4.5. Again, ANOVA is used to identify strong effects and their interactions on a response for each system

condition. Based on the results of ANOVA at significance level of 0.05, the significant interactions detected are shown in Table 5.9.

For the significant interactions for system condition HT, the interaction plots (Figures 5.1, 5.2, and 5.3) show that level 2 setting for factor B and factor C, and level 1 setting for factor D are good when factor A is set at level 1, while any setting for factors B, C, and D would be fine if factor A is set at level 3. In the figures, A1, A2, and A3 represent the level 1, level 2, and level 3 setting, respectively, for factor A. For factors B, C, and D, a lowercase letter represents the level 1 setting for that factor while an uppercase letter indicates use of the level 2 setting. Figure 5.4 shows that level 2 for factor B and level 1 for factor D are good settings for interaction BD.

These same results can be found by applying Duncan's test. Table 5.11 shows the results of Duncan's test (at 0.05 level of significance) for these four interactions. Overall, A3_BCd (Level 3 for factor A, level 2 for factor B, level 2 for factor C, and level 1 for factor D) can be concluded as the recommended factor level combination for system condition HT. The results of Duncan's test for the other system conditions (HL, LT, and LL) are presented in Table 5.12, Table 5.13, and Table 5.14, respectively. For system condition HL and LL, A3_bCd and A3BCd are identified as the recommended settings, while A3_BCd is recommended for system condition LT. Table 5.15 summarizes these favorable settings found for each of the four system conditions.

Table 5.9. Significant Interaction found by ANOVA

System Conditions	Significant Interactions ($\alpha = 0.05$)						
HT	AB	AC	AD	BD			
HL		AC	AD	BD			
LT	AB	AC	AD		CD		
LL						ABD	ACD

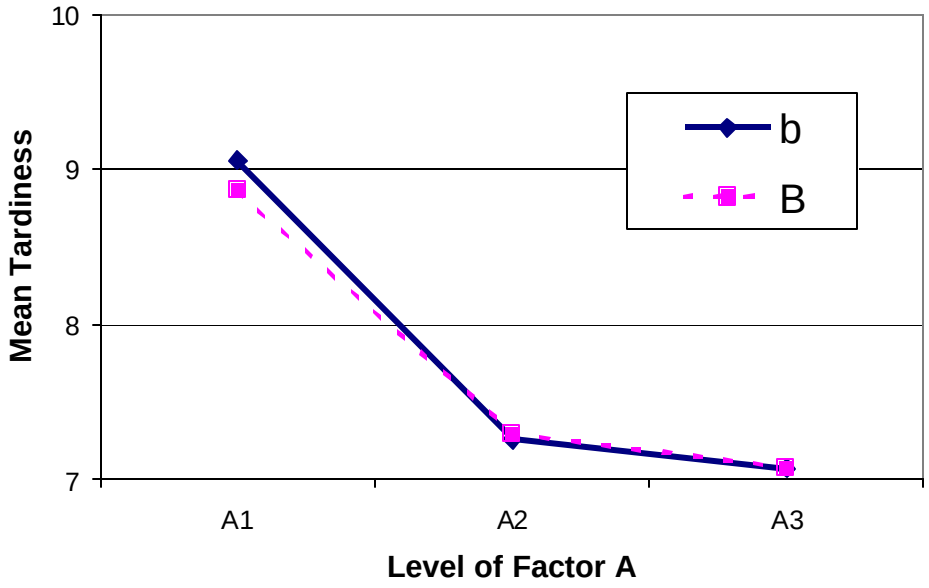

Figure 5.2. A-B Interaction under HT condition

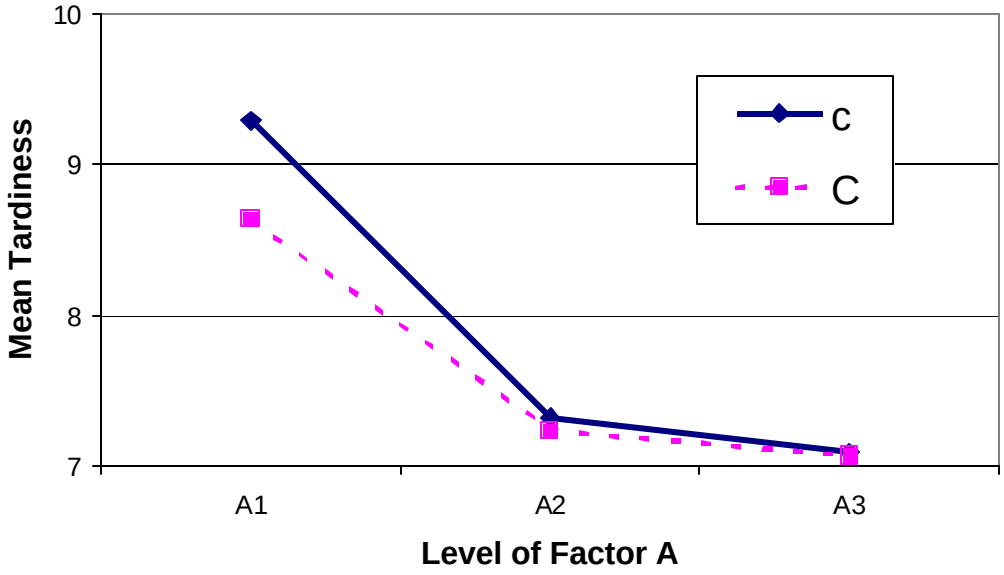

Figure 5.3. A-C Interaction under HT condition

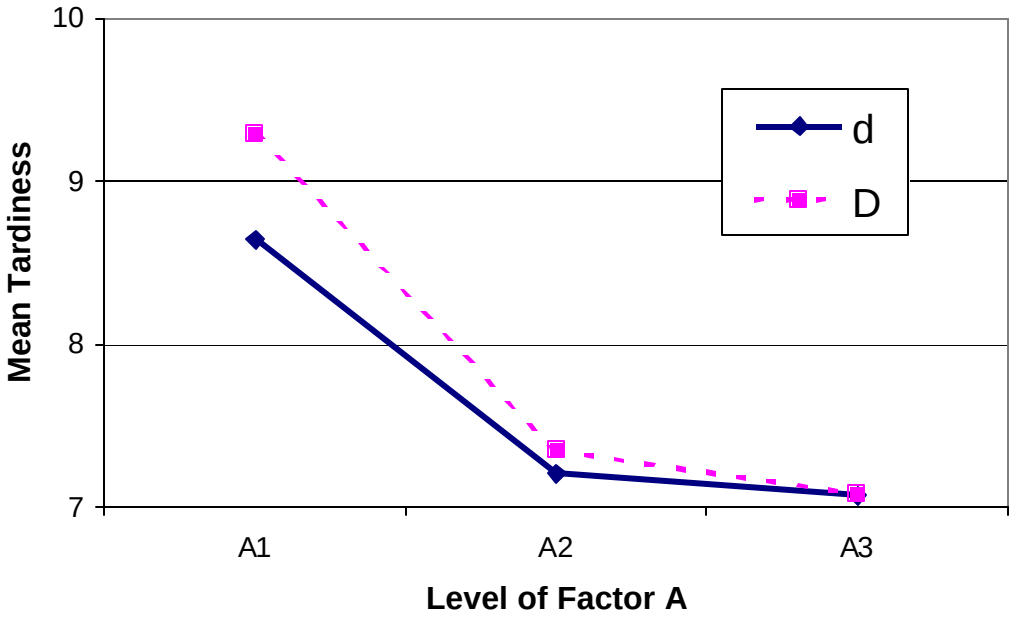

Figure 5.4. A-D Interaction under HT condition

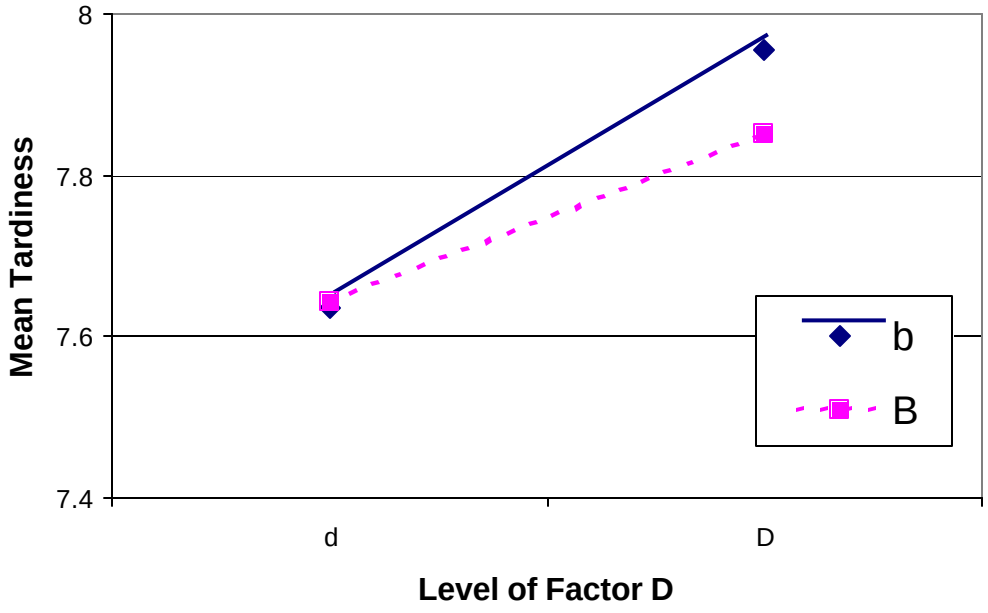

Figure 5.5. B-D Interaction under HT condition

Table 5.10. Experimental Results of Job Routing Problem
(0: level 1, 1: level 2, 2: level 3)

Experiment No.	A	B	C	D	HT	HL	LT	LL
1	0	0	0	0	9.017	4.622	2.275	0.794
2	0	0	0	1	9.708	5.186	2.428	0.875
3	0	0	1	0	8.367	4.327	2.136	0.726
4	0	0	1	1	9.14	4.947	2.418	0.869
5	0	1	0	0	8.919	4.605	2.241	0.781
6	0	1	0	1	9.511	5.044	2.334	0.804
7	0	1	1	0	8.262	4.326	2.104	0.747
8	0	1	1	1	8.787	4.705	2.297	0.79
9	1	0	0	0	7.252	3.462	1.889	0.45
10	1	0	0	1	7.406	3.627	1.933	0.471
11	1	0	1	0	7.067	3.442	1.917	0.442
12	1	0	1	1	7.294	3.465	1.966	0.462
13	1	1	0	0	7.225	3.513	1.888	0.458
14	1	1	0	1	7.395	3.621	1.913	0.456
15	1	1	1	0	7.276	3.399	1.845	0.441
16	1	1	1	1	7.289	3.563	1.943	0.431
17	2	0	0	0	7.017	3.34	1.819	0.433
18	2	0	0	1	7.155	3.43	1.875	0.444
19	2	0	1	0	7.088	3.257	1.803	0.425
20	2	0	1	1	7.031	3.387	1.885	0.441
21	2	1	0	0	7.132	3.371	1.843	0.436
22	2	1	0	1	7.042	3.439	1.854	0.434
23	2	1	1	0	7.048	3.301	1.815	0.433
24	2	1	1	1	7.087	3.29	1.883	0.43

Table 5.11. The Results of Duncan's Test for System Condition HT (a = 0.05)

Significant Interaction: AB

Duncan Grouping	Mean	Factor Setting
A	9.05825	A1_b
B	8.86949	A1_B
C	7.29607	A2_B
C C		
C	7.25464	A2_b
D	7.07698	A3_B
D D		
D	7.07288	A3_b

Significant Interaction: AC

Duncan Grouping	Mean	Factor Setting
A	9.28879	A1_c
B	8.63895	A1_C
C	7.31931	A2_c
C C		
C	7.23140	A2_C
D	7.08639	A3_c
D D		
D	7.06346	A3_C

Significant Interaction: AD

Duncan Grouping	Mean	Factor Setting
A	9.28647	A1_D
B	8.64127	A1_d
C	7.34583	A2_D
D	7.20488	A2_d
E	7.07863	A3_D
E E		
E	7.07122	A3_d

Significant Interaction: AD

Duncan Grouping	Mean	Factor Setting
A	7.9557	bD
A A		
A	7.8516	BD
A A		
A	7.6435	Bd
A A		
A	7.6348	bd

Recommended Factor Setting: A3_BCd

Table 5.12. The Results of Duncan's Test for System Condition HL (a = 0.05)

Significant Interaction: AC

Duncan

Grouping	Mean	Factor Setting
A	4.86425	A1_c
B	4.57623	A1_C
C	3.55569	A2_c
C		
D C	3.49349	A2_C
D		
D E	3.39225	A3_c
E		
E	3.30891	A3_C

Significant Interaction: BD

Duncan

Grouping	Mean	Factor Setting
A	4.0244	bD
A		
B A	3.9419	BD
B		
B	3.7525	Bd
B		
B	3.7417	bd

Significant Interaction: CD

Duncan

Grouping	Mean	Factor Setting
A	4.97050	A1_D
B	4.46997	A1_d
C	3.59515	A2_D
D	3.45403	A2_d
D		
E D	3.38385	A3_D
E		
E	3.31731	A3_d

Recommended Factor Setting: A3_bCd, A3BCd

Table 5.13. The Results of Duncan's Test for System Condition LT (a = 0.05)

Significant Interaction: AB

Duncan Grouping	Mean	Factor Setting
A	2.31410	A1_b
B	2.24422	A1_B
C	1.92627	A2_b
C C		
C	1.89729	A2_B
D	1.84869	A3_B
D D		
D	1.84575	A3_b

Significant Interaction: AC

Duncan Grouping	Mean	Factor Setting
A	2.31954	A1_c
B	2.23878	A1_C
C	1.91775	A2_C
C C		
C	1.90581	A2_c
D	1.84800	A3_c
D D		
D	1.84644	A3_C

Significant Interaction: AD

Duncan Grouping	Mean	Factor Setting
A	2.36930	A1_D
B	2.18902	A1_d
C	1.93873	A2_D
D	1.88484	A2_d
D D		
D	1.87442	A3_D
E	1.82002	A3_d

Significant Interaction: AD

Duncan Grouping	Mean	Factor Setting
A	2.06528	CD
A A		
A	2.05635	cD
A A		
B A	1.99255	cd
B B		
B	1.93670	Cd

Recommended Factor Setting: A3_BCd

Table 5.14. The Results of Duncan's Test for System Condition LL (a = 0.05)

Significant Interaction: ABD

Duncan Grouping	Mean	Factor Setting
A	0.87225	A1_bD
B	0.79729	A1_BD
C	0.76369	A1_Bd
C	0.75992	A1_bd
D	0.46636	A2_bD
D		
E D	0.44945	A2_Bd
E D		
E D	0.44622	A2_bd
E		
E	0.44382	A2_BD
E		
E	0.44259	A3_bD
E		
E	0.43452	A3_Bd
E		
E	0.43191	A3_BD
E		
E	0.42888	A3_bd

Significant Interaction: ACD

Duncan Grouping	Mean	Factor Setting
A	0.83961	A1_cD
A		
A	0.82992	A1_CD
B	0.78746	A1_cd
C	0.73615	A1_Cd
D	0.46341	A2_cD
D		
E D	0.45404	A2_cd
E D		
E D F	0.44677	A2_CD
E D F		
E D F	0.44163	A2_Cd
E F		
E F	0.43901	A3_cD
E F		
E F	0.43550	A3_CD
E F		
E F	0.43451	A3_cd
F		
F	0.42889	A3_Cd

Recommended Factor Setting: A3_bCd, A3_BCd

Table 5.15. Best Factor level Combinations for Various System Conditions

Conditions	Significant Interactions	Best Factor Level Combinations
HT	AB, AC, AD, BD	A3_BCd
HL	AC, AD, BD	A3_bCd, A3_BCd
LT	AB, AC, AD, CD	A3_BCd
LL	ABD, ACD	A3_bCd, A3_BCd

5.2.2. Discussion

Level 1 for factor A (A1) represents the case in which the policy table is constructed using only feature ID as the state determination criterion. Since this does not take into account any information regarding the current buffer status of the machine cells that are able to perform the job agent's next operation, it makes sense that the other two settings of factor A (A2 and A3), in which additional information is considered (number of jobs in queue (NIQ) and estimate work in queue (WIQ)), outperformed A1 for all of the four system conditions. To further compare the cases of A2 and A3, for system condition HT, HL, and LT, it was found that A3 is better than the A2. For the cases using the A2 setting, the job agents use the feature ID and NIQ as the state determination criteria, whereas, for the cases of A3, the cell agents make a further estimate of the total work of those jobs in the queue (WIQ). The WIQ measure provides more details for a job agent to more precisely determine the states they encounter. For the remaining system condition LL, Duncan's test revealed no significant difference among the settings of A2_bCd, A2_BCd, and A2_BCD, and any set of settings with A3. This indicates that the advantage from incorporating WIQ has less of an impact under the LL system condition.

In tables 5.11 and 5.13, for significant interaction AB, the cases with A3 setting (A3_b and A3_B) are grouped together (group D), and the cases with A2 setting (A2_b and A2_B) are grouped together (group C), while the setting A1_b and A1_B are assigned to different groups, group A and B, respectively. In other words, for the system conditions involving jobs with tight due-dates (HT and LT), the number of ranges (factor B) for the reward function is not important for those cases with the A2 or A3 setting. However, if factor A is set at level 1, the use of more ranges for rewards is better.

Therefore, if this approach is applied to a system where the machine agents have no capability of providing their buffer status, the use of more ranges for reward functions is suggested.

Toward reward function development, factor C defines the size of the interval for each range of the reward function. In table 5.11, 5.12, and 5.13, for significant interaction AC, the cases with A3 setting (A3_c and A3_C) are in the same group (group D in table 5.11 and 5.13, group E in table 5.12), and the cases with A2 setting (A2_c and A2_C) are in group C, while the setting A1_c and A1_C are in different groups, group A and B, respectively. Therefore, for system conditions HT, HL, and LT, the levels of factor C do not affect the job tardiness when factor A is set either at level 2 or level 3, while the use of wider ranges is recommended for the cases using the A1 setting. This means, again, if the machine agent is unable to provide information concerning NIQ and WIQ, it is better to design the reward function with wider ranges under most system conditions.

For factor D, Duncan's test shows that level 1 (d) is either the same as, or better than, level 2 (D) when factor A is set at level 1 under any system condition. That means the reward magnitude should be set small. According to the simulation results, around one fourth of the completed jobs are tardy under system condition HT (the most heavy loading system with tight due date jobs), this indicates that the job agent receives a reward for its routing decision with high possibility (around 75%). That may be why the use of a small value for the reward resulted in better performance than a large value in this case.

5.2.3. Mean Tardiness of Prior Operations

In this experiment, it is assumed that the machine agent has knowledge of the estimated mean processing time (EMPT) of the operations that the machine is able to perform but has no prior knowledge about how much the magnitude of job tardiness may be. Therefore, EMPT is used as the measure to set the tardiness ranges for the reward function. However, job tardiness varies under different system conditions, even for the same system conditions job tardiness may vary from time to time. Therefore, EMPT is not a good measure to set the tardiness ranges for the reward function because EMPT is not adjusted with the changing system conditions. To overcome this issue, a suggested approach is to use the mean tardiness for prior operations (MTPO) as the measure to set the tardiness ranges for the reward function.

As described in Chapter 4, using an allowance factor, the job agent can determine an intermediate due date for each required operation of a job. The intermediate due date is used by the job agent to check if the corresponding operation is behind and therefore assign a tardiness value for this operation. The mean tardiness for a specific operation can then be computed and updated whenever the operation is performed. In the system of this study, there are seven operations for seven features (one operation for each feature). Thus there would be seven mean tardiness values updated in real-time. Table 5.16 shows an example of using MTPO to set the ranges for measuring the tardiness of an operation. Figure 5.5 compares the performance of using EMPT and MTPO as the measure for setting the ranges of the reward function. MTPO makes the job mean tardiness drop by as much as 15% under system condition HL, compared with using EMPT. This result proved that EMPT is not a good measure for designing the reward function since the

EMPT value is not altered with the system changes. This result also indicates that MTPO is a good measure for setting ranges of the reward function since MTPO is updated as system condition changes.

Table 5.16. An Example of 10-range Reward Function

Range		Reward/Penalty
1	Tardiness = 0	r = 1 (or r = 10)
2	$0 <$ Tardiness $< n \times$ MTPO	r = -1
3	$n \times$ MTPO $\leq$ Tardiness $< 2n \times$ MTPO	r = -2
4	$2n \times$ MTPO $\leq$ Tardiness $< 3n \times$ MTPO	r = -3
5	$3n \times$ MTPO $\leq$ Tardiness $< 4n \times$ MTPO	r = -4
6	$4n \times$ MTPO $\leq$ Tardiness $< 5n \times$ MTPO	r = -5
7	$5n \times$ MTPO $\leq$ Tardiness $< 6n \times$ MTPO	r = -6
8	$6n \times$ MTPO $\leq$ Tardiness $< 7n \times$ MTPO	r = -7
9	$7n \times$ MTPO $\leq$ Tardiness $< 8n \times$ MTPO	r = -8
10	$8n \times$ MTPO $\leq$ Tardiness	r = -9

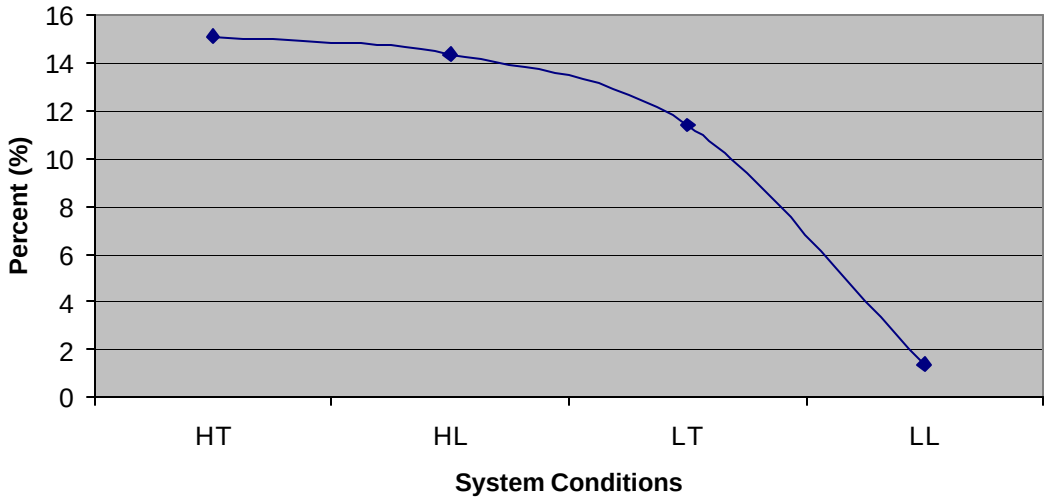

Figure 5.5. Performance Improvement (MTPO vs EMPT)

5.2.4. Traditional Routing Heuristics and the Q-learning Routing Policies

Table 5.17 provides a performance comparison for job routing by using the routing heuristics (NINQ and WINQ) and the routing policies learned by the Q-learning algorithm with EMPT and MTPO as a measure for setting tardiness ranges for the reward function and at the recommended setting (A3_BCd). It can be observed that the routing policy learned by the Q-learning algorithm MTPO as a measure for setting tardiness ranges and at A3_BCd is very competitive under system condition HT and LT. That is, the Q-learning performs well when the system is operating under tight due-dates. The percentages of tardy jobs are 22.3%, 10.3%, 11.7%, and 3.3% for the system condition HT, HL, LT, and LL, respectively. Although the mean tardiness of each job for system condition HL (2.827) is higher than the one for system condition LT (1.608), the percentage of tardy jobs for system condition LT (11.7%) is higher than the one for system condition HL (10.3%). This may indicate that in cases where there are a greater number of tardy jobs the Q-learning performs better. In the reward function, there are several levels of penalty to determine the job tardiness but only one level of reward. In other words, the reward function does not provide a measure differentiating the value of a decision is when it is good, but it does distinguish between cases is when the decision is bad. More than likely, that is why the Q-learning does not perform well for the light due date cases.

Table 5.17. Performance Comparison: Heuristics versus Q-Learning Policies

	HT	HL	LT	LL
NINQ (Heuristic)	6.324	2.367	2.171	0.379
WINQ (Heuristic)	5.943	2.113	2.094	0.341
EMPT (Recommended Setting: A3_BCd)	7.048	3.301	1.815	0.433
MTPO (Recommended Setting: A3_BCd)	5.983	2.827	1.608	0.427

CHAPTER VI

CONCLUSIONS

6.1. Summary and Conclusions

Reinforcement learning (RL) has recently become an active research interest within the field of machine learning. Although there have been several examples demonstrating the usefulness of RL, its application to manufacturing systems has not yet been fully explored. In addition, most of the current agent-based research in manufacturing systems focuses on the issues of negotiation and cooperation among agents, overlooking learning as a means for giving an agent an ability to increase its perceived intelligence for making decisions. This research investigated how the Q-learning algorithm can be used by job agents to generate policies for making real-time routing decisions and by machine agents to discover a policy for selecting a proper dispatching rule.

Several recommendations were derived from the results of this research. For applying Q-learning to dispatching rule selection, more states in the policy table and more ranges for the reward function essentially improve learning performance. When job due dates are tight, the use of wider ranges for determining the states and for determining penalties resulted in better performance than use of narrow ranges. In addition, the reward magnitude proved crucial under such conditions. If most of the completed jobs are tardy, a larger value for the reward magnitude is preferred.

When applying the Q-learning method to the job routing problem, it is strongly recommended that the current buffer status of the machines be included as one part of the state determination criteria. When the buffer status is included as one of the state determination criteria, then the number of ranges and size of each range for the reward function do not seem to have much effect on system performance in terms of mean job tardiness. However, if buffer status is not considered as part of the state determination criteria, increasing the number of ranges used and the width of each range is recommended.

The reward magnitude also proved crucial in this problem with the experimental results recommending the use of a small reward magnitude setting (compared to the penalty magnitude setting). The ratio of the number of tardy jobs to total number of completed jobs may need to be taken into account for setting this reward/penalty magnitude. In this study, the ratio for the worst case (HT condition) was 25%, where a small reward magnitude setting was recommended. If the ratio is large, for example, more than 50% of the jobs are tardy, then a larger reward setting is suggested so that learning can be reinforced from the fewer good decisions. In addition, it was determined that the use of the mean tardiness computed from previous job's operations (MTPO) proved much better than the use of the estimated mean processing time (EMPT) as the measure for setting tardiness ranges of the reward function. Therefore, a mechanism for collecting, recording and updating mean tardiness values for system operations is highly recommended.

The conclusions of this study are based solely on the experimental results of the simulation systems considered in this research. The simulation study was conducted

under system loading conditions with machine utilizations of 70% and 90%, and job due date tightness employing allowance factors of 1.5 and 2. These parameter settings for various system conditions are important and can be used as reference materials to apply the conclusions of this study to other systems. Therefore, an understanding of the loading conditions and allowance factors for any other system is required. On the other hand, conclusions regarding how the percentage of tardy jobs to total completed jobs influences the reward magnitude settings and how the use of MTPO benefits the learning performance are applicable for any other system.

6.2. Directions for Future Research

Future research will be needed in a number of areas to fully explore the application of reinforcement learning in the area of production scheduling. In this section, several issues for future research directions are addressed.

In this research, we dealt with the problem concerning fixed-sequencing routing flexibility (Table 2.2). That is, the operations of a job must be performed in a fixed sequence, but there can be more than one machine capable of processing any given operation. To further extend this study, flexible sequencing of the operations may be considered. This will increase the number of possible routes. To deal with the problem, the Q-learning algorithm can again be applied to construct a policy table for selecting an operation sequence. The selection of an operation sequence will be based on the current policy in use by the Q-learning algorithm. Once a sequence is selected, the approach in this research can be implemented for constructing a policy table of selecting machines. However, since one policy table for selecting machines needs to be learned for one

operation sequence. Therefore, if there are numerous possible operation sequences, then it will be very time-consuming for the agent to learn all the machine-selection policy tables for all the sequences.

In this study, the agent-negotiation schema is not complex. The job agent makes a routing decision based on the bids provided by the machine cell agents. Each decision takes only one round of message exchange (requesting – bidding). However, a complicated negotiation schema may require more than one round of message exchange to make a routing decision. For each message submission, the agent actually makes a negotiation decision and then a routing decision is derived from these negotiation decisions. The intermediate due date for an operation of a job can be used to determine the reward or penalty for a routing decision (using the proposed approach in this research). When applying reinforcement learning to make a negotiation decision, some problems need to be considered. First, decisions in the early rounds of negotiation may lead to either a good or a bad routing decision. In addition, since the negotiation decisions are made sequentially, one bad negotiation decision may result in a bad routing decision even though all the other negotiation decisions were good. Therefore, as more rounds of negotiation take place, it will be difficult to identify if a negotiation decision should be rewarded or penalized and to determine how much reward/penalty to apply for a decision.. Some negotiation schema involve employing a coordination agent who is responsible for solving the conflict among agents. In such cases, the decision-making policy derived by reinforcement learning must take into account the relationship of the coordination agent to the other agents based on the negotiation schema implemented in the system. The

learning policy table will become more complicated (more states and more actions) and more time will be needed for learning a proper policy.

The reinforcement learning approach applied in this research is called one-step tabular Q-learning method. It is one of the most widely used reinforcement learning methods (Sutton and Barto, 1999). Future work may focus on applying some other reinforcement learning methods such as the Sarsa algorithm, R-learning algorithm, or actor-critic methods to the same scheduling problems of this research. Details for these methods can be found in Sutton and Barto (1999). The issue of exploration and exploitation may be crucial for reinforcement learning. In this research, the exploration method implemented in this research is the ε-greedy method. Future research may also focus on implementing other exploration strategies. Details for these other exploration strategies can be found in the observations by Mahadevan and Kaelbling (1996).

The system objective in this research is minimizing mean tardiness, which is one of the most popular objectives for production scheduling problems (based on a review of the literature). Future studies may focus on applying reinforcement learning approaches to the scheduling problems for other popular objectives such as minimizing mean flow time and minimizing number of tardy jobs. The reward function proposed in this study must be modified to fit different objectives. For minimizing mean flow time, the difficulties will be in how to determine when an agent's decision should be rewarded or penalized. For minimizing number of tardy job, the reward function can be designed as assigning one positive unit for an early job and one negative unit for a late job. This case is much simpler than the ones with the objectives of minimizing mean flow time and mean tardiness.

REFERENCES

Ariz, Y. (1995). On-Line Scheduling in a Multi-Cell Flexible Manufacturing System. International Journal of Production Research, 33, 12, 3283-3300.

Aydin, M. E. and Oztemel, E. (2000). Dynamic Job-Shop Scheduling using Reinforcement Learning Agents. Robotics and Autonomous Systems, 33, 2, 169-178.

Bowden, R. O. and Bullington, S. F. (1996). Development of Manufacturing Control Strategies using Unsupervised Machine Learning. IIE Transactions, 28, 4, 319-331.

Brenner, W., Zarnekow, R., and Witting, H. (1998). Intelligent Software Agents: Foundations and Applications. Springer Verlag.

Brucker, P. (2001). Scheduling Algorithms. Springer Verlag.

Caskey, K. and Storch, R. L. (1996). Heterogeneous Dispatching Rules in Job and Flow Shops. Production Planning and Control, 7, 4, 351-361.

Chan, F. T. S. (2001). The Effects of Routing Flexibility on a Flexible Manufacturing System. International Journal of Computer Integrated Manufacturing, 14, 5, 431-445.

Chandra, J. and Talavage, J. (1991). Intelligent Dispatching for Flexible Manufacturing. International Journal of Production Research, 29, 11, 2259-2278.

Chang, P-T. and Lo, Y-T. (2001). Modeling of Job-Shop Scheduling with Multiple Quantitative and Qualitative Objectives and a GA/TS Mixture Approach. International Journal of Computer Integrated Manufacturing, 14, 4, 367-384.

Chiu, C. and Yih, Y. (1995). A Learning-Based Methodology for Dynamic Scheduling in Distributed Manufacturing Systems. International Journal of Production Research, 33, 11, 3217-3232.

Choi, R. H. and Malstrom, E. M. (1988). Evaluation of Traditional Work Scheduling Rules in a Flexible Manufacturing System with a Physical Simulator. Journal of Manufacturing Systems, 7, 1, 33-45.

Chryssolouris, G. and Subramaniam, V. (2001). Dynamic Scheduling of Manufacturing Job Shops using Genetic Algorithm. Journal of Intelligent Manufacturing, 12, 3, 281-293.

Cicirello and Smith (2001). 5th International Symposium on Autonomous Decentralized Systems, IEEE Computer Society Press, March, 2001.

Crites, R. H. and Barto, A.G. (1996). Improving Elevator Performance using Reinforcement Learning. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E. (Eds.), Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference, MIT Press, Cambridge, MA, 1017-1023.

Crowe, T.J. and Stahlman, E.J. (1995). A Proposed Structure for Distributed Shopfloor Control. Integrated Manufacturing Systems, 6, 6, 31-36.

Das, S. K. (1996). The Measurement of Flexibility in Manufacturing Systems. The International Journal of Flexible Manufacturing Systems, 8, 1, 67-93.

Das, T. K., Gosavi, A., Mahadevan, S., and Marchalleck, N. (1999). Solving Semi-Markov Decision Problems Using Average Reward Reinforcement Learning. Management Science, 45, 4, 560-574.

Dewan, P. and Joshi, S. (2000). Dynamic Single Machine Scheduling Under Distributed Decision Making. International Journal of Production Research, 38, 16, 3759-3777.

Dewan, P. and Joshi, S. (2001). Implementation of an Auction-Based Distributed Scheduling Model for a Dynamic Job Shop Environment. International Journal of Computer Integrated Manufacturing, 14, 5, 446-456.

Dilts, D.M., Boyd, N.P., and Whorms, H.H. (1991). The Evolution of Control Architectures for Automated Manufacturing Systems. Journal of Manufacturing Systems, 10, 1, 79-93.

Duffie, N.A. and Prabhu, V.V. (1994). Real-Time Distributed Scheduling of Heterarchical Manufacturing Systems. Journal of Manufacturing Systems, 13, 2, 94-107.

Fang, J. and Xi, Y. (1997). A Rolling Horizon Job Shop Rescheduling Strategy in the Dynamic Environment. International Journal of Advanced Manufacturing Technology, 13, 3, 227-232.

Ishii, N. and Talavage, J. (1991). A Transient-Based Real-Time Scheduling algorithm in FMS. International Journal of Production Research, 29, 12, 2501-2520.

Jahangirian, M. and Conroy, G. V. (2000). Intelligent Dynamic Scheduling System: the Application of Genetic Algorithms. Integrated Manufacturing Systems, 11, 4, 247-257.

Jeong, K.-C. and Kim, Y.-D. (1998). A Real-Time Scheduling Mechanism for a Flexible Manufacturing System: Using Simulation and Dispatching Rules. International Journal of Production Research, 36, 9, 2609-2626.

Jian, A. K. and Elmaraghy, H. A. (1997). Production Scheduling/Rescheduling in Flexible Manufacturing. International Journal of Production Research, 35, 1, 281-309.

Jones, A. T. and McLean, C. R. (1986). A Proposed Hierarchical Control Model for Automated Manufacturing Systems. Journal of Manufacturing Systems, 5, 1, 15-25.

Khoo, L. P., Lee, S. G., and Yin, X. F. (2000). A Prototype Genetic Algorithm-Enhanced Multi-Objective Scheduler for Manufacturing Systems. International Journal of Advanced Manufacturing Technology, 16, 2, 131-138.

Kim, M. H. and Kim, Y.-D. (1994). Simulation-Based Real-Time Scheduling in a Flexible Manufacturing System. Journal of Manufacturing Systems, 13, 2, 85-93.

Lin, S-C., Goodman, E. D., and Punch, W. F. (1997). A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems. Proceedings of the 7th International Conference on Genetic Algorithms. San Francisco, 481-488.

Lin, G. Y. and Solberg, J. J. (1991). Effective of Flexible Routing Control. The International Journal of Flexible Manufacturing Systems, 3, 3-4, 189-211.

Lin, G.Y. and Solberg, J.J. (1992). Integrated Shop Floor Control Using Autonomous Agents. IIE Transactions, 24, 3, 57-71.

Lin, G.Y. and Solberg, J.J. (1994). An Agent-Based Flexible Routing Manufacturing Control Simulation System. Proceedings of the 1994 Winter Simulation Conference, 970-977.

Liu, H. and Dong, J. (1996). Dispatching Rule Selection Using Artificial Neural Networks for Dynamic Planning and Scheduling. Journal of Intelligent Manufacturing, 7, 3, 243-150.

Mahadevan, S. and Kaelbling, L. P. (1996). The NSF Workshop on Reinforcement Learning: Summary and Observations. AI Magazine, Winter, 89-97.

Mahadevan, S., Khaleeli, N., and Marchalleck, N. (1997a). Designing Agent Controllers using Discrete-Event Markov Models. AAAI Fall Symposium on Model-Directed Autonomous Systems, MIT, Cambridge.

Mahadevan, S., Marchalleck, N., Das, T. K., and Gosavi, A. (1997b). Self-Improving Factory Simulation using Continuous-time Average-Reward Reinforcement Learning. Proceedings of the 4th International Machine Learning Conference, 202-210.

Mahadevan, S. and Theocharous, G. (1998). Optimizing Production Manufacturing using Reinforcement Learning. The 11th International FLAIRS Conference, AAAI Press, 372-377.

Mahmoodi, F., Mosier, C. T., and Guerin, R. E. (1996). The Effect of Combining Simple Priority Heuristics in Flow-Dominant Shops. International Journal of Production Research, 34, 3, 819-839.

Matsuura, H., Tsubone, H., and Kanezashi, M. (1993). Sequencing, Dsipatching, and Switching in a Dynamic Manufacturing Environment. International Journal of Production Research, 31, 7, 1671-1688.

Maturana, F., Shen, W., Norrie, D. H. (1999). MetaMorph: An Adaptive Agent-Based Architecture for Intelligent Manuacturing. International Journal of Production Research, 37, 10, 2159-2174.

Moon, D. H. and Christy, D. P. (1998). A Simulation Study for Dynamic Scheduling in a Hybrid Assembly/Job Shop Considering the JIT Context. Production Planning & Control. 9, 6, 532-541.

Morton, T. E. and Pentico, D. W. (1993) Heuristic Scheduling Systems With Applications to Production Systems and Project Management. John Wiley & Sons, Inc.

Murch, R. and Johnson, T. (1998). Intelligent Software Agents. Prentice Hall.

Okubo, H., Jiahua, W., and Onari, H. (2000). Characteristics of Distributed Autonomous Production Control. International Journal of Production Research, 28, 17, 4205-4215.

Nakasuka, S. and Yoshida, T. (1992). Dynamic Scheduling System Utilizing Machine Learning as A Knowledge Acquisition Tool. International Journal of Production Research, 30, 2, 411-431.

Okubo, H., Jiahua, W., and Onari, H. (2000). Characteristics of Distributed Autonomous Production Control. International Journal of Production Research, 28, 17, 4205-4215.

Ouelhadj, D., Hanachi, C., and Bouzouia, B. (1998) Multi-Agent Systems for Dynamic Scheduling and Control in Manufacturing Cells. Proceedings of the 1998 IEEE International Conference on Robotics & Automation, May 1998, Leuven, Belgium, 2128-2133.

Ouelhadj, D., Hanachi, C., Bouzouia, B., Moualek, A., and Farhi, A. (1999). A Multi-Contract Net Protocol for Dynamic Scheduling in Flexible Manufacturing Systems. Proceedings of the 1999 IEEE International Conference on Robotics & Automation, May 1999, Detroit, Chicago, 1114-1119.

Ottaway, T.A. and Burns, J.R. (2000) An Adaptive Production Control System Utilizing Agent Technology. International Journal of Production Research, 38, 4, 721-737.

Palmer, D. A. (1996). Integrating Genetic Algorithms, Clustering and Reinforcement Learning to Evolve Manufacturing Control Knowledge. Master Thesis, Mississippi State University.

Panwalkar, S. S. and Iskander, W. (1977). A Survey of Scheduling Rules. Operations Research, 25, 1, 45-62.

Park, S. C., Raman, N., and Shaw, M. J. (1997). Adaptive Scheduling in Dynamic Flexible Manufacturing Systems: A Dynamic Rule Selection Approach. IEEE Transactions on Robotics and Automation, 13, 4, 486-502.

Paternina-Arboleda, C. D. and Das, T. K. (2001). Intelligent Dynamic Control Policies for Serial Production Lines. IIE Transactions, 33, 1, 65-77.

Pierreval, H. and Mebarki, N. (1997). Dynamic Selection of Dispatching Rules for Manufacturing System Scheduling. International journal of Production Research, 35, 6, 1575-1591.

Pinedo, M. (1995). Scheduling Theory, Algorithms, and Systems. Prentice Hall.

Piramuthu, S., Shaw, M., and Fulkerson, B. (2000). Information-Based Dynamic Manufacturing System Scheduling. International Journal of Flexible Manufacturing Systems, 12, 2-3, 219-234.

Priore, P., Fuente, D. D. L., Gomez, A., and Puente, J. (2001a). A Review of Machine Learning in Dynamic Scheduling of Flexible Manufacturing Systems. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 15, 3, 251-263.

Priore, P., Fuente, D. D. L., Gomez, A., and Puente, J. (2001b). Dynamic Scheduling of Manufacturing Systems with Machine Learning. International Journal of Foundations of Computer Science, 12, 6, 751-762.

Ro, I.-K. and Kim, J.-I. (1990). Multi-Criteria Operational Control Rules in Flexible Manufacturing Systems (FMSs). International Journal of Production Research, 28, 1, 47-63.

Rossi, A. and Dini, G. (2000). Dynamic Scheduling of FMS using a Real-Time Genetic Algorithm. International Journal of Production Research, 38, 1, 1-20.

Saad, A., Kawamura, K., and Biswas, G. (1997). Performance Evaluation of Contract Net-Based Heterarchical Scheduling for Flexible Manufacturing Systems. Intelligent Autonomous and Soft Computing, 3, 3, 229-248.

Sabuncuoglu, I. and Karabuk, S. (1999). Rescheduling Frequency in an FMS with Uncertain Processing Times and Unreliable Machines. Journal of Manufacturing Systems, 18, 4, 268-283.

Saygin, C., Chen, F. F., and Singh, J. (2001). Real-Time Manipulation of Alternative Routings in Flexible Manufacturing Systems: A Simulation Study. International Journal of Advanced Manufacturing Technology, 18, 10, 755-763.

Shafaei, R. and Brunn, P. (1999a). Workshop Scheduling using Practical (Inaccurate) Data Part 1: The Performance of Heuristic Scheduling Rules in a Dynamic Job Shop Environment using a Rolling Time Horizon Approach. International Journal of Production Research, 37, 17, 3913-3925.

Shafaei, R. and Brunn, P. (1999b). Workshop Scheduling using Practical (Inaccurate) Data Part 2: An Investigation of the Robustness of Scheduling Rules in a Dynamic and Stochastic Environment. International Journal of Production Research, 37, 18, 4105-4117.

Shafaei, R. and Brunn, P. (2000) Workshop Scheduling using Practical (Inaccurate) Data Part 3: A Framwork to Integrate Job Releasing, Routing and Scheduling Functions to Create a Robust Predictive Schedule. International Journal of Production Research, 38, 1, 85-99.

Shaw, J. M. (1988) Dynamic Scheduling in Cellular Manufacturing Systems: A Framework for Network Decision Making. Journal of Manufacturing Systems, 7, 2, 83-94.

Shaw, M. J., Park, S., and Raman, N. (1992). Intelligent Scheduling with Machine Learning Capabilities: The Induction of Scheduling Knowledge. IIE Transactions, 24, 2, 156-168.

Shen, W. and Norrie, D.H. (1999). Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. Knowledge and Information Systems: an International Journal, 1, 2, 129-156.

Shen,W., Norrie, D.H., and Barthes J.-P. A. (2000). Multi-Agent System for Concurrent Intelligent Design and Manufacturing. Taylor & Francis.

Shmilovici, A. and Maimon, O. Z. (1992). Heuristic for Dynamic Selection and Routing of Parts in an FMS. Journal of Manufacturing Systems, 11, 4, 285-296.

Sim S. K., Yeo, K. T., and Lee, W. H. (1994). An Expert Neural Network System for Dynamic Job Shop Scheduling. International Journal of Production Research, 32, 8, 1759-1773.

Singh, S. P. and Bertsekas, D. (1997). Reinforcement Learning for Dynamic Channel Allocation in Cellular Telephone Systems. Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference, MIT Press, Cambridge, MA, 974-980.

Smith, R. (1980). The Contract Net Protocol: High Level Communication and Control in Distributed Problem Solver. IEEE Transactions on Computers, 29, 12, 1104-1113.

Sousa, P. and Ramos, C. (1996). A Holonic Approach for Task Scheduling in Manufacturing Systems. Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, April 1996, 2511-2516.

Sousa, P. and Ramos, C. (1998). A Dynamic Scheduling Holon for Manufacturing Orders. Journal of Intelligent Manufacturing, 9, 2, 107-112.

Sousa, P. and Ramos, C. (1999). A Distributed Architecture and Negotiation Protocol for Scheduling in Manufacturing Systems. Computers in Industry, 38, 2, 103-113.

Subramaniam, V., Lee, G. K., Ramesh, T., Hong, G. S., and Wong, Y. S. (2000a). Machine Selection Rules in a Dynamic Job Shop. International Journal of Advanced Manufacturing Technology, 16, 12, 902-908.

Subramaniam, V., Lee, G. K., Hong, G. S., Wong, Y. S., and Ramesh, T. (2000b). Dynamic Selection of Dispatching Rules for Job Shop Scheduling. Production Planning & Control, 11, 1, 73-81.

Sun, D. and Lin L. (1994). A Dynamic Job Shop Scheduling Framework: A Backward Approach. International Journal of Production Research, 32, 4, 967-985.

Suresh, V. and Chaudhuri, D. (1993). Dynamic Scheduling – A Survey of Reseach. International Journal of Production Economics, 32, 1, 53-63.

Sutton, R. S. (1996). Generalization in Reinforcement Learning: Successful Examples using Spare Coarse Coding. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E. (Eds.), Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference, MIT Press, Cambridge, MA, 1038-1044.

Sutton, R. S. and Barto, A. G. (1999). Reinforcement Learning: An Introduction. The MIT Press.

Tesauro, G. (1995). Temporal Difference Learning and TD-Gammon. Commun. of the ACM, 38, 3, 58-67.

Wang, Y.-C. and Usher, J. M. (2002) A Study of Reinforcement Learning to Dynamic Single-Machine Job Dispatching. Proceedings of the 6th International Engineering Design and Automation.

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

Weiss, G., 1999, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press.

Xue, D., Sun, J., and Norrie, D. H. (2001). An Intelligent Optimal Production Scheduling Approach using Constraint-based Search and Agent-based Collaboration. Computers in Industry, 46, 2, 209-231.

Yao, D. D. and Pei, F. F. (1990). Flexible Parts Routing in Manufacturing Systems, IIE Transactions, 22, 1, 48-55.

Zhang, W. and Dietterich, T. G. (1995). A Reinforcement Learning Approach to Job-Shop Scheduling. Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1114-1120.